UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS
CARRERA DE INGENIERÍA AUTOMOTRIZ

TRABAJO DE GRADO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO AUTOMOTRIZ

TEMA: “ELABORACIÓN DE UNA RED MULTIPLEXADA CAN EN BASE A MICROCONTROLADORES ARDUINO PARA EL CONTROL DE SISTEMAS EMBEBIDOS AUTOMOTRICES”

AUTOR: JEFFERSON JAVIER HUERA LOMAS

DIRECTOR: ING. RAMIRO ANDRÉS ROSERO AÑAZCO MSc.

Ibarra, 2023

CERTIFICADO

ACEPTACIÓN DEL DIRECTOR

En mi calidad de director del plan de trabajo de grado, previo a la obtención del título de Ingeniería Automotriz, nombrado por el Honorable Consejo Directivo de la Facultad de Ingeniería en Ciencias Aplicadas.

CERTIFICO:

Que una vez analizado el plan de grado cuyo título es **‘‘Elaboración de una red multiplexada CAN en base a microcontroladores Arduino para el control de sistemas embebidos automotrices’’** presentado por el señor: **Huera Lomas Jefferson Javier** con número de cédula 0401667449, doy fe que dicho trabajo reúne los requisitos y méritos suficientes para ser sometido a presentación pública y evaluación por parte de los señores integrantes del jurado examinador que se designe.

En la ciudad de Ibarra, a los 28 días del mes septiembre del 2023.

Atentamente

Ramiro Rosero Añazco

Firmado digitalmente por Ramiro Rosero Añazco
Fecha: 2023.10.02 11:27:37 -05'00'

Ing. Ramiro Andrés Rosero Añazco, MSc.

DIRECTOR DEL TRABAJO DE GRADO

UNIVERSIDAD TÉCNICA DEL NORTE
BIBLIOTECA UNIVERSITARIA

AUTORIZACIÓN DE USO Y PUBLICACIÓN A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

1. IDENTIFICACIÓN DE LA OBRA

En cumplimiento del Art. 144 de la Ley de Educación Superior, hago la entrega del presente trabajo a la Universidad Técnica del Norte para que sea publicado en el Repositorio Digital Institucional, para lo cual pongo a disposición la siguiente información:

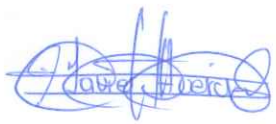
DATOS DE CONTACTO			
CEDULA DE IDENTIDAD:	040166744-9		
APELLIDOS Y NOMBRES:	Huera Lomas Jefferson Javier		
DIRECCIÓN:	San Pedro de Huaca, Parroquia Mariscal Sucre, Barrio Solferino		
EMAIL:	jjhueral@utn.edu.ec		
TELÉFONO FIJO:	223-4078	**TELÉFONO MÓVIL:**	0968997419
DATOS DE LA OBRA			
TÍTULO:	ELABORACIÓN DE UNA RED MULTIPLEXADA CAN EN BASE A MICROCONTROLADORES ARDUINO PARA EL CONTROL DE SISTEMAS EMBEBIDOS AUTOMOTRICES		
AUTOR:	Huera Lomas Jefferson Javier		
FECHA:	28 de septiembre del 2023		
SOLO PARA TRABAJOS DE GRADO			
PROGRAMA:	■ PREGRADO ☐ POSGRADO		
TITILO POR EL QUE OPTA:	INGENIERÍA AUTOMOTRIZ		
ASESOR/DIRECTOR:	Ing. Ramiro Andrés Rosero Añazco, MSc.		

2. CONSTANCIAS

El autor manifiesta que la obra objeto de la presente autorización es original y se la desarrolló, sin violar derechos de autor de terceros, por lo tanto, la obra es original y que es el titular de los derechos patrimoniales, por lo que asume la responsabilidad sobre el contenido de la misma y saldrá en defensa de la Universidad en caso de reclamación por parte de terceros.

Ibarra, a los 28 días del mes de septiembre del 2023

AUTOR:



Jefferson Javier Huera Lomas
C.I 0401667449

DEDICATORIA

El presente trabajo de grado se lo dedico principalmente a mis padres: Segundo Huera e Inés Lomas, quienes a lo largo de mi vida me han inculcado valiosos valores, son mi mayor inspiración para seguir adelante y superarme cada día, siendo los pilares que me han ayudado a realizar todos los proyectos que me he propuesto.

Mi papá, quien con su incansable perseverancia me ha enseñado que el trabajo honrado en algún momento trae grandes recompensas, siendo una persona admirable que me ha dado el ejemplo de ayudar a los demás y no rendirme.

Mi mamá, quien con su amor y apoyo incondicional han hecho de mi la persona que soy hoy en día, por estar siempre pendiente de mis situaciones, motivarme y encomendar mi futuro a Dios mediante la oración.

A mi hermano Kevin y compañero de vida, quien ha estado junto a mí en los buenos y malos momentos, y que me ha enseñado que la vida se encarga de encaminarnos hacia la felicidad.

A mi hermana Nathaly, quien con su amabilidad y cariño me ha motivado a no rendirme para culminar esta carrera Universitaria como parte de una promesa que guardo en el corazón.

Huera Lomas Jefferson Javier

AGRADECIMIENTO

En primer lugar, doy gracias a Dios por orientarme en las elecciones que he tomado en toda mi vida, las cuales me han llevado a este momento en el que puedo presentar mi trabajo de titulación, y también por darme fuerzas para afrontar las dificultades que se me han presentado.

Gracias a mis padres Segundo Huera e Inés Lomas por la confianza y apoyo brindado en mis decisiones personales. Agradezco todo el esfuerzo y sacrificio realizado para brindarme una educación de calidad, y también por la comprensión en los diferentes problemas que se me han presentado, en donde me han sabido dar valiosos consejos para salir a delante y crecer como persona.

Quiero agradecer a mis hermanos, Kevin y Nathaly, por todo el apoyo y el acompañamiento en los momentos felices y difíciles de mi vida.

Deseo expresar mis más sinceros agradecimientos a la Universidad Técnica del Norte por ser la institución que me abrió las puertas para seguir una carrera universitaria, de igual manera a la carrera de Ingeniería Automotriz y cada uno de los docentes por haberme guiado en el proceso de adquirir sólidos conocimientos que me servirán para desempeñarme en el campo profesional.

Agradezco al Ing. Andrés Cevallos, por haberme encaminado para desarrollar esta tesis, y de igual manera a mi director, el Ing. Ramiro Rosero, quien con sus enseñanzas, consejos y sugerencias me ha orientado en la elaboración del presente trabajo de grado.

Huera Lomas Jefferson Javier

ÍNDICE DE CONTENIDOS

PÁGINA

ÍNDICE DE CONTENIDOS ..... **vii**
ÍNDICE DE FIGURAS ..... **x**
ÍNDICE DE TABLAS ..... **xii**
ÍNDICE DE ANEXOS ..... **xiii**
ÍNDICE DE ECUACIONES ..... **xiv**
RESUMEN ..... **xv**
ABSTRACT ..... **xvii**
INTRODUCCIÓN ..... **xviii**
CAPÍTULO I **1**
1. PERFIL DEL PROYECTO **1**
1.1. OBJETIVOS 1
1.1.1. Objetivo general 1
1.1.2. Objetivos específicos 1
1.2. JUSTIFICACIÓN 1
1.3. ANTECEDENTES 2
1.4. PLANTEAMIENTO DEL PROBLEMA 3
1.5. ALCANCE 4
2. REVISIÓN BIBLIOGRÁFICA **6**
2.1. REDES MULTIPLEXADAS AUTOMOTRICES 6
2.1.1. Principios de electrónica en redes multiplexadas 7
2.1.1.1. Electrónica analógica y digital 7
2.1.1.2. Codificación numérica 7
2.1.1.3. Unidades de información 7
2.1.1.4. Trama de comunicación 8
2.1.1.5. Transmisión de mensaje 8
2.1.1.6. Velocidad de trasmisión de datos 8
2.1.2. Componentes de una red multiplexada 9
2.1.3. Topología de redes multiplexadas 10
2.1.4. Ventajas de la utilización de redes multiplexadas 11
2.2. PROTOCOLOS DE COMUNICACIÓN AUTOMOTRICES 12
2.2.1. Protocolos de comunicación 13
2.2.2. Redes inalámbricas 14
2.2.3. Otros protocolos de comunicación 14
2.3. RED CAN-BUS 15
2.3.1. Desarrollo de la red CAN-BUS 15
2.3.2. Componentes de una red CAN-BUS 16
2.3.2.1. Línea de comunicación o Bus de datos 16
2.3.2.2. Terminadores o resistencias 16
2.3.2.3. Unidad de control electrónico 16
2.3.3. Topología Bus 17
2.3.4. Tipos de redes CAN-BUS 18
2.3.5. Velocidad de transmisión de datos en redes CAN-BUS 19
2.3.6. Protocolo de comunicación CAN-BUS 19
2.3.6.1. Direccionamiento del mensaje 19
2.3.6.2. Control de acceso al Bus de datos 20
2.3.6.3. Formato de mensaje 20

2.3.7. Fallos en el protocolo de comunicación CAN-BUS 22
2.3.8. Ventajas de la red CAN-BUS 23
2.4. SISTEMAS EMBEBIDOS 23
2.4.1. Características de un sistema embebido 24
2.4.2. Hardware de un sistema embebido 24
2.4.3. Software en un sistema embebido 25
2.5. MICROCONTROLADORES PROGRAMABLES 25
2.5.1. Hardware Arduino 25
2.5.1.1. Características de una placa Arduino 26
2.5.2. Software Arduino IDE 28
2.5.2.1. Programar 28
2.5.2.2. Librerías 28
2.5.3. CAN-BUS Shield 29
2.5.3.1. Descripción general del hardware CAN-BUS Shield 29
2.5.3.2. Versiones de la placa CAN-BUS Shield 30
2.5.3.3. Funciones de la placa CAN-BUS Shield 30
2.6. SISTEMAS DE SEGURIDAD ACTIVA Y CONTROL DE LA CARROCERÍA 31
2.6.1. Seguridad activa 31
2.6.1.1. Sistema de luces 31
2.6.1.2. Sistema de iluminación adaptativa 33
2.6.1.3. Sistema de advertencia de colisión 33
2.6.2. Sistemas de control de la carrocería relacionados con la seguridad activa. 33
2.6.2.1. Sistema limpiaparabrisas 33
2.6.2.2. Sistema de dirección asistida electrónica 34
CAPÍTULO II 35
3. MATERIALES Y MÉTODOS 35
3.1. METODOLOGÍA 35
3.1.1. Diseño del sistema multiplexado de la red 36
3.1.1.1. Análisis del funcionamiento de una red CAN 37
3.1.1.2. Materiales y equipos 37
3.1.1.3. Diagramas del hardware que interviene en la conexión de red 42
3.1.1.4. Diseño del módulo CAN en base a microcontroladores Arduino 44
3.1.1.5. Selección del diseño base de la red CAN 46
3.1.2. Desarrollo de la red, algoritmo para establecer una comunicación entre módulos CAN por medio de programación 47
3.1.2.1. Materiales y equipos 48
3.1.2.2. Inicialización CAN (comunicación unidireccional) 48
3.1.2.3. Análisis del principio de comunicación bidireccional entre módulos CAN 51
3.1.2.4. Comunicación bidireccional entre dos módulos CAN 52
3.1.2.5. Comunicación bidireccional con más de dos módulos CAN 56
3.1.3. Diseño e implementación del sistema embebido a controlar en la red 59
3.1.3.1. Factores que intervienen en el desempeño de los sistemas embebidos automotrices 60
3.1.3.2. Sistemas de control y subsistemas del proyecto 61
3.1.3.3. Materiales y equipos 62
3.1.3.4. Funcionalidad individual de cada módulo CAN 65
3.1.3.5. Diseño del circuito electrónico de la red 66

3.1.3.6. Programación general del sistema embebido 69
CAPÍTULO III 71
4. RESULTADOS Y DISCUSIONES 71
4.1. RESULTADOS DEL DISEÑO Y CONEXIÓN DE MÓDULOS CAN AL BUS DE DATOS 71
4.1.1. Conexión Arduino Mega 2560 y placa CAN Bus Shield 71
4.1.2. Conexión de módulos CAN a la línea de comunicación 72
4.2. RESULTADOS DE LA COMUNICACIÓN ENTRE MÓDULOS CAN 72
4.2.1. Comunicación unidireccional y señal transmitida 72
4.2.2. Comunicación entre múltiples módulos CAN 75
4.2.2.1. Función attachInterrupt 75
4.2.2.2. Función millis 76
4.2.2.3. Interrupción por timer 77
4.2.3. Fórmula para obtener la frecuencia (Hz) considerando los periodos de transmisión 79
4.3. RESULTADOS DEL DISEÑO E IMPLEMENTACIÓN DEL SISTEMA DE CONTROL 80
4.3.1. Componentes de entrada y salida de datos del sistema 80
4.3.2. Configuración de los diagramas de conexión 83
4.3.3. Configuración del sistema de control de la red 88
4.3.3.1. Evaluación de subsistemas del módulo 1 88
4.3.3.2. Evaluación de subsistemas del módulo 2 89
4.3.3.3. Evaluación de subsistemas del módulo 3 90
4.3.3.4. Evaluación de subsistemas del módulo 4 92
4.3.4. Manejo e interacción de datos para la programación del sistema 93
4.3.4.1. Modificación de librerías 94
4.3.5. Construcción del prototipo de simulación de red CAN 95
CAPÍTULO IV 96
5. CONCLUSIONES Y RECOMENDACIONES 96
5.1. CONCLUSIONES 96
5.2. RECOMENDACIONES 98
6. BIBLIOGRAFÍA 99
ANEXOS 103

ÍNDICE DE FIGURAS

PÁGINA

Figura 2.1 Concepto de comunicación en sistemas electrónicos 6
Figura 2.2 Señal analógica y digital 7
Figura 2.3 Representación de bit, nibble y byte 8
Figura 2.4 Esquema de la interfaz interna de una unidad de control 9
Figura 2.5 Bus de datos en la red CAN 16
Figura 2.6 Componentes de una red CAN-BUS 17
Figura 2.7 Topología bus 18
Figura 2.8 Formato de mensaje CAN 20
Figura 2.9 Arduino Mega 2560 26
Figura 2.10 Hardware de la placa Arduino CAN-BUS Shield 29
Figura 2.11 Tipos de luces automotrices 31
Figura 3.1 Flujograma general del proyecto 36
Figura 3.2 Esquema metodológico para el diseño de red 37
Figura 3.3 Diagrama del microcontrolador Atmega 2560 42
Figura 3.4 Pines SPI del microcontrolador Atmega 2560 43
Figura 3.5 Diagrama microchip MCP 2515 43
Figura 3.6 Diagrama transceptor TJA 1050 44
Figura 3.7 Esquema general de un módulo CAN 45
Figura 3.8 Diseño del nodo o módulo CAN simple 45
Figura 3.9 Diagrama base de la red multiplexada CAN 46
Figura 3.10 Flujograma de red para establecer comunicación bidireccional 47
Figura 3.11 Conexión de dos módulos CAN 48
Figura 3.12 Diagrama del envío y recepción de mensajes unidireccional 49
Figura 3.13 Lectura de la comunicación unidireccional en el Bus monitor 50
Figura 3.14 Señal de la línea de comunicación unidireccional 51
Figura 3.15 Interfaz del envío y recepción de mensajes de un nodo CAN 53
Figura 3.16 Diagrama de programación para la comunicación entre 2 módulos CAN 53
Figura 3.17 Configuración de los pines para la lectura del Bus de datos 54
Figura 3.18 Comunicación entre 2 módulos empleando una rutina de interrupción 54
Figura 3.19 Lectura de mensajes enviados y recibidos en los dos módulos CAN 55
Figura 3.20 Diagrama de la fase de arbitraje 56
Figura 3.21 Proceso de arbitraje en la línea CAN HIGH 57
Figura 3.22 Diagrama de programación para la comunicación entre 4 módulos CAN 58
Figura 3.23 Lectura de la comunicación entre 3 módulos empleando el CAN Bus Analyzer 59
Figura 3.24 Lectura del Bus de datos empleando un timer para la comunicación 59
Figura 3.25 Esquema metodológico para el diseño del sistema embebido 60
Figura 3.26 Control de actuadores alimentados con una fuente externa 66
Figura 3.27 Diagrama de conexión de componentes del módulo 1 67
Figura 3.28 Diagrama de conexión de componentes del módulo 2 67
Figura 3.29 Diagrama de conexión de componentes del módulo 3 68
Figura 3.30 Diagrama de conexión de componentes del módulo 4 69
Figura 3.31 Lógica de programación para el manejo de información 70
Figura 4.1 Interconexión placa Arduino Mega y placa CAN Bus Shield 71
Figura 4.2 Esquema de conexión de los 4 módulos de la red 72

Figura 4.3 Transmisión de mensaje continuo 73
Figura 4.4 Decodificación de una trama de datos en formato extendido 74
Figura 4.5 Comunicación CAN únicamente con la función attachInterrupt 75
Figura 4.6 Comportamiento de la comunicación entre 4 módulos CAN únicamente con la función attachInterrupt 76
Figura 4.7 Coordinación de periodos de transmisión 76
Figura 4.8 Desfase del periodo de transmisión con la función millis 77
Figura 4.9 Secuencia continua de comunicación entre 4 módulos CAN con un timer 77
Figura 4.10 Comportamiento de la comunicación multidifusión entre 4 módulos CAN empleando un timer para el control de periodos de transmisión 78
Figura 4.11 Configuración de resistencias Pull Up y Pull Down 83
Figura 4.12 Configuración de circuito en cátado y ánodo común 83
Figura 4.13 Enlace del sistema general de la red con 4 módulos CAN 87
Figura 4.14 Estructura de la trama de datos del módulo 1 88
Figura 4.15 Señales condicionales de evaluación del sensor de lluvia 89
Figura 4.16 Estructura de la trama de datos del módulo 2 89
Figura 4.17 Señales de conmutación de estado de conducción 90
Figura 4.18 Señales de conmutación de luces direccionales 90
Figura 4.19 Estructura de la trama de datos del módulo 3 91
Figura 4.20 Señales de conmutación de sistema limpiaparabrisas 91
Figura 4.21 Señal de evaluación de proximidad 92
Figura 4.22 Estructura de la trama de datos del módulo 4 92
Figura 4.23 Modificación de la librería Servo.h 95
Figura 4.24 Prototipo de la red multiplexada CAN 95

ÍNDICE DE TABLAS

PÁGINA

Tabla 2.1 Tipos de topologías de redes multiplexadas automotrices 10
Tabla 2.2 Tipos de protocolos de comunicación automotrices 13
Tabla 2.3 Principales velocidades para la transmisión de datos 19
Tabla 2.4 Fallos comunes en la red CAN-BUS 22
Tabla 2.5 Comparativa de las versiones CAN-BUS Shield 30
Tabla 2.6 Descripción de los tipos de luces automotrices 32
Tabla 3.1 Especificaciones del microcontrolador Arduino Mega 2560 38
Tabla 3.2 Especificaciones técnicas de la placa CAN Shield V3.0 39
Tabla 3.3 Características del controlador CAN MCP2515 40
Tabla 3.4 Características del transceptor TJA 1050 41
Tabla 3.5 Componentes para la conexión física de la red 41
Tabla 3.6 Materiales y equipos para el desarrollo de la comunicación en la red 48
Tabla 3.7 Sistema de control general de la red CAN 61
Tabla 3.8 Dispositivos captadores o entradas del sistema embebido 62
Tabla 3.9 Actuadores o salidas del sistema embebido 64
Tabla 3.10 Funcionalidad de los módulos de la red 65
Tabla 4.1 Conversión de los periodos de transmisión constantes en frecuencia 79
Tabla 4.2 Elementos captadores y su relación con componentes automotrices 80
Tabla 4.3 Elementos actuadores y su relacion con compontentes automotrices 82
Tabla 4.4 Consumos de corriente generales de los sensores del sistema 85
Tabla 4.5 Consumo de corriente de los componentes alimentados con Arduino 85
Tabla 4.6 Componentes alimentados por una fuente externa 86
Tabla 4.7 Interacción del manejo de datos en los sistemas de control 93

ÍNDICE DE ANEXOS

ANEXO NUM **DESCRIPCIÓN** **PÁGINA**

I. DISPOSICIÓN DE LOS PINES DE LOS CHIPS DE LA PLACA CAN BUS SHIELD 104

II. PROGRMACIÓN DE LA COMUNICACIÓN UNIDIRECCIONAL 105

III. CÓDIGO DE PROGRAMACIÓN PARA LA COMUNICACIÓN CAN CON LA FUNCIÓN MILLIS 107

IV. TABLA DE LA CONFIGURACIÓN GENERAL DEL SISTEMA DE CONTROL 109

V. ESTRUCTURA DEL CÓDIGO DE PROGRAMACIÓN DEL MÓDULO 1 CON LA ADAPTACIÓN DE UN TIMER 111

VI. ESQUEMA DEL SISTEMA DE CONTROL 121

VII. CONSTRUCCIÓN DEL PROTOTIPO DE RED MULTIPLEXADA CAN 122

ÍNDICE DE ECUACIONES

PÁGINA

Ecuación [4.1] 79
Ecuación [4.2] 84

RESUMEN

La finalidad del presente trabajo de investigación es elaborar una red multiplexada CAN en base a microcontroladores Arduino, con la capacidad de controlar sistemas embebidos automotrices. Para ello, se diseñó un sistema multiplexado con 4 módulos sustentado en una configuración CAN-BUS, dichos módulos se constituyeron por placas CAN Bus Shield y Arduinos Mega 2560, mismos que al ensamblarlos y puentear los pines SPI (SCK, MOSI y MISO) son compatibles con la norma ISO 11898, por contar con el controlador MCP2515, transceptor TJA1050 y microprocesador Atmega2560. Luego, se desarrolló la lógica de programación empleando el software Arduino (IDE), estableciendo inicialmente una comunicación unidireccional para familiarizarse con el uso de las librerías SPI.h y mcp2515_can.h, y luego una comunicación bidireccional paulatina bajo el principio multidifusión, con los parámetros comunes de velocidad a 1 Mbit/s y formato extendido. La cual, se reduce al rango de 500 Kbits/s por la operación de la placa CAN Bus Shield a 8 MHz, generando que el tiempo de transmisión de una trama sea el doble del número de bits que la conforman, pero en microsegundos. En donde, se identificó fluctuaciones en los periodos de transmisión y counter, los cuales se controlaron implementando un Timer, con capacidad de ajustar dichos periodos hasta 5 ms. Seguidamente, se elaboró el sistema general de la red, el cual se conformó por componentes automotrices y compatibles a Arduino, para simular principalmente el funcionamiento de subsistemas de control de la carrocería, como: las luces de cruce, carretera, direccionales, emergencia, antiniebla, posición, freno y retroceso, en conjunto con la alerta acústica de proximidad y el sistema limpiaparabrisas, integrado con funcionalidad automática al igual que las luces adaptativas de carretera y selectivas antiniebla delanteras, además de subsistemas también relacionados con la seguridad activa, como las luces automáticas de la cabina, claxon, y como algo complementario un motor DC. Generando un sistema híbrido, en el que 3 módulos evaluaron y controlaron los subsistemas de la carrocería y el faltante los adicionales, teniendo una similar carga de procesamiento. El cual, se alimentó con dos fuentes, un adaptador de 9V y 2A para los 4 microcontroladores y sus componentes, al necesitar de un suministro de corriente de al menos 440 mA, y una fuente de poder de 5V y 5A para los actuadores de alimentación externa con un consumo normal de 700 mA. Tras ello, se adecuó los códigos con el principio de comunicación por eventos, en la emisión se incorporó un condicional que se inicializa al evaluar al menos una entrada activa para enviar mensajes, y en la recepción máscaras y filtros para aceptar únicamente mensajes de importancia, mismos que almacenan

en Buffers únicos para evitar colisión de datos. De manera que, al disponer de información de diferentes módulos, se realizó la programación de operaciones lógicas que dependan de la red, para controlar uno o más subsistemas, los cuales conformaron un sistema general con múltiples relaciones embebidas, siendo un prototipo de capacitación didáctica de redes CAN-BUS, con la capacidad de modificar y establecer parámetros únicos para el funcionamiento de los sistemas integrados y la implementación de otros subsistemas.

REPÚBLICA DEL ECUADOR

UNIVERSIDAD TÉCNICA DEL NORTE

Acreditada Resolución Nro. 173-SE-33-CACES-2020

EMPRESA PÚBLICA "LA UEMEPRENDE E.P."

Abstract

This research work aimed to develop a multiplexed CAN network based on Arduino microcontrollers, being able to control automotive embedded systems. For this purpose, a multiplexed system was designed with 4 modules based on a CAN-BUS configuration. These modules consisted of CAN Bus Shield boards and Arduino Mega 2560, which when assembled and jumpered the SPI pins (SCK, MOSI and MISO) are compatible with the ISO 11898 standard, because they have the MCP2515 controller, TJA1050 transceiver and Atmega2560 microprocessor. Then, the programming logic was developed using the Arduino software (IDE); initially establishing a unidirectional communication to become familiar with the use of the SPI.h and mcp2515_can. h libraries, and then a gradual bidirectional communication under the multicast principle, with the common parameters of speed at 1 Mbit/s and extended format, which is reduced to the range of 500 Kbits/s by the operation of the CAN Bus Shield board at 8 MHz, generating that the transmission time of a frame is twice the number of bits that make it up, but in microseconds. Fluctuations in the transmission and counter periods were identified, which were controlled by implementing a Timer, with the capacity to adjust these periods up to 5 ms. Then, the general system of the network was elaborated, which was constituted by automotive components and compatible to Arduino, to simulate mainly the operation of control subsystems of the bodywork, such as: low beam, high beam, directional, emergency, fog, position, brake and retro lights, in conjunction with the acoustic proximity alert and the windshield wiper system, integrated with automatic functionality as well as the adaptive road lights and selective front fog lights, in addition to subsystems also related to active safety, such as automatic cabin lights, horn, and as a complementary thing a DC motor; generating a hybrid system, in which 3 modules evaluated and controlled the body subsystems and the missing additional ones, having a similar processing load. This was powered with two sources, a 9V and 2A adapter for the 4 microcontrollers and their components, requiring a current supply of at least 440 mA, and a 5V and 5A power supply for the external power actuators with a normal consumption of 700 mA. After that, the codes were adapted with the event-driven communication principle; in the emission a conditional was incorporated that is initialized when evaluating at least one active input to send messages, and in the reception masks and filters to accept only messages of importance, which are stored in unique buffers to avoid data collision. In this way, having information from different modules, the programming of logical operations that depend on the network was performed, to control one or more subsystems, which formed a general system with multiple embedded relationships, being a prototype of didactic training of CAN-BUS networks, with the ability to modify and set unique parameters for the operation of integrated systems and the implementation of other subsystems.

Keywords: CAN multiplexed network, Arduino, programming, automotive embedded system, communication.

Reviewed by:
MSc. Luis Paspuezán Soto
CAPACITADOR-CAI
Octubre 3, 2023

INTRODUCCIÓN

Desde la invención y sustitución de componentes mecánicos por electrónicos en la década de los 60, la industria automotriz ha avanzado significativamente como parte de un importante sector de desarrollo tecnológico, en el cual la creación e implementación de sistemas de control han permitido cubrir necesidades de usuarios que se centran en seguridad, confort y rendimiento. En relación con lo mencionado, el pilar fundamental para tal desarrollo han sido las redes y protocolos de comunicación, dentro de las cuales destaca la red CAN (Red de Área del Controlador) desarrollada por Robert Bosch, al ser la promotora en la solución de problemas relacionados a sistemas sobrecargados de cableado, componentes y unidades de control, ya que su principio multimaestro permite intercambiar información de forma serial entre unidades electrónicas conectadas en paralelo a dos líneas de comunicación (CAN HIGH y CAN LOW).

Actualmente, el protocolo CAN es el más utilizado para el manejo de datos de diferentes subsistemas de un sistema general de red automotriz, el cual además integra otros protocolos de comunicación dependiendo del área funcional, pero conservando una relación común mediante un Gateway, en cuanto a lo expuesto es importante destacar que los protocolos VAN Bus, LIN Bus, MOST Bus y FlexRay son basados en CAN. Sin embargo, aunque se conoce que estas tecnológicas son la base del desarrollo, la industria automotriz ecuatoriana se está quedando atrás en cuanto a la preparación técnica de las tecnologías de interconexión de unidades de control y componentes, las cuales cada vez se vuelven más complejas por su capacidad de modificar equipamiento y relacionarse con nuevas áreas para mejorar la eficiencia del sistema general de la red, lo cual afecta directamente a la matriz productiva. Por lo cual, debido a la importancia que implica conocer el funcionamiento interno y externo de dicha red, en el presente trabajo se da conocer el proceso empleado para elaborar una red multiplexada CAN en la que los microcontroladores programables de Arduino son la base para estructurar la red física y lógica, siendo en la actualidad una alternativa viable para remplazar laboratorios sofisticados de demostración (Martinez-Santos et al., 2017, p. 209). De manera que, con dicha red se demuestre el comportamiento lógico de la comunicación CAN a través de herramientas de análisis, y el control de un sistema embebido automotriz con capacidad de modificar e incrementar funcionalidades.

CAPÍTULO I

1. PERFIL DEL PROYECTO

1.1. OBJETIVOS

1.1.1. OBJETIVO GENERAL

Elaborar una red multiplexada CAN en base a microcontroladores Arduino para el control de sistemas embebidos automotrices.

1.1.2. OBJETIVOS ESPECÍFICOS

- Diseñar el sistema multiplexado para el control de sistemas embebidos automotrices.
- Realizar el algoritmo y la programación de microcontroladores Arduino para comunicación CAN.
- Elaborar el sistema embebido a controlar enfocado en las entradas y salidas del mismo.
- Evaluar el funcionamiento y desempeño de la red CAN en pruebas dinámicas.

1.2. JUSTIFICACIÓN

La industria automotriz representa un importante sector de desarrollo debido a su crecimiento tecnológico, la creación e implementación de protocolos para aumentar la autonomía y realizar diferentes funciones permite a los usuarios tener una mayor confianza y comodidad (Martínez-Cruz et al., 2021). Sin embargo, las investigaciones se han centrado principalmente en el análisis de la naturaleza mecánica de un vehículo, más no de la parte electrónica.

El presente proyecto contribuye en el desarrollo del objetivo 7 del Plan de Creación de Oportunidades 2021 - 2025, en el que se busca “potenciar las capacidades de la ciudadanía y promover una educación innovadora, inclusiva y de calidad en todos los niveles” (Secretaría Nacional de Planificación, 2021). Además, el proyecto se relaciona con la Política 7.2 del objetivo antedicho, ya que se busca mejorar el modelo educativo del Ecuador empleando herramientas tecnológicas y aplicando ideas innovadoras, para que la ciudadanía tenga una educación de calidad, al igual que los modelos internacionales, los cuales se centran en la innovación tecnológica. Basándonos en esta política, este proyecto busca que los futuros profesionales del país tengan como base una herramienta, la cual promueva la modernización en lo que respecta a la tecnología de comunicación y control que utilizan los sistemas embebidos automotrices. Además, se alinea con la Política 7.4 en la cual se destaca

la importancia de reforzar el Sistema de Educación Superior por medio de la investigación de elevada importancia, es por tal motivo que con la elaboración de una red multiplexada CAN a base de microcontroladores Arduino se pretende ampliar las áreas de investigación de estas bases de desarrollo, generando en las futuras generaciones interés por conocer el funcionamiento y evaluación de los protocolos de comunicación actuales y futuros, lo que les permitirá generar propuestas que contribuyan al desarrollo de la matriz productiva, ya sea independientemente o en conjunto con otras entidades, ya que es importante competir con las nuevas tendencias tecnológicas.

1.3. ANTECEDENTES

En la década de los 60, los vehículos contaban con componentes netamente mecánicos, y con el propósito de aumentar la fiabilidad se inició con la sustitución de partes mecánicas por componentes electrónicos. Por lo cual, (Pérez Darquea, 2017) afirma que los microprocesadores y unidades electrónicas tuvieron su desarrollo a principios de la década de los setenta en vehículos de alta gama. También (Lara Rivero, 2014) menciona que se comenzó una fase comercial acelerada de integración de nuevas funciones en los vehículos desde 1970, como: sistemas de inyección de combustible, sistema ABS de sus siglas en inglés Anti-lock Braking System (1971), sistema de encendido electrónico (1973), entre otros.

El investigador (Tenesaca, 2013) indica que, a inicios de la década de los años 80, los vehículos contaban con unidades electrónicas conectadas independientemente para realizar una determinada función, y debido a la mayor demanda de comodidad, seguridad, rendimiento y consumo de combustible se necesitaba más unidades electrónicas y cableado. De acuerdo con esta situación (Lara Rivero, 2014) manifestó que se produjo el inconveniente de implementar un sistema tecnológico sobrecargado de unidades electrónicas, por lo que se buscó métodos para establecer una adecuada interconexión. La creación y desarrollo de una red multiplexada CAN, de sus siglas en inglés Controller Area Network (Red de Área del Controlador) se le atribuye a Robert Bosch a principios de los años 80, como una propuesta inicialmente empleada en la industria automotriz. Según (Puga Gutiérrez & Morales Recalde, 2022, p. 33) es una interconexión en tiempo real, empleada como una solución viable para transmitir y comunicar grandes conjuntos de datos entre unidades electrónicas de manera rápida y efectiva. Por lo tanto, la reducción del cableado permitió que los sistemas

complejos en vehículos modernos tengan una eficiente comunicación a pesar de contar con múltiples unidades electrónicas (Tenesaca, 2013).

La gran acogida del protocolo de comunicación CAN en la industria automotriz fue muy importante, esto por sus ventajas; disminución de cableado, facilidad de diagnóstico y alta inmunidad a ruidos e interferencias, lo que permitió su normalización (Corrigan, 2002). CAN fue presentado en 1986 en el congreso SAE, de sus siglas en inglés Society of Automotive Engineers llevado a cabo en Detroit y estandarizado en 1993 por la Organización Internacional de Normalización (ISO) como ISO 11898, abordando como es el intercambio de datos en una red común y su arquitectura de dos capas del modelo OSI/ISO (Johansson et al., 2005). En donde el modelo OSI de sus siglas en inglés Open Systems Interconnection fue desarrollado por la ISO como una interconexión de sistemas abiertos.

1.4. PLANTEAMIENTO DEL PROBLEMA

El mundo avanza a una velocidad extraordinaria en cuanto a la producción de vehículos y provisión de servicios, lo que genera oportunidades e inconvenientes para asumir nuevas competencias y poder integrarse en cadenas globales de valor. En la actualidad, la industria automotriz ecuatoriana está perdiendo mercado, lo cual denota un déficit de conocimiento en cuanto a las tecnologías de interconexión de unidades electrónicas, que ya se encuentran en el país y continúan llegando cada día, ya que las redes y protocolos de comunicación son la base del desarrollo, en donde destaca la red CAN al ser la más utilizada. Este problema afecta directamente a la matriz productiva y preferencias de usuarios, tanto por los precios competitivos como por la implementación de nuevas prestaciones vehiculares, que incrementan la dificultad de los sistemas de comunicación, lo cual genera dificultades para ofrecer servicios y solucionar problemas.

Las personas dedicadas al área automotriz conocen que un vehículo cuenta con un sistema de interconexión, en el cual interactúan módulos con entradas y salidas de datos para el correcto funcionamiento del vehículo, pero no conocen su funcionamiento interno, por lo que capacitarse en cuanto a este tipo de tecnologías permitirá que Ecuador no se deje absorber por los mercados mundiales. Con respecto a la problemática (Martinez-Santos et al., 2017, p. 209) mencionan que el uso de sistemas para el desarrollo de proyectos con Arduino ha permitido en muchos casos remplazar a los laboratorios de demostración. Por lo tanto, con el presente proyecto se pretende ayudar en el fortalecimiento del conocimiento de la problemática mencionada, mediante la creación de una red multiplexada CAN para que al

igual que los protocolos de comunicación utilizados en cualquier vehículo permita controlar sistemas embebidos automotrices, con la finalidad de que los futuros profesionales de la Universidad Técnica del Note tengan una herramienta que contribuya en su desarrollo y capacitación de tecnologías actuales, en cuanto a comunicación y control, fortaleciendo áreas de investigación necesarias y así contar con bases para afrontar y competir con las tecnologías que se están incorporando en el país y las próximas que se desarrollarán en el mundo.

1.5. ALCANCE

El presente proyecto pretende elaborar una red multiplexada CAN en base a microcontroladores Arduino para el control de sistemas embebidos automotrices. La primera instancia del proyecto es investigar y analizar el contexto teórico involucrado en el funcionamiento de una red CAN-BUS, de modo que se tenga referencias para diseñar una red con 4 módulos estructurados por microcontroladores Arduino Mega 2560 y placas CAN-BUS Shield V3.

Una vez diseñada la red base, se desarrollará un algoritmo de pasos lógicos que permita establecer una comunicación serial entre los módulos por medio de programación, empleando el entorno de desarrollo integrado (IDE) de Arduino, para que dichos módulos envíen y reciban información en tiempo real. Cada módulo contará con un identificador único que le permitirá interactuar en la red con los demás módulos para controlar el sistema a ser implementado en la red, el cual en el caso del presente proyecto será una hibridación de componentes automotrices y adaptados, enfocado principalmente en cubrir las funcionalidades de control de la carrocería con el sistema limpiaparabrisas, advertencia a la colisión y sistema de luces (posición, cruce, carretera, antiniebla, freno, retroceso, direccionales y emergencia). Para lo cual se empleará: la palanca conmutadora de luces y plumas del vehículo Chevrolet D-max del año 2005, un interruptor de luces de emergencia, un pulsador para luces de freno, dos interruptores para controlar las luces antiniebla delanteras y posteriores, y un interruptor conmutador deslizante de 3 posiciones para controlar el estado de conducción (avance, neutral y retroceso). Mientras que, los actuadores de dicho sistema son diodos emisores de luz y servomotores de rotación limitada respectivamente, que sumado a su funcionalidad normal integrarán un apartado de automatización de luces adaptativas de carretera y de la cabina mediante la evaluación de fotorresistores LDR, selectividad de luces antiniebla frontales dependiendo de la posición

de un potenciómetro que evalúa el giro del volante y autoajuste de la frecuencia de barrido del parabrisas dependiendo de la humedad o lluvia ambiental captada por el sensor MH-RD.

Dichos subsistemas se relacionan con la seguridad activa de un vehículo, en donde además se incluirá alertas acústicas del claxon y de proximidad por medio de un sensor ultrasónico, sensor óptico reflectivo y pulsador relacionados a las luces automáticas de la cabina y adicionalmente un motor DC de rotación continua como parte del sistema de tracción eléctrica, el cual se controlará con un potenciómetro para su aceleración. En relación con lo mencionado, la comunicación será un punto fundamental para ejecutar operaciones lógicas con datos provenientes de diferentes módulos, por lo cual se establecerá una lógica de programación para tener una trasmisión y recepción de mensajes controlada bajo el principio de comunicación multidifusión y por eventos.

La parte final del proyecto consiste en desarrollar un prototipo tipo maqueta de la red CAN en base a microcontroladores Arduino, que sea capaz de controlar sistemas embebidos automotrices por medio de la interacción de las entradas y salidas de datos, empleando el protocolo de comunicación CAN para evaluar el funcionamiento y desempeño del sistema, al someterlo a diferentes situaciones de pruebas dinámicas y así verificar su comportamiento por medio de dispositivos de análisis.

2. REVISIÓN BIBLIOGRÁFICA

2.1. REDES MULTIPLEXADAS AUTOMOTRICES

Desde la invención del automóvil se han ido implementando diferentes funciones para satisfacer necesidades de usuarios, como el alumbrado (1910), el encendido eléctrico (1912), entre otros. De modo que fue necesario implementar un equipo eléctrico básico, aunque no fue hasta inicios de los setenta que, los microprocesadores y unidades electrónicas llegarían a mejorar y ampliar las prestaciones de los vehículos (seguridad, confort, rendimiento y consumo de combustible) con sistemas innovadores. Sin embargo, el elevado número de circuitos eléctricos, electrónicos y cableado se volvía un problema, ya que cada unidad electrónica estaba conectada independientemente y desarrollando sus propias tareas, o sea, recibía información de sus exclusivos captadores y activaba sus actuadores en función de los requerimientos.

De acuerdo a lo mencionado, (Lara Rivero, 2014) manifiesta que a principios de la década de los 80 surgió el inconveniente de implementar sistemas tecnológicos sobrecargados de unidades de control, por lo que los técnicos de la época se dieron cuenta que, si las unidades de control pueden comunicarse entre sí, se conseguirían múltiples ventajas. Es así como se desarrollaron redes digitales que permiten transmitir información (datos), entre las diferentes unidades de control por medio de líneas y protocolos de comunicación, para que dicha información sea aprovechada por todo el sistema, permitiendo solucionar los problemas de peso y espacio de mazos de cables utilizados en los vehículos (Pérez, 2014, p. 19).

Figura 2.1 Concepto de comunicación en sistemas electrónicos
(Llanos López, 2022, p. 82)

En la figura 2.1 se presenta un esquema básico de la comunicación en la red interna de vehículos, lo cual evita la complejidad de la instalación eléctrica y electrónica del vehículo con un solo medio o canal de conexión (Bus), lo que deja atrás conexiones eléctricas independientes (Jaque, 2015). Además, por medio de los avances tecnológicos se ha ido mejorando la velocidad de intercomunicación, aumentando el procesamiento de datos y disminuyendo los módulos y cableado.

2.1.1. PRINCIPIOS DE ELECTRÓNICA EN REDES MULTIPLEXADAS

2.1.1.1. Electrónica analógica y digital

La electrónica analógica es la técnica en la cual las señales de tensión pueden variar y adoptar cualquier valor, en función del tiempo y dentro de un rango definido. Es decir, nunca se obtendrá un valor indeterminado, como es el caso de la señal en sensores de presión, temperatura, etc. En la figura 2.2 se presenta la forma de señales analógicas y digitales, evidenciando que en la electrónica digital las señales de tensión solo pueden adoptar uno de dos estados posibles (niveles lógicos), 1 (bit 1) y 0 (bit 0), denominados nivel alto y bajo respectivamente, los cuales tienen gran inmunidad a corrientes parasitas en la transmisión de datos (Sánchez, 2012).

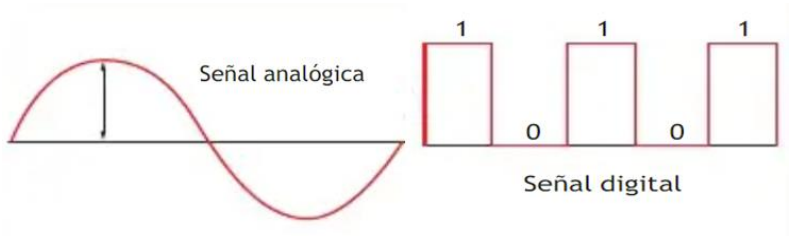


Figura 2.2 Señal analógica y digital
(Sánchez, 2012, p. 284)

2.1.1.2. Codificación numérica

La arquitectura de los circuitos emplea la electrónica digital, pero el mundo real es analógico. Por lo tanto, para manejar las entradas analógicas es necesario digitalizar toda la información para procesarla, obteniendo una salida digital que se convierte en su forma analógica original, para ser comprendida y poder realizar un propósito. Con los siguientes sistemas de numeración:

- **Binario. –** Es el sistema utilizado en circuitos digitales, es muy eficiente porque utiliza solamente dos estados o símbolos (0 y 1). Por lo cual, el valor del digito (bit) es base 2.
- **Octal. –** Hace uso de 8 números o símbolos (0 al 7), con un valor de dígito base 8. Al tener una potencia exacta se relaciona con la del sistema binario base 2.
- **Decimal. –** Utiliza 10 números o símbolos (0 al 9), con un valor de digito base 10.
- **Hexadecimal. –** Emplea 16 símbolos (0 al 9, A – F), con un valor de digito base 16.

2.1.1.3. Unidades de información

Los circuitos electrónicos utilizan el sistema binario y su unidad básica de información es el bit, la agrupación de estos se representan en la figura 2.3. Los datos pueden ser representados

únicamente en dos estados lógicos denominados biestables (1 y 0), indicando la presencia o ausencia de señal (tensión o corriente) respectivamente (Pérez, 2014).

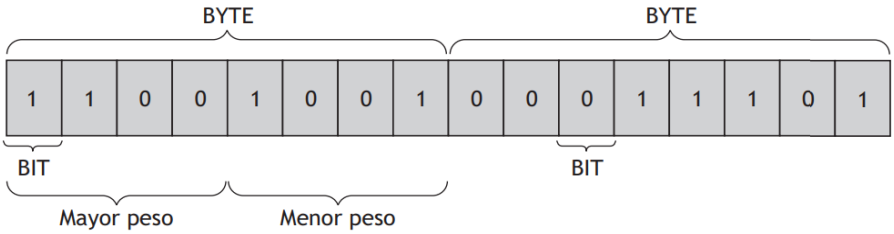


Figura 2.3 Representación de bit, nibble y byte
(Sánchez, 2012, p. 284)

- **Bit. –** Es la mínima unidad de información en el contexto del sistema binario al ser representado por ceros (0) y unos (1), permitiendo almacenar, procesar y transmitir datos mediante señales eléctricas digitales.
- **Nibble. –** Es un grupo de 4 bits con el que se puede empezar a transmitir datos.
- **Byte. –** También llamado octeto es una agrupación de 8 bits o biestables.
- **Palabra. –** Son agrupaciones indefinidas de bits.

2.1.1.4. Trama de comunicación

La trama de comunicación es una agrupación ordenada de bits o biestables, los cuales se organizan en múltiples campos o bloques. Cada campo de la trama de comunicación cumple un rol particular para la transmisión de información, dependiendo del protocolo que se utilice en las líneas y niveles de tensión, en donde las partes básicas son: inicio, dirección, mensaje, comprobación y final.

2.1.1.5. Transmisión de mensaje

La trama de comunicación o mensaje puede transmitirse de forma bidireccional por medio de un bus de datos (línea) y un protocolo de comunicación establecido entre las diferentes unidades de control conectadas a una red. Los mensajes se introducen a la red y al llegar a las unidades de control se convierten en código binario para ser comprendido por las mismas y usarlos si es de relevancia, se codifican si hay errores.

