INSTITUTO TECNOLÓGICO SUPERIOR DE IRAPUATO

ESTUDIOS CON RECONOCIMIENTO DE VALIDEZ OFICIAL
NÚMERO 11-00065

ESTUDIO, DISEÑO E IMPLEMENTACIÓN DE CONTROLADOR DE PROTOCOLO DE COMUNICACIÓN LIN BUS PARA APLICACIONES EN ACCESORIOS AUTOMOTRICES

OPCIÓN I: TESIS PROFESIONAL

PARA OBTENER EL GRADO DE

MAESTRÍA EN INGENIERÍA ELECTRÓNICA

PRESENTA:

ING. MARISOL DEL ROCÍO SOLÍS RAYAS

DIRECTOR DE TESIS:

DR. GILBERTO MUÑOZ MORENO

Co-Director DE TESIS:

DR. MARIO ALBERTO JUÁREZ BALDERAS

IRAPUATO, GTO. FEBRERO 2026

Educación
Secretaría de Educación Pública

TECNOLÓGICO NACIONAL DE MÉXICO

GUANAJUATO
TECNOLÓGICO DE ESTUDIOS SUPERIORES

Instituto Tecnológico Superior de Irapuato
Dirección General
Dirección de Académica

Irapuato, Guanajuato, 25/febrero/2026
Oficio No. CIPI-002-2026
ASUNTO: Autorización de impresión de tesis de maestría

DR. MARIO ALBERTO JUÁREZ BALDERAS
PRESIDENTE DEL CONSEJO DE POSGRADO
MAESTRÍA EN INGENIERÍA ELECTRÓNICA
PRESENTE

Por medio de la presente y a solicitud del comité tutorial integrado por:

Dr. Gilberto Muñoz Moreno
Dr. Mario Alberto Juárez Balderas
M.C. José Juan Alfaro Rodriguez

se autoriza la impresión de la tesis titulada **"Estudio, diseño e implementación de controlador de protocolo de comunicación LIN BUS para aplicaciones en accesorios automotrices"** realizada por el estudiante **C. Marisol del Rocío Solís Rayas** con número de control **MIP23110015** la cual ha sido desarrollada dentro del programa de la Maestría en Ingeniería Electrónica bajo la dirección del Dr. Gilberto Muñoz Moreno y la codirección del Dr. Mario Alberto Juárez Balderas y ha sido revisada y aprobada por el comité tutorial antes mencionado.

Sin otro en particular, le envio un cordial saludo.

A T E N T A M E N T E
Excelencia en Educación Tecnológica

EDUCACIÓN | TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO SUPERIOR DE IRAPUATO
MARGARITA OBA PUBLICA

M.C. ISAI GONZÁLEZ GAONA
DIRECTOR ACADÉMICO
PRESIDENTE DEL CIPI

ccp.
M.I. Ernasto Cabal Yapez — Titular de jefatura de División de Ing. Electrónica — Para su conocimiento y atención
M.C. Akira Torreblanca Ponce — Titular del Departamento de Investigación — Para su seguimiento

2026 año de Margarita Maza

INSTITUTO TECNOLÓGICO
SUPERIOR DE IRAPUATO

CONSTANCIA DE APROBACIÓN DE TESIS

La tesis Estudio, diseño e implementación de controlador de protocolo de comunicación LIN bus para aplicaciones en accesorios automotrices presentada para obtener el Grado de Maestro en Ingeniería Electrónica fue elaborada por la Ing. Marisol del Rocío Solís Rayas y aprobada el 27 de Febrero de 2026 por los suscritos, designados por el consejo de Posgrado de la Maestría en Ingeniería Electrónica del Tecnológico Nacional de México Campus Irapuato.

Dr. Gilberto Muñoz Moreno
(Director de tesis)

Dr. Mario Alberto Juárez Balderas
(Co-Director de tesis)

M.C. Jose Juan Alfaro Rodríguez
(Sinodal)

INSTITUTO TECNOLÓGICO
SUPERIOR DE IRAPUATO

CRÉDITOS INSTITUCIONALES

La presente tesis fue elaborada en el Laboratorio de Eléctrica y Electrónica de Potencia (LEEP) del Tecnológico Nacional de México / Instituto Tecnológico Superior de Irapuato, bajo la dirección del Dr. Gilberto Muñoz Moreno, en colaboración con el Dr. Mario Alberto Juárez Balderas, adscritos a este instituto.

AGRADECIMIENTOS

Al Tecnológico Nacional de México / Instituto Tecnológico Superior de Irapuato (TecNM/ITESI), por el apoyo institucional y el acceso a los recursos fundamentales para la elaboración de esta tesis. Asimismo, expreso mi reconocimiento al Consejo de Posgrado de la Maestría en Ingeniería Electrónica por permitirme desarrollar esta investigación.