2.1.1.6. Velocidad de trasmisión de datos

La velocidad es uno de los puntos principales para mejorar las redes multiplexadas, esta se mide de acuerdo a la cantidad de bits que pueden ser enviados en un segundo (bps) (Llanos López, 2022). En vehículos modernos se manejan velocidades de 10 Kbps a 100 Mbps.

- Low speed o transmisión lenta (< 125 Kbps).

- High speed o transmisión rápida (> 125 Kbps).

2.1.2. COMPONENTES DE UNA RED MULTIPLEXADA

Emisor o captador. – Es el componente de entrada que transforma los fenómenos físicos en voltaje a una frecuencia (señal eléctrica), este introduce la información ya sea análoga o digital al sistema, siendo normalmente sensores, pero también otra unidad de control.

Protocolo de comunicación. – El protocolo de comunicación o también llamado lenguaje es un conjunto de reglas que permiten la comunicación (transmisión de información) entre dos o más unidades electrónicas conectadas a una red (Domínguez & Ferrer, 2012).

Gateway. – Es el componente que controla la comunicación entre las unidades de control y cuando es necesario en el diagnóstico, este transforma o traduce mensajes de un protocolo a otro, formando nuevas tramas de datos de una red emisora a una receptora o viceversa (Sánchez, 2012).

Unidad de control. – La unidad de control representada en la figura 2.4 es también conocida como una computadora, la cual recibe señales de entrada (información) de un sensor o también de otra unidad de control, y la procesa de acuerdo a instrucciones definidas en su programación para generar órdenes de salida hacia actuadores u otras unidades de control, de modo que al final se realice una determinada función (Domínguez & Ferrer, 2012).

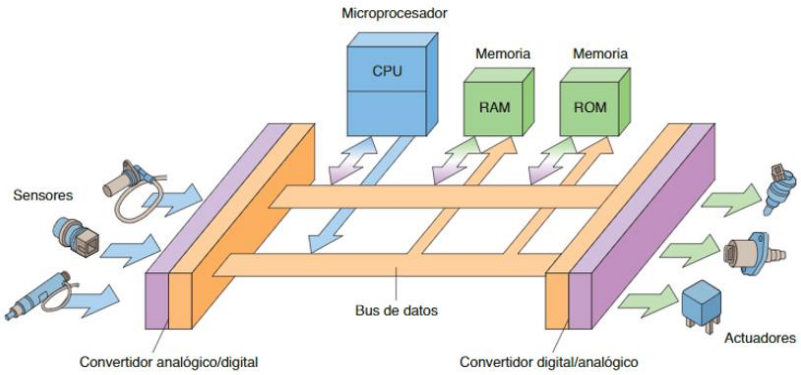


Figura 2.4 Esquema de la interfaz interna de una unidad de control
(Domínguez & Ferrer, 2012, p. 78)

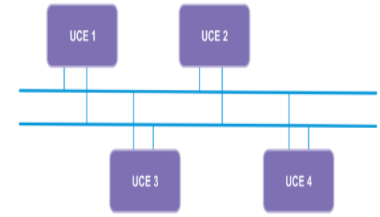
Medio de transmisión. – Es la línea de comunicación por donde circulan los datos (información).

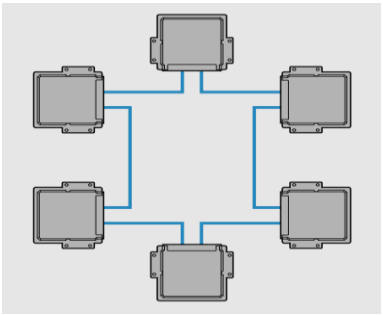
Receptor o actuador. – Es el componente de salida que recibe la información (órdenes) y la aprovecha para realizar determinadas funciones. Transforma señales eléctricas en fenómenos físicos, sin embargo, las órdenes también pueden ser aprovechadas por otra unidad de control.

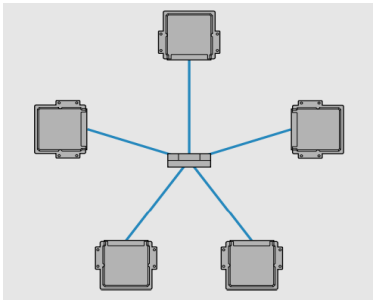
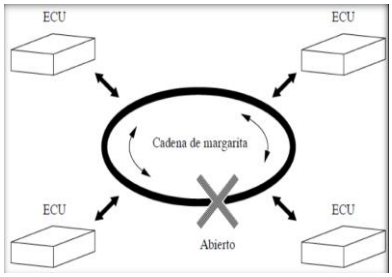
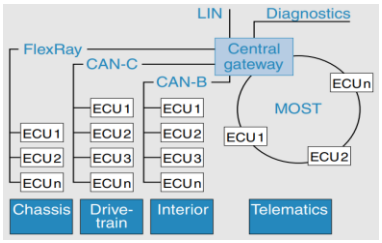
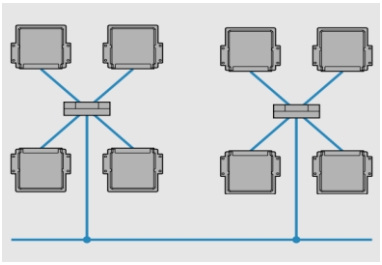
2.1.3. TOPOLOGÍA DE REDES MULTIPLEXADAS

La topología de redes multiplexadas se refiere a la configuración en la que se encuentran conectadas las unidades de control en una red de transmisión de datos. Según López Diguay (2021) la topología en general es física y lógica, la topología física se refiere a la estructura externar de la red como el cableado y antenas para la interconexión de componentes en la red, y la topología lógica es el lenguaje o protocolo de comunicación. Las cuales permiten la comunicación de sistemas y subsistemas, dependiendo los requerimientos de los mismos, para emplear los tipos de topologías que se presentan en la tabla 2.1 con sus características.

Tabla 2.1 Tipos de topologías de redes multiplexadas automotrices

Topología	Características
Punto a punto  α	• Conexión únicamente entre dos unidades de control. • Comunicación o transmisión de datos es directa mediante uno o dos cables trenzados (canal). • El ejemplo más práctico para este tipo de topología es la conexión entre el escáner y la ECU del vehículo.
Bus  β	• Es simple y de bajo costo, por su uso mínimo de cableado. • Las unidades de control se conectan a una línea de comunicación principal (cable), creando así una red. • Si una unidad de control falla la red no dejará de funcionar, únicamente se dejará de recibir datos de dicha unidad en toda la red (Reif, 2015). • No existe unidad de control principal. • Si la línea de comunicación es cortada, toda la red fallará. Por lo cual la integridad del mismo determina su buen funcionamiento.

Anillo
 
γ
	• Las unidades de control se conectan en serie, de tal manera que se forme un bucle cerrado al conectar el último nodo de la red con el primero.
	• No existe una unidad de control principal.
	• Al igual que la topología Bus, si la línea de comunicación presenta problemas, la red queda deshabilitada, pero depende de la configuración de anillo que se emplee.
	  ◦ Anillo simple. – Comunicación unidireccional.
	  ◦ Anillo doble. – Comunicación bidireccional.

Estrella  δ	• Las unidades de control se conectan a una sola unidad principal, lo cual genera un sistema cargado de cables dependiendo de la ubicación de los mismos (López Diguay, 2021). • Evita colisiones de mensajes y agiliza el tráfico de datos. • La unidad principal se encarga de la comunicación en el sistema. • Si la unidad de control principal presenta fallos, toda la red se verá afectada.
Daisy Chain  ε	• La conexión entre las unidades de control es en serie, pero a diferencia de la topología de anillo, esta no cierra el bucle. • Posee dos canales de comunicación con las mismas señales de datos, aumentando la seguridad en la red. • En caso de problemas en las líneas, la red se deshabilita. • Si una unidad de control se desconecta o falla, la red queda deshabilitada en ese punto (Mejía Morales et al., 2013).
Gateway  ζ	• Se encuentra en medio de dos redes con topologías y protocolos de comunicación diferentes. • Es una compuerta traductora. • Transforma los mensajes de un protocolo de comunicación a otro según sea necesario (Mejía Morales et al., 2013).
Híbrida  η	• Las unidades de control se conectan con combinaciones de diferentes topologías. • Genera una red más segura por sus conexiones. • Tiene relación con la topología mixta, ya que emplea diferentes topologías para generar conexiones múltiples entre nodos.

Fuente: α (Ashtari, 2022), β (Llanos López, 2017, p. 90), γ (Reif, 2015, p. 46), δ (Reif, 2015, p. 45), ε (Figueroa Peñafiel, 2015, p. 18), ζ (Reif, 2015, p. 61), η (Reif, 2015, p. 47)

2.1.4. VENTAJAS DE LA UTILIZACIÓN DE REDES MULTIPLEXADAS

El uso de redes multiplexadas en los vehículos ha permitido incrementar la seguridad, confort y también el ámbito tecnológico y económico.

- **Reducción de cableado:** Si las unidades de control aún se encontraran conectadas de manera independiente, se estima que la longitud del cableado sea de más de 2 km.

- **Menor peso:** Las redes multiplexadas han permitido utilizar menor longitud de cableado, lo que refleja una reducción importante de hasta 40 kg de peso. Sumándole la ausencia de duplicados de sensores y actuadores, y también menores dimensiones en módulos.
- **Menor número de sensores:** La información de un emisor (sensor) puede ser aprovechada por las diferentes unidades de control que la requieran, evitando duplicados.
- **Menor coste de mantenimiento y fabricación:** Es una ventaja económica por la simplificación del sistema de comunicación, reduciendo: cableado, unidades de control, sensores y actuadores. Lo cual, facilita las reparaciones.
- **Otras ventajas:** La comunicación entre unidades de control es por señales eléctricas (digital o binario), la información es compartida con mayor velocidad, fiabilidad y seguridad (Llanos López, 2022). Además, permite crear sistemas de control con más parámetros, lo cual se refleja en sistemas más exactos como la gestión del motor.

2.2. PROTOCOLOS DE COMUNICACIÓN AUTOMOTRICES

En la actualidad es muy común que un vehículo cuente con diferentes redes y protocolos de comunicación específicos para sus diferentes sistemas, ya que no todos los sistemas necesitan la misma velocidad y seguridad en cuanto a la transmisión de datos en tiempo real. Por lo cual, su implementación se la realiza dependiendo de las necesidades de la multiplexación del sistema (Wang et al., 2017). El protocolo de comunicación es una serie de criterios o reglas que se fijan y agregan a una red, para poder trasferir diferentes datos entre dos o más unidades de control por medio de una línea de comunicación, dicha información es envida en agrupaciones de bits (mensaje) de forma codificada por variaciones de voltaje, de modo que solo las unidades de control que manejen dicho protocolo puedan comprender y procesar el mensaje en caso de ser de utilidad.

- ✓ **Modelo de referencia ISO OSI. –** Permite describir y comparar diferentes protocolos de comunicación, ya que cada protocolo está estructurado por capas, las cuales combinan propiedades y funciones específicas, divididas principalmente en:
    - **Capa física:** Se fijan los parámetros eléctricos con los que las conexiones físicas permitirán la comunicación, como: nivel de señal, flujo de bits (datos).

- **Capa de comunicación:** Reglas para compartir información, manejando el mismo idioma, los datos serán aceptados, procesados y reenviados a la capa física.
- **Capa de aplicación:** Procesa y comparte información a la capa de comunicación.

2.2.1. PROTOCOLOS DE COMUNICACIÓN

Los protocolos o reglas de comunicación más utilizados en la lógica de redes automotrices se detallan en la tabla 2.2, a excepción del protocolo CAN, ya que se incluirá en el apartado de red CAN-BUS.

Tabla 2.2 Tipos de protocolos de comunicación automotrices

Denominación	Características
VAN BUS	• Velocidad de transmisión de datos media, con un máximo de 125 Kbps • Permite interconectar un máximo de 16 unidades de control con dos cables de 0,6 mm^2, denominados DATA y DATA B, por los que s se transmite señales espejo. • Emplea una topología Bus con configuraciones multimaestro, maestro-esclavo y combinaciones de estos (Llanos López, 2017). • Es empleado en sistemas de cierre centralizado, como las puertas de un vehículo.
LIN BUS	• Velocidad de transmisión baja (20 Kbps), para redes con menor flujo de datos. • Permite interconectar 16 unidades (1 maestra y 15 esclavas) por medio de un solo cable de 0,35 mm^2. • Maneja una configuración maestro – esclavo. • Ideal para sistemas electrónicos independientes de control de la carrocería.
MOST	• Velocidad: MOST 25 (24,8 Mbits/s), MOST 50 (50 Mbits/s) Y MOST 150 (150 Mbits/s). • Permite interconectar hasta 64 dispositivos por medio de un solo cable (unidireccional) de 2,3 mm. • Usa una topología en anillo. • Permite implementar una red de sistemas informáticos y entretenimiento.
FLEXRAY	• Dependiendo de la configuración, maneja altas tasas de transmisión de Bits (10 Mbits/s por un canal y hasta 20 Mbits/s por dos canales) • Utiliza topologías en bus, estrella y multi-estrella. • Funciona por medio de ciclos de comunicación que activan ventanas por tiempo (estática) y evento (dinámica), puede adaptarse al flujo de datos para permanecer y permanecer inactivo cuando no existe flujo de datos (Wang et al., 2017). • Es usado en sistemas de seguridad activa y pasiva, transmisión y controles avanzados, ya que maneja sistemas que no utilizan sistemas mecánicos (x-by-mire).

TTP/C	• Es un protocolo de alta velocidad, durante el arranque asíncrono 5 Mbits/s, funcionamiento normal 25 Mbits/s y en prototipos hasta 1 Gbits/s. • Permite interconectar hasta 64 unidades por dos canales. • Emplea topologías de estrella, bus, hibrida y bus-estrella. • Las unidades de la red realizan transmisiones en intervalos de tiempo, teniendo información del flujo de datos para sincronizarlos con relojes internos (Reif, 2015).

2.2.2. REDES INALÁMBRICAS

Las redes inalámbricas surgen en la búsqueda de nuevas tecnologías para la transmisión de datos por medio de ondas electromagnéticas, es decir, no se utiliza cableado, lo que aminora considerablemente el coste de su implementación en sistemas, pero aumenta la inseguridad.

- **Bluetooth. –** Es un sistema de corto alcance que permite transmitir información por radiofrecuencia entre aparatos móviles y fijos, emplea una configuración maestro-esclavo (maestro maneja la red y esclavo puede conectarse a otras redes). En donde cada dispositivo cuenta con un módulo Bluetooth que integran un microprocesador, transceptor y una antena.
- **Wifi. –** Es un sistema de largo alcance que permite la trasmisión de datos por medio de ondas de radio desde un emisor a un receptor, mediante modulación de frecuencia la señal análoga de las ondas de radio puede transformarse en señal digital (ceros y unos) y viceversa si el receptor actúa como emisor. Es un sistema más seguro que El Bluetooth.
- **GPS. –** Es un sistema que permite conocer la ubicación en tiempo real de cualquier dispositivo mediante radionavegación usando satélites, funcionan de manera sincronizada al comunicarse con un receptor para conocer su latitud, longitud y altitud.

2.2.3. OTROS PROTOCOLOS DE COMUNICACIÓN

- **Ethernet. –** Es una red de comunicación de alta velocidad, ya que se manejan velocidades que van de 10 Mbits/s a 100 Mbits/s, por medio de cables trenzados y fibra óptica respectivamente. En la automoción se emplean un módulo de Ethernet para manejar dichas velocidades y que las unidades de control verifiquen la línea libre (Wang et al., 2017).
- **MML Bus. –** MML **(**Mobile Multimedia Link) utiliza una topología en estrella y una configuración maestro-esclavo para interconectar unidades de control con una o dos líneas de fibra óptica, lo cual le permite disponer de una velocidad de 100 Mbits/s.

- **Intellibus.** – Es un protocolo similar a CAN que inicialmente fue desarrollado para uso militar, y ahora es empleado en vehículos con una velocidad de 12,5 Mbits/s.
- **BST Bus. –** El protocolo BST (Bosch Siemens Temic) desarrollado por la compañía Bosch, alcanza una velocidad de 250 Kbits/s para la transmisión de datos.
- **DSI Bus. –** Protocolo desarrollado por Motorola para comunicar dos unidades (maestra - esclava), a una velocidad de 150 Kbits/s con una topología punto a punto.
- **J1850. –** Protocolo inicialmente destinado para aplicaciones en vehículos americanos y diagnosis. Se encuentra normalizado como SAE J1850 y es un protocolo de mediana – baja velocidad al alcanzar 41,6 Kbits/s (Llanos López, 2017).
- **BEAN. –** Normalizado como SAE 920231, permite interconectar unidades de control con una topología de anillo doble con una velocidad mayor a 20 Kbits/s.

2.3. RED CAN-BUS

2.3.1. DESARROLLO DE LA RED CAN-BUS

La red CAN de sus siglas en inglés Controller Area Network es actualmente la red más utilizada en vehículos, fue desarrollada por Robert Bosch a comienzos de la década de los 80, y presentado en el congreso SAE (Society of Automotive Engineers) en el año de 1986 en Detroit (Estados Unidos), en donde comienzan los procesos para su normalización ISO. Debido a su versatilidad, el sistema CAN Bus fue implementado en 1991 para la producción en masa de un vehículo alemán de alta gama, denotando ser un sistema eficiente de interconexión en tiempo real entre sensores, actuadores y unidades de control para la transmisión de gran cantidad de datos (velocidad de 500 Kbits/s), capaz de realizar la comunicación y cubrir las diferentes aplicaciones en un vehículo (Paret, 2007). En el año 1993 la red CAN fue normalizada internacionalmente como ISO 11898-1, la gran acogida con fabricantes de vehículos asociados a la ISO ocasionó una mayor demanda de unidades de control con controladores CAN, lo que permitió su producción en masa y reducción de costes para la implementación de estos, ya que solucionaba el principal problema de los sistemas sobrecargados de cables y el peso de estos, con facilidad de conexión de diferentes unidades de control en paralelo a un Bus de datos de dos líneas, dejando a un lado sistemas de conexión independientes. Es una red muy confiable en cuanto a seguridad por su funcionalidad multimaestro lo que evita caídas de red, colisiones de mensajes y facilidad para la detección de errores, a tal punto de que a inicios del siglo 21 en muchos vehículos europeos la implementación de este ya era un estándar y para el año 2008 la mayor parte de fabricantes de vehículos ya lo utilizaban.

2.3.2. COMPONENTES DE UNA RED CAN-BUS

2.3.2.1. Línea de comunicación o Bus de datos

La línea de comunicación es la vía por la que se difunde la información y está formado por dos cables trenzados de 0,6 mm^2 para la comunicación entre las diferentes unidades de control o nodos, los cuales se denominan CAN High Y CAN Low, por estos cables se trasmite la misma información (señal digital) pero con diferencias de tensión el uno del otro, de manera opuesta y simétrica (figura 2.5). Es decir, se manejan dos estados y cada uno un bit, en donde de acuerdo a las normas ISO, las señales con mayor diferencia de tensión serán comprendidos como un bit dominante (0), y las señales menor diferencia de tensión un bit recesivo (1) (Llanos López, 2022).

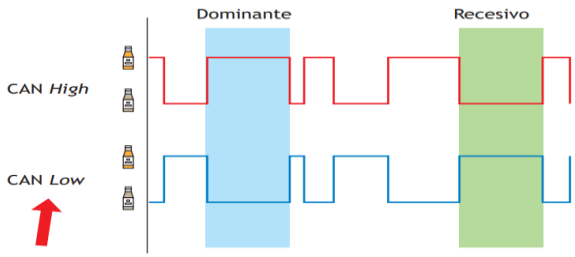


Figura 2.5 Bus de datos en la red CAN
(Sánchez, 2012, p. 293)

- Nivel de tensión, CAN High (2,5 V a 3,5 V) y CAN Low (1,5 V a 2,5 V).

2.3.2.2. Terminadores o resistencias

Las resistencias en una red CAN limitan el flujo de corriente para que el circuito mantenga los valores nominales establecidos, pero dependiendo de factores como el número de unidades de control y la distancia de cableado, las resistencias varían. Generalmente, se establece que en la red CAN se encuentran dos resistencias que se ubican al inicio y fin de la línea de comunicación en dos unidades de control principales, las cuales permiten cerrar el circuito de la red e impedir que se produzca la reflexión de las tramas de datos, cada una de estas tiene un valor de 120 Ohmios por lo que la resistencia de la red será 60 Ohmios (Gonzáles, 2019). Además, se puede encontrar una resistencia central de 66 Ohmios en unidades de control como la de gestión del motor.

- También existen resistencias internas en cada unidad de control, las cuales tendrán un valor de 2,6 kilo-ohmios y se comprobarán de manera independiente.

2.3.2.3. Unidad de control electrónico

Son computadoras que se implementan en sistemas vehiculares, a estas se les proporciona señales de entrada (información) que circula por el Bus de datos, y de acuerdo con la función

que tenga en el sistema procesa o ignora la información para generar órdenes de salida, las cuales serán trasmitidas por el Bus de datos como se evidencia en la figura 2.6 con su conexión y elementos principales, en los vehículos el número de unidades de control varía dependiendo de cada vehículo.

- **Controlador CAN. –** Se encuentra en la parte interna de cada unidad de control entre el microprocesador y el transceptor, este recibe los datos proporcionados por el transceptor para preparar la información y proporcionarla al microprocesador, así mismo envía la información proporcionada por el microprocesador hacia el transceptor, de manera que dicha información sea trasmitida por el Bus de datos hacia un componente que la necesite.
- **Transceptor CAN o receptor. –** El transceptor se encuentra ubicado entre el Bus de datos y el controlador, es el encargado de recibir la información que se transmite por el Bus, dicha información es comprobada por el transceptor para detectar anomalías en el mensaje como sobretensiones producidas por ruidos y filtrar señales parasitas (Llanos López, 2022). La información revisada solo puede traducirse en señales digitales (lenguaje binario) para que el controlador pueda comprender esos mensajes con niveles de tensión bajos, y también es el encargado de recibir los datos procesados del microcontrolador a través del controlador para traducir, amplificar la señal y transmitirla por el Bus de datos.

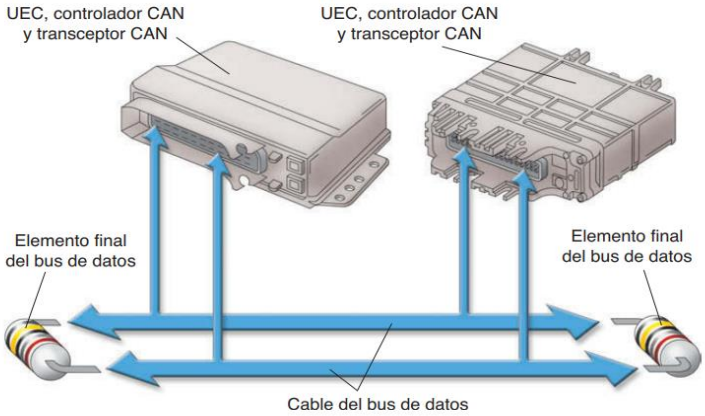


Figura 2.6 Componentes de una red CAN-BUS
(Domínguez & Ferrer, 2012, p. 89)

2.3.3. TOPOLOGÍA BUS

La topología Bus (figura 2.7) emplea una sola línea principal para el envío y recepción de datos entre diferentes unidades de control conectadas en paralelo a la misma, lo cual minimiza el uso de cableado. Si una unidad de control falla, la red no dejará de intercambiar información, solo se dejará de recibir los datos de dicha unidad en toda la red, sin embargo,

si el elemento principal es por ejemplo cortado, la red falla completamente (Reif, 2015). Es ampliamente utilizada en redes CAN por su simplicidad, bajo costo de implementación y versatilidad en cuanto a la implementación de nuevos módulos a la red, aunque de manera limitada por el número y distancia de conexión.

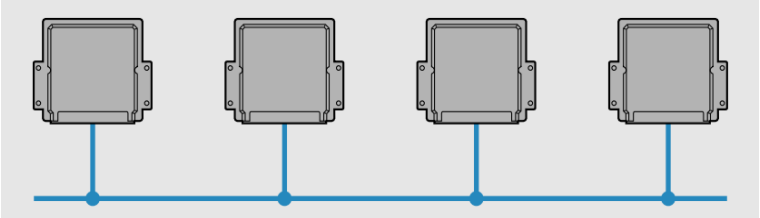


Figura 2.7 Topología bus
(Reif, 2015, p. 45)

- **Maestro-esclavo. –** La unidad maestra se encuentra conectada a la línea principal de la red, y las unidades esclavas se conectan a la maestra, creando un subsistema independiente.
- **Multimaestro. –** Varias unidades maestras se conectan directamente a la línea principal y envían mensajes según su prioridad, solo controlan sus subsistemas por salidas de datos.

2.3.4. TIPOS DE REDES CAN-BUS

En el sistema multiplexado CAN se usan buses de datos con diferentes características, ya que dependiendo del área los requisitos para una eficiente comunicación en tiempo real cambiarán. Puesto que, intervienen los factores de velocidad y seguridad.

- **CAN HS (High Speed). –** Se define como CAN de alta velocidad al trabajar en rangos de 125 Kbits/s hasta 1 Mbits/s. Se encuentra normalizada como ISO 11898-2 y debido a su velocidad de comunicación es ampliamente utilizada en sistemas de control electrónico como la gestión del motor (MEP y MEC), estabilización de vehículo y otros (Reif, 2015).
- **CAN LS (Low Speed). –** CAN de baja velocidad opera en rangos de 5 Kbits/s hasta 125 Kbits/s. Se encuentra normalizada como ISO 11898-3 y es empleada en sistemas de menor importancia en comparación al CAN HS como los diferentes sistemas de confort del vehículo (iluminación, ajuste de espejo, etc.) (Reif, 2015)

En la red CAN HS el protocolo de datos tarda alrededor de 0,25 ms y en la red CAN LS tarda 1ms, y las unidades de control intentan transmitir sus datos en un rango de 7 a 20 ms y 20 ms respectivamente.

2.3.5. VELOCIDAD DE TRANSMISIÓN DE DATOS EN REDES CAN-BUS

En una red CAN los datos se transmiten mediante agrupaciones de bits, por lo que la velocidad de transmisión determina el tiempo que tarda dicha agrupación en ser enviada completamente. En la tabla 2.3 se adjunta los valores referenciales del tiempo necesario para transmitir 1 bit y la longitud máxima de la línea de comunicación en función de la velocidad establecida.

Tabla 2.3 Principales velocidades para la transmisión de datos

Velocidad	Longitud máxima	Tiempo de transmisión bit
10 kbps	5000 m	100 µs
20 kbps	2500 m	50 µs
50 kbps	1000 m	20 µs
125 kbps	500 m	8 µs
250 kbps	250 m	4 µs
500 kbps	100 m	2 µs
800 kbps	50 m	1.25 µs
1 Mbps	30 m	1 µs

(Vela Xool, 2015, p. 22)

2.3.6. PROTOCOLO DE COMUNICACIÓN CAN-BUS

En la red multiplexada CAN, las unidades de control comparten el principio multimaestro, de modo que el protocolo de comunicación o lenguaje permite enviar y recibir mensajes entre todas las unidades de control sin depender de una unidad de control central. Actualmente, la red CAN y su protocolo de comunicación es el sistema más usado para aplicaciones en sistemas de comunicación de datos en vehículos (Sánchez et al., 2016).

La mayoría de los protocolos se basan en CAN, como es el caso de: VAN Bus, LIN Bus, MOST Bus y FlexRay. Cada uno de estos emplea un lenguaje diferente, por lo que para su correcta comunicación dentro del sistema se emplea un traductor o módulo Gateway (Puga Gutiérrez & Morales Recalde, 2022).

2.3.6.1. Direccionamiento del mensaje

El protocolo CAN se dirige a los mensajes enviados en el bus de datos, ya que cada mensaje tiene un identificador único. El cual clasifica la información para que las unidades de control verifiquen si dicho identificador se encuentra almacenado en su memoria para su aceptación, y así determinar si el mensaje es de utilidad. Al no depender de una unidad central, cada unidad de control puede llamar la información que necesite y se encuentre en el Bus de datos.

- **Formato estándar. –** También conocido como CAN 2.0 A, el identificador tiene 11 bits y puede distinguir entre 2048 diferentes mensajes CAN. En este formato se cuenta con una longitud de 130 bits como máximo.
- **Formato extendido. –** Conocido como CAN 2.0 B, el identificador está formado por 29 bits comprendidos en dos partes (11 y 18 bits), con una capacidad para distinguir mensajes CAN mayor (526 millones), y así mismo la longitud máxima de la misma con 150 bits.

2.3.6.2. Control de acceso al Bus de datos

- **Fase de arbitraje** Si el Bus de datos se encuentra en estado recesivo (1) se considera que este se encuentra libre, por lo cual se puede enviar un mensaje. La transmisión inicia con un Bit dominante (0) seguido del identificador, en el cual se aplica la fase arbitraje con la transmisión y monitoreo de los estados en el Bus de datos, perdiendo arbitraje con niveles altos o recesivos.
- **Asignación de prioridad. –** El identificador que presenta el valor binario más bajo es el que tiene mayor prioridad con el resto, en donde también interviene la velocidad en la que los bits dominantes del mensaje sobrescribirán a los recesivos (Reif, 2015).

2.3.6.3. Formato de mensaje

La transferencia datos en la red CAN se la puede realizar en dos formatos (estándar y extendido), los cuales conservan la misma organización de campos y marcos de datos, pero en el formato extendido se añade más bits por sus subcampos en los campos principales.

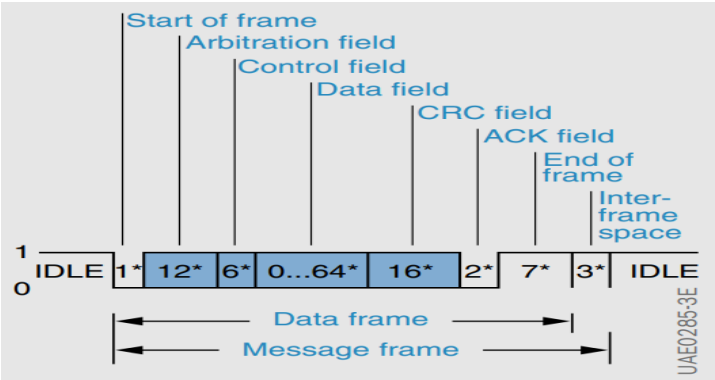


Figura 2.8 Formato de mensaje CAN
(Reif, 2015, p. 76)

En la figura 2.8 se muestra la organización del formato de un mensaje en una red CAN, en el cual intervienen 4 formatos de trama, el primero es la trama de datos o mensajes el cual contiene la información, el segundo es el cuadro remoto en donde las unidades de control solicitan la información que se relaciona con sus equipos de control, el tercero es la trama

de error, la cual comunica a las demás unidades de control de un fallo, y por último la de sobrecarga para retrasar el procesamiento de una trama con respecto a otra posterior.

- **Inicio de trama (SOF). –** Comienza con un bit dominante (0), esto cuando el bus de datos se encuentra libre (estado recesivo). Además, permite sincronizar las unidades de control.
- **Campo de estado o arbitraje. –** Se relaciona con la prioridad del mensaje en los identificadores de mensajes (11 bits y 29 bits), lo cuales comparten la característica de que al final van seguidos de un bit RTR (Solicitud de transmisión remota), el cual toma un valor dominante (0) en la trama de datos cuando envía información, y un valor recesivo (1) cuando requiere información del Bus o comprobaciones, por lo que determina el acceso.
    - CAN 2.0 A emplea una longitud de 12 bits, los primeros 11 son los identificadores del mensaje y al final el bit RTR. Tiene prioridad sobre el formato extendido cuando emplean el mismo identificador de 11 bits.
    - CAN 2.0 B utiliza 32 bits, inicia con 11 bits seguidos de un bit SRR (Solicitud remota sustitutiva) y también un bit IDE (extensión del identificador). Por lo cual, luego de los bits mencionados se añade el segundo identificador de 18 bits y al final el bit RTR (Puga Gutiérrez & Morales Recalde, 2022).
- **Campo de control. –** El presente campo está conformado por 6 bits, los cuales permiten conocer la longitud de datos que se transferirá en el campo de datos, y que las unidades receptoras determinen si el mensaje ha llegado completo. En el formato estándar este campo inicia con el bit IDE (siempre bit 0), seguido de un bit de reserva para extensiones futuras y los últimos 4 bits determinan el número de bytes a ser transmitidos. En el formato extendido se añade un bit de reserva en el bit IDE ya incluido.
- **Campo de datos. –** En el presente campo se almacena la información del mensaje, puede estar compuesto de 0 a 8 bytes. Por lo cual, consta de una longitud máxima de 64 bits. Si la cantidad de información es inferior a 8 bytes, se enviarán únicamente los bytes existentes.
- **Campo CRC. –** También conocido como campo de verificación, es el encargado de detectar interferencia o errores en la transmisión. Para la cual emplea 16 bits, los primeros 15 bits (comprobación de trama) verifican desde el inicio hasta el final de

la trama anterior si detecta algún error en los bits o en su posición, y el último bit o delimitador CRC indica el cierre del control y adopta un valor recesivo (1).

- **Campo ACK. –** También conocido como campo de confirmación, está conformado por dos bits y es el encargado de verificar si el mensaje ha sido recibido correctamente por una unidad de control, en caso de algún problema se solicitará un reenvío. El primer bit comprende la ranura ACK, que toma el valor recesivo (1) por defecto y cambia a un valor dominante (0) asegurando que el mensaje se ha recibido, y el otro bit es un delimitador que indica el final del ACK y siempre tiene un valor recesivo.
- **Fin de trama. –** Está comprendido por 7 bits con valores recesivos (1), los cuales indican el final del mensaje y a su vez indican que la línea de comunicación se encuentra libre.
- **Espacio entre tramas. –** El espacio se encuentra conformado por 3 bits recesivos, los cuales permiten separar tramas de datos y tramas remotas, pero no separan tramas de sobrecarga y error, ya que se necesita que se envíen lo más rápido posible. Si en el Bus se acumulan 10 bits recesivos, las unidades de control pueden disponer de la línea.

2.3.7. FALLOS EN EL PROTOCOLO DE COMUNICACIÓN CAN-BUS

Las unidades de control pueden detectar errores en el momento que se está trasmitiendo y receptando un mensaje, en caso de un fallo el mensaje debe ser reenviado, estas poseen contadores de errores, ya que constantes errores pueden ser comprendidos como fallos en el sistema (Martínez Requena, 2017). Los problemas más comunes se detallan en la tabla 2.4, siendo una de las causas que pueden hacer que las unidades se desconecten de la red.

Tabla 2.4 Fallos comunes en la red CAN-BUS

Descripción
• Cortocircuito entre las líneas CAN High y CAN Low
• Cortocircuito de la línea CAN High a masa o tierra
• Cortocircuito de línea CAN Low a masa o tierra
• Cortocircuito de línea CAN High a positivo
• Interrupción de línea CAN High
• Interrupción de línea CAN Low
• Cortocircuito de línea CAN High a masa o tierra con resistencia

Fuente: (Llanos López, 2022, p. 112)

2.3.8. VENTAJAS DE LA RED CAN-BUS

Debido a la versatilidad que refleja la incorporación de una red CAN-BUS en cuanto a seguridad, caídas de red, interferencias, colisiones y adaptabilidad a adversidades con sus velocidades de transmisión y detección de errores. Representan ventajas en sistema con características como:

- Principio de funcionamiento multimaestro y flexibilidad de configuración.
- Alta fiabilidad para transmitir datos con los cables trenzados, de modo que el protocolo reduce interferencias de señales parasitas y campos magnéticos del propio Bus de datos.
- Diseño simple y económico para la implementación en sistemas automotrices según los requerimientos de la red con una red de alta o baja velocidad.
- Detección y localización de averías. Además de un bajo consumo de energía.
- El software permite ampliar información suplementaria (funciones) en el protocolo.
- Comunicación priorizada para sistemas con mayor relevancia como la gestión del motor en comparación a sistemas de confort.
- Protocolo de comunicación efectivo para el buen desempeño del vehículo en tiempo real, manejando hasta 8 bytes de capacidad por mensaje y una velocidad límite de 1 Mbits/s.
- Método de entrada al Bus de datos no destructivo.

2.4. SISTEMAS EMBEBIDOS

Los sistemas embebidos son circuitos electrónicos de hardware y software que se interconectan a un sistema general (red), permiten solucionar problemas al realizar labores específicas que son inviables únicamente con tecnología hidráulica y mecánica. En el caso de la industria automotriz han permitido cubrir las distintas necesidades de los usuarios y el funcionamiento en general de un vehículo, ya que aportan inteligencia en una red que interconecta múltiples procesadores (Navet & Simonot-Lion, 2017).

En la década de los 60 se empezó con la sustitución y adaptación de componentes electrónicos en remplazo de componentes mecánicos, que tenían el inconveniente de ser robustos, consumir gran cantidad de energía y costosos. El autor Gustavo Galeano (2009) menciona que, debido a su versatilidad en cuanto al rango de aplicaciones en las distintas industrias, los sistemas embebidos ahora son mucho más compactos al integrar

microcontroladores, por lo que consumen menos energía y su precio es accesible al ser producidos en masa.

2.4.1. CARACTERÍSTICAS DE UN SISTEMA EMBEBIDO

Los sistemas embebidos automotrices son sistemas que funcionan de manera independiente, pero se relacionan con un sistema más grande para realizar ciertas funciones en conjunto, que sirven para el funcionamiento general del vehículo. Se destacan 3 características:

1. En un sistema embebido se ejecutan funciones específicas de acuerdo con las instrucciones de programación establecidas de manera repetitiva y no requieren que se ejecuten programas como en sistemas computacionales.
2. Los sistemas son compactos en su diseño al disminuir elementos adicionales para su funcionamiento, sus componentes son desarrollados con tamaños reducidos que permiten ahorrar espacio, costos de producción y consumo de energía sin disminuir su capacidad de procesamiento para realizar una determina función (Manosalvas Salazar, 2017).
3. Sistema reactivo que interactúa con el mundo exterior a través de sensores y actuadores en tiempo real, el cual ejecuta sus funciones con una interacción de usuario mínima y de manera segura, incluso al margen de cambios climáticos.

2.4.2. HARDWARE DE UN SISTEMA EMBEBIDO

El término Hardware se refiere a los componentes físicos que permiten construir un circuito integrado, siendo normalmente módulos que interactúan con sus entradas y salidas de datos para realizar una determinada función según su programación. Los sistemas integrados comparten características en cuanto a su diseño de dimensiones pequeñas, bajo costo, funcionalidad, capacidad de procesamiento y consumo de energía.

- **Microcontrolador/microprocesador. –** Es el componente que se encarga de efectuar las funciones o tareas establecidas en la programación con dispositivos de entrada y salida, por medio de su capacidad de cómputo en forma de circuito integrado con múltiples componentes internos.
- **Memorias. –** Permite almacenar el código de programación y datos del sistema, las cuales en microprocesadores o microcontroladores permiten dar lectura y escritura a los datos almacenados lo más rápido posible para ser utilizados.
- **Dispositivos de entrada y salida.** – Permiten a los microcontroladores/procesadores realizar una función específica por medio de señales eléctricas. Los dispositivos de

entrada se refieren a los sensores capaces de transformar fenómenos físicos en señales eléctricas y otros componentes que adoptan estados fijos y temporales de generación de señal. Los dispositivos de salida son los actuadores capaces de transformar las señales eléctricas (órdenes) en fenómenos físicos.

- **Cableado. –** Permiten transportar la energía eléctrica desde un punto a otro punto.

2.4.3. SOFTWARE EN UN SISTEMA EMBEBIDO

El software en un sistema embebido es el firmware que permite controlar dispositivos electrónicos al ejecutar instrucciones de forma repetitiva con un microprocesador interno o en un microcontrolador. También se puede ejecutar en un procesador de señales digitales (DSP) o en una matriz de puertas lógicas programables (FPGA). Dichas instrucciones o tareas programadas se las realiza en un programa o entorno de desarrollo integrado (IDE) que permite realizar la programación para el microcontrolador, el cual lo proporciona el fabricante.

La asociación de desarrollo AUTOSAR se creó en 2003 con el fin de desarrollar un estándar sobre interfaces de aplicación en arquitecturas de software automotriz. En sintaxis y semánticas en: seguridad, control de chasis, carrocería, comodidad, tren motriz en motor y transmisión (Sangiovanni-Vincentelli & Di Natale, 2007).

2.5. MICROCONTROLADORES PROGRAMABLES

En sus inicios los microcontroladores eran conocidos como microcomputadores, ya que integran un sistema cerrado que contienen las unidades básicas de una computadora, pero con características limitadas. Son dispositivos electrónicos con un circuito integrado programable y que, por su tamaño, puede incorporarse en el mismo dispositivo que gobierna para controlar una función establecida, mediante instrucciones previamente programadas y almacenadas en su memoria. Anteriormente, el desarrollo de circuitos se lo realizaba de manera manual, por lo que para realizar modificaciones había que realizar cambios en sus componentes, como soldar y desoldar. Por lo cual, para facilitar el desarrollo de circuitos electrónicos, ahora con el uso de tecnologías digitales se emplea tanto software como hardware (Banzi & Shiloh, 2022).

2.5.1. HARDWARE ARDUINO

El microcontrolador Arduino es un pequeño chip que se integra en un circuito compacto (placa), el cual puede ser programado en lenguaje C/C++ con la ayuda del software IDE traduciendo el lenguaje processing. La placa tiene pines digitales y analógicos, que permiten

conectar y controlar diferentes receptores y actuadores, por medio del procesamiento de la placa con la información entrante y dependiendo de la programación establecida en el microcontrolador y almacenada en su memoria. Debido a la popularidad de los microcontroladores Arduino, existen placas con diferentes características como: tamaño, diseño, cantidad de puertos de entrada y salida de información (análoga y digital), pero principalmente la capacidad de procesamiento del chip que integran (Pan & Zhu, 2018). Con lo cual se destaca la existencia de dos categorías de chips, los cuales se implementan en las placas dependiendo de las necesidades y son los siguientes:

- Standard: Atmega8/168/328
- Mega: Atmega16U2/1280/2560

2.5.1.1. Características de una placa Arduino

Las placas Arduino más empleadas en el desarrollo de proyectos son: Arduino UNO y Arduino MEGA 2560. Los cuales comparten características bases similares, sin embargo, en cuanto al microprocesador, capacidad de memorias y número de pines (analógicos y digitales), el Arduino MEGA 2560 (figura 2.9), lo supera.

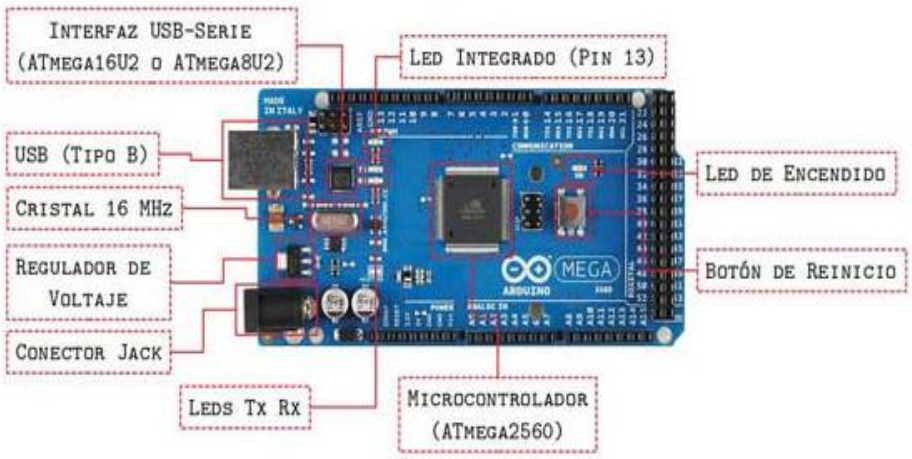


Figura 2.9 Arduino Mega 2560
(Guerra Carmenate, 2019)

- **Pines digitales. –** Como su nombre lo indica, maneja el ingreso y salida de señales digitales, es decir, dos estados o valores (0 y 5 voltios). El direccionamiento de datos dependerá de lo especificado en el código de programación.
- **PWM. –** Salidas de ancho de pulso modulado (**~)** con capacidad de 1 byte (8 bits), con capacidad para ser empleadas como salidas analógicas.
- **Rx y Tx. –** Son pines UART (Transmisor-Receptor Asíncrono Universal) de recepción (Rx) y transmisión de datos (Tx), los cuales son empleados para establecer

una comunicación serial. En los que, al igual que Arduino Uno, comunican dos o más ordenadores o microcontroladores con un nivel TTL de 0V/5V.

- **Bus SPI. –** De sus siglas en inglés Serial Peripheral Interface, es un bus de comunicación serial que maneja una arquitectura de maestro y esclavo en líneas independientes, es decir de maestro-esclavo y esclavo-maestro, por medio de un transmisor MOSI (Master Out Slave In), un receptor MISO (Master In Slave Out), una línea reloj (SCK) y selección de esclavo (SS), la cual permite seleccionar el esclavo para la comunicación por medio de las demás líneas (Llamas, 2016).
- **Bus I2C.** – De sus siglas en ingles Inter-Integrated Circuit es un bus que al igual que SPI, permite establecer una comunicación serial con una configuración maestro-esclavo, pero únicamente con dos cables: una línea para el envío de datos bidireccional (SDA) y una de pulsos de reloj (SCL). Los dispositivos se conectan a las mismas líneas comunes, pero los esclavos no pueden iniciar una comunicación.
- **Pines analógicos.** – Los pines de entrada y salida analógicos pueden manejar valores de tensión diferentes a dos estados o valores, ya que, por medio de un convertidor análogo-digital interno de 10 bits, se puede representar valores que van de 0 a 5 voltios en 1024 niveles de señal.
- **Alimentación.** – La placa Arduino UNO y Arduino MEGA 2560 soportan una tensión de 7 a 12 voltios de corriente continua mediante el puerto principal y voltaje operativo de 5 voltios. Pines de entrada y salida con alimentación de 5 voltios a 40 miliamperios y pin especial de 3,3 voltios a 50 miliamperios (Arduino, n.d.).
- **Memorias.** – Las placas Arduino tienen tres tipos de memorias, con capacidad que varía según el microcontrolador.
    - SRAM: Es una memoria volátil de lectura y escritura, la cual permite crear y escribir variables cuando se ejecutan las instrucciones.
    - EEPROM: Memoria no volátil, la cual permite eliminar y volver a grabar información, es solo de lectura y conserva datos incluso al reiniciarse.
    - FLASH: Memoria no volátil que permite almacenar información como el código de programación ya compilado, con una capacidad desde 1KB y 4MB.
- **Velocidad reloj.** – Se refiere al número de ciclos que el microprocesador de Arduino puede realizar por segundo, o la frecuencia de procesamiento medida en Hertz (Hz).
- **GND. –** Pin destinado para cerrar circuito a masa o tierra.
- **Reset. –** Permite reiniciar el procesador

- **Protección de sobrecarga. –** Las placas Arduino cuentan con una protección (fisibles reseteables) en caso de cortocircuitos de alrededor de 1 amperio (Puga Gutiérrez & Morales Recalde, 2022).

2.5.2. SOFTWARE ARDUINO IDE

El software Arduino IDE (Entorno de Desarrollo Integrado) es el programa más conocido y empleado para manejar código Arduino, ya que con el mismo se puede crear, trascribir y modificar códigos para dar vida a un circuito que integra un microcontrolador. El cual es de libre acceso y puede ser instalado en los 3 principales sistemas operativos (Windows, macOS, Linux), permitiendo que personas con poca experiencia puedan desarrollar bocetos de programación de manera intuitiva, siguiendo el modelo de lenguaje processing el cual es muy similar al lenguaje C y C++ (Banzi & Shiloh, 2022).

2.5.2.1. Programar

Es el desarrollo del código, en donde se describe las instrucciones con las cuales se regirá el microcontrolador, se desarrolla a través de un entorno de desarrollo. En el caso de Arduino la programación se la realiza en el software Arduino IDE, por lo cual se debe tener en cuenta las características del programa, como: sus librerías, lenguaje de alto nivel parecido a C++ y su estructura básica de programación para organizar las instrucciones:

- La declaración de variables puede realizarse antes de la configuración Setup ().
- **Setup (). –** En el programa es la línea de código que se ejecuta una sola vez, por lo cual esta función es utilizada para inicializar los pines y puertos en serie**.**
- **Loop (). –** Es la función que se ejecuta después de Setup (), pero no una sola vez sino de forma cíclica. Así, las líneas de código dentro de esta función permiten responder ante diferentes eventos incluidos en las instrucciones.

2.5.2.2. Librerías

Las librerías, o también conocidas como bibliotecas, son códigos desarrollados para facilitar la conexión, manejo de datos y ampliar la funcionalidad de hardware como: actuadores, Shields, pantallas, entre otros (Aljundi, 2023). Al momento de instalar el programa de Arduino IDE algunas librerías vienen incluidas, pero también desde el programa se pueden descargar directamente librerías que se encuentran enumeradas en el apartado de referencias de bibliotecas de Arduino, además de poder ser creadas por un usuario según sus propias necesidades de desarrollo.

2.5.3. CAN-BUS SHIELD

Los Shields son placas de circuito electrónico que se pueden utilizar junto con un Microcontrolador Arduino para expandir sus funciones al proporcionar nuevo hardware. El microcontrolador CAN-BUS Shield (figura 2.10) brinda capacidades de CAN Bus a la placa Arduino, permitiendo obtener información entre nodos y almacenamiento de datos, con amplias aplicaciones automotrices como el desarrollo de redes multiplexadas, módulo OBD II, etc. La placa puede encajar en diferentes placas Arduino con dimensiones similares (68x53 mm) sin la necesidad de soldadura, en el caso de la primera versión solo es compatible con Arduino Uno y versiones más recientes se añaden Arduino Mega, Arduino Leonardo y Linklt One (García Osés, 2015).

2.5.3.1. Descripción general del hardware CAN-BUS Shield

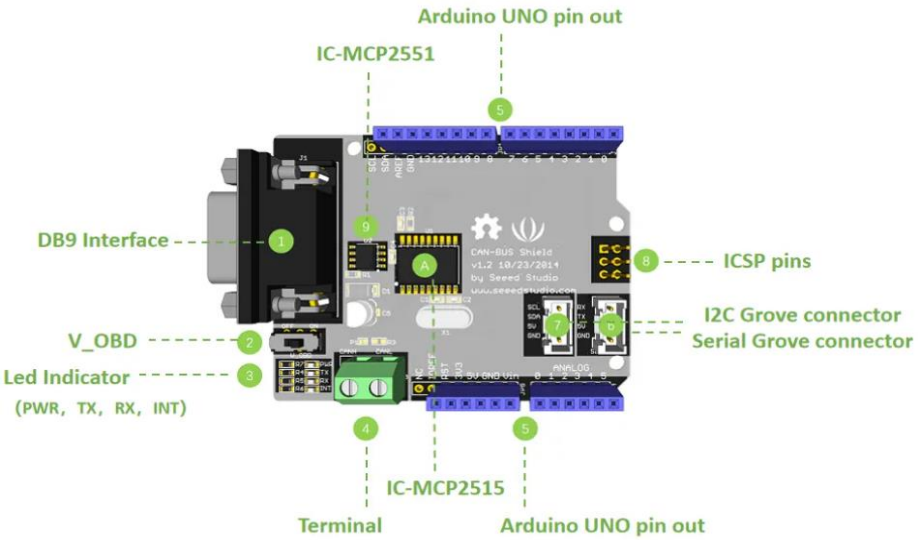


Figura 2.10 Hardware de la placa Arduino CAN-BUS Shield
(SeeedStudio, 2018)

El hardware de la placa CAN-BUS Shield está compuesto por puentes de terminales, que se relacionan con los pines de la placa Arduino para su acoplamiento. Según la versión de la placa, la ubicación de terminales, componentes y sus especificaciones pueden cambiar, pero tienen el mismo objetivo de permitir establecer una comunicación CAN con sus rangos de velocidad y seguridad, por lo que a continuación se presentan sus características:

- Cuenta con un controlador y un transceptor, que por lo general son el MCP2515 y el MCP2551 respectivamente.
- Velocidad de transmisión de datos de hasta 1 Mbps.
- Permite establecer una comunicación serial SPI, I2C y UART (Tx, Rx) Arduino.

- Leds referenciales de la transmisión, recepción e interrupción de la comunicación.
- Conectores: DB9 estándar de 9 pines, conector GPS y conector LCD.
- Terminal CAN High y CAN Low, como líneas de comunicación.
- Velocidad de reloj directamente proporcional con el microcontrolador Arduino base, si la placa cuenta con un cristal de 8 MHz la velocidad real será la mitad de la establecida en Arduino de 16 MHz.
- La alimentación de funcionamiento es la misma que la placa a ser utilizada (5V).
- Bajo consumo de corriente.
- Implementa un botón Reset.

2.5.3.2. Versiones de la placa CAN-BUS Shield

Debido a las características de la placa en cuanto a la velocidad de comunicación, alta confiabilidad y larga distancia de viaje de datos, se han mejorado las prestaciones en sus nuevas versiones. Al igual que otros componentes, los fabricantes varían, pero comparten características bases (tabla 2.5) para tener compatibilidad con la comunicación CAN.

Tabla 2.5 Comparativa de las versiones CAN-BUS Shield

Características	**Versiones**		
	Seeed Studio		**SunFlower**
	V1	V2	V3
Controlador	MCP 2515	MCP 2515	MCP 2515
Transceptor	MCP 2551	MCP 2551	TJA 1050
Asignación pines CAN	No compatible	Compatible con puente	Compatible con puente
Asignación pines OBD II	Estándar OBD II	Estándar OBD II	Estándar OBD II
Pin CS para ranura TF	Sin ranura TF	D4 o D5	D10
Ranura I2C	A4/A5	SDA/SCL	SDA/SCL
Ranura Serie	D0/D1	A0/A1	D0/D1
Pin INT	No alterable	D2 o D3	D2
Orientación	Vertical	Horizontal	Horizontal

Fuente: (SeeedStudio, n.d., p. 3)

2.5.3.3. Funciones de la placa CAN-BUS Shield

Permite establecer una comunicación CAN entre dos o más microcontroladores, es un circuito integrado de alto rendimiento con un transceptor MCP2551 de alta velocidad y un controlador MCP2515 independiente, que posibilita intercambiar información de forma bidireccional al igual que los módulos automotrices. Es compatible con estándares

industriales por el conector DB9, la velocidad que puede alcanzar para la trasmisión de datos y el direccionamiento del mensaje, al manejar tramas de datos estándar (11 bits) y extendida (29 bits) (SeeedStudio, 2018). Además, de poder controlar el tráfico de mensajes y aceptación por prioridad, con la ayuda de búfers internos que emplean filtros y máscaras que se compararan con datos almacenados de referencia.

2.6. SISTEMAS DE SEGURIDAD ACTIVA Y CONTROL DE LA CARROCERÍA

Los sistemas de seguridad y control permiten salvaguardar las vidas de los usuarios de vehículos y mejorar su experiencia de conducción, los avances tecnológicos de las redes de comunicación automotrices ofrecen la capacidad de establecer parámetros embebidos entre los diferentes componentes de la red, ampliando la funcionalidad de los mismos con sistemas cada vez más complejos que ayudan a mejorar de manera indirecta la eficiencia de otras áreas de control.

2.6.1. SEGURIDAD ACTIVA

La seguridad activa se refiere a las acciones directas o indirectas que genera el conductor para prevenir accidentes, incluyendo el correcto uso de los sistemas y la automatización de estos al presentarse situaciones específicas.

2.6.1.1. Sistema de luces

Figura 2.11 Tipos de luces automotrices
(Ingeniería y mecánica automotriz, 2020)

El sistema de luces proporciona mayor visibilidad al conductor en ambientes con poca luz o durante la noche, pero también proporciona información visual a otros conductores como la posición del vehículo, dimensiones y las intenciones del conductor. En relación con esto, es importante conocer los tipos de luces de un vehículo (figura 2.11) y de igual manera sus principales características (tabla 2.6), las cuales en la actualidad son controladas por unidades de control a través de redes y protocolos de comunicación, permitiendo generar sistemas embebidos.

Tabla 2.6 Descripción de los tipos de luces automotrices

Denominación	Descripción
Luces de posición	Luces de baja intensidad que indicar la presencia y dimensiones de un vehículo, normalmente se emplea dos faros de color blanco o ámbar en la parte frontal, faros laterales de color ámbar dependiendo de la longitud del vehículo y dos faros de color rojo en la parte posterior.
Luces de cruce	Luces de color blanco que se ubican en los extremos de la parte frontal de un vehículo, durante la conducción son las luces más utilizadas, ya que no deslumbran a los conductores de carriles opuestos ni peatones.
Luces de carretera	Luces de alta intensidad que se usan para alumbrar la vía cuando esta esta despejada de vehículos en sentido contrario, faros de color blanco ubicados en cada extremo de la parte frontal de un vehículo.
Luces antiniebla	Luces de color blanco o selectiva que debido a su ubicación e intensidad concentran su rayo de luz hacia la carretera, proporcionado al conductor mayor visibilidad y a los demás conductores visibilidad del vehículo en ambientes con neblina densa, polvo, entre otros. Por lo cual, pueden implementarse dos faros en la parte frontal y posterior.
Luces de dirección	Luces de color ámbar que indican la intención de giro hacia la izquierda o derecha de un conductor, ubicadas en los extremos de la parte frontal y posterior de un vehículo con un funcionamiento intermitente.
Luces de emergencia	Este sistema por lo general emplea las mismas luces de dirección para indicar de forma intermitente una situación de emergencia o precaución, en donde el conductor tiene la intención de estacionarse o se encuentra ya estacionado.
Luces de frenado	Luces de color rojo ubicadas en la parte posterior del vehículo, las cuales indican que un conductor está presionando los frenos y reduciendo la velocidad, por lo cual tienen mayor intensidad que las luces de posición.
Luces de retro	Luces de color blanco ubicadas en los extremos de la parte posterior de un vehículo, indican que se ha realizado el cambio a reversa, además de proporcionar mayor visibilidad al conductor al realizar dicha maniobra.

Fuente: (Instituto Ecuatoriano de Normalización, 2009)

2.6.1.2. Sistema de iluminación adaptativa

Sistema que permite automatizar los parámetros de funcionamiento del sistema de luces al presentarse situaciones específicas, mejorando la visibilidad y seguridad tanto al conductor como a los peatones.

- **Asistente de luz de carretera. –** Realiza una conmutación automática de las luces de carretera a las de cruce en presencia de población y cuando un vehículo se acerca en sentido contrario del carril o en el mismo sentido, de manera que se evite deslumbrar a las demás personas y conductores, reduciendo riesgos de colisión (Ros Marin & Barrera Doblado, 2011, p. 160).
- **Iluminación direccional.** – Sistema se ajusta automáticamente la posición o iluminación de los faros en función de la dirección o ángulo de giro del volante, de manera que se tenga una mejor visibilidad en ambiente con poca luz o en curvas e intersecciones.
- **Iluminación en función de condiciones.** – Sistema que ajusta y adapta la funcionalidad del sistema de luces en situaciones específicas que pueden variar por los fabricantes.

2.6.1.3. Sistema de advertencia de colisión

Tecnología diseñada para detectar la presencia de otro automóvil u objeto en puntos ciegos del automóvil al realizar una maniobra de conducción, por medio de sensores ultrasónicos o de radar que permiten determinar su distancia, y en caso de ser necesario activar por medio de una unidad de control alertas acústicas y visuales al conductor, para evitar colisiones.

2.6.2. SISTEMAS DE CONTROL DE LA CARROCERÍA RELACIONADOS CON LA SEGURIDAD ACTIVA

2.6.2.1. Sistema limpiaparabrisas

Es un conjunto de componentes que permite mantener la parte externa del parabrisas libre de suciedad para garantizar una adecuada visibilidad al conductor, por lo cual se relaciona con la seguridad activa. Integra un motor de corriente continua (DC) que acciona un mecanismo de brazos que se desplaza sobre el parabrisas, realizando movimiento de 0 a 90° o 120° y viceversa. Puede trabajar a diferentes velocidades según corresponda y algunos vehículos pueden implementar un limpiaparabrisas adicional en la parte trasera.

- **Control de visibilidad en función de condiciones. –** La automatización del sistema de limpiaparabrisas es una característica común en vehículos modernos, ya que, dependiendo de la presencia e intensidad de lluvia detectada por un sensor con placas resistivas sensibles a la humedad, el sistema activa y ajusta de manera automática la velocidad del motor DC, aunque también el usuario puede configurar la respuesta.

2.6.2.2. Sistema de dirección asistida electrónica

Sistema de asistencia al conductor que permite realizar maniobras en el volante con menos esfuerzo que la dirección hidráulica y mecánica, ya que emplea un motor eléctrico controlado por una unidad de control en respuesta a señales de sensores, velocidad de las ruedas, ángulo de dirección y de par. En consecuencia, proporciona una mayor flexibilidad y precisión en el ajuste de la dirección, generando un sistema seguro y eficaz.

CAPÍTULO II

3. MATERIALES Y MÉTODOS

En la elaboración de la red multiplexada CAN en base a microcontroladores Arduino para el control de sistemas embebidos automotrices, se utilizaron diferentes materiales (hardware y software) para la interconexión, alimentación, programación y control de los diferentes dispositivos conectados a la red. Al igual que los sistemas automotrices implementados en cualquier vehículo, el prototipo permite enviar y recibir información (mensajes) entre los módulos conectados a la red, de manera que se pueda realizar diferentes funciones establecidas en la programación de dispositivos adaptados, con sus entradas y salidas de datos. El sistema se enfocó en necesidades cotidianas de usuarios, permitiendo conocer el funcionamiento interno y externo de una red multiplexada CAN y su control en tiempo real. Por lo cual, en el presente capítulo se detalla el proceso metodológico para su realización.

3.1. METODOLOGÍA

El proceso metodológico (figura 3.1) para elaborar y desarrollar el trabajo de investigación, se encuentra estructurado con actividades secuenciales enfocadas en el cumplimiento de los objetivos específicos, incluyendo en las principales, subprocesos metodológicos para facilitar la comprensión. El flujograma comienza con la recopilación bibliográfica de los temas más relevantes que permitieron fundamentar el capítulo I, como material de apoyo para el desarrollo metodológico del trabajo de investigación, donde destaca el funcionamiento interno y externo de las redes multiplexadas, principalmente basadas en la red CAN y microcontroladores programables.

En segunda instancia se desarrolló el proceso empleado para diseñar el sistema multiplexado de la red, mediante el análisis de los parámetros requeridos para la operación de la red, materiales de conexión y los diagramas de los elementos que intervienen en la comunicación, con la finalidad de realizar un esquema general de la red CAN empleando una topología Bus y las resistencias de terminación características. Después, en el desarrollo de la red, se estructuró un algoritmo de pasos lógicos que sirva como guía para comprobar el funcionamiento de las placas, y establecer una comunicación multidifusión serial entre los 4 módulos de la red, utilizando el software Arduino IDE para su programación paulatina, de manera que sea posible comprobar el comportamiento de los mismos mediante herramientas de análisis y solucionar posibles errores.

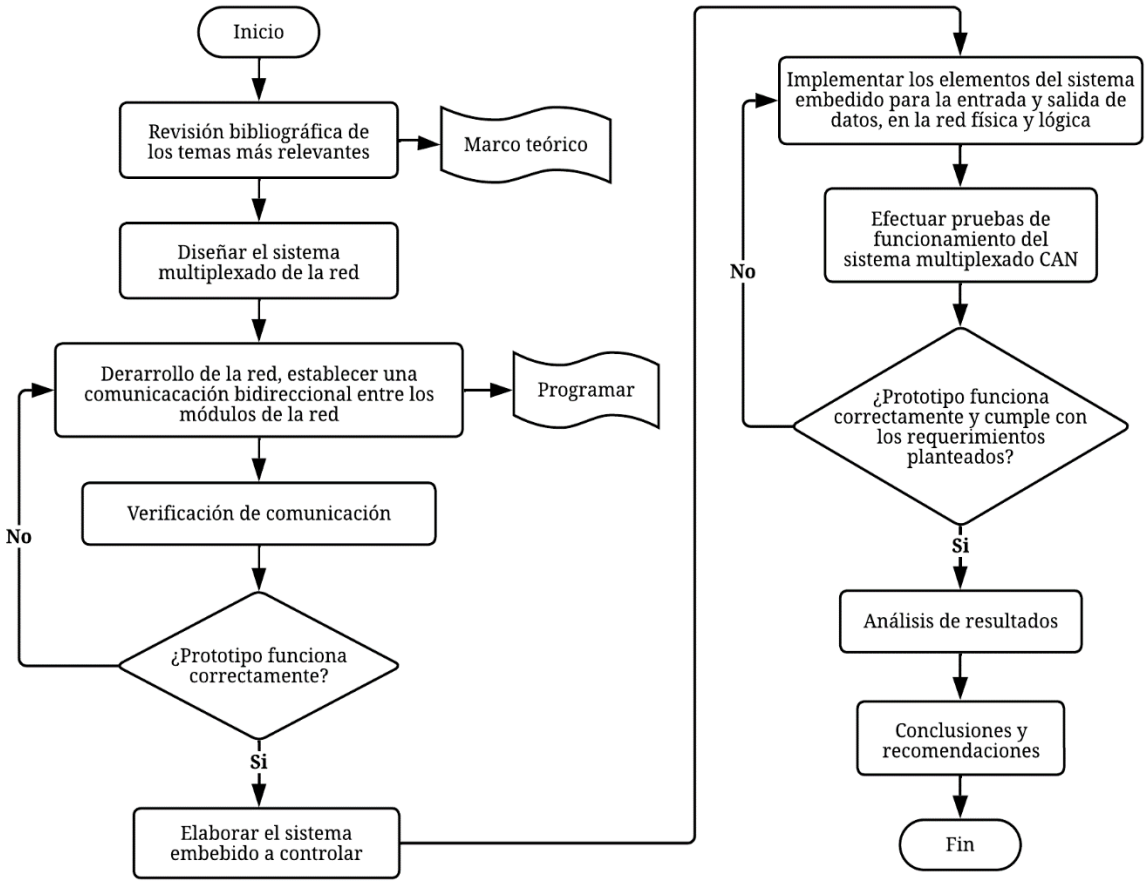


Figura 3.1 Flujograma general del proyecto

Fuente: Autor

Después de verificar la correcta comunicación entre los módulos de la red, se diseñó el sistema embebido a controlar, cubriendo lo propuesto en el alcance del presente proyecto con el sistema de control de la carrocería, automatización y tracción eléctrica. Por lo cual, se detallan los materiales en cuanto a sus entradas y salidas de datos, y la configuración de los diferentes módulos de la red para generar un arreglo eficiente con los componentes que cada uno integra, teniendo en cuenta parámetros de funcionamiento automotrices para su programación, como la implementación del principio de comunicación multidifusión por eventos. Por último, se evaluó la operación y desempeño de la red, integrando todo el sistema en un prototipo tipo maqueta con los equipos de análisis, permitiendo conocer el comportamiento de las señales digitales transmitidas por el Bus de datos.

3.1.1. DISEÑO DEL SISTEMA MULTIPLEXADO DE LA RED

Para diseñar el sistema base de la red CAN, se empleó el proceso representado en la figura 3.2, abordando los parámetros que intervienen en el funcionamiento de la red, los principales componentes (hardware) y las conexiones físicas y lógicas de la red.

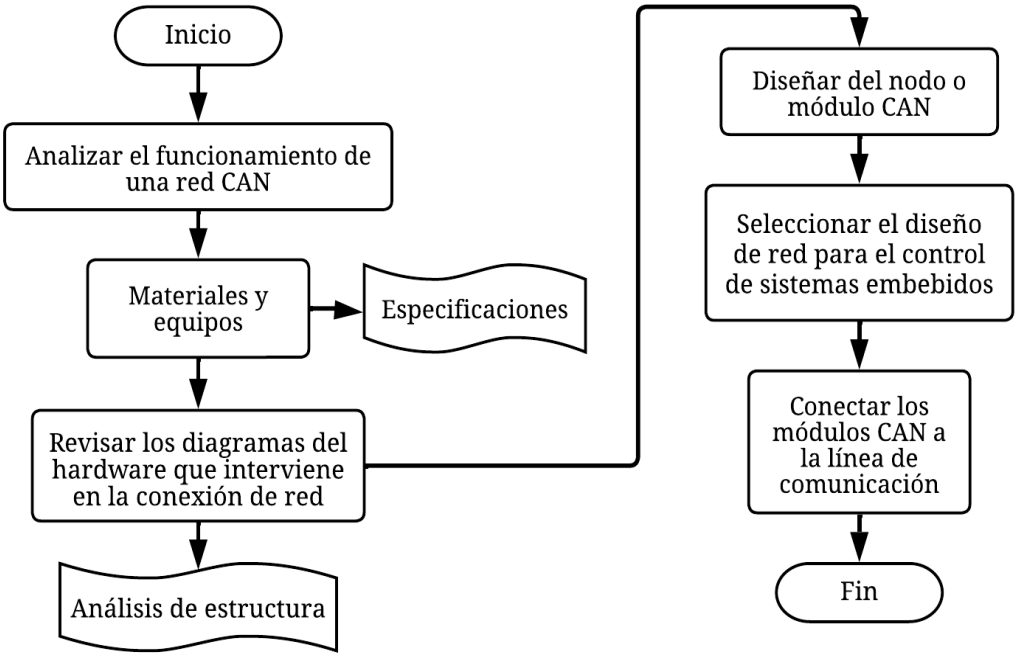


Figura 3.2 Esquema metodológico para el diseño de red

Fuente: Autor

3.1.1.1. Análisis del funcionamiento de una red CAN

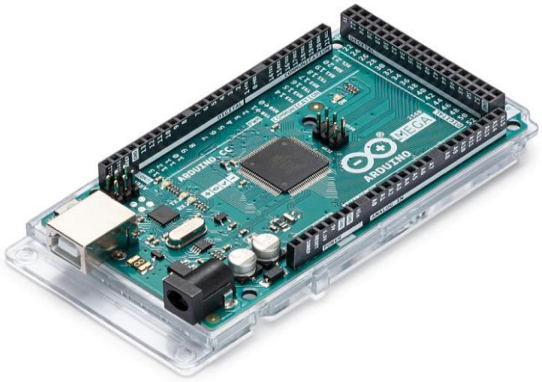
En este punto se realizó un análisis del funcionamiento de una red CAN, tomando en cuenta sus principales componentes, clasificación y protocolo de comunicación. Una red CAN emplea dos cables trazados con una topología Bus y dos resistencias de 120 ohmios en los módulos de sus extremos, por las líneas de comunicación se transmite la misma señal digital, pero de manera opuesta y simétrica, siendo CAN HIGH y CAN LOW con valores de tensión de (2,5 V a 3,5 V) y (1,5 V a 2,5 V) respectivamente. En donde, los parámetros para la comunicación dependen del sistema a ser implementado, tenido a disposición dos rangos de velocidades, las de alta velocidad de 125 Kbits/s hasta 1 Mbits/s y las de baja velocidad de 5 Kbits/s hasta 125 Kbits/s, además del manejo de identificadores ya sea en formato estándar (11 bits) o extendido (29 bits), los cuales determinan la prioridad cuando múltiples módulos intentan acceder al Bus de datos y la interacción para reconocer los mensajes enviado y recibidos, evitando colisiones y así generando una comunicación segura al no existir una unidad de control principal.

3.1.1.2. Materiales y equipos

Los materiales utilizados en el desarrollo de los módulos CAN y el sistema de red mutiplexado principalmente son: los microcontroladores Arduino Mega 2560, placas CAN Bus Shield y los componentes para su conexión física.

Arduino MEGA 2560

Tabla 3.1 Especificaciones del microcontrolador Arduino Mega 2560



θ

Microcontrolador	Atmega 2560	Características adicionales
Dimensiones	Longitud: 101.52 mm Ancho: 53.3 mm	**Comunicación serial** UART: Tx (pines 1,14,16,18) y Rx (pines 0,15,17,19). Bus SPI: líneas MISO, MOSI SCK y SS. Bus I2C: pin 20 (SDA) y 21 (SCL). **Corriente (CC)** Pines E/S de 5 V: 40 mA Pin de 3.3 V: 50 mA **Otros** **Peso:** 37 g **Pin GND:** Masa o tierra **Botón Reset:** Reiniciar microprocesador
Alimentación (V)	Funcionamiento: 5 V Recomendada: 7-12 V Mínimo y máximo: 6-20 V	
Velocidad reloj	16 MHz	
Pines de entrada y salida de datos	Digitales: 54 pines (0 – 53) de los que 15 pines (2 – 13; 44 – 46) pueden usarse como salidas PWM. Analógicos: 16 pines de entrada (A0 – A15)	
Memorias	SRAM: 8 KB EEPROM: 4 KB FLASH: 256 KB, 8 KB aprovechados en el arranque	

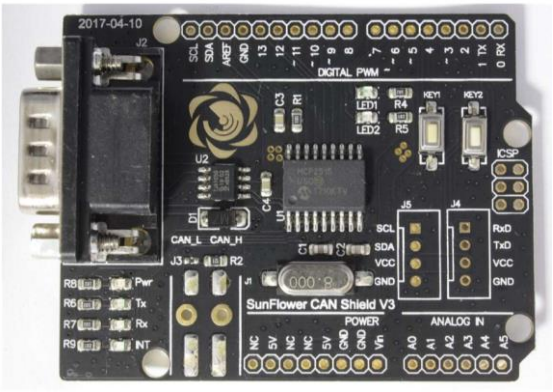
Fuente: (Arduino, 2023)**,** θ (Arduino, n.d.)

El Arduino MEGA 2560 es una placa de desarrollo muy versátil y con prestaciones técnicas (tabla 3.1) que amplían las posibilidades de realizar proyectos con grandes exigencias de software y hardware, funciona a una frecuencia de 16 MHz y cuenta con 54 pines de entrada y salida de datos digitales (0 – 53), de los cuales 15 pines (2 – 13; 44 – 46) pueden ser utilizados como salidas PWM con capacidad de 8 bits y también como salidas analógicas, así mismo cuenta con 16 pines de entrada de datos analógicos (A0 – A15), la cual al igual que Arduino UNO maneja 1024 niveles de señal. Además de 4 unidades UART, transmisores Tx (pines 1,14,16,18) y receptores Rx (pines 0,15,17,19), memorias, conectores para su alimentación y programación ICSP (Arduino, 2023).

CAN-BUS Shield V3.0

Es un Shield o mochila que permite ampliar las funciones del microcontrolador Arduino en cuanto a comunicación CAN, internamente implementa el controlador MCP 2515 y transceptor TJA 1050. Su diseño principalmente compatible con el Arduino UNO permite su fácil acoplamiento, teniendo en cuenta que los pines SPI de las dos placas integradas estén conectados al igual que la salida de interrupción a INTO. En la tabla 3.2 se describe sus principales especificaciones y una representación del mismo.

Tabla 3.2 Especificaciones técnicas de la placa CAN Shield V3.0



9

Controlador	MCP 2515	Características adicionales
Transceptor	TJA 1050	Maneja el protocolo CAN, con una velocidad de hasta 1 MBit/s
Dimensiones	Largo: 65 mm Ancho: 53 mm	**Comunicación** I2C, UART, SPI
Alimentación (V)	4.8 V – 5.2 V	
Corriente (CC)	30 mA	**Trama de datos** Estándar (11 bits) y extendida (29 bits) y marcos remotos
Puertos CAN	CAN High – CAN Low	
Compatibilidad	Principalmente Arduino UNO, Mega y Leonardo	**Temperatura funcionamiento** Rango de 85°C - 40°C
Velocidad reloj	8 MHz	**Conector** DB9, con pines estándar OBD II
Reset	Reinicio de la placa	**Peso:** 50 g
Stand-by	Bajo consumo de corriente (1 uA)	

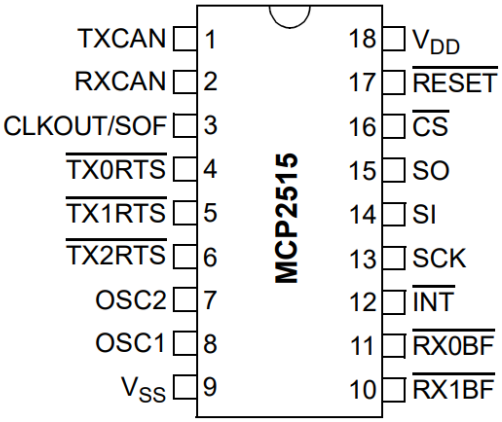
Fuente: (Marotronics, n.d.)**,** 9 (Marotronics, n.d.)

Controlador MCP 2515

Es el controlador CAN que integra la placa CAN-BUS Shield V3.0 basado en la versión 2.0B, por lo cual en la trama de datos emplea un arbitraje de recepción. En la tabla 3.3 se incluye el esquema del microchip MCP 2515 y sus principales características, dicho controlador por medio de la interfaz SPI permite enviar y recibir tramas de datos en formato

estándar y extendido entre microcontroladores (comunicación de bajo nivel en el bus), lo cual genera el desarrollo de múltiples aplicaciones de forma simplificada (Vela Xool, 2015). Además, es capaz de alcanzar velocidades de hasta 1 Mbps.

Tabla 3.3 Características del controlador CAN MCP2515



ι

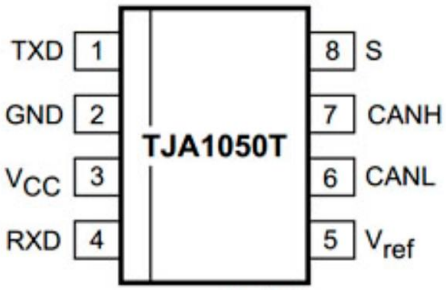
Campo de datos	0 a 8 bytes	Características adicionales
Velocidad de transmisión	La versión CAN 2.0B alcanza 1Mbps	Recepción de mensajes por 2 búfers (Rx) para su aceptación: • 2 máscaras • 6 filtros Transmisión de mensajes por 3 búfers (Tx), por prioridad y cancelación
Direccionamiento del mensaje	Datos estándar (11 bits), extendidos (29 bits) y marcos de datos	
Interfaz SPI	Es de alta velocidad (10 MHz)	
Monitoreo	SOF (inicio de fotograma)	
Modo One-Shot	Garantiza que se intente la transmisión del mensaje	

Fuente: (Microchip, 2021, p. 1), ι (Microchip, 2021, p. 1)

Transceptor TJA 1050

Es un transceptor CAN de alta velocidad que se integra en la placa CAN-BUS Shield V3.0, el cual se encuentra entre el bus de datos físico y el controlador MCP2515 para generar una interfaz eléctrica (niveles de voltaje), que permita una transmisión diferencial entre ambos componentes (Rodríguez Rodríguez et al., 2018). En la tabla 3.4 se adjunta el esquema del transceptor TJA 1050 y sus principales características, en donde se destaca su compatibilidad a la norma ISO 11898, por lo que maneja estándares en cuanto a la comunicación CAN en la capa física y enlace de datos, para transmitir información con una velocidad de hasta 1 Mbps y alta seguridad por influencia mínima de emisión electromagnética en las señales de salida, las cuales se manejan por la línea de comunicación (CAN High y CAN Low).

Tabla 3.4 Características del transceptor TJA 1050



κ

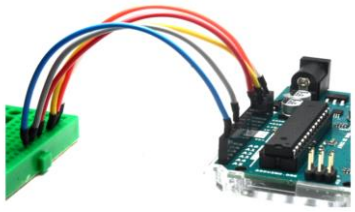
Rango de operación (V)	4,75 V a 5,25 V	Características adicionales
Velocidad	Se relaciona con el controlador MCP 2515, manejando una velocidad hasta 1 Mbps	Se encuentra montado superficialmente a la placa CAN Shield
Búfer	Protección del controlador CAN de los picos de señales en el Bus.	A prueba de cortocircuitos a masa y batería
Pines Bus	CAN High y CAN Low	**Compatibilidad**
Tiempo de espera	Función con de transmisión dominante (TXD)	ISO 11898 (protocolo CAN)
Retardo del bucle	TDX a RXD: 255 ns	**Emisión electromagnética**
Protección	Térmica	Baja

Fuente: (AWERS, n.d.), κ (AWERS, n.d.)

Componentes de conexión

En la conexión física se aprovecharon los materiales presentados en la tabla 3.5, los cuales permiten conectar con puentes de conexión los pines SPI del Arduino Mega 2560 y la placa CAN Bus Shield. Además de formar las líneas de comunicación características del protocolo CAN con resistencias es sus extremos.

Tabla 3.5 Componentes para la conexión física de la red

Denominación	Descripción
Cables Jumper  λ	Los cables Jumper o Dupont tienen conectores en sus extremos, los cuales pueden ser machos (M) y hembras (H). Estos cables tienen un calibre pequeño de aproximadamente 2,54 mm, y son empleados en las conexiones de dispositivos electrónicos por su sencilles.
Cable multifilar eléctrico  μ	Este cable se constituye por varios hilos finos de cobre, los cuales conforman un conductor firme con una capa exterior que aísla la conexión, tiene la capacidad de flexionarse sin romperse, siendo ideal para conexiones de diversas índoles

Resistencias  ν	Son elementos que limitan el flujo de corriente para que los valores nominales en un circuito se mantengan estables. En el caso de una red CAN y prototipado se emplean resistencias de 120 ohmios en el inicio y fin.

Fuente: λ (DynamoElectronics, n.d.), μ (Ingecom, n.d.) , ν (BAUBOR, n.d.)

3.1.1.3. Diagramas del hardware que interviene en la conexión de red

De acuerdo a lo propuesto en el alcance, se usó microcontroladores Arduino Mega 2560 y placas CAN Bus Shield V3 para formar módulos CAN, por lo que conocer los diagramas de dichos componentes permite tener una mejor perspectiva de las conexiones internas y externas de los mismos al montarlos.

ATMEGA 2560

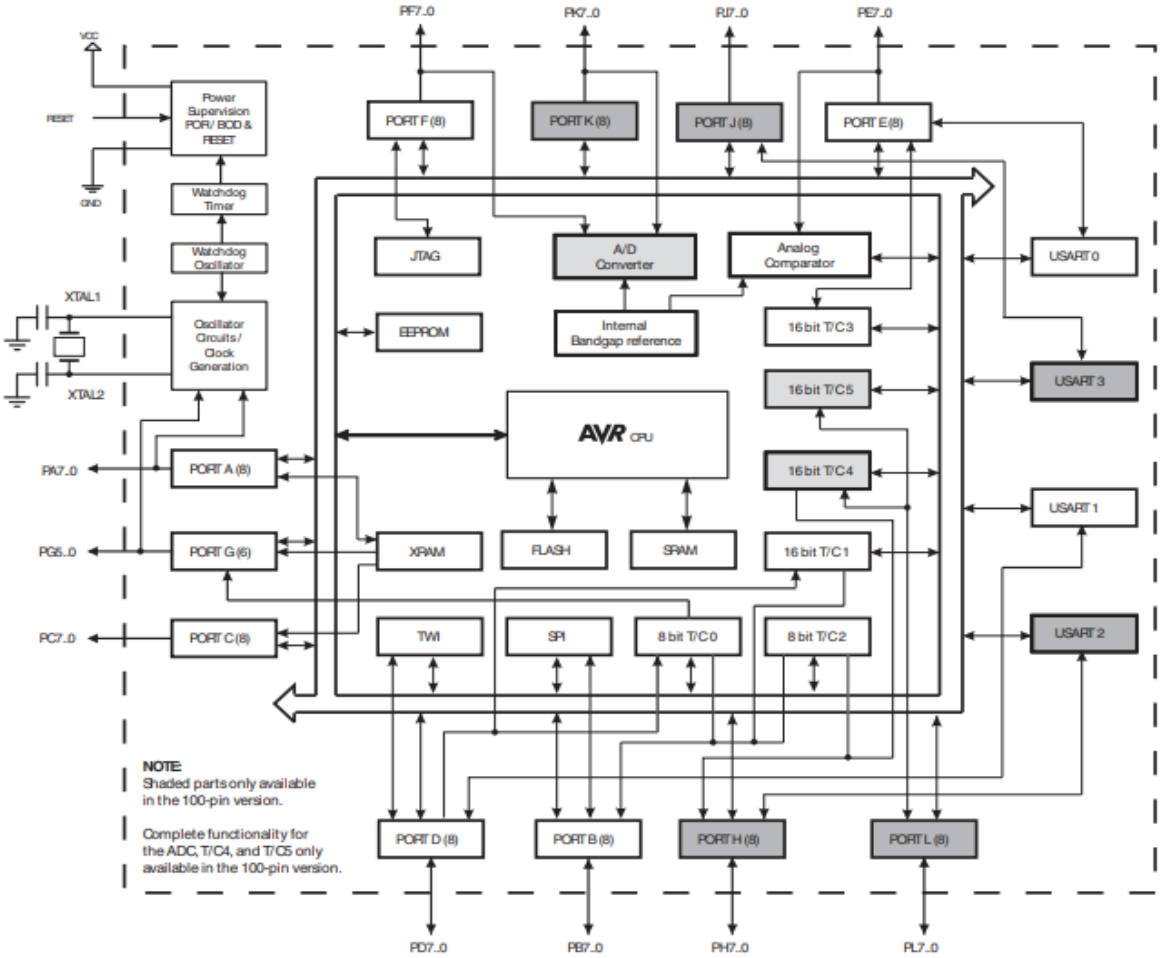


Figura 3.3 Diagrama del microcontrolador Atmega 2560
(Microchip, 2014, p. 5)

En la figura 3.3 se muestra el diagrama del microcontrolador Atmega 2560 de Atmel, el cual se encuentra estructurado por diferentes conjuntos de puertos que se conectan hacia una línea

principal del CPU (unidad central de procesamiento). En el mismo se evidencia que en el B interviene la comunicación SPI utilizando el temporizador/contador en intervalos de 8 bits, necesario para el trabajo, y tiene la siguiente estructura en la placa Arduino Mega 2560: 22 MISO, 21 MOSI, 20 SCK y 19 SS, como se observa en la figura 3.4.

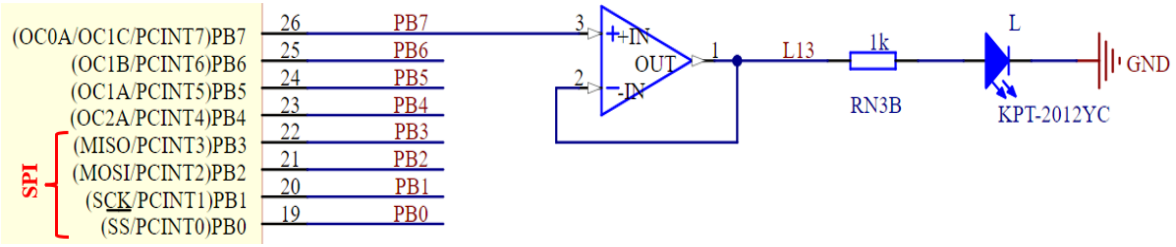


Figura 3.4 Pines SPI del microcontrolador Atmega 2560
(Arduino, n.d.)

MCP 2515

En la figura 3.5 se aprecia el diagrama de conexión del controlador MCP 2515 y en el anexo I (figura AI.1) el detalle de sus pines, estructurado principalmente por la lógica de control y registros usados para modelar el funcionamiento del dispositivo, a la que se conectan: el módulo CAN, interfaz SPI y la generación de tiempos. El módulo CAN incluye el protocolo, filtros, máscaras y búfers de transmisión y recepción de mensajes, además el bloque del Bus SPI se relaciona con el microcontrolador Atmega 2560 para generar una comunicación entre los nodos de la red con la lectura y escritura de registros, y comandos específicos de SPI.

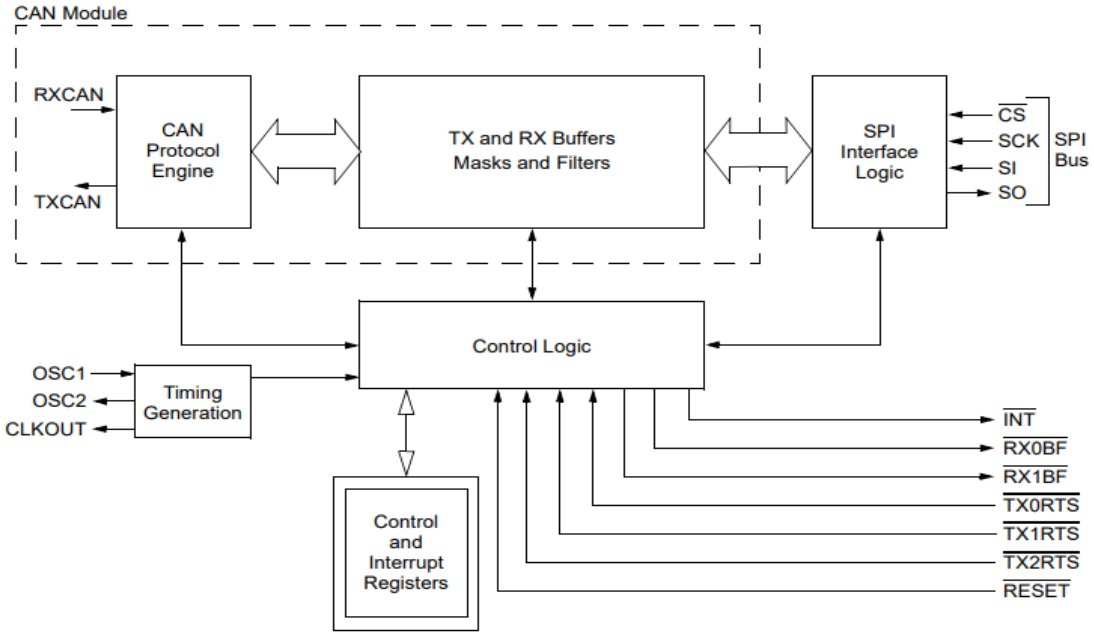


Figura 3.5 Diagrama microchip MCP 2515
(Microchip, 2021, p. 3)

TJA 1050

En la figura 3.6 se evidencia el diagrama del transceptor TJA 1050 y en el anexo I (figura AI.2) el detalle de sus pines, el cual se estructura por un receptor del Bus diferencial de alto y bajo nivel CAN, para generar y entregar niveles lógicos a la entrada RXD del controlador CAN. Cuando este requiere transmitir información, envía tramas de datos a la entrada lógica TXD, donde interviene un temporizador de salida en tiempo dominante (permanente), pero si excede el valor interno, este se deshabilita llevando el bus en estado recesivo hasta restablecerse con un flanco positivo en TXD. Luego un controlador de salida se encarga de enviar dicha información hacia el bus por arbitraje.

- En los pines CAN High y CAN Low se reciben y emiten estados dominantes y recesivos

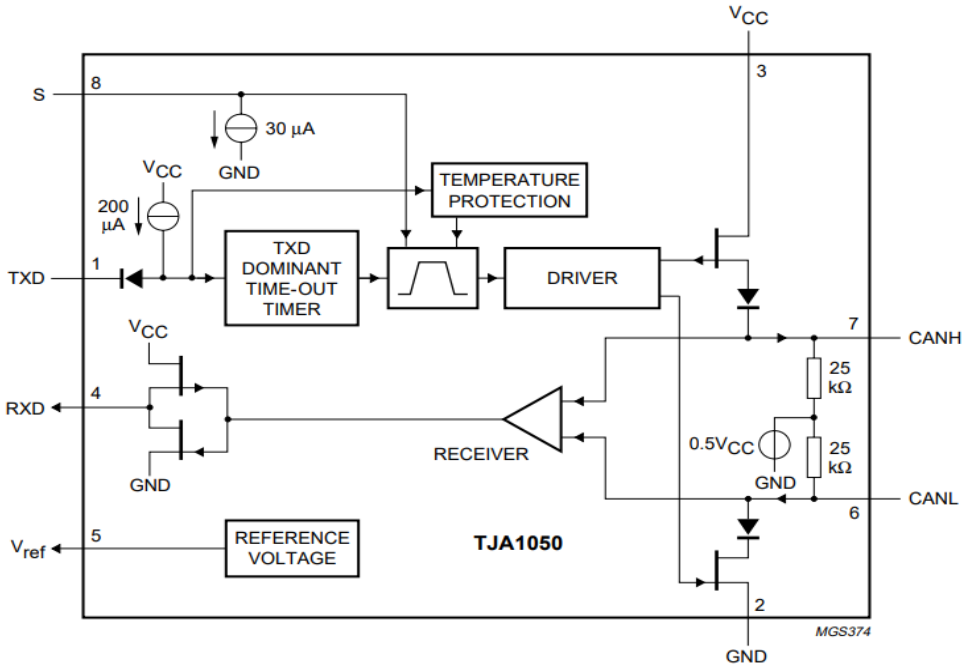


Figura 3.6 Diagrama transceptor TJA 1050
(NXP, 2022, p. 2)

3.1.1.4. Diseño del módulo CAN en base a microcontroladores Arduino

En el desarrollo de una red CAN interviene la conexión de dos o más módulos a una línea de comunicación, los cuales internamente comparten características similares, siendo el microprocesador, controlador y transceptor los componentes indispensables para establecer una comunicación bajo el protocolo CAN, y de igual manera su estructura interna y externa (figura 3.7) para establecer una comunicación.