Un agradecimiento especial a mi director de tesis, el Dr. Gilberto Muñoz Moreno, y a mi codirector, el Dr. Mario Alberto Juárez Balderas, por compartir su conocimiento, orientación y paciencia. Su dedicación y motivación constante han sido pilares fundamentales en mi formación profesional. De igual manera, agradezco al sinodal, el M.C. José Juan Alfaro Rodríguez, cuyas valiosas aportaciones mejoraron significativamente la calidad de este trabajo.

A mi familia, por su apoyo incondicional y motivación desde el primer día. En especial a mi hermana, Antonella Solís, por su ayuda al prestarme su dispositivo móvil para poder realizar las pruebas finales de este proyecto.

A mis compañeros de maestría por su compañerismo durante todo el camino; especialmente a Samuel Ortega Cano, quien, a pesar de enfrentar la misma exigencia de combinar el trabajo con el estudio, siempre estuvo dispuesto a brindarme su apoyo y orientación.

Finalmente, me agradezco a mí misma. Compaginar el desempeño profesional en la industria automotriz catalogada como una de las más demandantes a nivel global con los estudios de posgrado, ha representado el mayor reto de mi vida y una prueba de mi perseverancia.

RESUMEN

Desde sus inicios, la industria automotriz ha estado intrínsecamente ligada a la exploración de la comodidad y la seguridad de los usuarios. La búsqueda por ofrecer una experiencia de conducción más cómoda y segura ha impulsado la integración de una multitud de dispositivos electrónicos en los vehículos. Estos dispositivos, que van desde simples sensores hasta complejos sistemas de control, requieren comunicarse entre sí de manera fluida para garantizar un funcionamiento óptimo. En las primeras décadas de la industria automotriz, la electrónica se limitaba a funciones básicas como el encendido del vehículo y la iluminación. Sin embargo, con el paso del tiempo, la demanda de características más avanzadas ha llevado a un aumento exponencial en la complejidad de los sistemas electrónicos a bordo. Esta creciente complejidad ha planteado un desafío significativo: ¿Cómo lograr una comunicación eficiente y confiable entre tantos dispositivos diferentes, sin necesidad de un enredo de cables que complique la fabricación y el mantenimiento de los vehículos? La respuesta a esta pregunta se encuentra en los protocolos de comunicación. Estos protocolos establecen un conjunto de reglas y estándares que permiten a los diferentes dispositivos electrónicos intercambiar información de manera estructurada y segura.

En función de la criticidad de los mensajes transmitidos, existen diversas redes de comunicación vehicular. Entre ellas, destaca la Red de Interconexión Local LIN (Local Interconnect Network), clasificada como SAE “Society Automotive Engineers” Clase A. Esta red resulta idónea para aplicaciones no críticas, como el control de ventanillas, techo solar, iluminación interior y demás accesorios. El bus LIN se ha posicionado como un elemento fundamental en el diseño automotriz moderno, facilitando la comunicación eficiente entre múltiples componentes electrónicos y contribuyendo significativamente a la funcionalidad y confort del vehículo. Con el objetivo de delimitar el ámbito de este estudio, se ha centrado la atención en el protocolo LIN-bus. El propósito final consiste en desarrollar un controlador LIN-BUS capaz de gestionar accesorios automotrices con comunicación LIN a partir de una aplicación móvil y conexión Bluetooth.

ABSTRACT

Over the last few years, the global automotive industry has undergone significant electrical and electronic (E/E) transformation. These technological changes are driven by the integration of numerous Electronic Control Units (ECUs). The Local Interconnect Network (LIN) Bus is a key automotive standard used for non-critical E/E vehicle systems, such as interior lighting, switches, and rain sensors. This paper presents the design of an automotive device that simplifies this technology: a low-cost LIN Bus automotive controller. The primary objective is to develop a controller for this system at a minimal investment. We met this goal by specific objectives: researching accessible electronic components, creating a schematic diagram, executing the necessary programming, and performing testing to validate the LIN Bus signal acquisition. The project used a two-phase method, covering both hardware and software. The hardware phase focuses on designing and fabricating a portable LIN Bus controller using a prototype Printed Circuit Board (PCB). The software phase involves programming a microcontroller to manage LIN communication and establishing a Bluetooth® connection with a mobile application developed in Android Studio®. By creating a portable ECU for controlling and inspecting automotive parts, this project allows engineers to perform faster analysis and product design validations without disrupting production lines. One of the most important benefits is the minimal investment required for the total cost of the controller. This solution could significantly benefit automotive manufacturers by accelerating project timelines, meeting the industry standards and preventing potential production shortages accelerating product development.