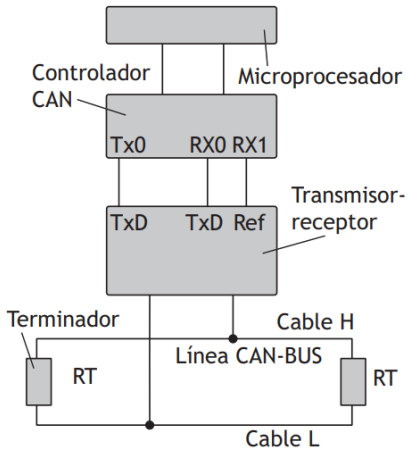


Figura 3.7 Esquema general de un módulo CAN
(Sánchez, 2012, p. 292)

En el caso del presente trabajo, los componentes que permiten construir un módulo CAN son: la placa Arduino Mega 2560 con su microcontrolador Atmega 2560, y la placa CAN-BUS Shield con el controlador MCP 2515 y el transceptor de alta velocidad TJA 1050.

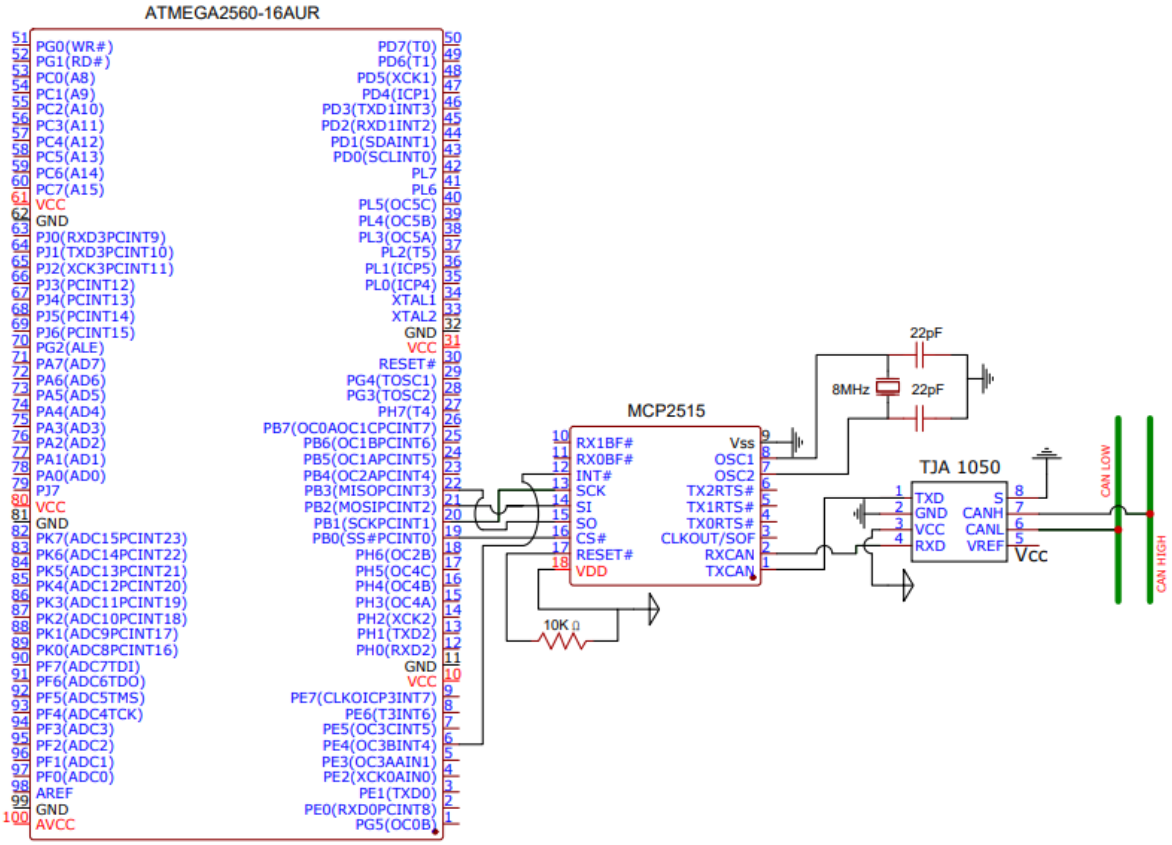


Figura 3.8 Diseño del nodo o módulo CAN simple
Fuente: Autor

Para realizar el esquema de la figura 3.8 se utilizó la herramienta online EasyEDA, en la cual se evidencia que el controlador MCP2515 se encarga de la transmisión y recepción de mensajes CAN, ya que hace uso de la interfaz SPI para conectarse al microcontrolador

Atmega 2560, este mismo se conecta al transceptor TJA 1050 que es el encargado de convertir señales digitales en unas adecuadas para ser trasmitidas por el bus de datos y viceversa para que el controlador comprenda dichas señales (Feliciano Fuentes, 2019). Además, los pines UART pueden ser utilizados como entradas (Tx) y salidas (Rx) de propósitos generales para recuperar la comunicación SPI.

3.1.1.5. Selección del diseño base de la red CAN

Debido a las características de los componentes del módulo CAN diseñado en el apartado anterior, este es compatible con la norma ISO 11898, por lo que permite establecer una comunicación bajo el principio multimaestro o maestro-esclavo, al emplear una topología Bus para transmitir información de forma serial y bidireccional.

La estructura de la red fue conformada por 4 módulos CAN en paralelo al Bus de datos, con dos resistencias de 120 Ω en los extremos de la línea de comunicación o módulos terminales como se muestra en la figura 3.9, esta configuración es característica de las redes CAN, ya que permite que todas las unidades de control puedan acceder a los mensajes disponibles en el Bus de datos, el cual se compone por las líneas CAN High y CAN Low normalmente trenzadas, permitiendo la transmisión segura al reducir las interferencias electromagnéticas ocasionadas por agentes externos, además, las resistencias de terminación evitan el rebote de señal al manejar un valor aproximado de 60 Ω. Por lo cual, en el presente trabajo se utilizó dicha configuración para asegurar una comunicación segura, eficiente y con capacidad de soportar grandes cargas en el manejo de datos.

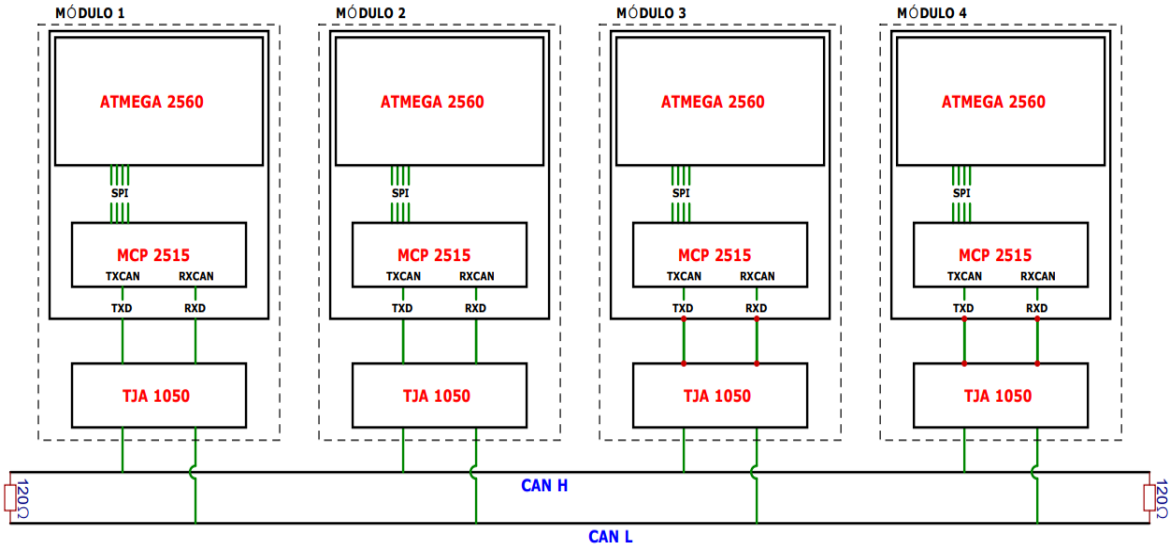


Figura 3.9 Diagrama base de la red multiplexada CAN

Fuente: Autor

3.1.2. DESARROLLO DE LA RED, ALGORITMO PARA ESTABLECER UNA COMUNICACIÓN ENTRE MÓDULOS CAN POR MEDIO DE PROGRAMACIÓN

En la figura 3.10 se detalla el flujograma que se utilizó para establecer una comunicación paulatina entre los 4 módulos CAN, especificando los materiales y equipos necesarios para desarrollar la programación y analizar de manera bidireccional el comportamiento de la red.

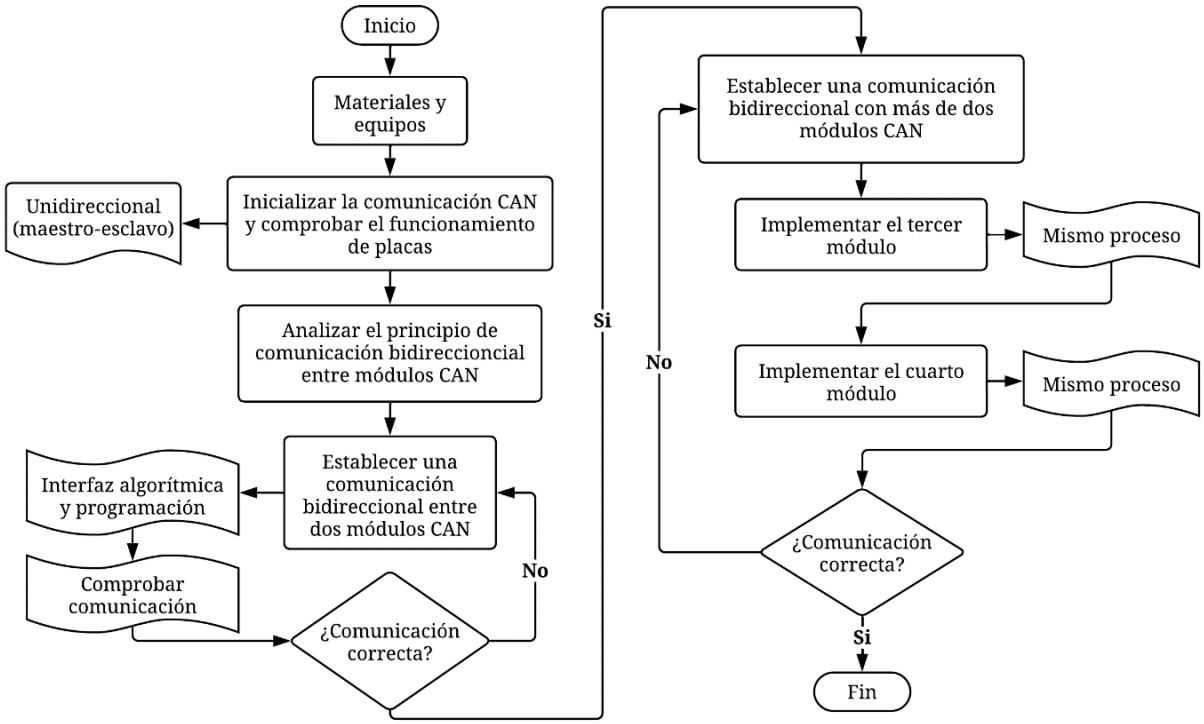


Figura 3.10 Flujograma de red para establecer comunicación bidireccional
Fuente: Autor

En lo que se refiere a comunicación CAN, se estableció un algoritmo de pasos lógicos para una red unidireccional (maestro-esclavo), y luego se llevó a cabo un análisis de la comunicación bidireccional entre módulos CAN, pues fue el principio en el cual se basó el desarrollo de la red. Para establecer dicha interacción se siguió una interfaz algorítmica para su programación paulatina, de manera que inicialmente se establezca una comunicación en tiempo real entre dos módulos CAN y luego se vayan añadiendo los faltantes, simulando la gestión de comunicación multibroadcast. En donde, según García Osés (2015) toma relevancia el tema del arbitraje de los identificadores para controlar el acceso al Bus de datos, ya que en estos se establece el grado de prioridad, evitando colisiones cuando dos o más módulos intentan transmitir información al mismo tiempo, y también la recepción de mensajes empleando máscaras y filtros.

3.1.2.1. Materiales y equipos

En la tabla 3.6 se detalla el software y el hardware para el desarrollo de la red

Tabla 3.6 Materiales y equipos para el desarrollo de la comunicación en la red

Denominación	Descripción
Software Arduino IDE	Permite desarrollar código de programación para dar vida a sistemas que integran un microcontrolador o microprocesador, por medio de la ejecución de instrucciones lógicas establecidas en dicho código.
Cable USB tipo A/B	Establece una comunicación bidireccional entre el microcontrolador y el computador, se utiliza para cargar el código de programación compilado y monitorear el comportamiento de los datos.
CAN-BUS Analyzer (Microchip)	Es una herramienta de análisis de redes CAN basadas en CAN 2.0 B, ya que es compatible con la norma ISO11898-2 (velocidad CAN de hasta 1 Mbit/s). Siendo muy versátil para el rastreo, depuración y trasmisión de marcos de datos en redes CAN (Microchip, 2022).
G-SCOPE-2 (Super Tools)	Es un dispositivo que permite setear y diagnosticar el comportamiento de sistemas, redes y componentes automotrices, ya que cuenta con dos interfaces de configuración para su funcionamiento: 100% electrónico y automotriz (Globaltech, 2019).
Hantek 6074BC	Es un osciloscopio digital de 4 canales que emplea un software para el diagnóstico y análisis de señales eléctricas desde un computador, cuenta con una frecuencia de operación de 70 MHz.

3.1.2.2. Inicialización CAN (comunicación unidireccional)

En relación con el flujograma de la figura 3.10, inicialmente se estableció una comunicación unidireccional entre dos módulos CAN (figura 3.11), verificando así el funcionamiento del conjunto de placas integradas y el manejo de parámetros de las librerías incluidas en los códigos de programación.

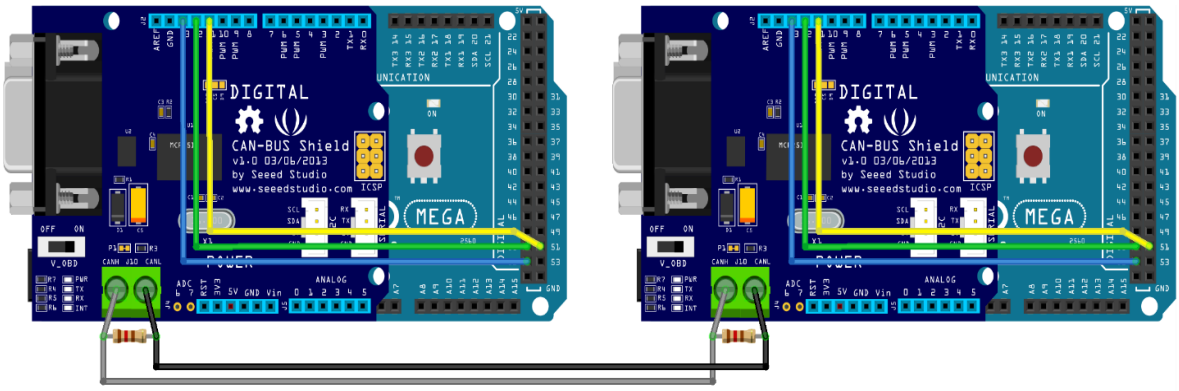


Figura 3.11 Conexión de dos módulos CAN

Fuente: Autor

Procedimiento de inicialización CAN (maestro - esclavo)

Para establecer una comunicación CAN bajo el principio maestro – esclavo se llevó a cabo el procedimiento del diagrama 3.12, de manera que un módulo se comporte únicamente como emisor de mensajes y el otro como receptor. Para lo cual, utilizando el entorno de desarrollo integrado de Arduino, se desarrolló los códigos de programación, los cuales comienzan incluyendo la librería SPI.h h para permitir una comunicación con dispositivos periféricos, y también la librería mcp2515_can.h de Seeed Studio al ser la biblioteca más actualizada para comunicación CAN, teniendo relación únicamente con el controlador y no importa que transceptor implemente. Luego, se realizó la declaración de variables definiendo el pin 10 como puerto de selección CS, comunicación CAN-BUS en modo normal, velocidad de 1000 kbps (high speed), la cual se reduce a la mitad por la operación a 8 MHz del Shield CAN Bus. Además, se añadió una impresión de confirmación CAN OK en el caso de una inicialización correcta y también una alerta de error para reintentar la conexión.

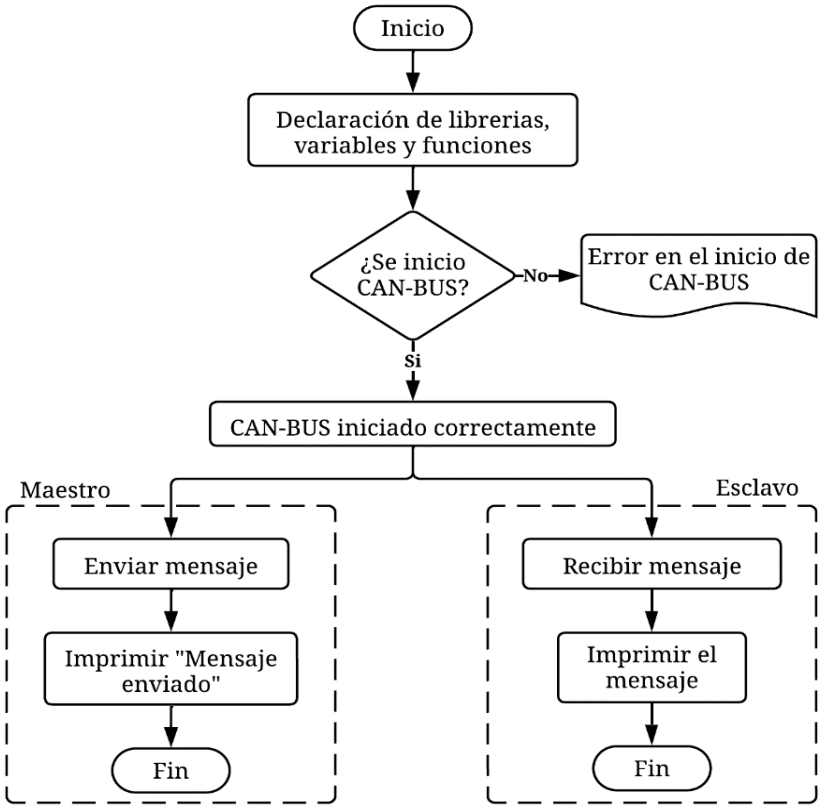


Figura 3.12 Diagrama del envío y recepción de mensajes unidireccional
Fuente: Autor

Tras una correcta inicialización, se configuró un módulo como maestro emisor de mensajes de 8 bytes, o sea, un campo de datos con un máximo de 64 bits al emplear la función CAN.sendMsgBuf (Id, Ext, Len, Buf), siendo:

- **Id. –** Es el identificador de donde provienen el mensaje
- **Ext. –** Representa el formato de la trama de datos, estándar (0) y extendido (1)
- **Len. –** Longitud del mensaje, hasta 8 bytes
- **Buf. –** Contenido del mensaje

Por otra parte, el módulo esclavo funciona como receptor de mensajes, por lo cual se estableció una lógica de programación diferente con la integración de las funciones CAN.readMsgBuf (&Len, Buf) y CAN.getCanId, las cuales permiten obtener los marcos de datos enviados por el emisor, teniendo en cuenta que es posible configurar máscaras y filtros para establecer una comunicación más segura al solo aceptar y procesar mensajes de interés. Sin embargo, en este punto eso no se implementó, por ello los códigos de programación (anexo II) son similares a los ejemplos planteados en la librería mcp2515_can.h.

Verificación de comunicación entre placas

En la figura 3.13 se evidencia que los dos módulos CAN inician exitosamente la comunicación CAN, por lo que el módulo que se configuró como emisor transmite la trama de datos {0, 0, 100, 255, 256, 514, 1, 1} con el identificador 0x132001. Dicho campo de datos se encuentra inicialmente en codificación decimal, y al ser leído por el módulo esclavo se maneja el sistema hexadecimal, en donde se evidencia que el 4 byte (255) es el máximo valor que puede ser enviado e interpretado con 8 bits en el sistema binario. Por lo tanto, en la lectura del byte 5 (256) se resetea la numeración asignando un valor de 0 y de igual manera al duplicarlo como el byte 6 (512+2), que adoptó un valor de 2. Además, se denota que el identificador se encuentra en formato extendido y no varía su sistema numérico hexadecimal.

Figura 3.13 Lectura de la comunicación unidireccional en el Bus monitor

Fuente: Autor

De igual manera se verificó la comunicación en el Bus de datos usando el osciloscopio digital G-SCOPE-2, ya que tiene una funcionalidad enfocada en el diagnóstico de redes de comunicación automotrices, y se lo configuró de la siguiente manera:

- CH1 (CAN HIGH) y CH2 (CAN LOW)
- Velocidad de transmisión 500 KBPS y filtro de paso bajo en 1 MHz (10X)
- En los dos canales, una amplitud de 500 mV y un tiempo de 10 µs por división
- Lectura de firmas eléctricas de CH1 para CAN HIGH a una velocidad de 500 KBPS

En la lectura digital de los dos canales (figura 3.14) se exhibe las firmas eléctricas en sistema hexadecimal, en donde se obtuvo los mismos datos establecidos en la configuración del módulo emisor al rescatar el valor del identificador, número de bytes enviados, contenido de los 8 bytes y al final la verificación de trasmisión CRC. Además, se evidencia los valores de tensión máximos y mínimos de cada canal, siendo estos correctos, ya que según el protocolo CAN los niveles de tensión entre bits dominantes y recesivos para la línea CAN HIGH es de 2,5 V a 3,5 V y CAN LOW 1,5 V a 2,5 V.

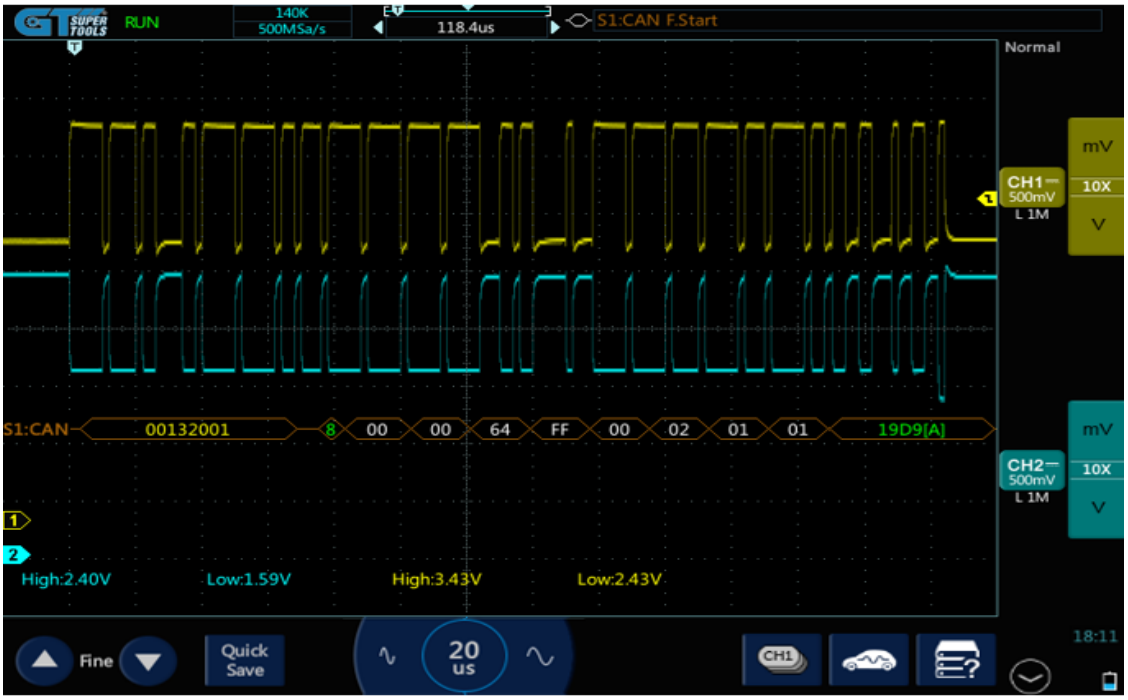


Figura 3.14 Señal de la línea de comunicación unidireccional

Fuente: Autor

3.1.2.3. Análisis del principio de comunicación bidireccional entre módulos CAN

La comunicación bidireccional, como su nombre lo indica, permite intercambiar información en dos direcciones, enviar y recibir datos de forma serial entre dos o más módulos mediante protocolos de comunicación según las necesidades del sistema de red. En los vehículos, el protocolo más utilizado es el CAN (Controller Area Network), ya que

proporciona una comunicación con elevada velocidad y bajo consumo de energía, en donde los datos de entrada y salida son manejados de forma serial y por su configuración multimaestro genera la posibilidad de establecer interacciones entre los datos de múltiples nodos, desarrollando sistemas embebidos. Por ejemplo, el sistema ECS (control electrónico de estabilidad) y el sistema ABS (sistema de frenos antibloqueo) operan de manera conjunta para garantizar a los usuarios una conducción segura y estable, cuando el ECS detecta que el vehículo presenta inestabilidad envía una señal al ABS para disminuir la fuerza de frenado y así recuperar tracción, del mismo modo cuando el ABS detecta que una rueda está por bloquearse envía una señal al ESC para que ajuste la potencia del motor o aplique un frenado en las ruedas restantes, evidenciando una comunicación bidireccional. Es importante destacar que al intentar comunicar dos o más módulos CAN, estos pueden intercambiar información dependiendo de las necesidades de la red para su configuración.

- **Comunicación por eventos:** Al presentarse una situación específica, los módulos envían mensajes en respuesta a los mismos, de manera que los demás nodos se enteren de dicha situación y realicen acciones en función de eso si es necesario.
- **Comunicación multidifusión:** Esta configuración permite que cualquier módulo de la red pueda transmitir información a los demás módulos o de manera selectiva, de manera que estos empleen dicha información para tomar decisiones dependiendo si es te utilidad o no, por tal motivo es recomendable implementar máscaras y filtros.
- **Comunicación maestro-esclavo:** Un módulo maestro interactúa con varios esclavos al requerir información específica, dependiendo de la solicitud estos transmiten mensajes de respuesta, por lo cual es empleada para los subsistemas CAN.

3.1.2.4. Comunicación bidireccional entre dos módulos CAN

De acuerdo a lo visto en el análisis de comunicación bidireccional, en este apartado la configuración de los módulos es bajo el concepto “broadcast”, es decir, cualquiera puede adoptar la funcionalidad de emisor o receptor. Por lo tanto, en los códigos de programación se incluyó las funciones respectivas, las cuales intervienen alternadamente con la configuración de la función attachInterrupt, ya que al detectar mensajes disponibles en el Bus de datos se genera una rutina de interrupción, de manera que el módulo pueda leer las tramas de datos y caso contrario adopta la funcionalidad emisora para transmitir un mensaje.

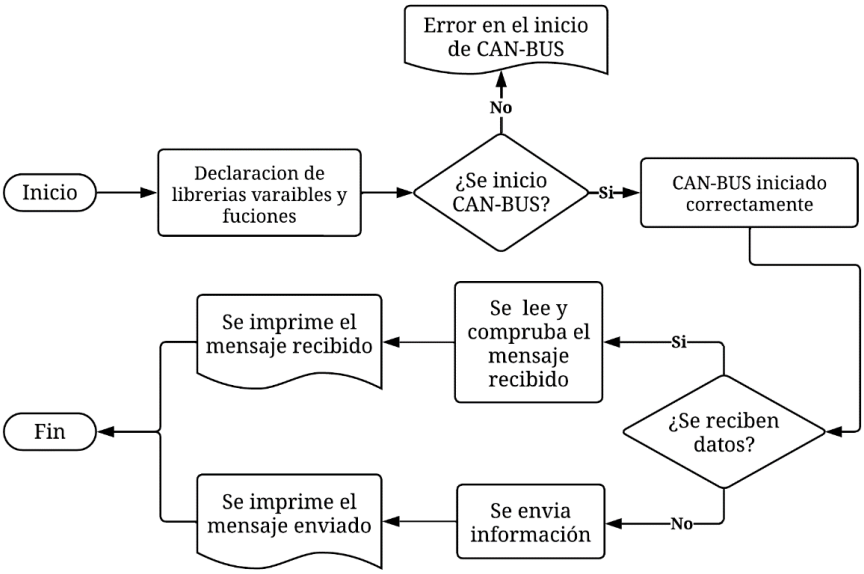


Figura 3.15 Interfaz del envío y recepción de mensajes de un nodo CAN
(Feliciano Fuentes, 2019, p. 58)

Los parámetros que se estableció en dicha función inician por la referencia de señal de disparo con un valor de 0, seguido de la función (MCP2515_ISR) que se llama al presentarse una interrupción y el modo de activación con la denominación FALLING que actúa al pasar la señal de un estado alto a bajo. Al existir una interrupción se verifica la disponibilidad de mensajes mediante la función CAN.checkReceive, pero en caso de no detectar ningún mensaje, el módulo enviará su propio mensaje al Bus de datos, como se evidencia en la interfaz de la figura 3.15, con la facultad de enviar y recibir mensajes.

Interfaz algorítmica para la programación

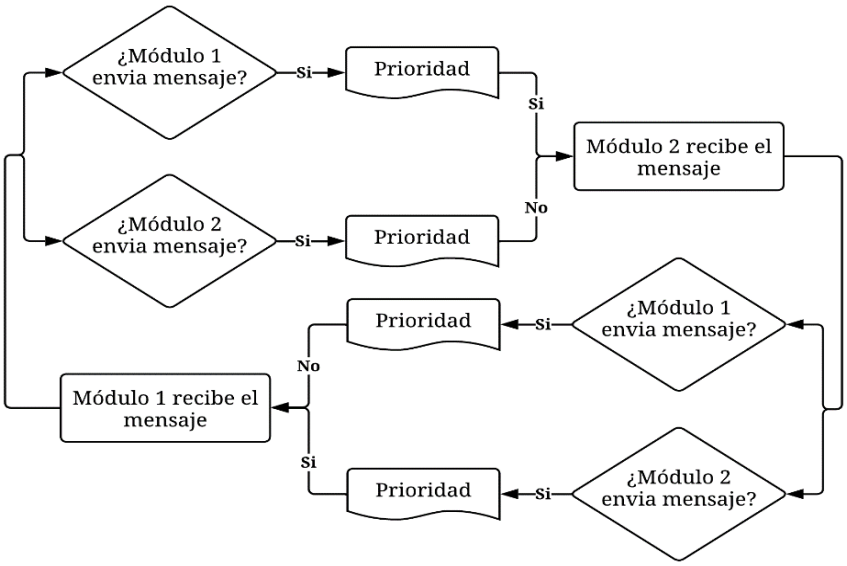


Figura 3.16 Diagrama de programación para la comunicación entre 2 módulos CAN
Fuente: Autor

El diagrama presentado en la figura 3.16 expone el comportamiento de una comunicación bidireccional entre 2 módulos CAN, esto basándose en la interfaz de pasos lógicos de la figura 3.15, generando una interacción secuencial en cuanto al envio y recepción de datos por tener una parametrización base similar.

Comprobación de comunicación

Haciendo uso del software Arduino IDE y la herramienta CAN Bus Analyzer, se comprobó el comportamiento de la comunicación entre los dos módulos, ya que dicha herramienta permite obtener más información. Para su utilizarlo, fue necesario instalar el software de dicha herramienta desde el sitio oficial de microchip y realizar las conexiones físicas, por lo cual se consultó la guía del CAN Bus Analyzer, que según Microchip (2022) tiene una configuración de los pines CAN High, CAN Low y masa de su conector DB9 como se muestra en la figura 3.17, en donde también se añade la distribución de la placa CAN Bus Shield, denotando que son diferentes. Por lo tanto, se llevó a cabo una adaptación de los pines del Can Bus Analyzer hacia la línea de comunicación mediante puentes de conexión.

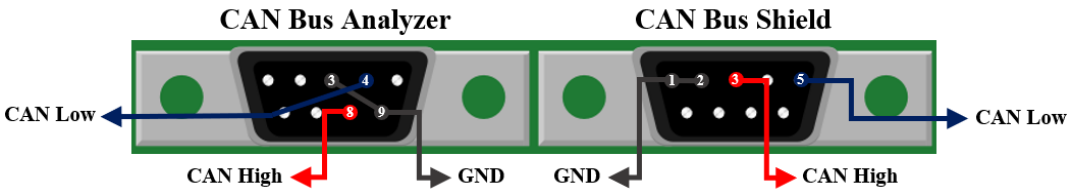


Figura 3.17 Configuración de los pines para la lectura del Bus de datos
Fuente: Autor

Empleando solamente la función attachInterrupt es posible generar una comunicación serial entre dos módulos, sin embargo, utilizando la herramienta de análisis (figura 3.18), se comprobó que los periodos de transmisión son inestables, lo cual genera inestabilidad en el counter de transmisión, y en consecuencia la frecuencia de acceso al Bus de datos. Aunque la configuración de velocidad y baudios en la programación de los módulos reduce dicho comportamiento, esto no se ajusta a los requerimientos de estabilidad.

Fixed Trace

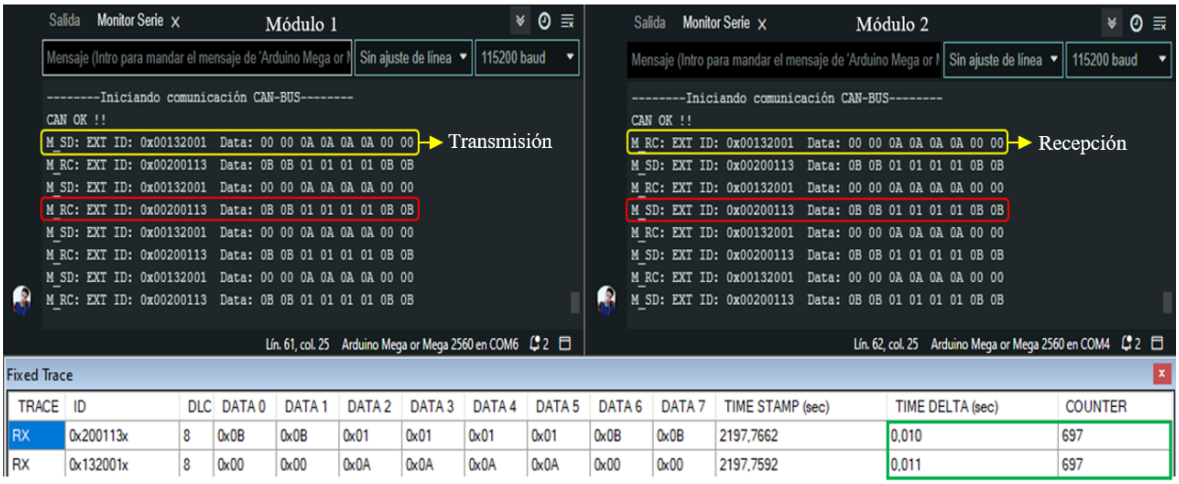
TRACE	ID	DLC	DATA 0	DATA 1	DATA 2	DATA 3	DATA 4	DATA 5	DATA 6	DATA 7	TIME STAMP (sec)	TIME DELTA (sec)	COUNTER
RX	0x132001x	8	0x00	0x00	0x0A	0x0A	0x0A	0x0A	0x00	0x00	112,7342	0,010	1530
RX	0x200113x	8	0x0B	0x0B	0x01	0x01	0x01	0x01	0x0B	0x0B	112,7252	0,015	1319

Figura 3.18 Comunicación entre 2 módulos empleando una rutina de interrupción
Fuente: Autor

Por lo tanto, para controlar los periodos de transmisión se buscó alternativas de interrupción que permitan establecer una comunicación multidifusión en tiempo real, las opciones planteadas en este punto fueron:

- Delay (): Detiene la ejecución del programa en el tiempo establecido
- Millis (): No detiene la ejecución del programa y emplea temporizadores

Tras realizar pruebas con las funciones mencionadas, se evidenció que en ambos casos los periodos de transmisión y el counter se estabiliza en comparación a solamente emplear la función attachInterrupt. Por lo cual, en este punto se escogió a millis() para establecer una comunicación en tiempo real, teniendo el comportamiento mostrado en el monitoreo del monitor de Arduino y analizador (figura 3.19), el cual se relaciona con lo planteado en la interfaz de la figura 3.16, ya que al iniciar correctamente la comunicación CAN y al encontrarse el Bus de datos sin mensajes ambos módulos intentaran transmitir.



TRACE	ID	DLC	DATA 0	DATA 1	DATA 2	DATA 3	DATA 4	DATA 5	DATA 6	DATA 7	TIME STAMP (sec)	TIME DELTA (sec)	COUNTER
RX	0x200113x	8	0x0B	0x0B	0x01	0x01	0x01	0x01	0x0B	0x0B	2197,7662	0,010	697
RX	0x132001x	8	0x00	0x00	0x0A	0x0A	0x0A	0x0A	0x00	0x00	2197,7592	0,011	697

Figura 3.19 Lectura de mensajes enviados y recibidos en los dos módulos CAN

Fuente: Autor

En el acceso al Bus de datos interviene el proceso de arbitraje con los identificadores, siendo estos los que determinan la prioridad dependiendo de su valor binario, por lo cual al conectar más de dos módulos se profundizará la importancia de los identificadores. En este caso, como se relacionó solamente dos módulos, la red funciona de manera secuencial, empezando con el módulo con el identificador de nivel más bajo (0x132001), y denotando que los periodos de transmisión tienen un grado de estabilidad, pero no son constantes, ya que varían (+1 o -1) del periodo establecido y de igual manera la frecuencia, lo cual puede afectar a la sincronización. No obstante, según lo propuesto por Tenesaca (2013) también puede ser producto de las características físicas de la línea de comunicación, agentes externos con

interferencias electromagnéticas, retardos en los controladores y propagación, debido a esto, los controladores implementan un proceso de resincronización por muestreo al presentarse desajustes que afectan el mecanismo de arbitraje. En el anexo III, se adjunta el código empleado, el cual es similar a excepción del identificador y el campo de datos.

3.1.2.5. Comunicación bidireccional con más de dos módulos CAN

Al igual que en el apartado anterior de comunicación entre dos módulos CAN, se utilizó el mismo principio de programación en cuanto a la inicialización, envió y recepción de mensajes, pero tomando mayor relevancia en los identificadores, ya que la prioridad se establece en el nivel binario de cada uno. Cuando dos o más módulos intentan transmitir información al mismo tiempo, compiten por acceder al Bus de datos, y es ahí donde el identificador con el nivel binario más bajo toma relevancia para acceder primero, de manera que al implementar más de dos nodos con un periodo de transmisión similar este proceso debería comportarse como una especie de turnos, aunque como se mencionó anteriormente puede existir factores que afecten dicho comportamiento.

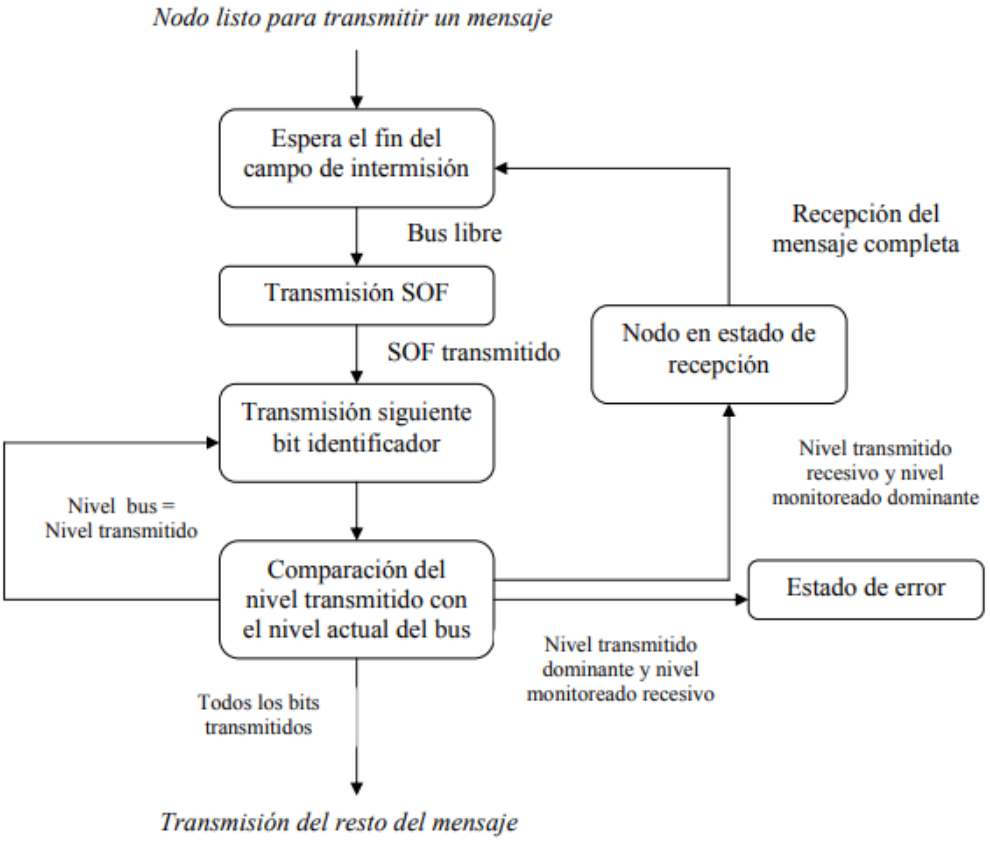


Figura 3.20 Diagrama de la fase de arbitraje
(Vergara Vargas, 2005, p. 40)

Como se muestra en la figura 3.20, los módulos conectados a la red intentan enviar información cuando el bus de datos se encuentra disponible, al finalizar el campo de interrupción cada uno envía un bit dominante (SOF) seguido del identificador. De acuerdo con la figura 3.21, los nodos trasmiten su identificador en sistema binario desde el bit más significativo (izquierda a derecha), en donde la transmisión de bits dominantes (0) determina la prioridad del nodo. Durante el periodo de arbitraje, los módulos verifican el estado digital del Bus de datos y lo comparan con el bit que se está trasmitiendo, de manera que al tener uno similar (dominante-dominante o recesivo-recesivo) continúa transmitiendo el siguiente bit, pero si el nodo transmite un bit recesivo (1) y el nivel del bus se encuentra en dominante pierde el arbitraje, por lo que debe detener el envío y adoptar la funcionalidad de receptor hasta que la línea este libre para intentarlo nuevamente.

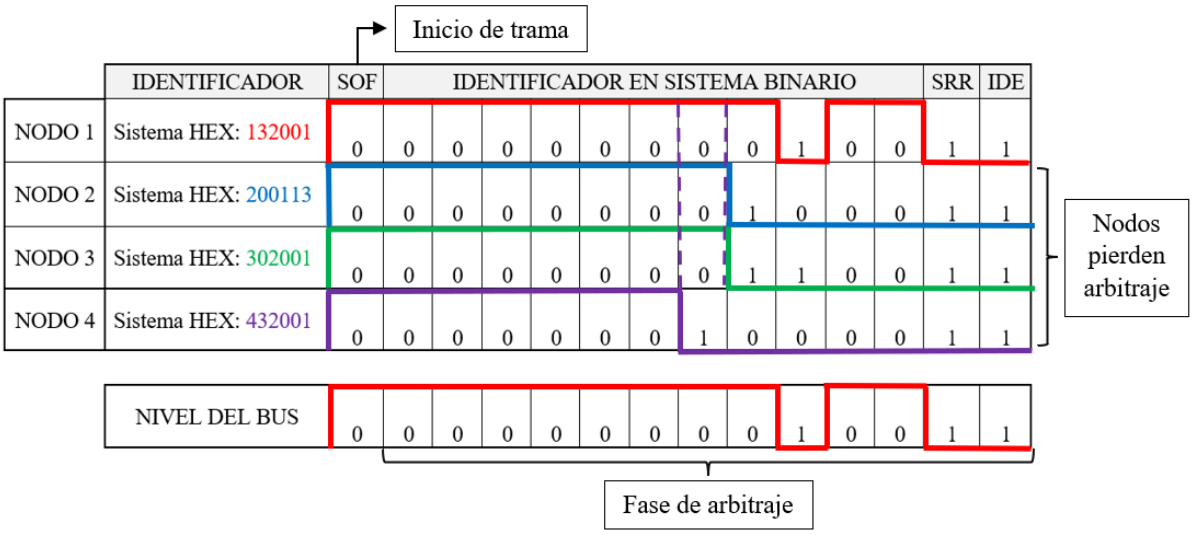


Figura 3.21 Proceso de arbitraje en la línea CAN HIGH

Fuente: Autor

En el proceso de arbitraje de los 4 módulos que se incorporó en el proyecto (figura 3.21), se evidencia que cada uno cuenta con un identificador en formato extendido (29 bits) y que de acuerdo con CAN 2.0B se emplea 32 bits en este campo, los 11 primeros representan la primera parte del identificador que permite determinar la prioridad del mensaje, luego se transmite un bit SRR en estado recesivo (1) para datos regulares y un bit IDE recesivo, ya que se utiliza una identificación extendida. En el caso de que la primera parte del identificador sea igual en dos nodos se emplea la segunda parte de 18 bits para determinar la prioridad, el acceso al Bus de datos permite definir el nivel del bit RTR de cada uno, siendo dominante para el módulo que gana arbitraje por lo que continuará transmitiendo los demás bits del marco de datos, y los demás un bit recesivo requiriendo información.

Interfaz algorítmica para la programación

Tomando en referencia el diagrama de la figura 3.21, la implementación de un tercer y cuarto módulo genera mayor carga en la gestión del Bus de datos, por lo cual, es recomendable configurar el tipo de comunicación que manejaran los módulos. Sin embargo, en este punto no se estableció dicho parámetro, ya que se buscaba simular una comunicación constante, como un comportamiento de máxima exigencia en la red basándose en la interfaz algorítmica de la figura 3.22, permitiendo conocer de mejor manera como interactúa la emisión y captura de datos al añadir más de dos módulos de acuerdo con su grado de prioridad, teniendo en cuenta la configuración de programación es similar a la del apartado anterior.

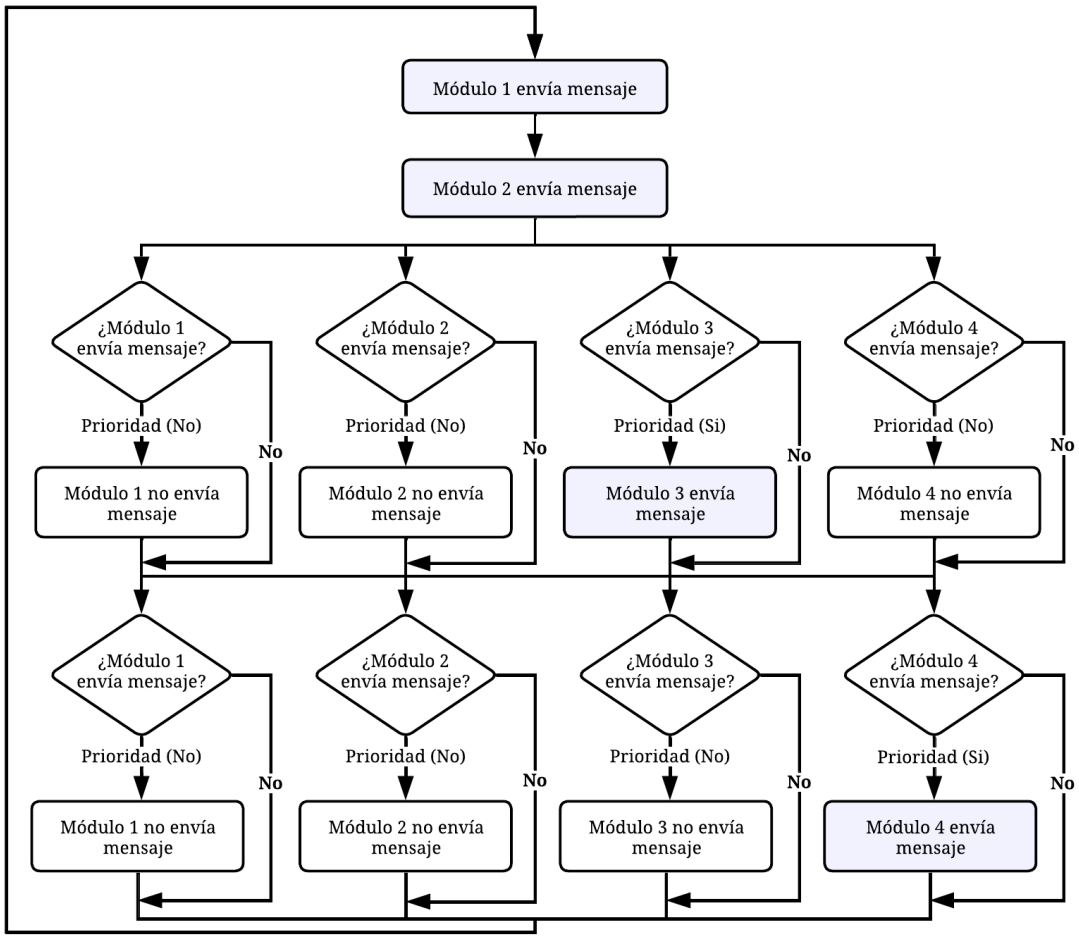


Figura 3.22 Diagrama de programación para la comunicación entre 4 módulos CAN

Fuente: Autor

Comprobación de comunicación

De acuerdo a lo mencionado anteriormente, se implementó un tercer módulo en la red física y lógica, asignando el identificador propuesto en la figura 3.21. La lectura de la línea de comunicación (figura 3.23) se la realizó tras la alimentación simultánea de los 3 módulos mencionados con un adaptador AC/DC, en donde se constató que los periodos de transmisión establecidos (20 ms) varían, lo cual provoco problemas de sincronización y

frecuencia variable, este comportamiento toma mayor relevancia, ya que al conectar solo dos módulos esto no era muy notorio.

Fixed Trace

TRACE	ID	DLC	DATA 0	DATA 1	DATA 2	DATA 3	DATA 4	DATA 5	DATA 6	DATA 7	TIME STAMP (sec)	TIME DELTA (sec)	COUNTER
RX	0x132001x	8	0x00	0x00	0x0A	0x0A	0x0A	0x0A	0x00	0x00	1638,2912	0,022	429
RX	0x200113x	8	0x0B	0x0B	0x01	0x01	0x01	0x01	0x0B	0x0B	1638,2833	0,021	428
RX	0x302001x	8	0x0D	0x0D	0x03	0x03	0x03	0x03	0x0D	0x0D	1638,2862	0,021	428

Figura 3.23 Lectura de la comunicación entre 3 módulos empleando el CAN Bus Analyzer

Fuente: autor

Las variaciones en relación con el tiempo de funcionamiento denotan un comportamiento de resincronización cuando se realiza la inserción de un nodo de mayor prioridad, el orden de lectura del Bus de datos toma poca relevancia, ya que el mecanismo de arbitraje solamente actúa cuando dos o más módulos intentan enviar datos simultáneamente.

En este punto se buscó una nueva alternativa para controlar los periodos de transmisión en cada módulo según los requerimientos de la red, siendo un Timer la mejor alternativa, puesto que permite la ejecución de un código en un intervalo específico sin bloquear el funcionamiento del programa principal. Por lo cual, se adaptó esta opción a la lógica de programación ya estructurada, con las bibliotecas que permiten establecer una comunicación CAN, en consecuencia, se hicieron múltiples pruebas para comprobar la estabilidad de la red al implementar los 4 módulos propuestos en el alcance como parte del sistema general.

Fixed Trace

TRACE	ID	DLC	DATA 0	DATA 1	DATA 2	DATA 3	DATA 4	DATA 5	DATA 6	DATA 7	TIME STAMP (sec)	TIME DELTA (sec)	COUNTER
RX	0x132001x	8	0x00	0x00	0x0A	0x0A	0x0A	0x0A	0x00	0x00	255,2232	0,010	1342
RX	0x200113x	8	0x0B	0x0B	0x01	0x01	0x01	0x01	0x0B	0x0B	255,2242	0,020	672
RX	0x302001x	8	0x0D	0x0D	0x03	0x03	0x03	0x03	0x0D	0x0D	255,2293	0,020	672
RX	0x432001x	8	0x15	0x0F	0x01	0x00	0x00	0x01	0x0F	0x15	255,1602	0,100	134

Figura 3.24 Lectura del Bus de datos empleando un timer para la comunicación

Fuente: Autor

El comportamiento que se obtuvo con la alternativa mencionada se presenta en la figura 3.24, en donde se evidencia que los periodos de transmisión son constantes, al igual que el número de mensajes transmitidos. Sin embargo, no se descarta que puede ocurrir errores de sincronización o colisiones en algún momento, por algún tipo de desajuste o interferencia, como lo manifestado en la comunicación entre dos módulos.

3.1.3. DISEÑO E IMPLEMENTACIÓN DEL SISTEMA EMBEBIDO A CONTROLAR EN LA RED

En el diseño del sistema de control, se empleó el esquema de la figura 3.25, el cual se enfoca en cumplir el alcance propuesto con sus entradas y salidas de datos, por lo cual, este inicia

por revisar los factores a tener en cuenta para establecer una correcta configuración física y lógica del sistema, de manera que este funcione eficientemente con el manejo de datos.

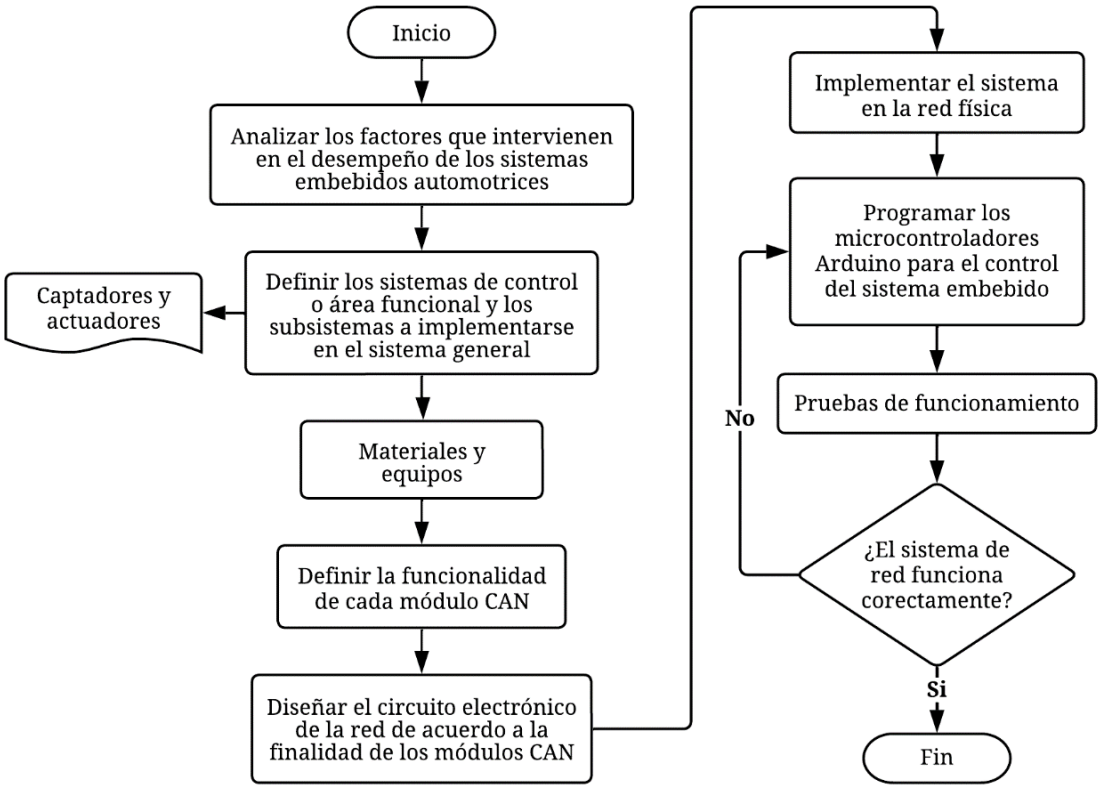


Figura 3.25 Esquema metodológico para el diseño del sistema embebido
Fuente: Autor

3.1.3.1. Factores que intervienen en el desempeño de los sistemas embebidos automotrices

De manera general, los factores que intervienen en el buen funcionamiento de los sistemas embebidos y su capacidad de respuesta son el software y hardware que lo componen. En el software, el procesador y la memoria determinan la rapidez en la que se realizan cálculos y operaciones lógicas, por lo cual el desempeño del sistema y la anulación de errores dependen de los mismos, pero también interviene la integridad de las líneas de comunicación. Por esta razón, el hardware se relaciona directamente con el desempeño del sistema, existiendo factores internos que pueden afectar a la red como las características de los componentes integrados, posición e integridad de los mismos para captar o realizar una función de manera precisa, y también factores externos como variaciones de temperatura, interferencias electromagnéticas y conexiones.

3.1.3.2. Sistemas de control y subsistemas del proyecto

En la actualidad todos los vehículos vienen integrados con subsistemas de control que componen un sistema general, ya que las unidades electrónicas se encargan de controlar diferentes áreas funcionales, que pueden variar su complejidad con las relaciones embebidas de la red y así mismo su equipamiento. En relación con lo mencionado, el sistema de control del presente proyecto (tabla 3.7) abarca funcionalidades cotidianas con dependencias lógicas y automatización.

Tabla 3.7 Sistema de control general de la red CAN

SISTEMA DE CONTROL DE LA CARROCERÍA	
Sistema de luces	
Descripción	**Relación embebida**
Principales luces de un vehículo, como: posición, cruce, carretera, neblineros, freno, retroceso, direccionales y emergencia.	Luces adaptativas, que realizan una conmutación automática de luces de carretera a las de cruce al presentarse una referencia de luz acercándose en sentido contrario. Además, de encender de manera selectiva las luces antiniebla dependiendo de la intensión de giro del volante en curvas o intersecciones en situaciones de baja visibilidad y velocidad.
Sistema limpiaparabrisas	
Descripción	**Relación embebida**
Movimiento de brazos en un rango de 0 a 90° o 120°, los cuales están en contacto con el parabrisas para mejorar la visibilidad con la ayuda de agua	Ajuste automático de la velocidad y frecuencia del sistema limpiaparabrisas tras la detección externa de un ambiente húmedo o lluvioso cuando el sistema está en la funcionalidad automática
Sistema de advertencia de colisión	
Descripción	
Evaluación de la cercanía con objetos en la zona de punto ciego del vehículo al seleccionar la marcha atrás, en donde se genere una alerta acústica que incrementa su frecuencia de activación en función de la distancia con un objeto cercano, es decir, a menor distancia mayor frecuencia.	
OTROS SISTEMAS DE CONTROL	
Luces de cortesía	
Descripción	**Relación embebida**
Luces automáticas que se activan al evaluar que el umbral de la luminosidad interior de un vehículo se encuentra por debajo del programado.	Encendido automático de luces en la cabina del vehículo como cortesía al detectar una baja luminosidad en el ambiente y la puerta abriéndose, y así mismo mantenerse encendidas al cerrar la puerta y detectar la presencia de un usuario
Claxon	
Descripción	
Señal acústica que advierte a peatones y conductores de una situación de atención o peligro	

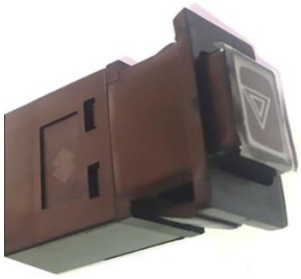
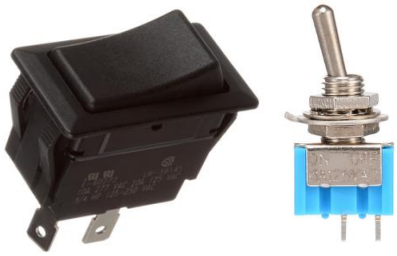
Tracción de motor DC	
Descripción	**Relación embebida**
Tracción al eje posterior del vehículo por medio de un motor DC de rotación continua, el cual puede variar su velocidad y sentido de giro.	Evaluación de estado de conducción como parámetro para el sentido de giro, y velocidad rotacional por la aceleración del mismo para la función embebida del sistema de luces adaptativas antiniebla.

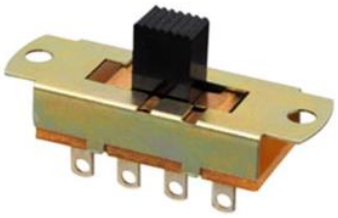
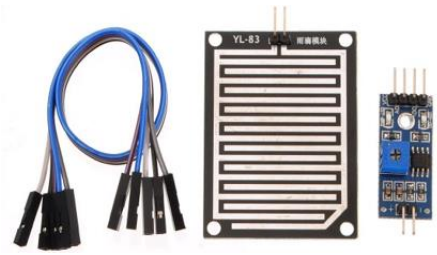
Fuente: Autor

3.1.3.3. Materiales y equipos

En la tabla 3.8 se detalla los dispositivos que se implementaron para captar señales digitales y analógicas en el sistema general de red, como una descripción de su funcionamiento y características.

Tabla 3.8 Dispositivos captadores o entradas del sistema embebido

Denominación	Descripción
Palanca conmutadora de luces y plumas del vehículo Chevrolet D-max del año 2005 	Es un conjunto conmutador conformado por dos palancas (luces y limpiaparabrisas). La primera permite seleccionar el estado de las luces, teniendo las posiciones de apagado, posición, cruce, carretera y direccionales. Por otra parte, la palanca de limpiaparabrisas permite seleccionar una frecuencia baja y alta de movimiento, y posiciones momentáneas de un barrido rápido MIST y roció o lavaparabrisas PULL.
Interruptor de luces de emergencia 	Es un switch de dos posiciones estables que se bloquean al seleccionar alguno de los dos estados (encendido y apagado), permitiendo controlar el accionamiento de las luces intermitentes de un vehículo.
Switch ON/OFF 	Interruptor de dos posiciones únicas (encendido y apagado), internamente cuenta con un mecanismo de acople y desacople, el cual permite que de manera estable la energía fluya y se interrumpa respectivamente. Además, dependiendo del componente, la resistencia a cambios de tensión e intensidad de corriente aumentará y disminuirá.

Conmutador deslizante de 3 posiciones 	Es un switch que internamente cuenta con un mecanismo que se mueve horizontalmente para establecer una conexión eléctrica o evaluación al encontrarse en alguna de las dos posiciones laterales, ya que su centro denota un estado neutral o apagado.
Potenciómetro de 10k ohmios 	Un potenciómetro es un dispositivo divisor de voltaje, cuenta internamente con una espiral resistiva que varía su valor dependiendo de la posición de un contacto deslizante que es ajustado por medio de una perilla. Es decir, modifica la resistencia de los extremos, lo cual permite ajustar parámetros de proyectos dependiendo de la señal de salida
Pulsador 	Como su nombre mismo lo dice, es un dispositivo electromecánico que permite cerrar o abrir un circuito por periodos momentáneos de tiempo. Por lo cual, al dejar de pulsar dicho botón, este vuelve de manera automática a su posición original.
Sensor ultrasónico HC-SR04 	Permite censar la distancia de objetos por medio de ondas ultrasónicas, un emisor emite pulsos ultrasónicos que se impactan en el objeto y se reflejan hacia un receptor, siendo el tiempo de dicha acción el parámetro necesario para calcular la distancia.
Sensor de luz LDR 	Es un fotorresistor que censa la intensidad de luz del ambiente para variar el valor de su resistencia interna, siendo su comportamiento inversamente proporcional y de la siguiente manera: • Su resistencia disminuye a mayor iluminación • Su resistencia aumenta a menor iluminación
Sensor detector de lluvia MH-RD 	Es un sensor compuesto por una placa y un conjunto de almohadillas conductoras, las cuales varían su conductividad eléctrica al entrar en contacto con humedad o lluvia, provocando una variación en su resistencia interna, generando mediciones analógicas entre 0 y 1023 niveles de señal o digitales con la presencia o ausencia de lluvia. Además, puede ser calibrado dependiendo de las necesidades.

Sensor óptico reflexivo 	Se compone por un remitente de luz infrarroja y un fototransistor para detectar la cantidad reflejada al estar en presencia de un objeto o superficie, por lo cual varía su resistencia, siendo ideal para evaluar la existencia o ausencia de objetos en una posición y distancia específica.

De igual manera, en la tabla 3.9 se presentan los actuadores que forman parte del sistema de control, los cuales permiten realizar acciones específicas según los parámetros programados en los módulos de control, ya que son estos los que generan órdenes de salida y la interacción de los sistemas embebidos.

Tabla 3.9 Actuadores o salidas del sistema embebido

Denominación	Descripción
Leds 	Son diodos que emiten luz en presencia de corriente eléctrica, su accionamiento es inmediato, ya que su principio de funcionamiento se basa en electroluminiscencia, y pueden adoptar diferentes colores dependiendo del material semiconductor y procesos.
Servomotor MG90s 	Es un componente electromecánico compacto que permite controlar el ángulo o posición de un eje mediante señales de control PWM, pero de manera limitada al tener un rango de movimiento de 0 a 180 grados, además de soportar cargas de hasta 2.2 Kg
Servomotor MG995 rotación continua 	Al igual que el servomotor estándar, es un componente electromecánico, pero sin limitaciones en cuanto al rango de movimiento, ya que está diseñado para girar de manera continua en ambos sentidos según se requiera con cargas de hasta aproximadamente 15 kilogramos.
Buzzer 	Es un dispositivo electromecánico que permite generar sonidos por medio de vibraciones, las cuales son causadas por un elemento piezoeléctrico al alimentar el dispositivo con corriente eléctrica.

Los dispositivos detallados en las tablas 3.8 y 3.9 conformaron el sistema general de la red, siendo una configuración híbrida entre componentes automotrices y adaptados. Su funcionalidad es similar a los elementos reales de un vehículo, por lo cual, para su evaluación y desempeño se optó por implementarlos en un banco de simulación con un modelo de vehículo a escala 1/12.

3.1.3.4. Funcionalidad individual de cada módulo CAN

Para cubrir la funcionalidad del sistema propuesto, se estableció que los 4 módulos de la red se encarguen de captar y controlar al menos un dispositivo en cada sección, como lo expuesto en la tabla 3.10. Es decir, no limitar a que un módulo sea solamente emisor o receptor, de manera que se establezca una comunicación multidifusión.

Tabla 3.10 Funcionalidad de los módulos de la red

Control de la carrocería		
Módulo	**Captación**	**Control**
1	• Fotorresistor LDR para el control de luz adaptativa de carretera a cruce • Pulsador para el control de luces de freno • Sensor detector de lluvia para el control automático de limpiaparabrisas • Interruptores de antiniebla frontales y posteriores	• Leds para el encendido de las luces de cruce, carretera, direccionales, emergencia, y luces frontales de posición y antiniebla
2	• Conmutador tipo palanca mencionado en la tabla 3.7, para el control de las luces principales • Conmutador deslizante que evalúa el estado de conducción (avance, neutral y retroceso) • Interruptor luces de emergencia	• Servomotores para el sistema limpiaparabrisas, y led para la referencia de activación de rocío PULL • Buzzer de alerta de proximidad
3	• Sensor ultrasónico para la advertencia acústica de proximidad en una conducción en reversa • Conmutador tipo palanca para el control de limpiaparabrisas mencionado en la tabla 3.7, e interruptor para activar el sistema automático • Pulsador para el control del claxon	• Leds para el encendido de las luces de freno, retroceso y las luces posteriores de posición y antiniebla
Otros sistemas de control		
4	• Sensor de luz LDR para el control de luz automática de la cabina • Sensor óptico reflectivo para la evaluación de presencia o ausencia de usuario en la cabina • Potenciómetro para la evaluación de posición del volante de la dirección • Potenciómetro para acelerar de un motor DC • Pulsador que evalúa apertura o cierre de la puerta	• Servomotor de rotación continua para darle tracción al eje posterior del vehículo • Luz automática de la cabina • Buzzer claxon

Fuente: Autor

3.1.3.5. Diseño del circuito electrónico de la red

Debido a la disposición hibrida del sistema, la mayoría de las conexiones de los componentes de entrada y salida, no requirieron una alimentación superior a la proporcionada por el microcontrolador Arduino. Por lo cual, el esquemático se lo realizó teniendo en cuenta la configuración adecuada de cada componente con elementos de protección y conmutación. Sin embargo, en el caso de que se necesite implementar actuadores automotrices o componentes que requieran una fuente externa de alimentación, es necesario implementar elementos adicionales como: transistores del tipo NPN (unión bipolar), fusibles y relés, según lo propuesto en la figura 3.26 en una conexión de faros. En donde, la señal generada por el microcontrolador Arduino permite la activación del circuito.

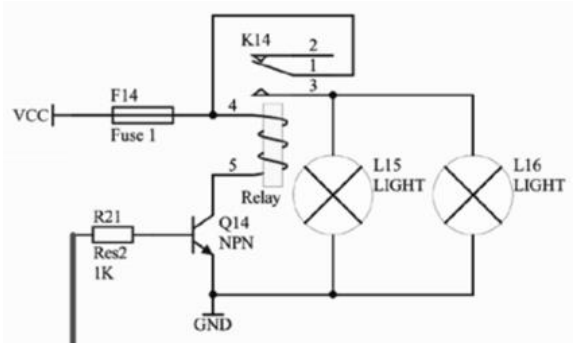


Figura 3.26 Control de actuadores alimentados con una fuente externa
(Yee et al., 2017, p. 4)

En la etapa de elaboración del esquemático o diagrama de conexión de los 4 módulos, se utilizó el software Altium Designer por su versatilidad en cuanto a herramientas de diseño y manejo. En el cual, se realizaron varios diseños de librerías para incluir la placa del microcontrolador Arduino y los componentes planteados en la tabla 3.10 según el módulo encargado de su evaluación o funcionamiento.

Esquemático módulo 1

El diagrama de conexión del módulo 1 (figura 3.27), se integró parámetros de configuración de circuitos electrónicos, para que los elementos se adapten a los requerimientos de evaluación y control, como la estructura PULL DOWN de los captadores de señales digitales, la cual genera un cambio de estado dominante (0) a recesivo (1) al presentarse una entrada activa, como también para la conexión de un fotorresistor que manejan 1023 niveles de voltaje. Además, los actuadores que en este caso son leds compartieron una configuración de conexión a ánodo y cátodo común, permitiendo encender dos o más leds simultáneamente, pero en las luces que requieren una alimentación externa como las de carretera con 9 Voltios, es necesario añadir un transistor NPN como switch a GND.

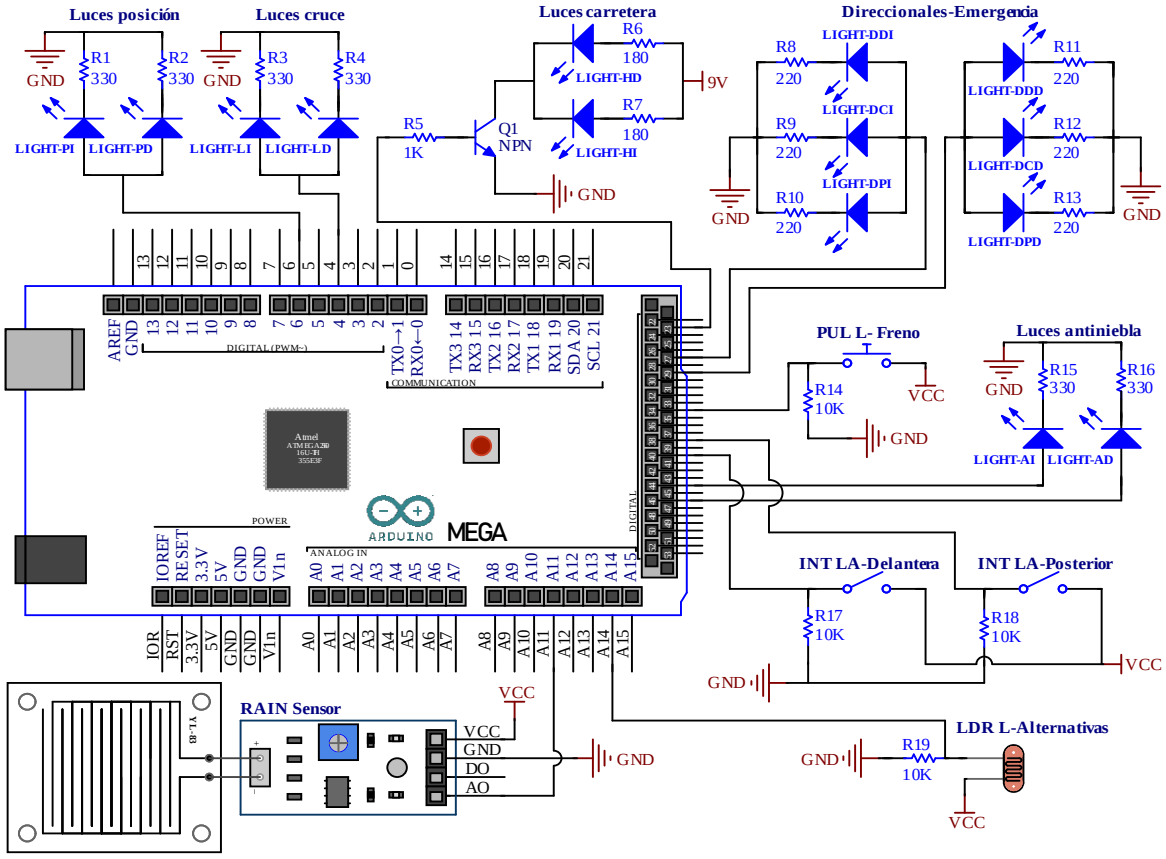


Figura 3.27 Diagrama de conexión de componentes del módulo 1

Fuente: Autor

Esquemático módulo 2

En el caso del módulo 2 (figura 3.28) de igual manera se aplicó una configuración PULL DOWN a excepción del conmutador para la evaluación de estado de conducción con una estructura en PULL UP.

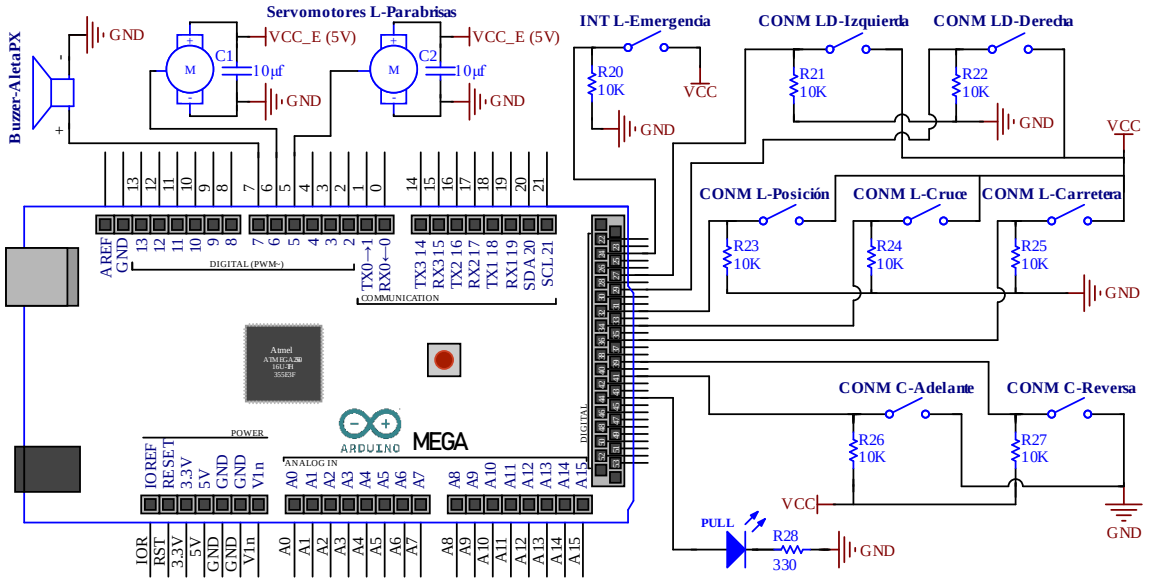


Figura 3.28 Diagrama de conexión de componentes del módulo 2

Fuente: Autor

Además, según lo planteado, se añade un Buzzer, un led para la referencia de la función de rocío o lavaparabrisas y dos servomotores que simulan el comportamiento del mecanismo de barrio de limpiaparabrisas, los cuales demandaron de una alimentación estable de 5V para funcionar de manera estable al ser controlados por el microcontrolador Arduino.

Esquemático módulo 3

La configuración del diagrama de conexión del módulo 3 (figura 3.29) es similar a los esquemáticos de los módulos anteriores, ya que implementa la estructura PULL DOWN para la evaluación de entradas, configuración a ánodo y cátodo común para diodos emisores de luz de baja intensidad y los alimentados por una fuente externa que son controlados por medio de un transistor NPN y Arduino.

Además, se implementó un sensor ultrasónico que emplea dos entradas digitales, para emitir y receptar pulsos que permiten medir la distancia con objetos cercanos.

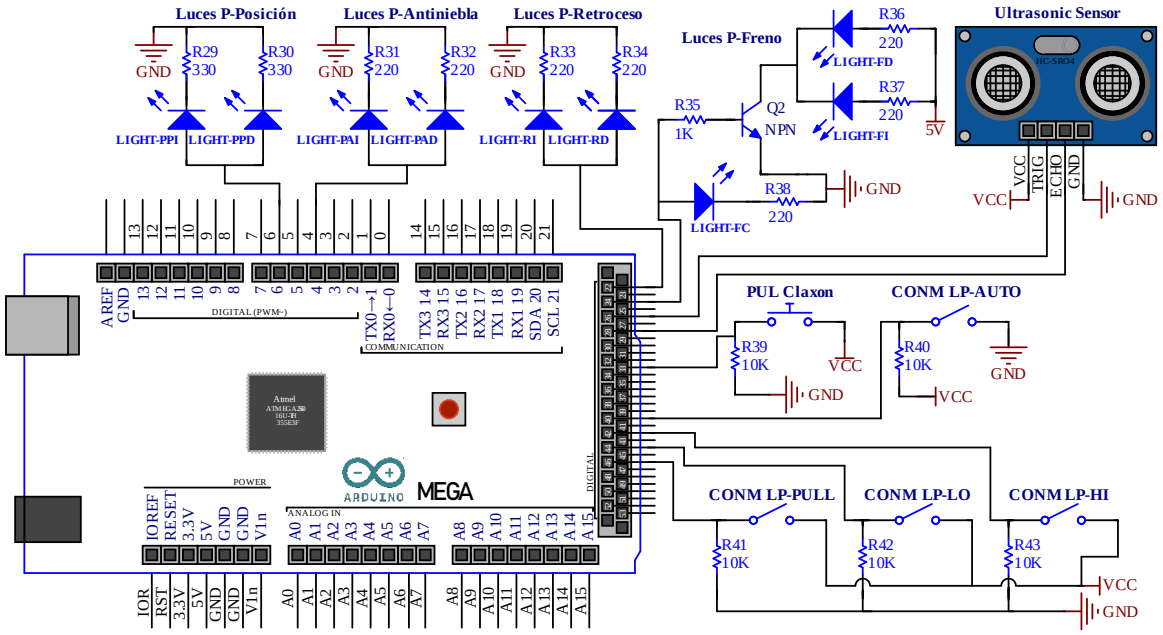


Figura 3.29 Diagrama de conexión de componentes del módulo 3

Fuente: Autor

Esquemático modulo 4

El diagrama de conexión o esquemático del módulo 4 (figura 3.30) se centró en componentes adicionales que se relacionan a los implementados en los 3 módulos anteriores de control de la carrocería, por lo cual sus características de configuración son similares a excepción de la adición de potenciómetros que funcionan como divisores de voltaje (0 a 1023 niveles), Buzzer, sensor infrarrojo, LDR, y un motor DC de rotación continúa alimentado por una fuente externa de 5V, al igual que los leds 8 smd light rojos y servomotores MG90S.

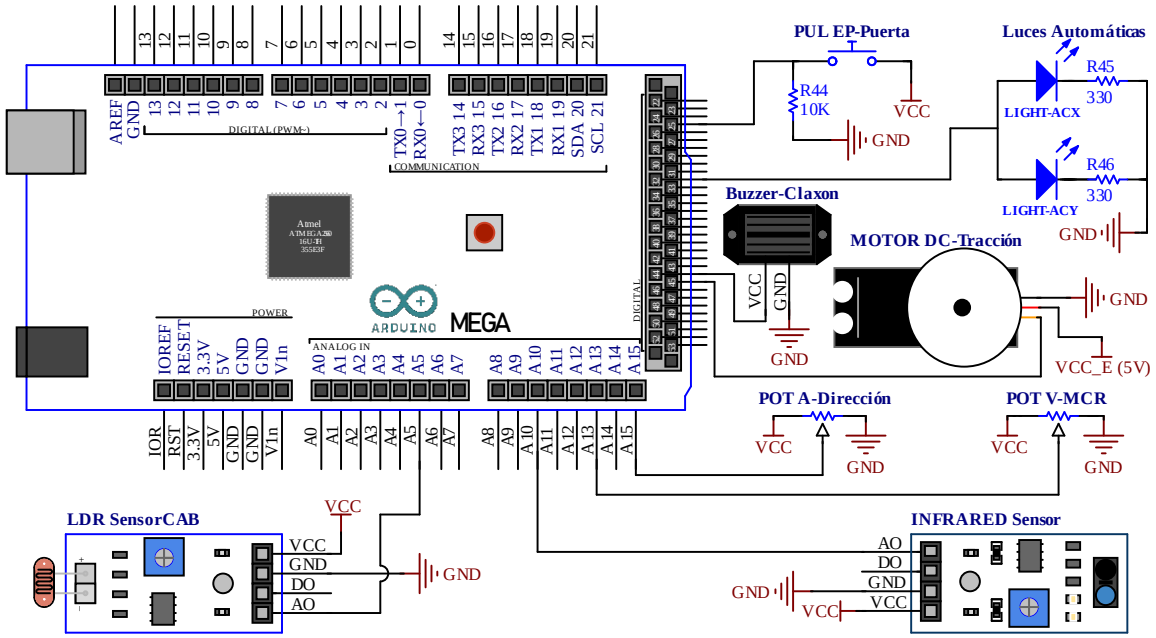


Figura 3.30 Diagrama de conexión de componentes del módulo 4

Fuente: Autor

3.1.3.6. Programación general del sistema embebido

Según lo propuesto en el alcance y diseño de la red, el sistema se conformó por 4 módulos CAN, los cuales en el apartado anterior de comunicación bidireccional se configuraron de tal manera que todos envíen y reciban mensajes en periodos de tiempo específicos, simulando una situación de máxima exigencia con el principio multidifusión. Sin embargo, una red CAN BUS necesariamente no trabaja así, ya que este comportamiento sobrecargaría el Bus de datos con mensajes en algún momento innecesarios, por lo cual, se modificó la lógica de los códigos integrando el principio de comunicación por eventos, para que estos solo transmitan mensajes cuando dispongan de información útil que compartir

Empleando el flujograma de la figura 3.31, se realizaron los ajustes para cambiar la lógica en el manejo de datos de los módulos en su funcionalidad emisora y receptora, en relación con lo primero se estableció que la evaluación de componentes captadores como pulsadores, interruptores, actuadores o incluso sensores se controle por medio de la inicialización de una bandera, la cual permite dar paso a la transmisión de mensajes al presentarse una señal específica en al menos una de dichas entradas. En donde, la información ocupa uno o más bytes en una posición exclusiva de los 8 bytes de la trama de datos, para que los demás módulos de control puedan acceder y procesar dicha señal cuando sea de utilidad.

Por otra parte, en la recepción de mensajes se incorporó máscaras y filtros para que los módulos solo acepten mensajes de utilidad, los cuales son almacenados en Buffers únicos para evitar colisión de datos de diferentes módulos emisores.

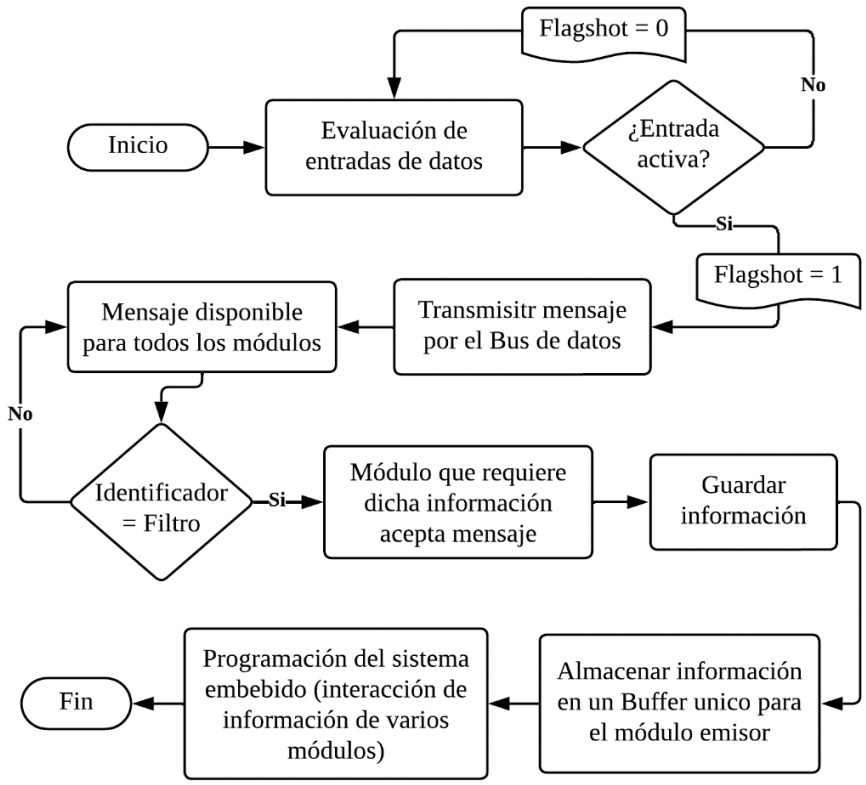


Figura 3.31 Lógica de programación para el manejo de información

Fuente: Autor

En relación con lo mencionado, se realizó cambios de variable en las señales activas de pulsadores, interruptores y conmutadores con valores diferentes a un estado dominante (0) o recesivo (1) antes de transmitir el mensaje por el Bus de datos, permitiendo diferenciar estados de activación y aumentado la seguridad en cuanto a señales parasitas de fuentes externas. Por otro lado, en la evaluación de potenciómetros y sensores que mapean mediciones analógicas en 1023 niveles de voltaje, se hizo una conversión para manejar datos en un rango de 0 a 255, ya que es el valor mínimo y máximo que puede ser transmitido en un byte, pero si se requiere transmitir uno más preciso, y que ocupa un mayor número de bits, se puede separar el valor en múltiples bytes antes de ser transmitido, y luego al receptarlos reconstruir el valor original con todas sus divisiones.

En el anexo IV y V se añade una tabla resumen de los subsistemas de control con sus señales referenciales de funcionamiento y la estructura de código de programación que se manejó en los 4 módulos respectivamente, la cual cuenta con una disposición de funciones ordenadas para la evaluación de entradas y operación de subsistemas, en donde interactúan los datos de la red en variables globales, ya que la recepción y almacenamiento de mensajes se lo hace en Buffers únicos para cada módulo, los cuales se actualizan con la llegada de un nuevo mensaje evitando problemas de direccionamiento y colisiones.

CAPÍTULO III

4. RESULTADOS Y DISCUSIONES

En este capítulo se presentan los resultados alcanzados a partir de la investigación y desarrollo metodológico del presente trabajo, por lo cual se detalla la organización base de la red con los módulos propuestos, la comunicación unidireccional y bidireccional, además de la configuración y funcionamiento del sistema general de la red multiplexada CAN. En donde, la evaluación de la comunicación fue realizada por la herramienta CAN Bus Analyzer y los osciloscopios G-SCOPE-2 y Hantek 6074BC, obteniendo información directamente del Bus de datos, para comprobar su comportamiento y así solucionar posibles errores, permitiendo elaborar una red multiplexada CAN segura, estable y eficiente.

4.1. RESULTADOS DEL DISEÑO Y CONEXIÓN DE MÓDULOS CAN AL BUS DE DATOS

4.1.1. CONEXIÓN ARDUINO MEGA 2560 Y PLACA CAN BUS SHIELD

Para realizar la conexión de la placa CAN Bus Shield se consultó la disposición de los pines SPI del Arduino Mega 2560, ya que por su diseño es mayormente compatible con el Arduino UNO. Por lo cual, fue necesario realizar puentes de conexión (figura 4.1) de los pines 13 SCK, 12 MISO y 11 MOSI de la placa CAN Shield a los pines: 52 SCK, 50 MISO y 51 MOSI del microcontrolador, excepto el pin 10 CS/SS por la configuración de la librería mcp2515_can.h, la cual se utilizó en el apartado de programación para establecer la comunicación entre los módulos de la red, aunque también es posible realizar dicha conexión en los pines ICSP (In Circuit Serial Programming).

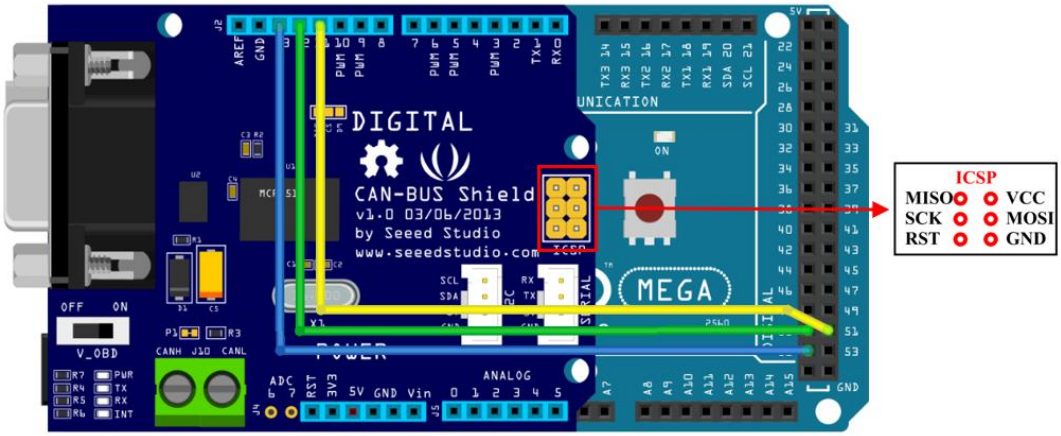


Figura 4.1 Interconexión placa Arduino Mega y placa CAN Bus Shield
Fuente: Autor

4.1.2. CONEXIÓN DE MÓDULOS CAN A LA LÍNEA DE COMUNICACIÓN

Según lo establecido en la selección del diseño de red, la topología que se aplicó fue del tipo Bus para la conexión de los 4 módulos CAN en paralelo (figura 4.2), por medio de dos cables (CAN High y CAN Low) que normalmente deben ir trenzados y en sus extremos dos resistencias de 120 ohmios, siendo una estructura compatible con la norma ISO 11898. En relación con esto, cada módulo cuenta con una resistencia que puede ser habilitada al soldar el Jumper J3. Sin embargo, se optó por implementar resistencias externas por recomendaciones de usuarios en el desempeño de la red, y facilidad de sustitución en caso de ser necesario.

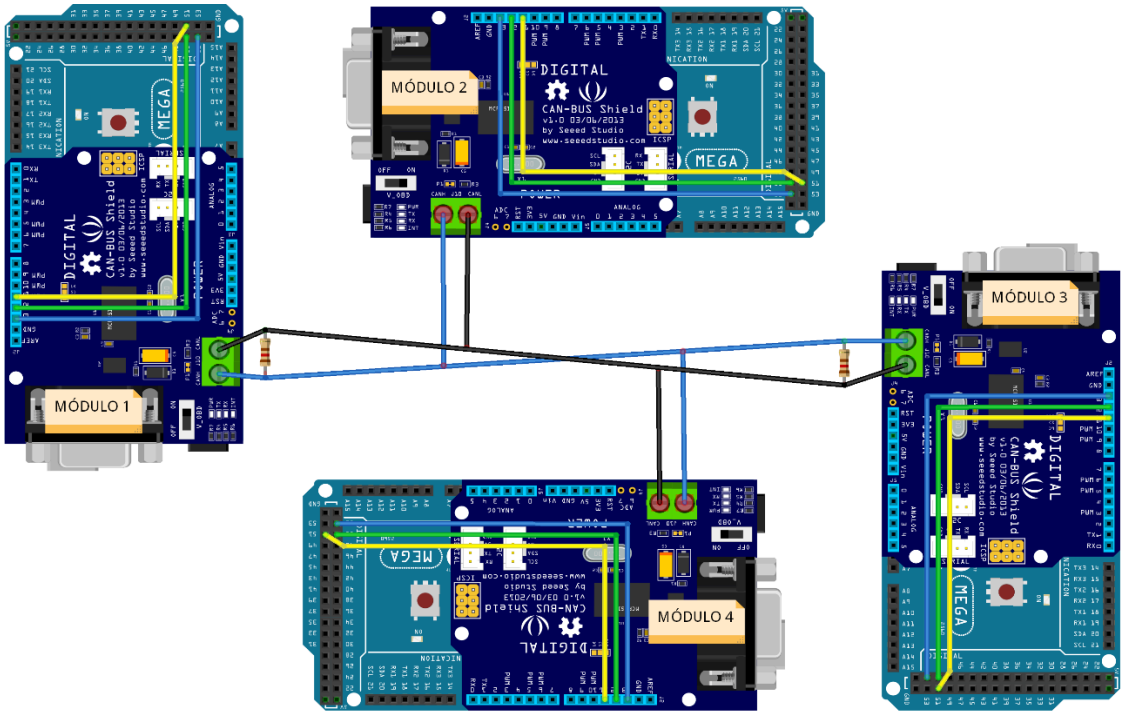


Figura 4.2 Esquema de conexión de los 4 módulos de la red
Fuente: Autor

4.2. RESULTADOS DE LA COMUNICACIÓN ENTRE MÓDULOS CAN

4.2.1. COMUNICACIÓN UNIDIRECCIONAL Y SEÑAL TRANSMITIDA

La configuración maestro – esclavo permitió conocer el uso de las librerías con sus principales funciones de envío y recepción de datos, siendo una configuración muy estable al establecer periodos de transmisión por medio de una interrupción en el programa como un delay(), o también obviando el uso de algún tipo de interrupción, lo cual genera una comunicación multidifusión constante (figura 4.3) en el void loop. En donde, la velocidad

de transmisión y el formato de la trama de datos determina el tiempo que tarda en enviarse por la línea de comunicación un mensaje completo.

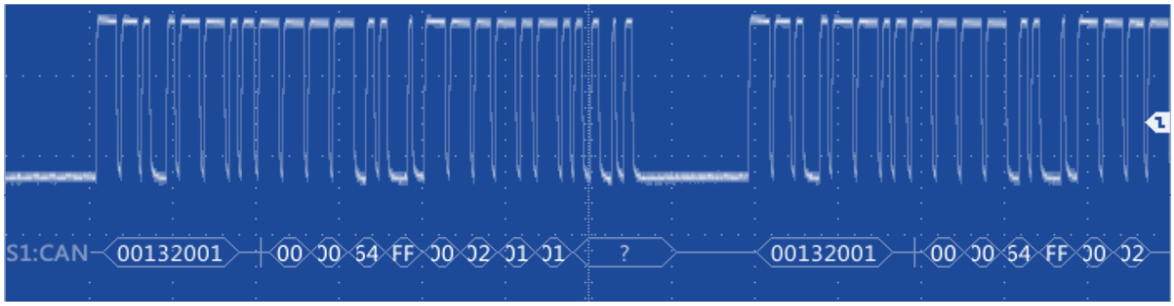


Figura 4.3 Transmisión de mensaje continuo
Fuente: Autor

Según lo propuesto, la red la red se configuró a una velocidad de 1 Mbps. Sin embargo, debido a la operación de la placa CAN Bus Shield a 8MHz, la velocidad de operación se reduce al rango de 500 Kbits/s mostrado en la tabla 2.3. En consecuencia, el mensaje con identificador 0x132001 que se configuró en el apartado de comunicación unidireccional, tardó 264 microsegundos en transmitirse, ya que en la figura 4.4 se puede identificar que el mensaje cuenta con 132 bits hasta el primer bit de fin de trama y la interacción de sus campos de datos, en donde, por medio de la lectura de las firmas eléctricas se evidencia la estructura del formato extendido establecido y la inserción de bits de relleno o stuffing que se añaden al transmitir 5 bits continuos de similar estado.

- **Inicio de trama. –** Comienzo de transmisión con un bit dominante (0)
- **Identificador ID-A. –** 11 primeros bits más representativos del ID (00000000100)
- **Bit SRR. –** Bit que debe ser recesivo (1)
- **Bit IDE. –** Se transmite un bit recesivo (1) para tramas en formato extendido
- **Identificador ID-B. –** Segunda parte del ID con 18 bits (110010000000000001)
- **Bit RTR. –** Bit en estado dominante (0) al ser una trama de datos normal
- **Bits r0, r1. –** Bits de reserva normalmente en estado dominante (0)
- **Bits DLC. –** Longitud de bytes del campo de datos, al ser 8 se representa en: 1000
- **Campo de datos. –** 8 bytes con el mensaje {0, 0, 100, 255, 256, 514, 1, 1}
- **Bits CRC. –** Verificación de transmisión de datos (001100111011001)
- **Delimitador CRC. –** Tras la correcta verificación debe ser un bit recesivo (1)
- **Bit ACK. –** Bit de referencia a la funcionalidad, transmisor (1) y receptor (0)
- **Delimitador ACK. –** Bit en estado recesivo (1) normalmente
- **Fin de trama (EOF). –** Se emite 7 bits recesivos (1), como referencia de fin

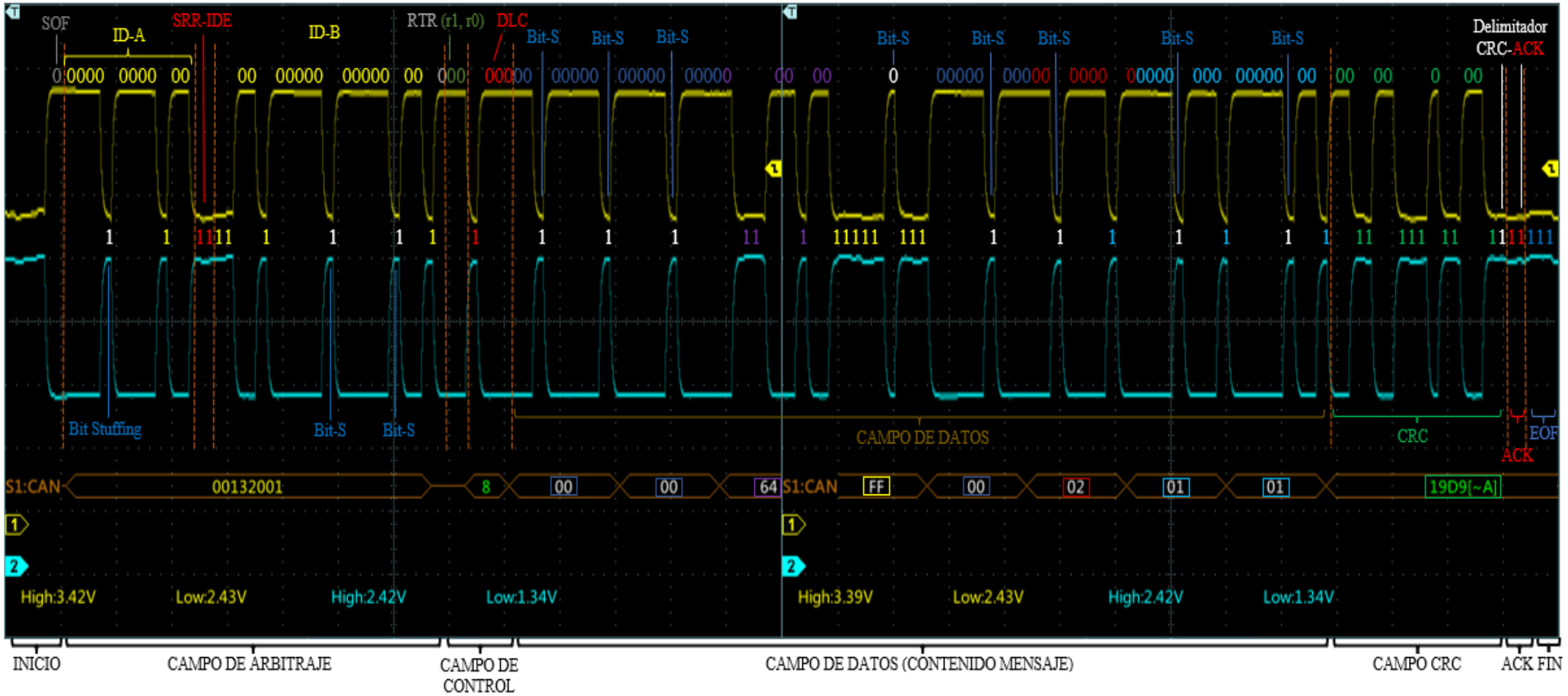


Figura 4.4 Decodificación de una trama de datos en formato extendido

Fuente: Autor

4.2.2. COMUNICACIÓN ENTRE MÚLTIPLES MÓDULOS CAN

La interacción entre los diferentes módulos de la red puede ser configurada dependiendo de las necesidades de la misma, en este caso se estableció una comunicación multidifusión constante para simular la máxima exigencia de la red. Es decir, todos los módulos envían y reciben mensajes por medio de la línea de comunicación, por lo cual para establecer dicha configuración se implementó de manera paulatina los módulos, y por medio de las herramientas de análisis se concluyó que la frecuencia de acceso al Bus de datos determina la estabilidad de comunicación, ya que las fluctuaciones pueden generar colisiones de datos y afectar el mecanismo de arbitraje, en consecuencia se buscó alternativas para controlar los periodos de transmisión de los diferentes módulos de la red.

4.2.2.1. Función attachInterrupt

Este tipo de interrupción permite la comunicación entre módulos CAN sin establecer un periodo de transmisión fijo, por lo cual esto varía con la configuración de velocidad CAN y serial de Arduino, o por acción de un dispositivo externo conectado a la red, como lo planteado en el trabajo de titulación de Feliciano Fuentes (2019), en donde empleando un clúster de instrumentos se genera una comunicación estable con 3 módulos CAN (Arduino + Can Bus Shield), ya que dicho componente se encarga de controlar la frecuencia de acceso al Bus de datos de los módulos y no de manera independiente.

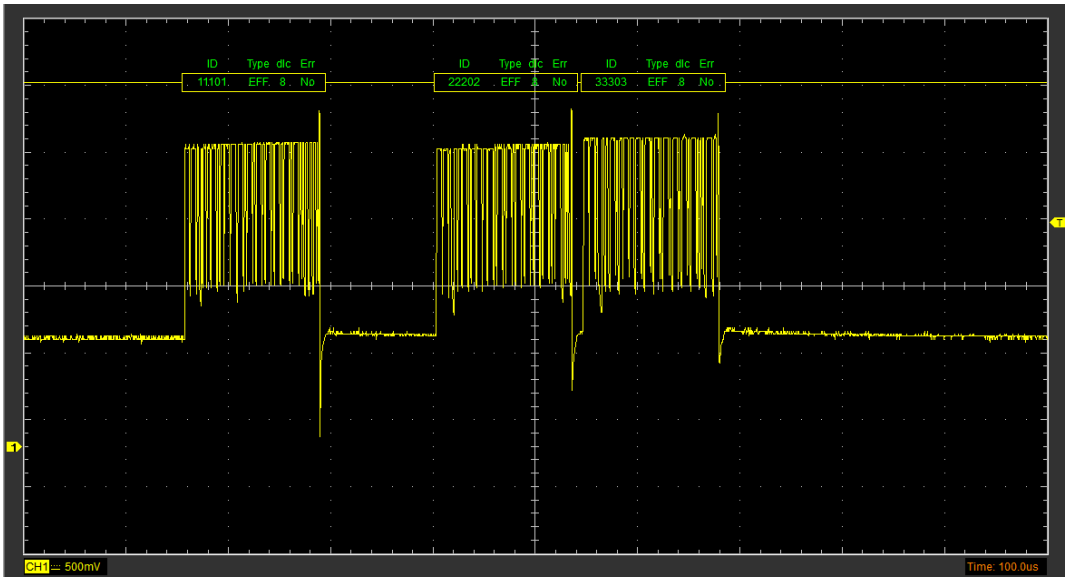


Figura 4.5 Comunicación CAN únicamente con la función attachInterrupt

Fuente: Autor

Dado que el presente trabajo es una red independiente, se realizaron pruebas con el primer planteamiento, al implementar de igual manera 3 módulos se genera una comunicación

medianamente estable (figura 4.5), pero al prolongar el tiempo de funcionamiento las variaciones de los periodos de transmisión incrementan y así mismo el counter de mensajes. Por lo que, al implementar el cuarto módulo la comunicación no fue posible, ya que las variaciones en los puntos mencionados incrementan notoriamente (figura 4.6). Por ende, se descartó esta opción base para la comunicación de la red.

Fixed Trace

TRACE	ID	DLC	DATA 0	DATA 1	DATA 2	DATA 3	DATA 4	DATA 5	DATA 6	DATA 7	TIME STAMP (sec)	TIME DELTA (sec)	COUNTER
RX	0x302001x	8	0x0D	0x0D	0x03	0x03	0x03	0x03	0x0D	0x0D	1004,1172	0,003	2816
RX	0x432001x	8	0x0F	0x0F	0x01	0x00	0x00	0x01	0x0F	0x0F	1004,0812	0,006	3110
RX	0x132001x	8	0x00	0x00	0x0A	0x0A	0x0A	0x0A	0x00	0x00	1004,0812	0,092	447
RX	0x200113x	8	0x0B	0x0B	0x01	0x01	0x01	0x01	0x0B	0x0B	1004,0812	0,006	2402

Figura 4.6 Comportamiento de la comunicación entre 4 módulos CAN únicamente con la función attachInterrupt

Fuente: Autor

4.2.2.2. Función millis

La implementación de esta alternativa permite establecer periodos de transmisión modificables sin interrumpir la ejecución del programa base, generando una interconexión más estable en comparación a solo emplear la función attachInterrupt. Sin embargo, como se evidenció en el apartado de comunicación entre dos módulos CAN, los periodos de transmisión establecidos presentan pequeñas fluctuaciones (+1 o -1), ocasionando que en la línea de comunicación él acceso al Bus de datos varíe, aunque también en ciertas ocasiones los tiempos se coordinan e interviene el proceso de arbitraje (figura 4.7).

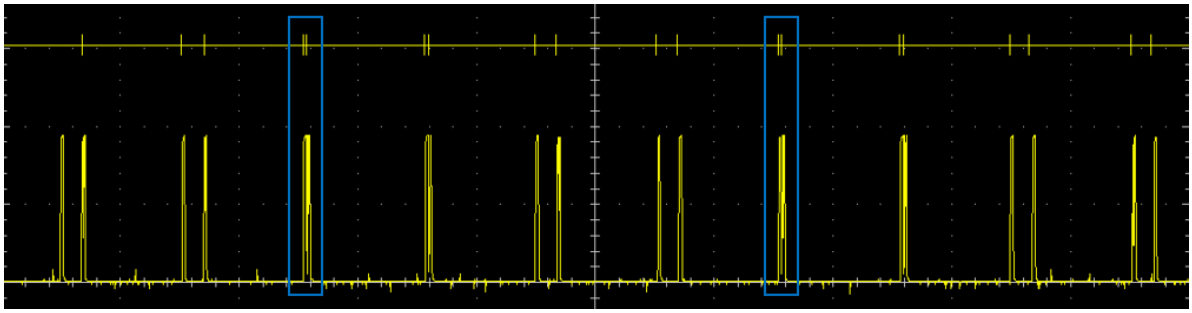


Figura 4.7 Coordinación de periodos de transmisión

Fuente: Autor

De igual manera, al implementar más de dos módulos, la comunicación presenta variaciones en la frecuencia de acceso al Bus de datos (figura 4.8). Sin embargo, a pesar de este comportamiento, no deja de ser una opción viable para el manejo de una comunicación multidifusión en la que no se tome en cuenta las fluctuaciones de transmisión, ya que también dispone de la característica de no detener la ejecución del programa principal. No obstante, la exactitud de los periodos y frecuencia es un parámetro muy importante, por lo cual en el presente trabajo también se descartó esta opción.

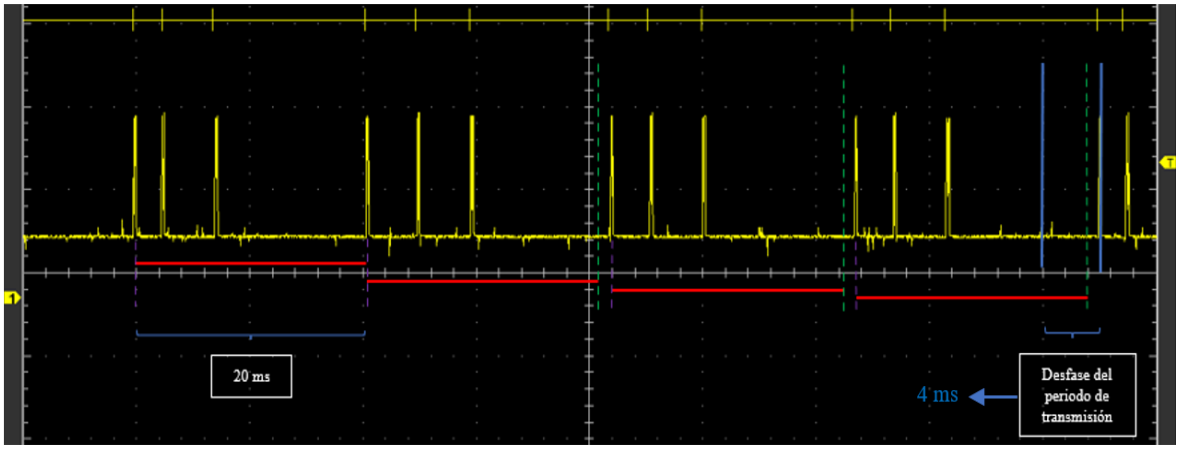


Figura 4.8 Desfase del periodo de transmisión con la función millis

Fuente: Autor

4.2.2.3. Interrupción por timer

Según lo planteado en la metodología, la opción que permite controlar y establecer una comunicación multidifusión estable es el Timer, ya que se comprobó que no bloquea la ejecución del código principal y mantiene los periodos de transmisión establecidos en una red multimaestro, esto por medio de la lógica de programación que integra, para llamar a la función emisora al transcurrir un tiempo especificado y adoptar la funcionalidad de receptor cuando exista mensajes en la línea.

Rolling Trace

TRACE	ID	DLC	DATA 0	DATA 1	DATA 2	DATA 3	DATA 4	DATA 5	DATA 6	DATA 7	TIME STAMP (sec)	TIME DELTA (sec)
RX	0x132001x	8	0x00	0x00	0x0A	0x0A	0x0A	0x0A	0x00	0x00	412,5152	0,008
RX	0x302001x	8	0x0D	0x0D	0x03	0x03	0x03	0x03	0x0D	0x0D	412,5072	0,001
RX	0x200113x	8	0x0B	0x0B	0x01	0x01	0x01	0x01	0x0B	0x0B	412,5062	0,001
RX	0x132001x	8	0x00	0x00	0x0A	0x0A	0x0A	0x0A	0x00	0x00	412,5052	0,010
RX	0x132001x	8	0x00	0x00	0x0A	0x0A	0x0A	0x0A	0x00	0x00	412,4952	0,002
RX	0x432001x	8	0x15	0x0F	0x01	0x00	0x00	0x01	0x0F	0x15	412,4933	0,006
RX	0x302001x	8	0x0D	0x0D	0x03	0x03	0x03	0x03	0x0D	0x0D	412,4873	0,001
RX	0x200113x	8	0x0B	0x0B	0x01	0x01	0x01	0x01	0x0B	0x0B	412,4862	0,001
RX	0x132001x	8	0x00	0x00	0x0A	0x0A	0x0A	0x0A	0x00	0x00	412,4852	0,010
RX	0x132001x	8	0x00	0x00	0x0A	0x0A	0x0A	0x0A	0x00	0x00	412,4752	0,008
RX	0x302001x	8	0x0D	0x0D	0x03	0x03	0x03	0x03	0x0D	0x0D	412,4673	0,001
RX	0x200113x	8	0x0B	0x0B	0x01	0x01	0x01	0x01	0x0B	0x0B	412,4662	0,001
RX	0x132001x	8	0x00	0x00	0x0A	0x0A	0x0A	0x0A	0x00	0x00	412,4652	0,010
RX	0x132001x	8	0x00	0x00	0x0A	0x0A	0x0A	0x0A	0x00	0x00	412,4552	0,008
RX	0x302001x	8	0x0D	0x0D	0x03	0x03	0x03	0x03	0x0D	0x0D	412,4473	0,001

Figura 4.9 Secuencia continua de comunicación entre 4 módulos CAN con un timer

Fuente: Autor

La estabilidad en cuanto a los periodos de transmisión establecidos en el apartado de comunicación entre más de dos módulos CAN, ocasionó que el proceso de arbitraje intervenga cuando la red sé sincrónica (figura 4.9). Sin embargo, este comportamiento no se conserva todo el tiempo, por lo que se infiere que la frecuencia determina el buen funcionamiento de la red. Además, en la figura 4.10 se observa las firmas electrónicas de dicha comunicación, la cual al conectar y desconectar módulos se ajusta al intento de sincronización y comunicación independiente respectivamente.

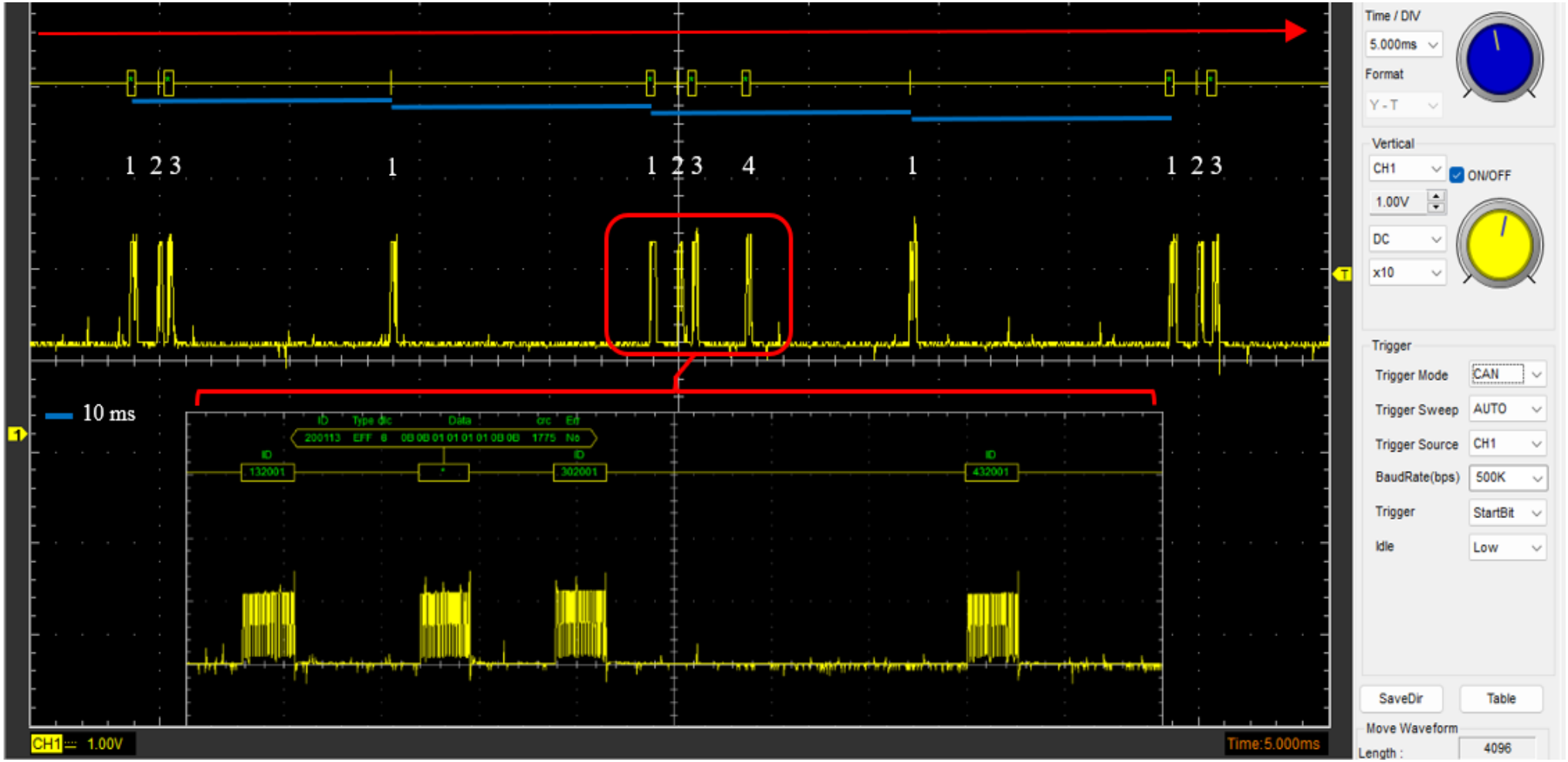


Figura 4.10 Comportamiento de la comunicación multidifusión entre 4 módulos CAN empleando un timer para el control de periodos de transmisión

Fuente: Autor

4.2.3. FÓRMULA PARA OBTENER LA FRECUENCIA (HZ) CONSIDERANDO LOS PERIODOS DE TRANSMISIÓN

El proceso de transformación del periodo de transmisión en el valor de la frecuencia o número de veces que sucede una acción en el lapso de 1 segundo, se lo realizó empleado la siguiente fórmula:

$$f = \frac{(1000)\text{ms}}{(T)\text{ms}} \qquad [4.1]$$

Donde:

f: Frecuencia

T: Periodo de transmisión

Mediante la configuración del código de programación, al utilizar un timer y su comprobación en el Bus de datos, es factible emplear la fórmula [4.1] para obtener la frecuencia de acceso de los 4 módulos CAN, ya que en este punto se maneja periodos de transmisión constantes, que según lo establecido en la figura 3.24 del apartado de comunicación con más de dos módulos CAN son 10, 20 y 100 ms.

Con la aplicación de la fórmula en un periodo de 10 ms:

$$f = \frac{(1000)\text{ms}}{(10)\text{ms}}$$

$$f = 100\ \text{Hz}$$

Se obtiene el valor de frecuencia de 100 Hz, en consecuencia, los resultados de los demás módulos se presentan de manera conjunta en la tabla 4.1, en donde se evidencia que el valor de la frecuencia es inversamente proporcional a los periodos de transmisión. Es decir, cuando los periodos son bajos la frecuencia aumenta y viceversa.

Tabla 4.1 Conversión de los periodos de transmisión constantes en frecuencia

Módulo	Periodo de transmisión (ms)	Frecuencia (Hz)
1	10	100
2	20	50
3	20	50
4	100	10

Fuente: Autor

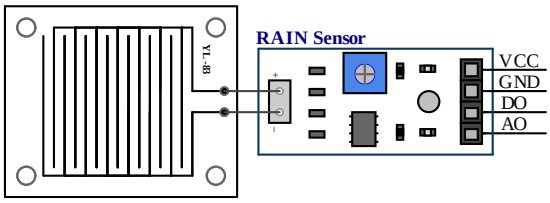
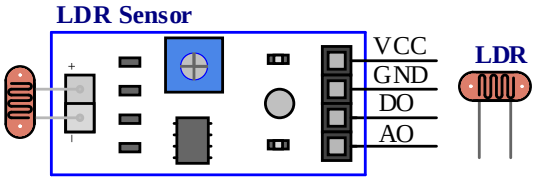
Además, tras pruebas en la modificación de los periodos de transmisión, se determinó que el valor mínimo y estable que se puede establecer es 5 ms con una frecuencia de 200 Hz.

4.3. RESULTADOS DEL DISEÑO E IMPLEMENTACIÓN DEL SISTEMA DE CONTROL

4.3.1. COMPONENTES DE ENTRADA Y SALIDA DE DATOS DEL SISTEMA

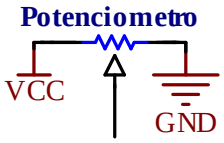
Según lo propuesto en el alcance y metodología del presente proyecto, la estructura del sistema de control es una hibridación entre componentes automotrices reales y elementos de simulación a menor escala compatibles con los microcontroladores Arduino, como lo detallado en la tabla 4.2 y 4.3, ya que el principio de funcionamiento es el mismo al tener en cuenta parámetros de funcionamiento de los sistemas en la programación de los mismos.

Tabla 4.2 Elementos captadores y su relación con componentes automotrices

Sensor MH-RD para la detección de lluvia

Los sensores de lluvia automotrices al igual que el MH-RD detecta si existe lluvia en el ambiente
El sensor MH-RD pone en uso un conjunto de almohadillas sensibles a la humedad que varían su resistencia interna. / El sensor automotriz emplea un procedimiento opto eléctrico con diodos luminosos, fotodiodos y un prisma, que permiten reflejar rayos lumínicos, los cuales son transmitidos con máxima reflexión cuando la superficie del cuerpo está seca y a medida que existe humedad la capacidad de reflexión disminuye.
Fotorresistor LDR para la detección de luz exterior

El foto resistor LDR y sensor de luminosidad censan la intensidad de luz del ambiente, pero su principio de funcionamiento es diferente
El fotorresistor varía su resistencia en función de la cantidad de luz, a mayor luz menor resistencia y viceversa / El sensor automotriz emplea un procedimiento opto eléctrico que mediante un cristal antepuesto registra la cantidad de luz del ambiente y la intensidad de luz en la parte delantera del vehículo. Con capacidad de diferenciar la luz ambiental y artificial.
Pulsadores, interruptores y conmutadores de evaluación

En general, el comportamiento de estos componentes es similar en cuanto a la evaluación física y digital del estado de una entrada, su objetivo es cerrar un circuito o captar señales referenciales para ser transmitidas por la red CAN BUS, ya que en la actualidad están desapareciendo las conexiones directas. Por lo cual, en este proyecto toda señal se transmitió por la red, teniendo en cuenta que la mayoría de conmutadores son automotrices.

Potenciómetro para censar el ángulo de dirección



Un sensor de ángulo de dirección automotriz y un potenciómetro correctamente configurado permiten evaluar la posición de giro de un volante

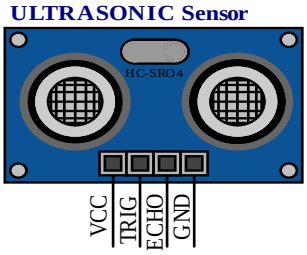
El potenciómetro aplica un contacto físico para generar divisiones de voltaje, y la ubicación de dicho contacto determina el ángulo de giro	El sensor automotriz a diferencia de un potenciómetro no cuenta con un contacto físico, ya que trabaja bajo el principio opto eléctrico por medio de dos anillos (absoluto y de incremento), que al mover la columna de dirección la luz interna de fotodiodos y fototransistores permite generar señales digitales al encontrar ventanas o ranuras características de los anillos.

Potenciómetro para ajustar la aceleración del motor

Un potenciómetro y un sensor de aceleración emplean el mismo principio de funcionamiento

El potenciómetro emplea una resistencia interna para evaluar la ubicación de un elemento que se encuentra constantemente en contacto físico con la pista resistiva	Los sensores automotrices emplean dos pistas resistivas que varían de manera opuesta su resistencia dependiendo de la ubicación del contacto físico, pero también existen sensores que emplea imanes para generar un campo magnético cuando la aleta del cuerpo de aceleración se mueve, permitiendo producir una señal eléctrica dependiendo de la diferencia de potencial.

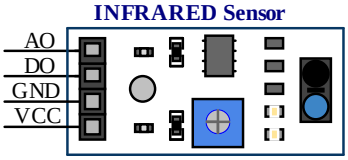
Sensor HC-SRO4 para detección de proximidad



El sensor HC-SRO4 emplean el mismo principio de funcionamiento de algunos sensores de proximidad automotrices para censar la distancia con objetos cercanos

El sensor HC-SRO4 emplea ondas ultrasónicas, un emisor emite los pulsos y se reflejan en el objeto hacia un receptor	Algunos sensores automtorices al igual que el sensor HC-SRO4 utilizan ondas ultrasónicas, pero también existen sensores electromagnéticos con un mayor rango de evaluación al emplear frecuencias electromagnéticas en el espectro.

Sensor óptico reflexivo

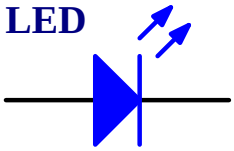
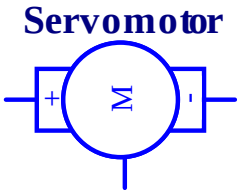
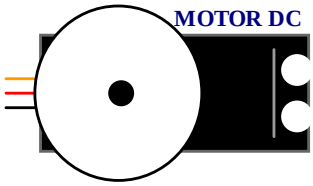
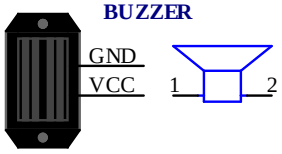


El sensor óptico reflexivo y sensores de presencia en el asiento monitorean si se encuentra un usuario en un área específica, pero estos últimos cuentan con más formas de evaluación

El sensor óptico emplea una luz infrarroja que se refleja en presencia de un objeto hacia un receptor, variado así el valor de una resistencia	Los sensores automotrices usa diferentes formas de evaluación a la del sensor óptico, como la detección infrarroja de la radiación del usuario, evaluación de peso en el asiento, ondas ultrasónicas similares a sensores de proximidad o fluctuaciones en el campo magnético de una detección por capacitancia.

Como se evidencia en la tabla 4.2, los componentes del sistema de red tienen el mismo objetivo de los elementos automotrices con los que se relacionan. No obstante, en la mayoría de los casos su principio de operación es diferente para transformar un evento físico en una señal eléctrica, pero por sus características se obtuvo las mediciones necesarias para simular el funcionamiento de subsistemas planteados, por medio de actuadores controlados por órdenes lógicas.

Tabla 4.3 Elementos actuadores y su relacion con compontentes automotrices

Leds para emitir luz en aplicaciones como ilimunación y señalización  Los leds y faros automotrices tienen la función de emitir luz, por lo cual su aplicabilidad fue similar teniendo en cuenta los parámetros de funcionamiento del sistema de luces. Sin embargo, su intensidad y rangos de operación son diferentes, además de que se puede encontrar varias prestaciones de faros como: halógenos, xenón o leds.
Servomotor MG90S para controlar el funcionamieto del limpiaparabrisas  El servomotor MG90S cuenta con un rango de movimiento de 0 a 180 grados, lo cual permitió simular el comportamiento de barrido de brazos en contacto con el parabrisas, ya que dicho mecanismo suele trabajar en un rango de 0 a 90 o 120 grados, pero con un motor DC de baja potencia sin limitaciones de movimiento y ajuste controlado.
Servomotor MG996R para el control de tracción del eje posterior del vehículo  El servomotor de rotación continua se relaciona con un motor DC, estos transforman la energía eléctrica en trabajo mecánico por acción de un campo magnético pero sin limitaciones de movimiento, siendo el componente que se acopló al eje posterior del vehículo, los cuales por sus prestaciones y control electrónico de velocidad son más utilizados que los motores AC.
Buzzer de alerta acustica  Al igual que el claxon automotriz, el Buzzer reaccionan al flujo de corriente para generar sonidos en referencia a alertas acústicas por medio de vibraciones. Los cuales pueden aumentar su complejidad con secuencias de frecuencias para emitir melodías

4.3.2. CONFIGURACIÓN DE LOS DIAGRAMAS DE CONEXIÓN

En el desarrollo de los esquemáticos del sistema de control, la configuración de conexión de los elementos captadores y actuadores generó seguridad en cuanto a interferencias y falsos contactos en la evaluación de señales de importancia, problemas como cortocircuitos e ineficiencia energética por parte del suministro de energía al sobrecargar el sistema.

Conexión de pulsadores, interruptores y conmutadores

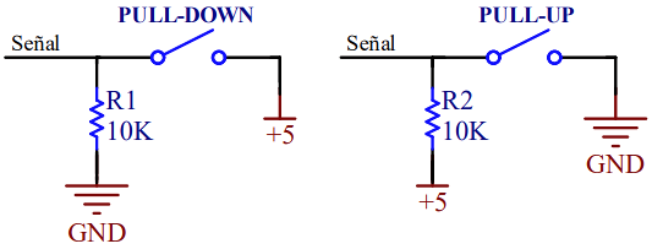


Figura 4.11 Configuración de resistencias Pull Up y Pull Down

Fuente: Autor

La configuración de conexión de la mayoría de elementos captadores del sistema de control fue la PULL DOWN, ya que su estructura emplea una resistencia entre la toma de señal y la referencia a masa o tierra para evaluar su estado lógico, cuando existe una entrada activa dicho estado cambia de dominante (0) a recesivo (1). Por lo cual, al contar con una línea común GND, la configuración de resistencias, normalmente de 10 kΩ, se realizó de manera más accesible, además de generar una protección mutua entre los elementos y el microcontrolador. Sin embargo, en ciertas ocasiones es conveniente utilizar la configuración PULL UP para evaluar un estado lógico dominante (0) cuando no existe una entrada activa, por lo que sus esquemas de conexión son opuestos (figura 4.11).

Conexión de diodos emisores de luz

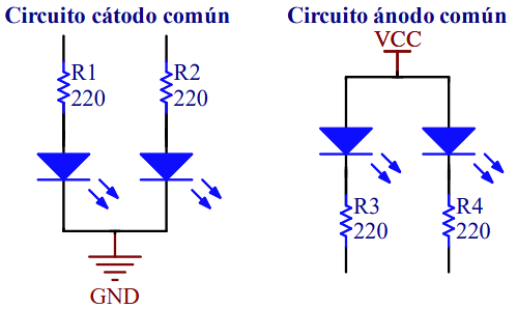


Figura 4.12 Configuración de circuito en cátado y ánodo común

Fuente: Autor

Teniendo en cuenta la disposición esquemática del sistema de luces en los vehículos, se implementó la configuración ánodo y cátado común (figura 4.12), según sea necesario, permitiendo activar múltiples leds de manera segura con resistencias independientes

conectadas en paralelo a la línea GND, mismas que determinan la intensidad de luz que se emite por su principio de oposición al flujo de corriente. En el caso del presente proyecto se emplearon resistencias de 220 Ω y 330 Ω, siendo valores recomendados para la seguridad del led, pero si se requiere una intensidad menor se puede implementar resistencias mayores o salidas analógicas.

Conexión de captadores y actuadores que no requieren alimentacion externa

La correcta alimentación de las placas integradas, en relación con los componentes que se implementaran, se refleja en el buen funcionamiento de los mismos. Por lo cual, se calculó y consultó los consumos de corriente de los componentes a ser implementados en el sistema propuesto con la siguiente formula:

$$I = V/R \qquad [4.2]$$

Donde:

I: Intensidad de corriente medida en Amperios (A)

V: Tensión o diferencia de potencial medido en Voltaje (V)

R: Resistencia eléctrica medida en Ohmios (Ω)

Teniendo en cuenta que se maneja una tensión estable de 5 Voltios, se determinó el valor de consumo de corriente de los captadores digitales y analógicos planteados en la tabla 3.10 empleando la formula [4.2], los cuales cuentan con configuraciones de resistencias en Pull Up y Pull Down según lo estructurado en los diagramas de conexión del apartado de diseño del circuito electrónico de la red.

Mediante la aplicación de la fórmula con una tensión de 5 Voltios y 10K Ohmios:

$$I = \frac{5V}{10000\Omega}$$

$$I = 0.0005A$$

Se obtiene un valor de consumo de corriente de 0.5 mA, el cual está por debajo de la capacidad máxima que puede suministrar el microcontrolador Arduino a través de sus pines digitales y analógicos. En este sentido, mediante la indagación de valores referenciales de consumo de corriente de sensores (tabla 4.4), estos no exceden la corriente que puede suministrar el microcontrolador mediante los pines VCC con una alimentación estable, por lo cual su conexión es factible.

Tabla 4.4 Consumos de corriente generales de los sensores del sistema

Componente	Alimentación de funcinamiento	
	Tensión (VCC)	Corriente
Sensor de lluvia MH-RD	3.3 V – 5 V	15 mA
Sensor LDR – Fotorresistor	5 V	0.5 mA
Sensor ultrasonico HC-SR04	5 V	15 mA
Sensor óptico reflectivo	3.3 V - 5 V	60 mA

De igual manera, la alimentación y funcionamiento de componentes actuadores como los diodos emisores de luz que normalmente trabajan en un rango de 15 mA al acoplar resistencias de alrededor de 220 ohmios en una alimentación estable de 5 Voltios, ya que estos generan caídas de tensión de 1.8 a 2 Voltios durante su operación. Además, en el caso de los Buzzers su consumo es de aproximadamente 30 mA, pero añadiendo resistencias este valor se reduce al igual que su intensidad acústica.

Tabla 4.5 Consumo de corriente de los componentes alimentados con Arduino

Módulo	Consumo de corriente		
	Captadores	Actuadores	Total
1	17 mA	130 mA	147 mA
2	3,5 mA	45 mA	48,5 mA
3	17,5 mA	105 mA	122,5 mA
4	62 mA	60 mA	122 mA

Fuente: Autor

Teniendo en cuenta los valores de tensión y corriente necesaria en cada módulo (tabla 4.5), para los componentes implementados, sin contar los de suministro de energía externa, la alimentación de las placas se efectuó por medio de un adaptador de corriente (AC/DC) de 9 Voltios y 2 Amperios, ya que proporciona valores de tensión estables en cada módulo, como también la corriente necesaria para la operación conjunta de los componentes conectados a lo 4 módulos con un total de 440 mA, que puede subir dependiendo de las situaciones.

Al alimentar los microcontroladores con una fuente estable, estos pueden proporcionar 20 mA en sus pines digitales, PWM y analógicos, y como máximo 40 mA, a excepción de los VCC, que dependen del regulador y de la capacidad de corriente con la cual se suministra al microcontrolador. Con respecto a lo anterior, alimentar las placas con una tensión por debajo de 7 Voltios en el Jack externo, genera que el regulador interno trabaje ineficientemente, lo cual se denota en salidas de tensión por debajo de los 5V en los pines VCC, como también variaciones en las salidas y entradas de datos, por tal motivo, se agregó el adaptador mencionado.

Alimentación de componentes con una fuente externa

Los componentes que fueron alimentados por una fuente externa se detallan en la tabla 4.6 con sus valores nominales de tensión y consumo de corriente aproximados, en donde se evidencia que a excepción de los diodos de luz blanca, los componentes requieren de una alimentación de 5 Voltios estables.

Tabla 4.6 Componentes alimentados por una fuente externa

Descripción	Tensión	Consumo de corriente
Servomotor MG90S	4.8 V a 6.0 V	100 – 250 mA
Led de luz roja de 8 smd	5 V	15 – 30 mA
Servomotor MG995	4.8 V a 7.2 V	500 – 900 mA
Led de luz blanca	9 V	20 mA

Los microcontroladores Arduino pueden siministran la tensión requerida, pero no la corriente de operación de los actuadores en conjunto con al menos 700 mA, la cual puede aumentar por las cargas. Razón por la cual, se implementó la fuente de poder conmutada TPS-0505, con capacidad de suministrar una alimentación de 5 Voltios y corriente de hasta 5 amperios, permitiendo cubrir los requerimientos de los componentes de manera estable al relacionar la referencia GND de la fuente y las placas integradas Arduino.

En el diseño de los circuitos electrónicos se mencionó que en el caso de implementar alimentación externa era necesario añadir componentes adicionales, no obstante, en los servomotores la configuración detallada no fue necesaria, ya que internamente cuentan con un circuito integrado que permite controlar el movimiento del motor con la librería Servo.h, pero sí se añadió condensadores de 10 μf en paralelo a la alimentación para reducir interferencias, picos de corriente y mejorar el rendimiento.

En el caso de los diodos emisores de luz se empleó transistores tipo NPN (unión bipolar) que actúan como interruptores para conectar la referencia GND de los leds con la línea GND común de la fuente y los microcontroladores, de manera que la señal digital generada por Arduino permita el cierre del circuito, dando como resultado la integración del sistema general de red expuesto en la figura 4.13, con su estructura de conexión codificada en colores, y su esquemático en el anexo VI (figura AVI.1). En donde se involucran componentes automotrices y compatibles a Arduino, a los cuales se les implementó las configuraciones mencionadas anteriormente para su correcto funcionamiento.

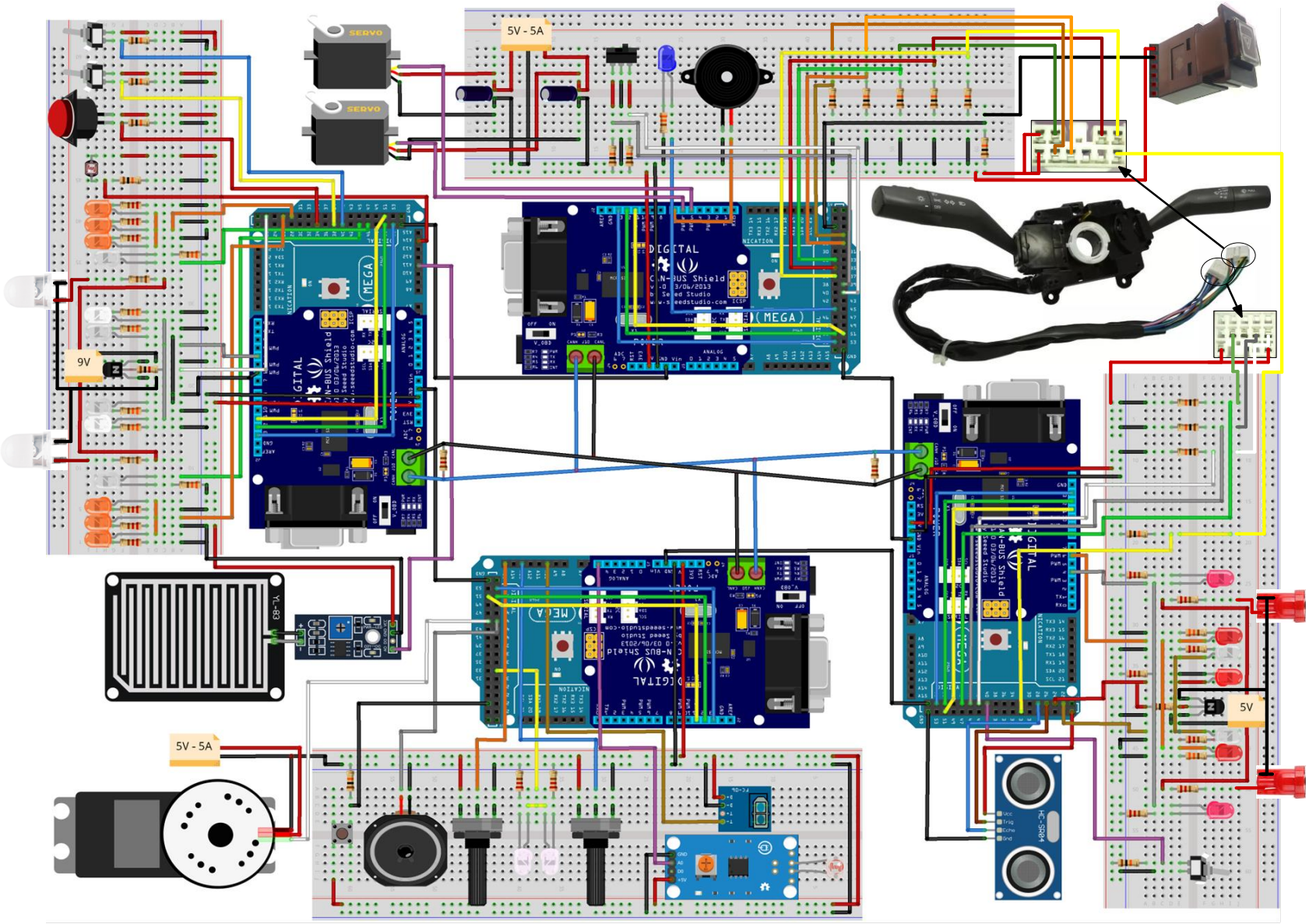


Figura 4.13 Enlace del sistema general de la red con 4 módulos CAN

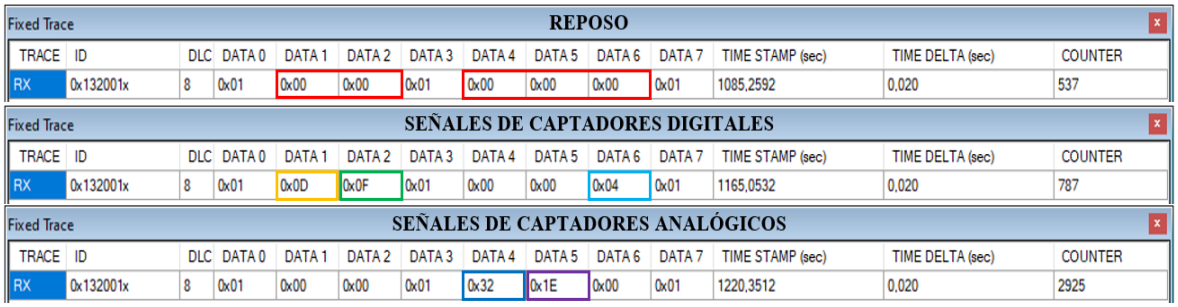
Fuente: Autor

4.3.3. CONFIGURACIÓN DEL SISTEMA DE CONTROL DE LA RED

Según lo propuesto, la configuración de los elementos captadores y actuadores del sistema se encuentra distribuida de tal manera que, las señales referenciales de activación de los subsistemas sean transmitidas cuando sea necesario, de acuerdo a la asignación de posiciones de bytes resumidas en el anexo [5], con la característica común de tener un estado dominante (0) en reposo o señal desactivada, y respetando los periodos de transmisión establecidos en cada módulo.

4.3.3.1. Evaluación de subsistemas del módulo 1

En la figura 4.14, se evidencia la transmisión de las señales de activación de los diferentes subsistemas del módulo 1 teniendo en cuenta la naturaleza del componente captador, en donde al dejar de tener una entrada activa se envía un mensaje adicional que confirma el estado dominante común de desactivación o reposo.



Fixed Trace — **REPOSO**

TRACE	ID	DLC	DATA 0	DATA 1	DATA 2	DATA 3	DATA 4	DATA 5	DATA 6	DATA 7	TIME STAMP (sec)	TIME DELTA (sec)	COUNTER
RX	0x132001x	8	0x01	0x00	0x00	0x01	0x00	0x00	0x00	0x01	1085,2592	0,020	537

Fixed Trace — **SEÑALES DE CAPTADORES DIGITALES**

TRACE	ID	DLC	DATA 0	DATA 1	DATA 2	DATA 3	DATA 4	DATA 5	DATA 6	DATA 7	TIME STAMP (sec)	TIME DELTA (sec)	COUNTER
RX	0x132001x	8	0x01	0x0D	0x0F	0x01	0x00	0x00	0x04	0x01	1165,0532	0,020	787

Fixed Trace — **SEÑALES DE CAPTADORES ANALÓGICOS**

TRACE	ID	DLC	DATA 0	DATA 1	DATA 2	DATA 3	DATA 4	DATA 5	DATA 6	DATA 7	TIME STAMP (sec)	TIME DELTA (sec)	COUNTER
RX	0x132001x	8	0x01	0x00	0x00	0x01	0x32	0x1E	0x00	0x01	1220,3512	0,020	2925

Figura 4.14 Estructura de la trama de datos del módulo 1

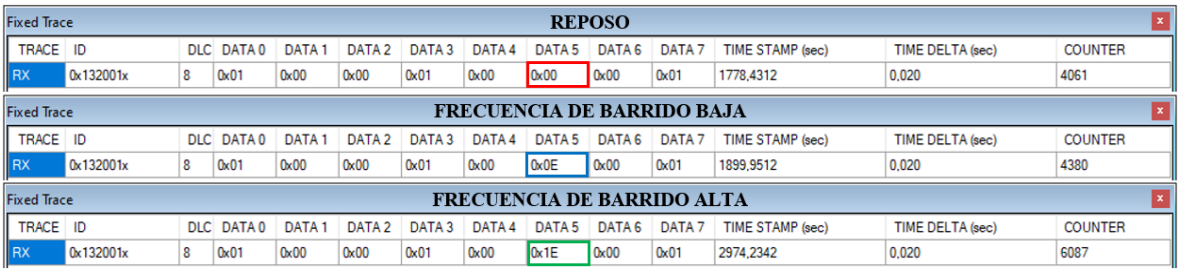
Fuente: Autor

Se estableció que el módulo con identificador 0X132001 envíe mensajes con un periodo de transmisión de 20 ms cuando alguna de sus entradas se encuentra activa:

- Byte [0]. – Sin función
- Byte [1]. – Interruptor, luces antiniebla delanteras
- Byte [2]. – Interruptor, luces antiniebla posteriores
- Byte [3]. – Sin función
- Byte [4]. – Luces adaptativas, fotorresistor LDR
- Byte [5]. – Control automático de limpiaparabrisas, sensor de lluvia MH-RD
- Byte [6]. – Pulsador, luces de freno
- Byte [7]. – Sin función

La evaluación y transmisión de la señal de activación de entradas digitales adoptan un estado y valor estable, pero en el caso de captadores que generan señales analógicas se maneja 255

niveles de voltaje tras el mapeo de los 1023 niveles originales, por lo que, para evitar problemas en la validación de parámetros en subsistemas, se optó por realizar cambios de variable manejando los rangos de niveles de voltaje. De manera que, al evaluar una luminosidad ≥ 200 se inicialice la variable Flagshot = 1 para transmitir en el byte [4] el valor común de 0x50 decimal y 0x32 hexadecimal.



Fixed Trace — REPOSO

TRACE	ID	DLC	DATA 0	DATA 1	DATA 2	DATA 3	DATA 4	DATA 5	DATA 6	DATA 7	TIME STAMP (sec)	TIME DELTA (sec)	COUNTER
RX	0x132001x	8	0x01	0x00	0x00	0x01	0x00	0x00	0x00	0x01	1778,4312	0,020	4061

Fixed Trace — FRECUENCIA DE BARRIDO BAJA

TRACE	ID	DLC	DATA 0	DATA 1	DATA 2	DATA 3	DATA 4	DATA 5	DATA 6	DATA 7	TIME STAMP (sec)	TIME DELTA (sec)	COUNTER
RX	0x132001x	8	0x01	0x00	0x00	0x01	0x00	0x0E	0x00	0x01	1899,9512	0,020	4380

Fixed Trace — FRECUENCIA DE BARRIDO ALTA

TRACE	ID	DLC	DATA 0	DATA 1	DATA 2	DATA 3	DATA 4	DATA 5	DATA 6	DATA 7	TIME STAMP (sec)	TIME DELTA (sec)	COUNTER
RX	0x132001x	8	0x01	0x00	0x00	0x01	0x00	0x1E	0x00	0x01	2974,2342	0,020	6087

Figura 4.15 Señales condicionales de evaluación del sensor de lluvia

Fuente: Autor

De igual manera, en el byte [5] se efectuó un mapeo de la señal generada por el sensor de lluvia MH-RD en 10 niveles (figura 4.15), adoptando un estado dominante (0) con un nivel ≤ 1, intensidad de lluvia baja en el rango ≥ 2 y ≤ 7 con un valor de 0x14 decimal y 0x0E hexadecimal e intensidad de lluvia alta en el rango ≥ 8 y ≤ 10 con un valor de 0x30 decimal y 0x1E hexadecimal.

4.3.3.2. Evaluación de subsistemas del módulo 2

En el caso del módulo con identificador 0x200113, el periodo de transmisión que se estableció fue de 40 ms, ya que solamente se centra en la evaluación de captadores que generan señales digitales (figura 4.16), los cuales conservan la característica de transmitir un estado dominante (0) cuando no cuentan con una estrada activa.

Fixed Trace — REPOSO

TRACE	ID	DLC	DATA 0	DATA 1	DATA 2	DATA 3	DATA 4	DATA 5	DATA 6	DATA 7	TIME STAMP (sec)	TIME DELTA (sec)	COUNTER
RX	0x200113x	8	0x02	0x00	0x02	0x00	0x00	0x00	0x00	0x00	1243,3092	0,040	1572

Fixed Trace — SEÑALES DE CAPTADORES DIGITALES

TRACE	ID	DLC	DATA 0	DATA 1	DATA 2	DATA 3	DATA 4	DATA 5	DATA 6	DATA 7	TIME STAMP (sec)	TIME DELTA (sec)	COUNTER
RX	0x200113x	8	0x02	0x07	0x02	0x0F	0x1D	0x06	0x08	0x0A	1579,4132	0,040	3445

Figura 4.16 Estructura de la trama de datos del módulo 2

Fuente: Autor

- Byte [0]. – Sin función
- Byte [1]. – Conmutador, estado de conducción
- Byte [2]. – Sin función
- Byte [3]. – Interruptor, luces de emergencia
- Byte [4]. – Conmutador, intensión de giro con luces direccionales

- Byte [5]. – Conmutador, control luces de posición
- Byte [6]. – Conmutador, control luces de cruce
- Byte [7]. – Conmutador, control luces de carretera

Tal y como se aprecia en la figura 4.17, en el byte [1] se transmitió 3 valores de evaluaciones digitales, las cuales indican la activación de funciones que no trabajan al mismo tiempo como el estado de conducción, teniendo una posición de reposo (0), avance 0x07 y retroceso 0x03, por lo cual no fue necesario emplear otro byte.

Fixed Trace — **REPOSO - NEUTRAL**

TRACE	ID	DLC	DATA 0	DATA 1	DATA 2	DATA 3	DATA 4	DATA 5	DATA 6	DATA 7	TIME STAMP (sec)	TIME DELTA (sec)	COUNTER
RX	0x200113x	8	0x02	0x00	0x02	0x00	0x00	0x00	0x00	0x00	2061,6132	0,040	4901

Fixed Trace — **AVANCE - DRIVE**

TRACE	ID	DLC	DATA 0	DATA 1	DATA 2	DATA 3	DATA 4	DATA 5	DATA 6	DATA 7	TIME STAMP (sec)	TIME DELTA (sec)	COUNTER
RX	0x200113x	8	0x02	0x07	0x02	0x00	0x00	0x00	0x00	0x00	2147,4392	0,040	5003

Fixed Trace — **RETROCESO**

TRACE	ID	DLC	DATA 0	DATA 1	DATA 2	DATA 3	DATA 4	DATA 5	DATA 6	DATA 7	TIME STAMP (sec)	TIME DELTA (sec)	COUNTER
RX	0x200113x	8	0x02	0x03	0x02	0x00	0x00	0x00	0x00	0x00	2192,1142	0,040	6105

Figura 4.17 Señales de conmutación de estado de conducción

Fuente: Autor

De igual manera, el byte [4] se aprovecha para transmitir 3 estados de la conmutación de las luces direccionales (figura 4.18), indicando la posición de reposo común ya mencionado, y las intenciones de giro hacia la izquierda (27 decimal, 1B hexadecimal) o derecha (29 decimal, 1D hexadecimal).

Fixed Trace — **REPOSO**

TRACE	ID	DLC	DATA 0	DATA 1	DATA 2	DATA 3	DATA 4	DATA 5	DATA 6	DATA 7	TIME STAMP (sec)	TIME DELTA (sec)	COUNTER
RX	0x200113x	8	0x02	0x00	0x02	0x00	0x00	0x00	0x00	0x00	3349,5182	0,040	6377

Fixed Trace — **INTENCIÓN DE GIRO (IZQUIERDA)**

TRACE	ID	DLC	DATA 0	DATA 1	DATA 2	DATA 3	DATA 4	DATA 5	DATA 6	DATA 7	TIME STAMP (sec)	TIME DELTA (sec)	COUNTER
RX	0x200113x	8	0x02	0x00	0x02	0x00	0x1B	0x00	0x00	0x00	3426,5052	0,040	6609

Fixed Trace — **INTENCIÓN DE GIRO (DERECHA)**

TRACE	ID	DLC	DATA 0	DATA 1	DATA 2	DATA 3	DATA 4	DATA 5	DATA 6	DATA 7	TIME STAMP (sec)	TIME DELTA (sec)	COUNTER
RX	0x200113x	8	0x02	0x00	0x02	0x00	0x1D	0x00	0x00	0x00	3472,0992	0,040	7242

Figura 4.18 Señales de conmutación de luces direccionales

Fuente: Autor

4.3.3.3. Evaluación de subsistemas del módulo 3

La estructura de señales activas del módulo 3 se encuentra representada en la figura 4.19, las cuales son mayormente digitales. Este módulo, al igual que el anterior, se lo configuró con un periodo de transmisión de 40 ms, pero con el identificador 0x302001, en donde se aprecia que el byte [0] se le asignó la referencia del pulsador del claxon adoptando un valor de 0x05, siendo la única entrada que emplea esa posición en los 4 módulos.

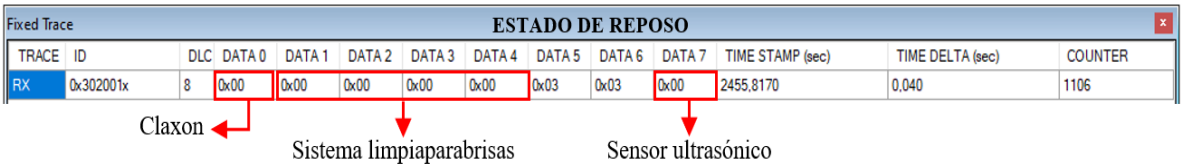


Figura 4.19 Estructura de la trama de datos del módulo 3

Fuente: Autor

- Byte [0]. – Pulsador, claxon
- Byte [1]. – Interruptor, activación de limpiaparabrisas automático
- Byte [2]. – Conmutador, control de rocío de líquido (posición de Pull)
- Byte [3]. – Conmutador, control de velocidad baja de barrido
- Byte [4]. – Conmutador, control de velocidad alta de barrido
- Byte [5]. – Sin función
- Byte [6]. – Sin función
- Byte [7]. – Sensor ultrasónico, medición de proximidad

Como se mencionó en el apartado del diseño del sistema embebido, la evaluación para el funcionamiento del sistema limpiaparabrisas se llevó a cabo a través de la palanca conmutadora y un switch (figura 4.20), de tal manera que la posición de rocío (26 decimal, 1A hexadecimal) pueda ser activada en conjunto con la posición MIST (barrido único), señal de funcionalidad automática (23 decimal, 17 hexadecimal), velocidad baja (45 decimal, 2D hexadecimal) o alta (46 decimal, 2E hexadecimal). En donde, la posición MIST y LO adoptan el mismo valor en el byte [3] por tener continuidad en entrada.

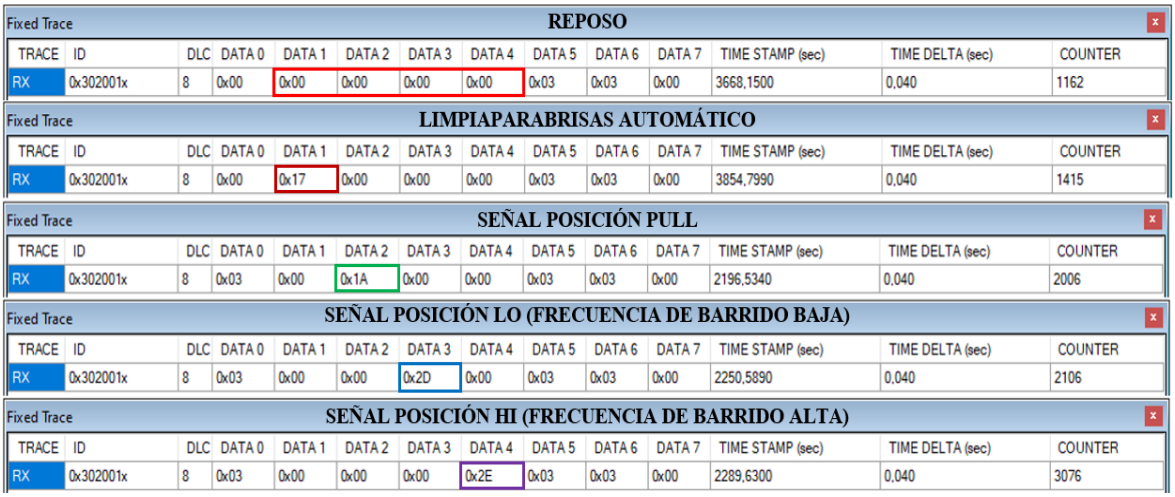


Figura 4.20 Señales de conmutación de sistema limpiaparabrisas

Fuente: Autor

Por otra parte, en el byte [7] se transmitió la evaluación de la distancia evaluada en centímetros por el sensor ultrasónico (figura 4.21), ya que el principio de funcionamiento del mismo no genera señales digitales o analógicas. Por lo cual, al no tener un rango o mapeo

en la evaluación se incorporó un condicional que permite dar paso a la transmisión en tiempo real de valores ≤ 100, caso contrario se adopta un estado de reposo dominante (0).

Fixed Trace — REPOSO (FUERA DE RANGO)

TRACE	ID	DLC	DATA 0	DATA 1	DATA 2	DATA 3	DATA 4	DATA 5	DATA 6	DATA 7	TIME STAMP (sec)	TIME DELTA (sec)	COUNTER
RX	0x302001x	8	0x03	0x00	0x00	0x00	0x00	0x03	0x03	0x00	2644,5630	0,040	4013

Fixed Trace — MEDICIÓN DE CERCANÍA

TRACE	ID	DLC	DATA 0	DATA 1	DATA 2	DATA 3	DATA 4	DATA 5	DATA 6	DATA 7	TIME STAMP (sec)	TIME DELTA (sec)	COUNTER
RX	0x302001x	8	0x03	0x00	0x00	0x00	0x00	0x03	0x03	0x0C	2691,2370	0,040	4210

Figura 4.21 Señal de evaluación de proximidad

Fuente: Autor

4.3.3.4. Evaluación de subsistemas del módulo 4

Por último, como se mencionó en la configuración de la funcionalidad general de la red, el módulo con identificador 0x432001 aborda subsistemas complementarios a los demás módulos seguridad pasiva y gestión del vehículo. Por lo cual, se utilizó características similares en cuanto a evaluación y control, teniendo el manejo de señales digitales y analógicas (figura 4.22).

Fixed Trace — ESTADO DE REPOSO

TRACE	ID	DLC	DATA 0	DATA 1	DATA 2	DATA 3	DATA 4	DATA 5	DATA 6	DATA 7	TIME STAMP (sec)	TIME DELTA (sec)	COUNTER
RX	0x432001x	8	0x04	0x00	0x00	0x00	0x04	0x00	0x00	0x04	1352,5773	0,010	2152

Fixed Trace — SEÑAL DE CAPTADOR DIGITAL

TRACE	ID	DLC	DATA 0	DATA 1	DATA 2	DATA 3	DATA 4	DATA 5	DATA 6	DATA 7	TIME STAMP (sec)	TIME DELTA (sec)	COUNTER
RX	0x432001x	8	0x04	0x00	0x0A	0x00	0x04	0x00	0x00	0x04	1069,0823	0,010	2561

Fixed Trace — SEÑALES DE CAPTADORES ANALÓGICOS

TRACE	ID	DLC	DATA 0	DATA 1	DATA 2	DATA 3	DATA 4	DATA 5	DATA 6	DATA 7	TIME STAMP (sec)	TIME DELTA (sec)	COUNTER
RX	0x432001x	8	0x04	0x0C	0x00	0x08	0x04	0xE3	0x30	0x04	1145,6563	0,010	8282

Figura 4.22 Estructura de la trama de datos del módulo 4

Fuente: Autor

- Byte [0]. – Sin función
- Byte [1]. – Luces automáticas, sensor LDR
- Byte [2]. – Posición de puerta de conductor, pulsador
- Byte [3]. – Detección de usuarios, sensor infrarrojo
- Byte [4]. – Sin función
- Byte [5]. – Potenciómetro, ángulo de dirección
- Byte [6]. – Potenciómetro, aceleración motor DC rotación continua
- Byte [7]. – Sin función

La evaluación de la única señal digital es por medio de un pulsador, el cual adopta un estado dominante (0) cuando se encuentra presionado, y un valor 0x10 decimal o 0x0A hexadecimal en su posición mecánica de reposo, indicando el cierre y apertura de la puerta

respectivamente. Por otra parte, al igual que en los módulos anteriores, en la evaluación y transmisión de captadores analógicos se realizó mapeo de niveles de tensión, el byte [1] adopta un valor de 0x0C (12 decimal) al examinar una luminosidad ≤ 150, en el byte [3] un valor de 0x08 al evaluar una reflexión ≥ 200, y en el caso de los potenciómetros se transmite las variaciones de los 255 niveles de voltaje previamente mapeado. En donde, se estableció que la señal de aceleración (byte [6]) de le asigne un valor diferente de 0x00 al comenzar la transmisión, y en el byte [5] rangos según el sentido de giro, al adoptar valores ≥ 215 cuando es a la izquierda y a la derecha con valores ≤ 185, siendo el rango intermedio una posición central aproximada.

4.3.4. MANEJO E INTERACCIÓN DE DATOS PARA LA PROGRAMACIÓN DEL SISTEMA

La información proporcionada por los diferentes módulos de la red CAN, genera la dependencia de condicionales para el desarrollo de sistemas embebidos, en donde se emplea compuertas lógicas (AND, OR, NOT, etc) para la configuración de parámetros relacionados con la operación de sistemas de control automotrices. Los cuales se organizaron en funciones (tabla 4.7), permitiendo relacionar datos según sea necesario.

Tabla 4.7 Interacción del manejo de datos en los sistemas de control

	Función	**ID**	**Byte**	**Relación de señal activa**
MÓDULO 1	Luces emergencia y direccionales	0x200113	[3]	Situación de emergencia, luces intermitentes del vehículo encendidas en parqueo
			[4]	Intención de giro de vehículo, selectividad de intermitentes (izquierda y derecha)
	Luces antiniebla delanteras	0x132001	[1]	Situación de baja luminosidad por neblina, mejorar la visibilidad de conductores
		0x432001	[5]	Selectividad de luminosidad de acuerdo al sentido de giro en intersecciones
			[6]	Condicional de aceleración del servomotor DC, como validación para selectividad de luces a baja velocidad
	Luces posición frontales	0x200113	[5]	Luces de posición frontales del vehículo que se activan en conjunto con las posteriores
	Luces cruce y carretera	0x132001	[4]	Conmutación automática, luces de carretera a luces de cruce tras la evaluación de LDR
		0x200113	[6]	Encender luces de cruce o bajas
			[7]	Encender luces de carretera y apagar luces de cruce con la señal activa del byte [6] y [7], pero con la evaluación de LDR por debajo del nivel establecido
		0x132001	[5]	Ajuste de frecuencia de barrido dependiendo del rango de lluvia al estar encendido el sistema automático

MÓDULO 2	Sistema limpiaparabrisas	0x302001	[1]	Condicional para activar el sistema automático de limpiaparabrisas
			[2]	Esparcimiento de agua o lavaparabrisas, que puede ser activada en conjunto con el sistema automático o con la conmutación manual
			[3]	Frecuencia de barrido de parabrisas baja
			[4]	Frecuencia de barrido de parabrisas alta
	Alerta acústica de proximidad		[7]	Frecuencia de activación de alerta acústica, a menor distancia mayor frecuencia acústica
MÓDULO 3	Sistema de advertencia a la colisión	0x200113	[1]	Condicional para activar la evaluación de proximidad, al encontrarse el estado de conducción en reversa
	Luces de retro	0x200113	[1]	Alerta visual activada en conjunto con el sistema de alerta a la colisión y tracción de motor DC en retro
	Luces posición posteriores		[5]	Luces de posición posteriores que se enciende de manera conjunta con las luces de posición frontales
	Luces de freno	0x132001	[6]	Luces de freno que se activan de manera independiente como una alerta visual de reducción de velocidad.
	Luz antiniebla posterior		[2]	Luces antiniebla se encienden para mejorar la visibilidad del vehículo a otros conductores
MÓDULO 4	Luces automáticas de la cabina	0x432001	[1]	Evaluación de luminosidad exterior, como parámetro para activar las luces automáticas de la cabina
			[2]	Estado de apertura o cierre de puerta, como condicional para la activación las luces automáticas
			[3]	Detección de presencia de usuario en el vehículo, lo cual permite relacionar la posición de la puerta y señal LDR para la activación embebida de las luces automáticas de la cabina
	Claxon	0x302001	[0]	Alerta acústica de frecuencia constante, que alerta de una situación a peatones y conductores
	Control de tracción eléctrica	0x200113	[1]	Selectividad de estado de conducción, como parámetro que condiciona el estado de reposo del motor y sentido de giro constante (avanzar o retroceder)
		0x432001	[6]	Aceleración del motor de rotación continua tras seleccionar el sentido de giro

Fuente: Autor

4.3.4.1. Modificación de librerías

Debido a la configuración implementada para el control de periodos de transmisión con el Timer 1, en los códigos propuestos se generó una incompatibilidad entre las librerías mcp_can2515.h y Servo.h, provocando errores en la comunicación CAN y control de los servomotores implementados. Por lo cual, se realizó una modificación en el archivo ServoTimers, localizado en la carpeta del paquete de librerías descargado en conjunto con el software Arduino IDE, con la siguiente dirección:

- ✓ C:\Users\JAVIERHUERA\AppData\Local\Arduino15\libraries\Servo\src\avr

La modificación realizada fue la eliminación de la dependencia del Timer 1 en la librería Servo.h (figura 4.23), para lo cual se empleó el software Dev-C++.

```
// Say which 16 bit timers can be used and in what order
#if defined(__AVR_ATmega1280__) || defined(__AVR_ATmega2560__)
#define _useTimer5
#define _useTimer3
#define _useTimer4
typedef enum { _timer5, _timer3, _timer4, _Nbr_16timers } timer16_Sequence_t;
```

Figura 4.23 Modificación de la librería Servo.h

Fuente: Autor

4.3.5. CONSTRUCCIÓN DEL PROTOTIPO DE SIMULACIÓN DE RED CAN

Conforme a lo establecido, el prototipo de red multiplexada CAN (figura 4.24), se implementó en un modelo de vehículo a escala 1/12 como se adjunta en el anexo VII, ya que el sistema general integra componentes captadores y actuadores que forman un sistema híbrido, permitiendo demostrar el funcionamiento lógico de la red a través de las herramientas CAN Bus Analyzer y osciloscopio digital. Además, de su operación física en tiempo real de los subsistemas implementados.

Figura 4.24 Prototipo de la red multiplexada CAN

Fuente: Autor

CAPÍTULO IV

5. CONCLUSIONES Y RECOMENDACIONES

5.1. CONCLUSIONES

- El diseño del sistema multiplexado se sustentó en una configuración de red CAN-BUS de 4 módulos, la cual emplea una topología Bus de dos canales (CAN HIGH y CAN LOW) y dos resistencias de 120Ω en los extremos de dos módulos, los cuales se conectaron en paralelo a las líneas y se constituyeron en base a Arduinos Mega 2560 y placas CAN Bus Shield, mismos que al ensamblar y puentear los pines SPI (SCK, MOSI y MISO) permiten tener una compatibilidad con la norma ISO 11898, ya que integran el controlador MCP 2515, transceptor TJA 1050 y microprocesador Atmega 2560.
- El principio de comunicación multidifusión, en el cual se basó la red, permitió establecer una comunicación bidireccional. En donde, los periodos de transmisión se controlaron por medio de un Timer, para generar estabilidad en la frecuencia de acceso al Bus de datos, counter de transmisión y operación del proceso de arbitraje cuando la red se sincroniza. Los cuales, se pueden configurar hasta 5 ms, ampliando la posibilidad de implementar sistemas con velocidades de reacción menor.
- Se comprobó que los módulos, con una configuración de 1 Mbps de velocidad y formato extendido de mensaje, reducen su velocidad a 500 Kbits/s, ya que la placa CAN Bus Shield opera a 8MHz, que es la mitad del Arduino. Generando, que la trama de 132 bits establecida en la comunicación unidireccional se transmita en 264 μs. Además de ofrecer una mayor flexibilidad para asignar identificadores (536.870,912 posibilidades), lo cual aumentó su capacidad de direccionamiento y compatibilidad con estándares industriales.
- La configuración híbrida del sistema de control fue una opción viable para la simulación de sistemas automotrices, ya que los componentes compatibles a Arduino y automotrices tienen el mismo propósito en la evaluación de eventos físicos y control, solo que, a menor escala y precisión, pero en algunos casos diferente operación. Además, debido a su estructura, se requirió de dos fuentes de alimentación, un adaptador 9V y 2A para los 4 módulos CAN y cubrir el consumo de corriente de los elementos captadores y actuadores de al menos 440 mA, y una fuente de 5V y 5A para los actuadores que superan la corriente que puede proporcionar Arduino.
- En la programación del sistema embebido, se aplicó el principio de comunicación por eventos, de manera que, con la inicialización de una bandera, cada módulo comprueba

si existe al menos una entrada activa en sus captadores para transmitir mensajes, lo cual evitó que el Bus de datos se sobrecargue con información innecesaria, y en la recepción máscaras y filtros, para aceptar solo mensajes de importancia y almacenarlos en Buffers únicos para evitar colisiones de datos. En donde, la información de cada byte de interés se manejó en variables globales, evitando así intermitencias en la ejecución de funciones embebidas en más de un subsistema de control

- Por último, se concluye que el prototipo de red multiplexada CAN con una configuración hibrida, presenta una lógica de software y hardware adecuada para demostrar el funcionamiento interno y externo de una red CAN a menor escala, ya que el control de sistemas embebidos automotrices depende de los parámetros establecidos en la programación. Siendo una herramienta didáctica de aprendizaje, en la que interactúa las características principales de una red CAN: velocidad, estabilidad y seguridad.

5.2. RECOMENDACIONES

- Como parte de trabajos futuros en la implementación de sistemas y subsistemas más complejos a la red como los sistemas: airbag y seguridad, asistencia al conductor (ADAS), confort, entre otros. Se aconseja evitar la implementación de funciones que bloquen la ejecución del código de programación, como lo es el uso de delay(), y evaluar el consumo de los periféricos a ser añadidos para evitar reinicios o el mal funcionamiento del sistema por la falta de corriente proporcionada por la fuente.
- Se recomienda realizar investigaciones para relacionar la red multiplexada CAN desarrollada en el presente trabajo, con otros protocolos de comunicación a través de un Gateway, ya que los vehículos internamente emplean múltiples redes y protocolos como: LIN, MOST, FlexRay, entre otros. Los cuales, tienen sus propias características de funcionamiento como la topología y velocidades, mismas que pueden variar dependiendo del subsistema o área funcional, siendo parte del sistema general.
- La evaluación de comunicación e interacción de datos entre los módulos de la red se la realiza por medio de las herramientas analizadoras como el caso del CAN Bus Analyzer y osciloscopio digital. Sin embargo, en ausencia de dichos dispositivos o similares, se dificulta el ejecutar comprobaciones de señales referenciales en la red. En este sentido, es posible desarrollar una interfaz por comunicación serial a un PC o una pantalla que permita imprimir el estado de la red y el control digital, como parte de una funcionalidad embebida del sistema, al igual que los dispositivos automotrices.
- Al momento de alimentar los microcontroladores Arduino por el Jack, se recomienda que el adaptador de corriente (AC/DC) proporcione una tensión mayor o igual a 7V, ya que una tensión por debajo de dicho umbral genera que el regulador interno trabaje ineficientemente, entregando en los pines de salida VCC tensiones por debajo de 5V.
- En el caso de implementar componentes que requieran alimentación externa para cubrir los requisitos o rangos de operación, es importante realizar una conexión común a GND entre el sistema general que controla el microcontrolador Arduino y la fuente para evitar problemas de funcionamiento.

6. BIBLIOGRAFÍA

Aljundi, L. (2023). *Get to know Arduino Libraries | Arduino Documentation |*. https://docs.arduino.cc/learn/starting-guide/software-libraries

Arduino. (n.d.). *Arduino Mega 2560 Rev3*. Retrieved February 24, 2023, from https://docs.arduino.cc/hardware/mega-2560

Arduino. (2023). *Arduino® MEGA 2560 Rev3*. https://docs.arduino.cc/resources/datasheets/A000067-datasheet.pdf

Ashtari, H. (2022). *Diagramas de topología de red y prácticas recomendadas de selección para 2022*. https://www.spiceworks.com/tech/networking/articles/what-is-network-topology/

AWERS. (n.d.). *TJA1050 Transceptor CAN de alta velocidad SMD*. Retrieved March 5, 2023, from https://tienda.sawers.com.bo/tja1050-transceptor-can-de-alta-velocidad

Banzi, M., & Shiloh, M. (2022). *Getting Started With Arduino, 4th Edition* (4th ed.). Marker Media Inc.

BAUBOR. (n.d.). *RESISTENCIA - 120 ohm - 1/4W* . Electronica y Telecomunicaciones. Retrieved March 3, 2023, from http://baubor.com/electronics/index.php?id_product=12&controller=product

Corrigan, S. (2002). *Introducción a la red de área del controlador (CAN)*. TEXAS INSTRUMENTS.

Domínguez, E., & Ferrer, J. (2012). *Circuitos eléctricos auxiliares del vehículo* (1st ed.). Editex, S.A. https://anyflip.com/iwkp/rzaq/basic

DynamoElectronics. (n.d.). *Cables protoboard o arduino*. Retrieved March 3, 2023, from https://dynamoelectronics.com/tienda/cable-montaje-protoboard-o-arduino-10cm/

Feliciano Fuentes, L. G. (2019). Identificación y control de parámetros de clúster de instrumentos automotriz mediante Red Can [Universidad Autónoma Metropolitana (México). Unidad Azcapotzalco.]. In *Exploraciones, intercambios y relaciones entre el diseño y la tecnología*. https://doi.org/10.16/CSS/JQUERY.DATATABLES.MIN.CSS

Figueroa Peñafiel, H. J. (2015). *Estudio y análisis del sistema multiplexado del vehículo híbrido toyota prius* [GUAYAQUIL / UIDE / 2015]. https://repositorio.uide.edu.ec/handle/37000/840

Galeano, G. (2009). *Programación de sistemas embebidos en C* (L. Buitrago (ed.)). Alfaomega Colombiana S.A.

García Osés, A. (2015). *Diseño de una red CAN bus con Arduino*. https://hdl.handle.net/2454/19115

Globaltech. (2019). *G-SCOPE-2 Osciloscopio automotriz de 2 canales 100MHz*. https://globaltech-car.com/wp-content/uploads/2019/06/gscope2.pdf

Gonzáles, V. (2019). *Diagnosis CAN BUS, Paso a Paso*. Diagnosis Tips. https://www.diagnosistips.com/can-bus/

Guerra Carmenate, J. (2019). *Arduino Mega 2560 el hermano mayor de Arduino UNO*.

Programarfacil. https://programarfacil.com/blog/arduino-blog/arduino-mega-2560/

Ingecom. (n.d.). *Cable Multifilar Flexiplus AWM – Calibre 18 AWG*. Retrieved June 5, 2023, from https://ingecomsas.com/producto/cable-multifilar-flexiplus-awm-calibre-18-awg-centelsa-105c-600v/

Ingeniería y mecánica automotriz. (2020). *¿Qué es el sistema de iliminación automotriz y cómo se clasifican?* https://www.ingenieriaymecanicaautomotriz.com/que-es-el-sistema-de-iluminacion-automotriz-y-como-se-clasifica/

Instituto Ecuatoriano de Normalización. (2009). *NORMA TÉCNICA ECUATORIANA NTE INEN 1155:2009 VEHÍCULOS AUTOMOTORES. DISPOSITIVOS PARA MANTENER O MEJORAR LA VISIBILIDAD Segunda revisión*. https://ia802905.us.archive.org/0/items/ec.nte.1155.2009/ec.nte.1155.2009.pdf

Jaque, R. I. V. (2015). *Multiplexado*. https://www.academia.edu/16544990/Multiplexado

Johansson, K. H., Törngren, M., & Nielsen, L. (2005). Vehicle Applications of Controller Area Network. In *Handbook of Networked and Embedded Control Systems* (pp. 741–765). Birkhäuser Boston. https://doi.org/10.1007/0-8176-4404-0_32

Lara Rivero, A. Á. (2014). De sistema mecánico a sistema tecnológico complejo El caso de los automóviles. *Contaduría y Administración*, *59*(2), 11–39. https://doi.org/10.1016/S0186-1042(14)71253-7

Llamas, L. (2016). *El bus SPI en Arduino*. https://www.luisllamas.es/arduino-spi/

Llanos López, M. J. (2017). *Circuitos eléctricos auxiliares del vehículo 2.ª edición* (2nd ed.). Paraninfo.

Llanos López, M. J. (2022). *Circuitos eléctricos auxiliares del vehículo 3.ª edición 2022* (3rd ed.). Paraninfo.

López Diguay, J. A. (2021). *Estudio y análisis del sistema multiplexado de un vehículo Audi Q5* [Escuela Superior Politécnica de Chimborazo]. http://dspace.espoch.edu.ec/handle/123456789/16005

Manosalvas Salazar, C. A. (2017). *Diseño de un sistema embebido para el control de ingreso y salida de vehículos a través de internet, en el acceso principal de la ESPOCH* [Escuela Superior Politécnica de Chimborazo]. http://dspace.espoch.edu.ec/handle/123456789/7621

Marotronics. (n.d.). *MCP2515 Can Bus controlador shield V3.0 SunFlower para Arduino*. Retrieved March 2, 2023, from https://www.marotronics.de/MCP2515-Can-Bus-controller-shield-V30-SunFlower-fuer-Arduino

Martínez-Cruz, A., Ramírez-Gutiérrez, K. A., Feregrino-Uribe, C., & Morales-Reyes, A. (2021). Security on in-vehicle communication protocols: Issues, challenges, and future research directions. *Computer Communications*, *180*, 1–20. https://doi.org/10.1016/j.comcom.2021.08.027

Martinez-Santos, J. C., Acevedo-Patino, O., & Contreras-Ortiz, S. H. (2017). Influence of Arduino on the Development of Advanced Microcontrollers Courses. *IEEE Revista Iberoamericana de Tecnologias Del Aprendizaje*, *12*(4), 208–217. https://doi.org/10.1109/RITA.2017.2776444

Martínez Requena, A. (2017). *Introducción a CAN bus: Descripción, ejemplos y aplicaciones de tiempo real* [Universidad Politécnica de Madrid]. https://msde.etsisi.upm.es/wp-content/uploads/2017/09/Memoria_TFM_Adrian_Martinez_Requena.pdf

Mejía Morales, P. R., Poma Montaño, J. L., & Ramón Pineda, J. L. (2013). *Diseño y construcción una interfaz didáctica de redes y multiplexado CAN para aplicaciones en el automóvil* [Universidad Politécnica Salesiana]. https://dspace.ups.edu.ec/handle/123456789/5144

Microchip. (2014). *ATmega640/V-1280/V-1281/V-2560/V-2561/V*. https://ww1.microchip.com/downloads/aemDocuments/documents/OTH/ProductDocuments/DataSheets/ATmega640-1280-1281-2560-2561-Datasheet-DS40002211A.pdf

Microchip. (2021). *MCP2515*. 96. https://ww1.microchip.com/downloads/aemDocuments/documents/APID/ProductDocuments/DataSheets/MCP2515-Family-Data-Sheet-DS20001801K.pdf

Microchip. (2022). *CAN Bus Analyzer User's Guide*. 24. https://ww1.microchip.com/downloads/en/DeviceDoc/51848B.pdf

Navet, N., & Simonot-Lion, F. (2017). *Automotive Embedded Systems Handbook* (N. Navet & F. Simonot-Lion (eds.)). CRC Press. https://doi.org/10.1201/9780849380273

NXP. (2022). *High-Speed CAN Transceiver TJA1050*. https://www.nxp.com/assets/block-diagram/en/TJA1050.pdf

Pan, T., & Zhu, Y. (2018). Getting Started with Arduino. In *Designing Embedded Systems with Arduino* (4th ed., pp. 3–16). Springer Singapore. https://doi.org/10.1007/978-981-10-4418-2_1

Paret, D. (2007). Multiplexed Networks for Embedded Systems. In *Multiplexed Networks for Embedded Systems*. Wiley. https://doi.org/10.1002/9780470511770

Pérez Darquea, D. G. (2017). Evolution of Electronic Devices in an Automobile. *INNOVA Research Journal*, *3*(2), 1–7. https://dialnet.unirioja.es/servlet/articulo?codigo=6340313

Pérez, J. M. (2014). *Circuitos Eléctricos Auxiliares Del Vehículo* (C. C. Lara (ed.); 1st ed.). Paraninfo.

Puga Gutiérrez, A. M., & Morales Recalde, J. S. (2022). *Desarrollo de un dispositivo basado en microcontrolador Arduino para la adquisición de datos a través de la red CAN para conocer las condiciones que suscitan accidentes vehiculares* [Tesis de pregrado, Universidad Técnica del Norte]. http://repositorio.utn.edu.ec/handle/123456789/12304

Reif, K. (2015). *Automotive Mechatronics* (K. Reif (ed.)). Springer Fachmedien Wiesbaden. https://doi.org/10.1007/978-3-658-03975-2

Rodríguez Rodríguez, A., Vento Álvarez, J. R., & Inouye Rodriguez, R. (2018). Implementation of an OBD-II diagnostics tool over CAN-BUS with Arduino. *Sistemas y Telemática*, *16*(45). https://doi.org/10.18046/SYT.V16I45.2747

Ros Marin, J. A., & Barrera Doblado, O. (2011). *Sistemas electricos y de seguridad y confortabilidad*. Paraninfo, S.A.

Sánchez, E. F. (2012). *Circuitos eléctricos auxiliares del vehículo* (1st ed.). Macmillan Profesional.

Sánchez, L., Molano, M., Fabela, M. de J., Madrid, M., Hernández, J., Vázquez, D., & Flores, O. (2016). *Revisión documental del protocolo CAN como herramienta de comunicación y aplicación en vehículos.* http://imt.mx/archivos/Publicaciones/PublicacionTecnica/pt474.pdf

Sangiovanni-Vincentelli, A., & Di Natale, M. (2007). Embedded system design for automotive applications. *Computer*, *40*(10), 42–51. https://doi.org/10.1109/MC.2007.344

Secretaría Nacional de Planificación. (2021). *Plan de Creación de Oportunidades 2021-2025*. Gobierno de la república del Ecuador. https://www.planificacion.gob.ec/plan-de-creacion-de-oportunidades-2021-2025/

SeeedStudio. (n.d.). *CAN-BUS Shield V2.0*. Retrieved March 5, 2023, from https://media.digikey.com/pdf/data sheets/seeed technology/can_bus_shield_v2.0_web.pdf

SeeedStudio. (2018). *CAN-BUS Shield V2*. https://www.seeedstudio.com/CAN-BUS-Shield-V2.html

Tenesaca, T. (2013). *Estudio para simular una Red CAN con aplicación en comunicación de dispositivos electrónicos en el automóvil* [Universidad del Azuay]. https://dspace.uazuay.edu.ec/handle/datos/2214

Vela Xool, R. A. (2015). *Desarrollo de un sistema educativo para monitoreo e interacción con señales en el Bus Can* [Tecnológico Nacional de México]. https://rinacional.tecnm.mx/jspui/handle/TecNM/2461

Vergara Vargas, I. A. (2005). *Análisis e investigación sobre las caraterísticas fundamentales del protocolo controller area Network (CAN)* [SANGOLQUÍ / ESPE / 2005]. http://repositorio.espe.edu.ec/jspui/handle/21000/703

Wang, Y., Tian, D., Sheng, Z., & Jian, W. (2017). *Connected Vehicle Systems* (1st ed.). CRC Press. https://doi.org/10.4324/9781315232928

Yee, N., Chand, P., & Foehst, S. (2017). Student Designed CANBus Simulator Used as Teaching Aid in Autotronics Course. *2017 4th Asia-Pacific World Congress on Computer Science and Engineering (APWC on CSE)*, 82–87. https://doi.org/10.1109/APWConCSE.2017.00023

ANEXOS

ANEXO I

DISPOSICIÓN DE LOS PINES DE LOS CHIPS DE LA PLACA CAN BUS SHIELD

Name	PDIP/ SOIC Pin #	TSSOP Pin #	QFN Pin #	I/O/P Type	Description	Alternate Pin Function
TXCAN	1	1	19	O	Transmit output pin to CAN bus	—
RXCAN	2	2	20	I	Receive input pin from CAN bus	—
CLKOUT	3	3	1	O	Clock output pin with programmable prescaler	Start-of-Frame signal
$\overline{TX0RTS}$	4	4	2	I	Transmit buffer TXB0 Request-to-Send; 100 kΩ internal pull-up to V_{DD}	General purpose digital input, 100 kΩ internal pull-up to V_{DD}
$\overline{TX1RTS}$	5	5	3	I	Transmit buffer TXB1 Request-to-Send; 100 kΩ internal pull-up to V_{DD}	General purpose digital input, 100 kΩ internal pull-up to V_{DD}
$\overline{TX2RTS}$	6	7	5	I	Transmit buffer TXB2 Request-to-Send; 100 kΩ internal pull-up to V_{DD}	General purpose digital input, 100 kΩ internal pull-up to V_{DD}
OSC2	7	8	6	O	Oscillator output	—
OSC1	8	9	7	I	Oscillator input	External clock input
V_{SS}	9	10	8	P	Ground reference for logic and I/O pins	—
$\overline{RX1BF}$	10	11	9	O	Receive buffer RXB1 interrupt pin or general purpose digital output	General purpose digital output
$\overline{RX0BF}$	11	12	10	O	Receive buffer RXB0 interrupt pin or general purpose digital output	General purpose digital output
$\overline{INT}$	12	13	11	O	Interrupt output pin	—
SCK	13	14	12	I	Clock input pin for SPI interface	—
SI	14	16	14	I	Data input pin for SPI interface	—
SO	15	17	15	O	Data output pin for SPI interface	—
$\overline{CS}$	16	18	16	I	Chip select input pin for SPI interface	—
$\overline{RESET}$	17	19	17	I	Active-low device Reset input	—
V_{DD}	18	20	18	P	Positive supply for logic and I/O pins	—
NC	—	6,15	4,13	—	No internal connection	—
EP	—	—	21	—	Exposed Thermal Pad, connect to V_{SS}	—

Legend: I = Input; O = Output; P = Power

Figura AI.1 Descripción de los pines del controlador MCP 2515

SYMBOL	PIN	DESCRIPTION
TXD	1	transmit data input; reads in data from the CAN controller to the bus line drivers
GND	2	ground
V_{CC}	3	supply voltage
RXD	4	receive data output; reads out data from the bus lines to the CAN controller
V_{ref}	5	reference voltage output
CANL	6	LOW-level CAN bus line
CANH	7	HIGH-level CAN bus line
S	8	select input for high-speed mode or silent mode

Figura AI.2 Descripción de los pines del transceptor TJA 1050

ANEXO II

PROGRMACIÓN DE LA COMUNICACIÓN UNIDIRECCIONAL

Módulo maestro

```
#include <SPI.h>          // Librería que permite establecer una
comunicación con el Bus de datos
#include "mcp2515_can.h"  // Librería que permite comunicación entre el
microcontrolador y la placa Shield
const int SPI_CS_PIN = 10;
mcp2515_can CAN(SPI_CS_PIN); // Pin CS para la selección de comunicación SPI

void setup() {
  Serial.begin(115200);
  SPI.begin();
  Serial.println("--------Iniciando comunicación CAN-BUS--------");
  while (CAN_OK != CAN.begin(CAN_1000KBPS)) {       // Iniciar controlador y
comunicación CAN a una velocidad de 1000KBPS
      Serial.println("Error en el inicio de CAN-BUS");  // Mensaje de
alerta, fallo al iniciar la comunicación CAN
      Serial.println("Reintentando !!!");
      delay(100);
  }
  Serial.println("CAN OK !!");   // Confirmación de inicio CAN-BUS
  CAN.setMode(MODE_NORMAL);    // Controlador CAN en modo normal
}
void loop(){
  unsigned char data[8] = {0, 0, 100, 255, 256, 514, 1, 1}; //Contenido del
campo de datos (información bytes)
  unsigned char sndStat = CAN.MCP_CAN::sendMsgBuf(0x132001, 1, 8, data);  //
Función para enviar el mensaje
  if (sndStat == CAN_OK){
    Serial.println("Mensaje enviado");   // Confirmación de mensaje enviado
  }
  else {
    Serial.println("Error, mensaje no enviado!!");// Alerta de mensaje no
enviado
  }
  delay(100);   // Enviar mensajes cada 100 milisegundos
}
```

Módulo esclavo

```
#include <SPI.h>          // Librería que permite establecer una
comunicación con el Bus de datos
#include "mcp2515_can.h"  // Librería que permite comunicación entre el
microcontrolador y la placa Shield
//DECLARACIÓN VARIABLES
unsigned char Length = 8;       // Longitud del campo de datos (0-8 bytes)
unsigned char Buf[8];          // Almacenamiento del mensaje
char DataString[64];          // Variable que arrastra la cadena de datos
const int SPI_CS_PIN = 10;
mcp2515_can CAN(SPI_CS_PIN); // Pin CS para la selección de comunicación SPI
void setup() {
  Serial.begin(115200);
  SPI.begin();
  Serial.println("--------Iniciando comunicación CAN-BUS--------");
  while (CAN_OK != CAN.begin(CAN_1000KBPS)) {       // Iniciar controlador y
comunicación CAN a una velocidad de 1000KBPS
      Serial.println("Error en el inicio de CAN-BUS");  // Mensaje de
alerta, fallo al iniciar la comunicación CAN
      Serial.println("Reintentando!!!");
      delay(100);
  }
  Serial.println("CAN OK!!");   // Confirmación de inicio CAN-BUS
  CAN.setMode(MODE_NORMAL);    // Controlador CAN en modo normal
}
void loop(){
  if (CAN_MSGAVAIL == CAN.checkReceive()){   // Comprobar si existe mensaje
    CAN.readMsgBuf(&Length, Buf);     // Función para recibir contenido de
mensaje
    unsigned long Identifier = CAN.getCanId();  // Obtener el identificador
    sprintf(DataString, "EXT ID: 0x%.8lX  Data:", (Identifier));
    Serial.print(DataString); //Imprimir cadena de datos leidos
(identificador)
    for(int i = 0; i<Length; i++){
      sprintf(DataString, " %.2X", Buf[i]);
      Serial.print(DataString);  // Imprimir cadena de datos (contenido de
bytes)
    }
    Serial.println();
  }
}
```

ANEXO III

CÓDIGO DE PROGRAMACIÓN PARA LA COMUNICACIÓN CAN CON LA FUNCIÓN MILLIS

Módulo base

```
#include <SPI.h>          // Librería que permite establecer una
comunicación con el Bus de datos
#include "mcp2515_can.h"  // Librería que permite comunicación entre el
microcontrolador y la placa Shield
//DECLARACIÓN VARIABLES
unsigned long Last = 0;
unsigned char Length = 8;       // Longitud del campo de datos (0-8 bytes)
unsigned char Buf[8];           // Almacenamiento del mensaje
char DataString[64];          // Variable que arrastra la cadena de datos

void MCP2515_ISR();
const int SPI_CS_PIN = 10;
mcp2515_can CAN(SPI_CS_PIN); // Pin CS para la selección de comunicación SPI

void setup() {
  Serial.begin(115200);
  Serial.println("--------Iniciando comunicación CAN-BUS--------");
  while (CAN_OK != CAN.begin(CAN_1000KBPS)) {     // Iniciar controlador y
comunicación CAN a una velocidad de 1000KBPS
      Serial.println("Error en el inicio de CAN-BUS"); // Mensaje de alerta,
fallo al iniciar la comunicación CAN
      Serial.println("Reintentando !!!");
      delay(100);
  }
  Serial.println("CAN OK !!");   // Confirmación de inicio CAN-BUS
  CAN.setMode(MODE_NORMAL);    // Controlador CAN en modo normal
  attachInterrupt(0, MCP2515_ISR, FALLING);
  Last = millis();
}

void MCP2515_ISR()       // Función empleada al presentarse una interrupción
{
  if (CAN_MSGAVAIL == CAN.checkReceive()){   // Comprobar si existe mensaje
```

```
    CAN.readMsgBuf(&Length, Buf);      // Función para recibir contenido de
mensaje
    Serial.print("M_RC: ");         // Imprimir M_RC (mensaje recibido)

    unsigned long Identifier = CAN.getCanId();  // Obtener el identificador
    sprintf(DataString, "EXT ID: 0x%.8lX  Data:", (Identifier));
    Serial.print(DataString); //Imprimir cadena de datos leidos
(identificador)

    for(int i = 0; i<Length; i++){
      sprintf(DataString, " %.2X", Buf[i]);
      Serial.print(DataString);      // Imprimir cadena de datos (contenido
de bytes)
    }
    Serial.println();
  }
}

void loop()
{
  if((millis() - Last) >= 10){   // Verificación del periodo de envío 10ms
    unsigned char data[8] = {0, 0, 10, 10, 10, 10, 0, 0}; //Contenido del
campo de datos (información bytes)
    CAN.MCP_CAN::sendMsgBuf(0x132001, 1, 8, data);  // Función para enviar
el mensaje
    Serial.print("M_SD: ");       // Imprimir M_SD (mensaje enviado)

    unsigned long canId = CAN.getCanId();  //Recuperar identificador enviado
    sprintf(DataString, "EXT ID: 0x%.8lX  Data:", (canId));
    Serial.print(DataString);    // Imprimir cadena de datos
(identificador)

    for (int i = 0; i < 8; i++) {
      sprintf(DataString, " %.2X", data[i]);
      Serial.print(DataString);    // Imprimir cadena de datos final
    }
    Serial.println();
    Last = millis();  // Resetear el tiempo
  }
}
```

ANEXO IV

TABLA DE LA CONFIGURACIÓN GENERAL DEL SISTEMA DE CONTROL

CONTROL DEL CHASIS											
ID	**DLC**	**DATA 0**	**DATA 1**	**DATA 2**	**DATA 3**	**DATA 4**	**DATA 5**	**DATA 6**	**DATA 7**	**SUBSITEMA DE CONROL**	**DESCRIPCIÓN DE LA SEÑAL**
0X132001	8	-	-	-	-	-	-	0x00	-	Luz de freno	Luz de freno apagada
	8	-	-	-	-	-	-	0x04	-		Luz de freno encendida
	8	-	0x00	0x00	-	-	-	-	-	Luces antiniebla delanteras y posteriores	Luz antiniebla en estado de reposo
	8	-	0x0D	-	-	-	-	-	-		Luz antiniebla delantera encendida
	8	-	-	0x0F	-	-	-	-	-		Luz antiniebla posterior encendida
	8	-	-	-	-	0x00	-	-	-	Sensor LDR exterior	Conmutación de luces de carretera desactivada
	8	-	-	-	-	0x32	-	-	-		Activar conmutación automática de luces de carretera
	8	-	-	-	-	-	0x00	-	-	Sensor de lluvia	Reposo del sistema automático de limpiaparabrisas
	8	-	-	-	-	-	0x0E	-	-		Ajuste de frecuencia y velocidad baja
	8	-	-	-	-	-	0x1E	-	-		Ajuste de frecuencia y velocidad alta
0X200113	8	-	0x07	-	-	-	-	-	-	Estado de conducción	Vehículo en estado de avance
	8	-	0x00	-	-	-	-	-	-		Vehículo en estado neutral
	8	-	0x03	-	-	-	-	-	-		Vehículo en estado de retroceso
	8	-	-	-	0x00	-	-	-	-	Luces emergencia	Luces intermitentes apagadas
	8	-	-	-	0x0F	-	-	-	-		Luces intermitentes encendidas
	8	-	-	-	-	0x1D	-	-	-	Luces direccionales	Encender direccionales (derecha)
	8	-	-	-	-	0x00	-	-	-		Direccionales apagados
	8	-	-	-	-	0x1B	-	-	-		Encender direccionales (izquierda)

	8	-	-	-	-	-	0x00	0x00	0x00	Sistema de luces (posición, cruce y carretera)	Estado de reposo de luces de posición, cruce y carretera
	8	-	-	-	-	-	0x06	-	-		Encender luces de posición
	8	-	-	-	-	-	-	0x08	-		Encender luces de cruce
	8	-	-	-	-	-	-	-	0x0A		Encender luces de carretera
0x302001	8	0x00	-	-	-	-	-	-	-	Buzzer claxon	Claxon en estado de reposo
	8	0x05		-	-	-	-	-	-		Claxon encendido
	8	-	0x00	0x00	0x00	0x00	-	-	-	Sistema limpiaparabrisas	Reposo del sistema limpiaparabrisas
	8	-	0x17	-	-	-	-	-	-		Activar sistema automático
	8	-	-	0x1A	-	-	-	-	-		Esparcimiento de líquido encendido
	8	-	-	-	0x2D	-	-	-	-		Barrido de parabrisa en velocidad LO
	8	-	-	-	-	0x2E	-	-	-		Barrido de parabrisa en velocidad HI
	8	-	-	-	-	-	-	-	0x00	Alerta a la colisión con objetos	Sistema de alerta a la colisión desactivado
	8	-	-	-	-	-	-	-	0x≤100		Alerta acústica según la proximidad con objetos
0x432001	8	-	0x00	0x00	0x00	-	-	-	-	Sistema de luces automáticas de la cabina	Reposo de luces de cortesía
	8	-	0x0C	-	-	-	-	-	-		Señal referencial de luminosidad baja en el ambiente
	8	-	-	0x0A	-	-	-	-	-		Señal referencial que indica la abertura de la puerta de conductor
	8	-	-	-	0x08	-	-	-	-		Señal referencial de presencia de usuario en la cabina del vehículo
	8	-	-	-	-	-	0x<185	-	-	Ángulo de dirección volante	Giro de volante hacia la izquierda
	8	-	-	-	-	-	0x00	-	-		Señal referencial de posición central de volante, rango aproximado
	8	-	-	-	-	-	0x≥215	-	-		Giro de volante hacia la derecha
	8	-	-	-	-	-	-	0x00	-	Aceleración motor DC rotación continua	Motor detenido (reposo)
	8	-	-	-	-	-	-	>1 & 0x≤255	-		Aceleración y desaceleración paulatina del motor DC

ANEXO V

ESTRUCTURA DEL CÓDIGO DE PROGRAMACIÓN DEL MÓDULO 1 CON LA ADAPTACIÓN DE UN TIMER

Estructura empleada en los 4 módulos

```
#define nodo 1
#include <SPI.h>
#include "mcp2515_can.h"

//DECLARACIÓN VARIABLES PARA COMUNICACIÓN CAN
const int SPI_CS_PIN = 10;
mcp2515_can CAN(SPI_CS_PIN); // Set CS pin
int Flagshot = 0;            //Bandera, referencia de comienzo de
transmisión
unsigned char Buf_NODO2[8];  // Buffer para recibir mensajes del módulo con
identificador 0x200113
unsigned char Buf_NODO3[8];  // Buffer para recibir mensajes del módulo con
identificador 0x302001
unsigned char Buf_NODO4[8];  // Buffer para recibir mensajes del módulo con
identificador 0x432001
//DECLARACIÓN FUNCIONES
void MCP2515_ISR();
void LUCES_POSICION();
void LUCES_CRUCE_CARRETERA();
void LUCES_ANTINIEBLA();
void LUCES_DIRECCIONALES_EMERGENCIA();
//VARAIBLES DE TRANSMISIÓN
int EST_LDRHI = 0;
int EST_RAINS = 0;
//VARIABLES DE RECEPCIÓN
int VAL_LPOS = 0;
int VAL_LCRUCE = 0;
int VAL_LCRTERA = 0;
int VAL_LEMRG = 0;
int VAL_INTGIRO = 0;
int VAL_ANGDIR = 0;
int VAL_ACEM_DC = 0;
int VAL_LP_AUTO = 0;
int EST_INTAN = 0;
```

```
//DECLARACIÓN DE SALIDAS DIGITALES
const int LEDS_POSICION = 6;
const int LEDS_CRUCE = 4;
const int LEDS_CARRETERA = 23;
const int LEDS_DIREMG_RH = 27;
const int LEDS_DIREMG_LF = 29;
//Leds luces antiniebla
const int LED_ANT_RH = 44;
const int LED_ANT_LF = 46;

//Control leds intermitentes direcionales_emergencia
unsigned long DataTime = 0; // Variable para almacenar el tiempo en
milisegundos
const long Int_ON = 500; // Intervalo de tiempo en milisegundos para el
parpadeo del LED
bool Cordinar_Intermitentes = false; // Variable global para controlar la
coordinación de los Leds
//FILTROS
const unsigned long FILTER_ID_1 = 0x200113;
const unsigned long FILTER_ID_2 = 0X302001;
const unsigned long FILTER_ID_3 = 0x432001;

//CONFIGURACIÓN TIMER
void ConF_Timer(){
  TCCR1A = 0;              // Modo normal
  TCCR1B = (1 << WGM12);
  TCCR1B |= (1 << CS11) | (1 << CS10);  // Configuración de pre-escalador
  OCR1A = 625*8;     // Establecer periodo de transmisión en 20 ms
  TIMSK1 = (1 << OCIE1A);
}
//ENVÍO DE MENSAJE SEGÚN EL PERIODO DEL TIMER Y CONDICIONAL
void send() {
  PUL_FRENO();
  INT_ANTDEL();
  INT_ANTPOS();
  LDR_SENSOR_ADP();
  RAIN_SENSOR_AT();
  if(Flagshot == 1){
    Flagshot = 0;
//  detachInterrupt(0);  // Desactivar temporalmente la interrupción externa
```

```
    unsigned char data[8] = {1, INT_ANTDEL(), INT_ANTPOS(), 1, EST_LDRHI,
EST_RAINS, PUL_FRENO(), 1};  //Campo de datos (8 bytes)
    CAN.MCP_CAN::sendMsgBuf(0x132001, 1, 8, data);
//  attachInterrupt(0, MCP2515_ISR, FALLING);  // Volver a activar la
interrupción externa
  }
}
//INICIALIZACIÓN
void setup() {
  Serial.begin(115200);
  attachInterrupt(0, MCP2515_ISR, FALLING);
  Serial.println("--------Iniciando comunicación CAN-BUS--------");
  if (CAN_OK == CAN.begin(CAN_1000KBPS)) {    // Iniciar controlador y
comunicación CAN a una velocidad de 1000KBPS
    Serial.println("CAN OK !!");     // Confirmación de inicio CAN-BUS
  }
  ConF_Timer();      // Función de configuración de Timer1
  // Inicialización de pines de salida
  pinMode(LEDS_POSICION, OUTPUT);
  pinMode(LEDS_CRUCE, OUTPUT);
  pinMode(LEDS_CARRETERA, OUTPUT);
  pinMode(LEDS_DIREMG_RH, OUTPUT);
  pinMode(LEDS_DIREMG_LF, OUTPUT);
  pinMode(LED_ANT_RH, OUTPUT);
  pinMode(LED_ANT_LF, OUTPUT);

  CAN.init_Mask(0, 1, 0XFFFFFFFF); // Máscara de filtro 1
  CAN.init_Filt(0, 1, FILTER_ID_1); // ID de filtro 1
  CAN.init_Mask(1, 1, 0XFFFFFFFF); // Máscara de filtro 2
  CAN.init_Filt(1, 1, FILTER_ID_2); // ID de filtro 2
  CAN.init_Mask(2, 1, 0XFFFFFFFF); // Máscara de filtro 3
  CAN.init_Filt(2, 1, FILTER_ID_3); // ID de filtro 3
}

void MCP2515_ISR()         // Función empleada al presentarse una
interrupción
{
  if (CAN_MSGAVAIL == CAN.checkReceive()){        // Comprobar si existe
mensaje
    unsigned char Length = 8;    // Longitud del campo de datos (0-8 bytes)
    unsigned char Buf[8];   // Almacenamiento temporal del mensaje
```

```
    CAN.readMsgBuf(&Length, Buf);    // Función para recibir contenido de
mensaje

    unsigned long Identifier = CAN.getCanId();
    if (Identifier == FILTER_ID_1) {
      memcpy(Buf_NODO2, Buf, sizeof(Buf));  // Copiar el mensaje al
Buf_NODO2
      if(Buf_NODO2[3] == 15){
        VAL_LEMRG = 9;
      }else{VAL_LEMRG = 0;}

      if(Buf_NODO2[4] == 27){
        VAL_INTGIRO = 11;
      }
      else if(Buf_NODO2[4] == 29){
        VAL_INTGIRO = 13;
      }else{VAL_INTGIRO = 0;}

      if(Buf_NODO2[5] == 6){
        VAL_LPOS = 6;
      }else{VAL_LPOS = 0;}

      if(Buf_NODO2[6] == 8){
        VAL_LCRUCE = 8;
      }else{VAL_LCRUCE = 0;}

      if(Buf_NODO2[7] == 10){
        VAL_LCRTERA = 10;
      }else{VAL_LCRTERA = 0;}
    }

    if (Identifier == FILTER_ID_2) {
      memcpy(Buf_NODO3, Buf, sizeof(Buf));  // Copiar el mensaje al
Buf_NODO3
      VAL_LP_AUTO = Buf_NODO3[1];
    }
    if (Identifier == FILTER_ID_3) {
      memcpy(Buf_NODO4, Buf, sizeof(Buf));  // Copiar el mensaje al
Buf_NODO4
      VAL_ANGDIR = Buf_NODO4[5];
      VAL_ACEM_DC = Buf_NODO4[6];
```

```
    }
  }
}
void loop() {
  int PUL_FRENO();
  int INT_ANTDEL();
  int INT_ANTPOS();
  int LDR_SENSOR_ADP();
  int RAIN_SENSOR_AT();
  LUCES_CRUCE_CARRETERA();
  LUCES_DIRECCIONALES_EMERGENCIA();
  LUCES_ANTINIEBLA();
  LUCES_POSICION();
}
ISR(TIMER1_COMPA_vect) {
  send();
}

//FUNCIONES DE EVALUACIÓN DE ENTRADAS DE DATOS
int LDR_SENSOR_ADP(){  //Función que evalúa la entrada del fotorresistor LDR
  static int PreLDR_ADP = 1;
  int REF_CESTLDR = 1;
  const int LDR_ADP = A14;
  pinMode(LDR_ADP, INPUT);
  int ANG_RDLDR_ADP = analogRead(LDR_ADP);
  int TRAN_VLDR_ADP = map(ANG_RDLDR_ADP, 0, 1023, 0, 255);
  if(TRAN_VLDR_ADP < 200){
    REF_CESTLDR = 0;
  }
  if(TRAN_VLDR_ADP >= 200){
    EST_LDRHI = 50; //Señal que valida la conmutación automática de las
luces de carretera a cruce
    Flagshot = 1;
  }
  else{
    if(PreLDR_ADP == 1 && REF_CESTLDR == 0){
      EST_LDRHI = 0; //Señal de reposo de la conmutación de luces
adaptativas
      Flagshot = 1;
    }
  }
```

```
  PreLDR_ADP = REF_CESTLDR;
  return 0;
}

int RAIN_SENSOR_AT(){  //Función que evalúa la entrada del Sensor de lluvia

  static int Pres_RAIN = 1;
  int REF_ESTC_RAIN = 1;
  const int RAIN_SENSOR = A11;
  pinMode(RAIN_SENSOR, INPUT);
  int ANG_RDRAIN = analogRead(RAIN_SENSOR); //Lectura analógica del sensor
  int TRAN_SSRAIN = map(ANG_RDRAIN, 0, 1023, 10, 0);
  if(TRAN_SSRAIN  < 2){
    REF_ESTC_RAIN = 0;
  }
  if(TRAN_SSRAIN  >= 2 && TRAN_SSRAIN <= 7  ){
    EST_RAINS = 14;  //Señal que valida la activación de los servomotores en
una frecuencia baja
    Flagshot = 1;
  }
  else if (TRAN_SSRAIN  >= 8 && TRAN_SSRAIN <= 10 ){
    EST_RAINS = 30;  //Señal que valida la activación de los servomotores en
una frecuencia alta
    Flagshot = 1;
  }
  else{
    if(Pres_RAIN == 1 && REF_ESTC_RAIN == 0){
      EST_RAINS = 0; //Señal de reposo para el sistema automático de
limpiaparabrisas
      Flagshot = 1;
    }
  }
  Pres_RAIN = REF_ESTC_RAIN;
  return 0;
}

int PUL_FRENO(){   //Función que evalúa la entrada del pulsador de freno
  const int PUL_FR = 34;
  pinMode (PUL_FR, INPUT);
  static int PrePULFRS = HIGH;
  int LasPULFRS = digitalRead(PUL_FR);
```

```
  int VAL_FR = 0;
  if (LasPULFRS == HIGH) {
    VAL_FR = 4;       //Señal de activación, pulsador presionado
    Flagshot = 1;
  } else {
    if (PrePULFRS == HIGH && LasPULFRS == LOW) {
      VAL_FR = 0; //Señal de reposo, pulsador en posición original
      Flagshot = 1;
    }
  }
  PrePULFRS = LasPULFRS;
  return (VAL_FR);
}

int INT_ANTDEL(){    //Función que evalúa la entrada del interruptor de
luces antiniebla delanteras
  const int Int_LAD = 38;
  pinMode (Int_LAD, INPUT);
  static int Pre_LAD = HIGH;
  int PosINTLAD = digitalRead(Int_LAD);
  int VAL_LANTDEL = 0;
  if (PosINTLAD == HIGH) {
    VAL_LANTDEL = 13; //Señal de activación, interruptor en posición ON
    Flagshot = 1;
  }
  else {
    if (Pre_LAD == HIGH && PosINTLAD == LOW) {
      VAL_LANTDEL = 0;  //Señal de reposo, interruptor en posición OFF
      Flagshot = 1;
    }
  }
  Pre_LAD = PosINTLAD;
  EST_INTAN = VAL_LANTDEL;
  return(VAL_LANTDEL);
}
int INT_ANTPOS(){   //Función que evalúa la entrada del interruptor de luces
antiniebla posteriores
  const int Int_LAP = 40;
  pinMode (Int_LAP, INPUT);
  static int Pre_LAP = HIGH;
  int PosINTLAP = digitalRead(Int_LAP);
```

```
  int VAL_LANTPOS = 0;
  if (PosINTLAP == HIGH){
    VAL_LANTPOS = 15;  //Señal de activación, interruptor en posición ON
    Flagshot = 1;
  }
  else {
    if (Pre_LAP == HIGH && PosINTLAP == LOW) {
      VAL_LANTPOS = 0;   //Señal de reposo, interruptor en posición OFF
      Flagshot = 1;
    }
  }
  Pre_LAP = PosINTLAP;
  return (VAL_LANTPOS);
}

// FUNCIONES DE LOS SUBSISTEMAS DE CONTROL
void LUCES_DIRECCIONALES_EMERGENCIA(){
  unsigned long Current_TM = millis(); // Obtener el tiempo actual en
milisegundos

  if ((VAL_LEMRG == 9) && (VAL_INTGIRO == 11 || VAL_INTGIRO == 13)) {
    if (!Cordinar_Intermitentes) { // Verificar si los Leds no están
coordinados
      DataTime = Current_TM; // Actualizar el tiempo previo
      Cordinar_Intermitentes = true; // Marcar los Leds como coordinados
      digitalWrite(27, HIGH);
      digitalWrite(29, HIGH);

    }
    else if (Current_TM - DataTime >= Int_ON) {
      DataTime = Current_TM; // Actualizar el tiempo previo
      digitalWrite(27, !digitalRead(27));
      digitalWrite(29, !digitalRead(29));
    }
  }
  else if (VAL_LEMRG == 9 && VAL_INTGIRO != 11 && VAL_INTGIRO != 13) {
    Cordinar_Intermitentes = false; // Reiniciar el estado de coordinación
de los LEDs

    if (Current_TM - DataTime >= Int_ON) {
      DataTime = Current_TM; // Actualizar el tiempo previo
```

```
      digitalWrite(27, !digitalRead(27));
      digitalWrite(29, !digitalRead(29));
    }
  }
  else if (VAL_INTGIRO == 11 && VAL_INTGIRO != 13 && VAL_LEMRG != 9) {
    Cordinar_Intermitentes = false; // Reiniciar el estado de coordinación
de los LEDs

    if (Current_TM - DataTime >= Int_ON) {
      DataTime = Current_TM; // Actualizar el tiempo previo
      digitalWrite(29, LOW);
      digitalWrite(27, !digitalRead(27));
    }
  }
  else if (VAL_INTGIRO == 13 && VAL_LEMRG != 9 && VAL_INTGIRO != 11) {
    Cordinar_Intermitentes = false; // Reiniciar el estado de coordinación
de los LEDs

    if (Current_TM - DataTime >= Int_ON) {
      DataTime = Current_TM; // Actualizar el tiempo previo
      digitalWrite(27, LOW);
      digitalWrite(29, !digitalRead(29));
    }
  }
  else {
    Cordinar_Intermitentes = false; // Reiniciar el estado de coordinación
de los LEDs
    digitalWrite(27, LOW);
    digitalWrite(29, LOW);
  }
}

void LUCES_ANTINIEBLA(){
  if(EST_INTAN == 13 && VAL_ACEM_DC >= 76 && (VAL_ANGDIR <= 185 ||VAL_ANGDIR
== 0 || VAL_ANGDIR >= 215)){
    analogWrite (44, 80);
    analogWrite (46, 80);
  }
  else if (EST_INTAN == 13 && VAL_ANGDIR == 0){
    analogWrite (44, 80);
    analogWrite (46, 80);
```

```
  }
  else if(EST_INTAN == 13 && VAL_ANGDIR <= 185 && VAL_ACEM_DC <= 215){
    analogWrite (44, 0);
    analogWrite (46, 150);
  }
  else if(EST_INTAN == 13 && VAL_ANGDIR >= 215 && VAL_ACEM_DC <= 185){
    analogWrite (44, 150);
    analogWrite (46, 0);
  }
  else{
    analogWrite (44, 0);
    analogWrite (46, 0);
  }
}
void LUCES_POSICION(){
  if(VAL_LPOS == 6){
    analogWrite (6, 15);
  }
  else{
    analogWrite (6, 0);
  }
}
void LUCES_CRUCE_CARRETERA(){

  if(VAL_LCRUCE == 8 && VAL_LCRTERA != 10 && EST_LDRHI != 50){
    analogWrite (4, 100);
    digitalWrite (23, LOW);
  }
  else if(VAL_LCRUCE == 8 && VAL_LCRTERA == 10 && EST_LDRHI != 50){
    digitalWrite (23, HIGH);
    analogWrite (4, 0);
  }
  else if(VAL_LCRUCE == 8 && VAL_LCRTERA == 10 && EST_LDRHI == 50){
    digitalWrite (23, LOW);
    analogWrite (4, 100);
  }
  else{
    analogWrite (4, 0);
    digitalWrite (23, LOW);
  }
```

ANEXO VI

ESQUEMA DEL SISTEMA DE CONTROL

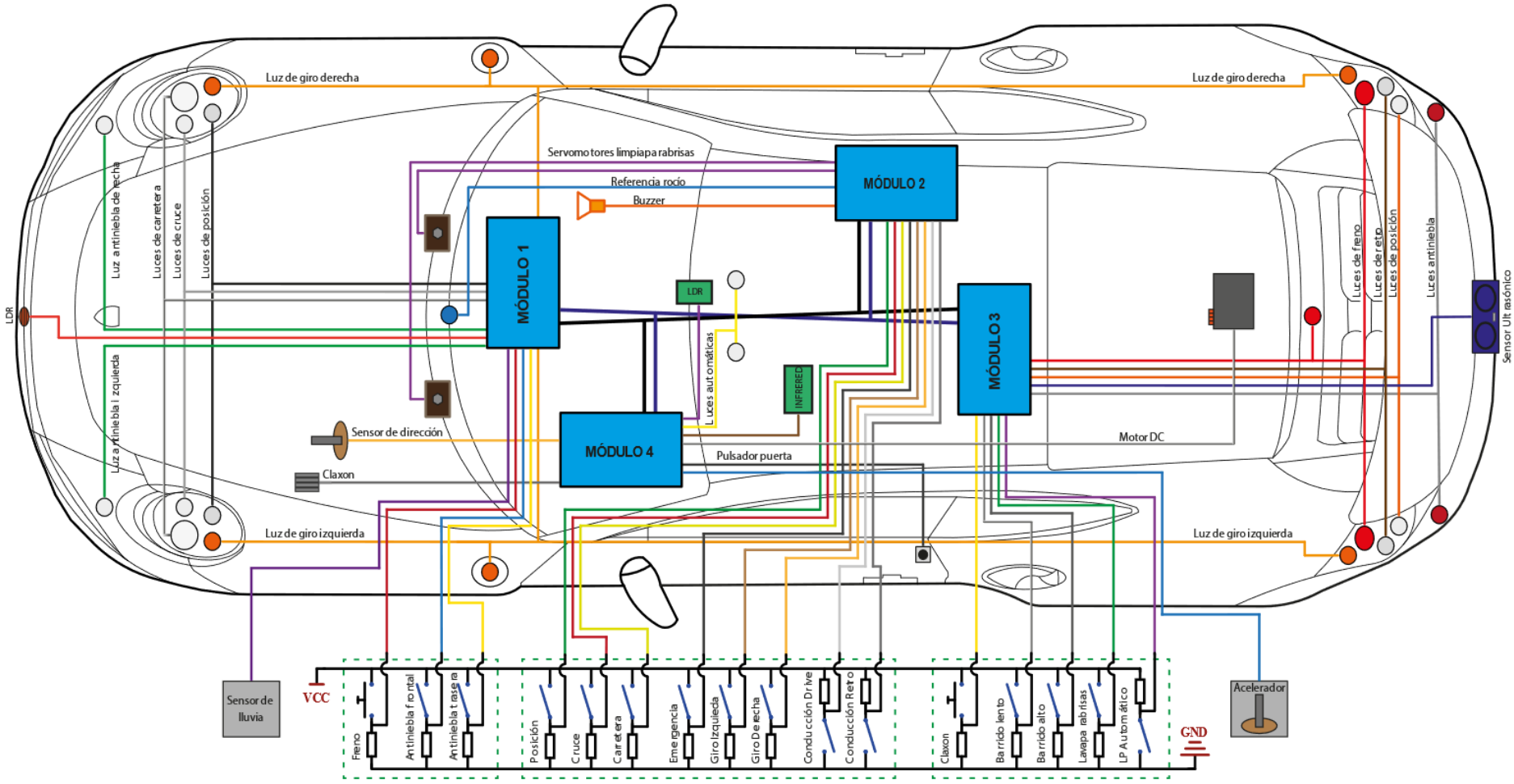


Figura AVI.1 Red multiplexada CAN del prototipo

ANEXO VII

CONSTRUCCIÓN DEL PROTOTIPO DE RED MULTIPLEXADA CAN

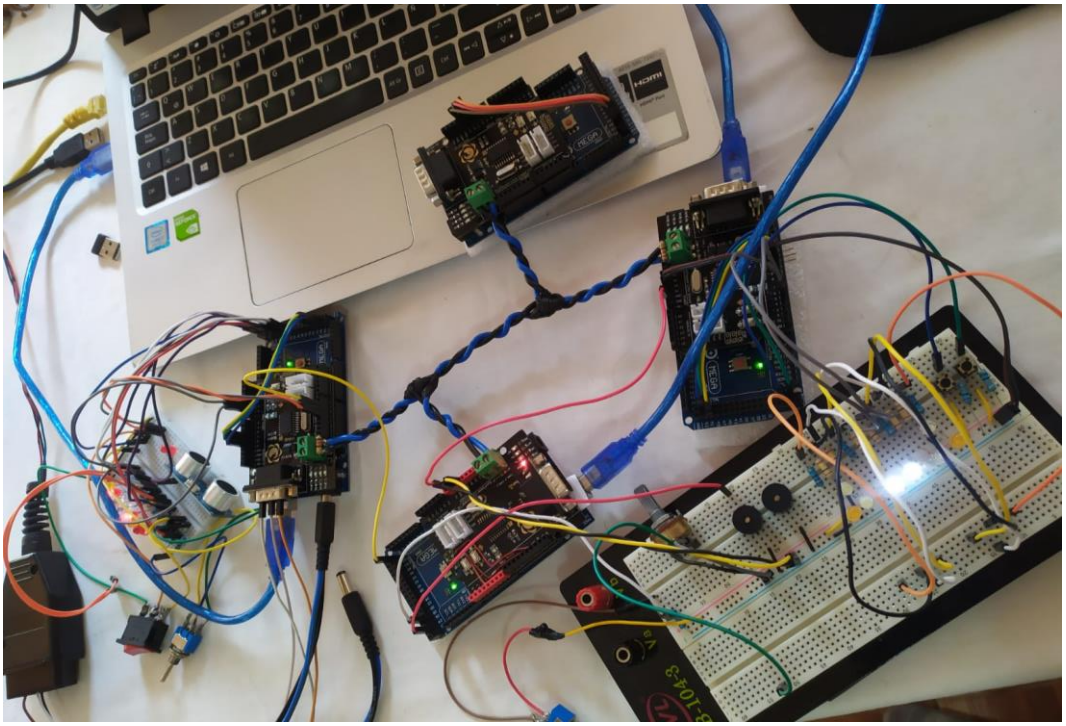


Figura AVII.1 Estructura base de la red multiplexada CAN

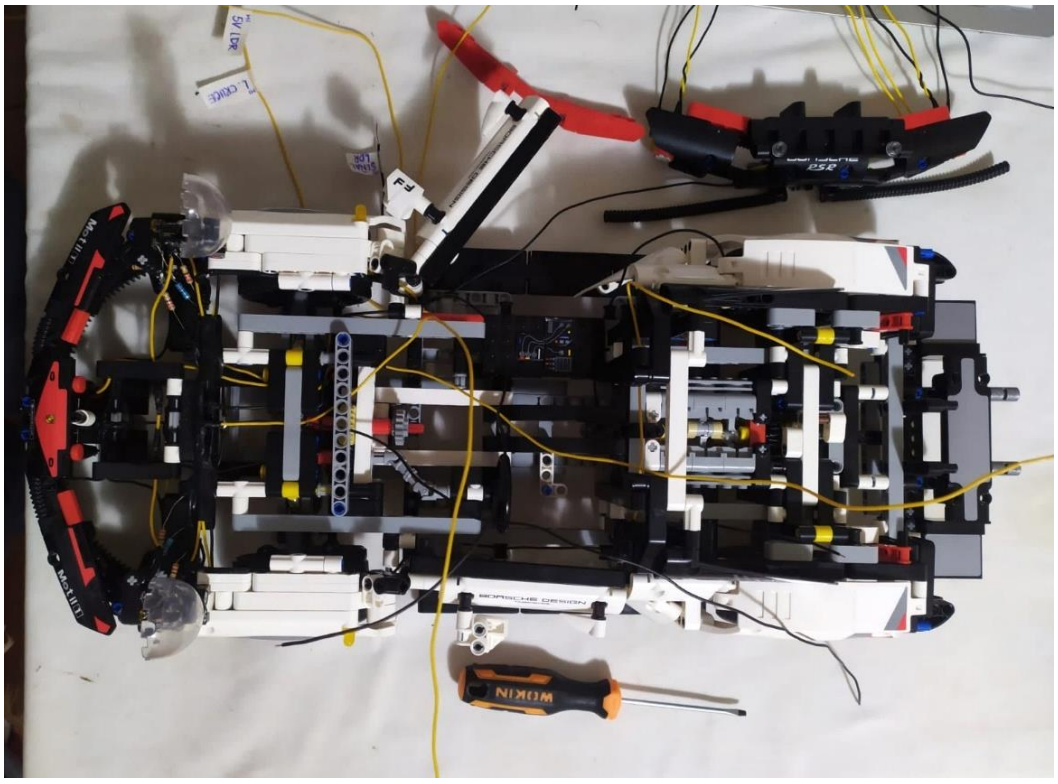


Figura AVII.2 Montaje de componentes captadores y actuadores en el vehículo a escala

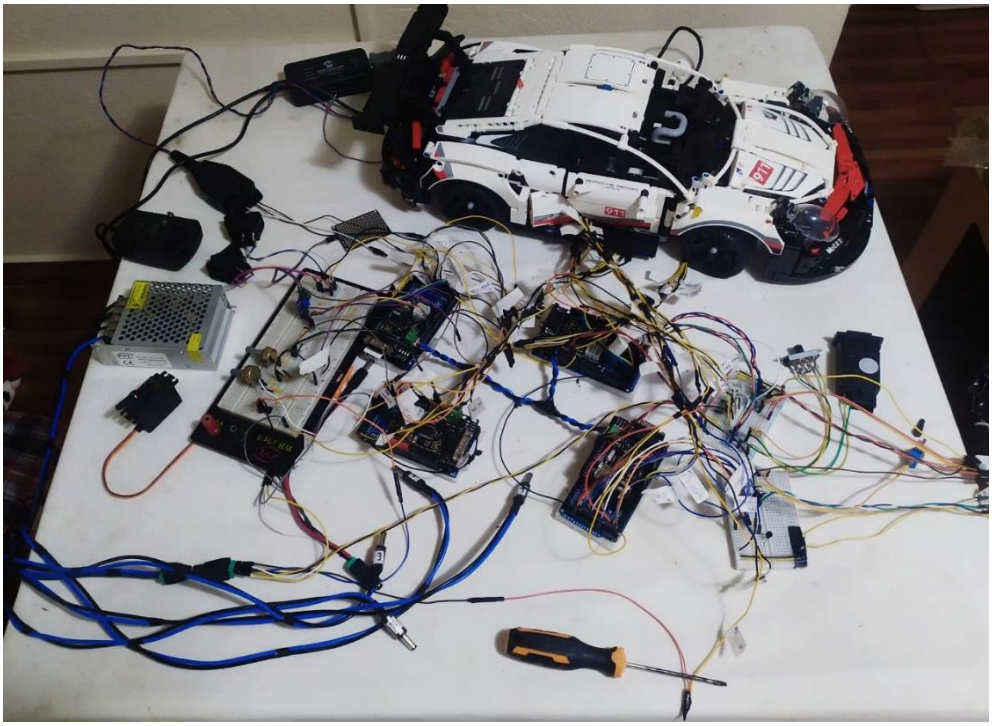


Figura AVII.3 Pruebas de funcionamiento de subsistemas

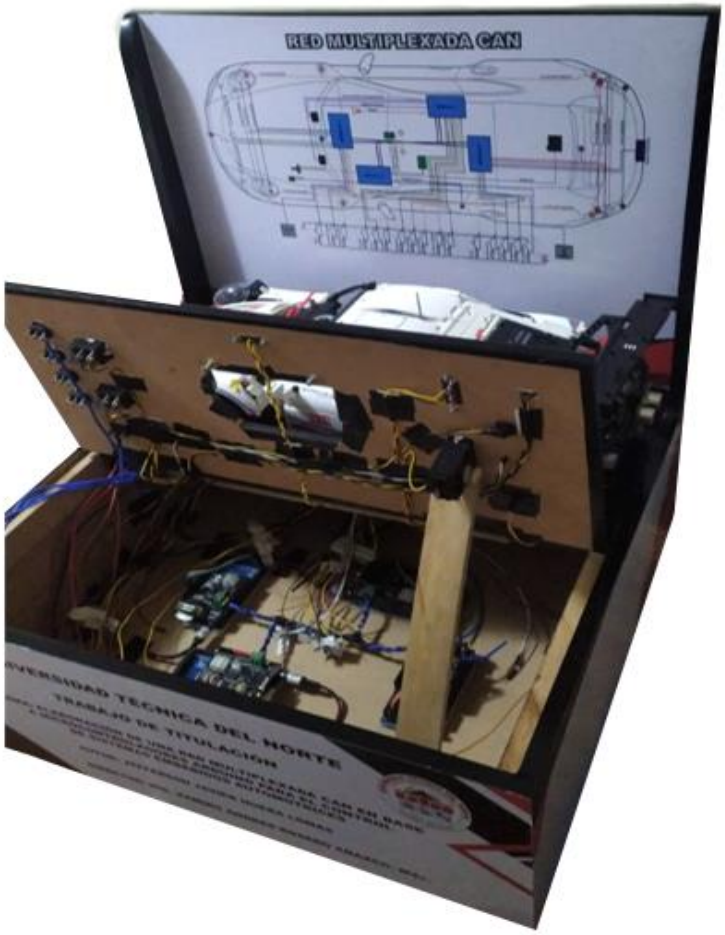


Figura AVII.4 Implementación del sistema general de la red en la estructura del prototipo