NOTACIÓN

Protocolos y Estándares Automotrices

- **CAN** – Controller Area Network (Red de Área de Controladores)
- **IPC** – Institute for Printed Circuits (Instituto de Circuitos Impresos)
- **I2C** – Inter-Integrated Circuit (Circuito Inter-Integrado)
- **LDF** – LIN Description File (Archivo de Descripción de LIN)
- **LIN** – Local Interconnect Network (Red de Interconexión Local)
- **MOST** – Media Oriented Systems Transport (Transporte de Sistemas Orientados a Medios)
- **NCF** – Node Capability File (Archivo de Capacidad del Nodo)
- **SAE** – Society of Automotive Engineers (Sociedad de Ingenieros Automotrices)
- **UART** – Universal Asynchronous Receiver-Transmitter (Transmisor-Receptor Asíncrono Universal)
- **UDS** – Unified Diagnostic Services (Servicios de Diagnóstico Unificados)
- **SPI** – Serial Peripheral Interface (Interfaz Periférica Serial)

Sistemas de Control y Seguridad

- **ABS** – Antilock Brake System (Sistema de Frenos Antibloqueo)
- **ECU** – Electronic Control Unit (Unidad de Control Electrónica)
- **ESP** – Electronic Stability Program (Programa de Estabilidad Electrónica)

Hardware y Electrónica

- **GPIO** – General Purpose Input/Output (Entrada/Salida de Propósito General)
- **IC** – Integrated Circuit (Circuito Integrado)
- **PCB** – Printed Circuit Board (Placa de Circuito Impreso)
- **SoC** – System on Chip (Sistema en un Chip)

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1. INTRODUCCIÓN

El sector automotriz en México es uno de los más sólidos a nivel mundial, lo que lo convierte en un pilar fundamental para la economía del país. Solo en 2024, sus contribuciones al Producto Interno Bruto (PIB) alcanzaron una cifra de 4.5% (INEGI y asociación Mexicana de la Industria Automotriz, ver Figura 1).

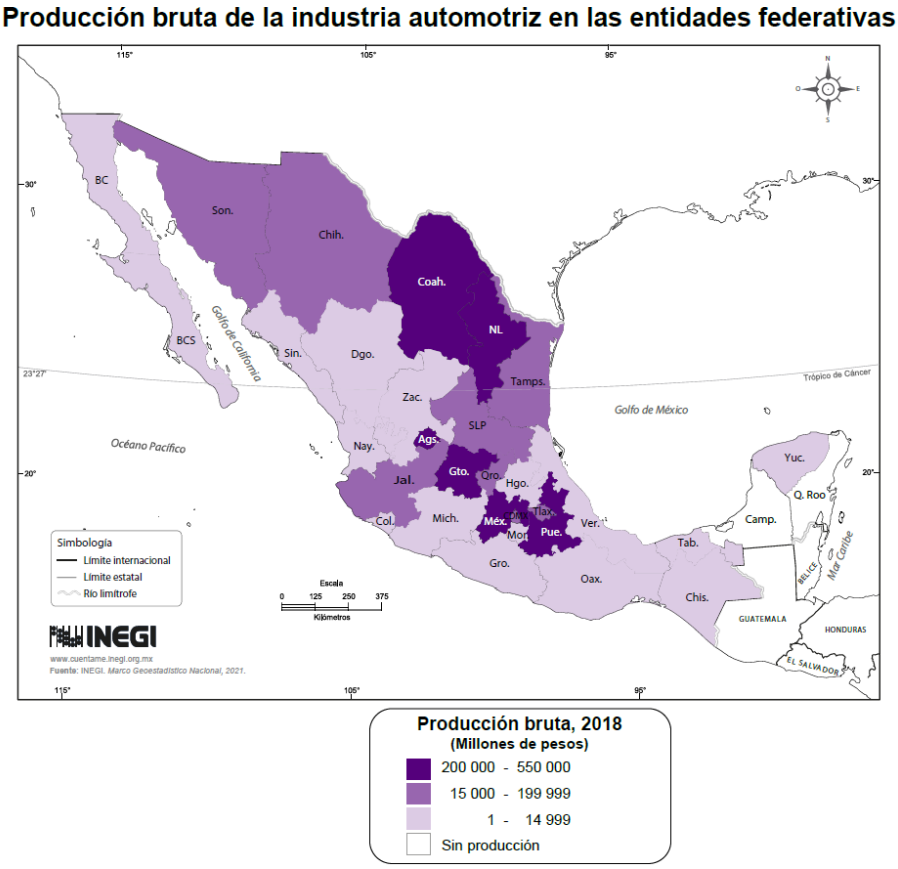


Figura 1. Producción bruta de la industria automotriz en las entidades federativas.

Gracias a este impacto, tanto la producción de vehículos como la de autopartes se han consolidado como sectores clave para el desarrollo económico de México (ver Figura 2) [13]. Este éxito no solo se basa en la capacidad de producción, sino también en el trabajo del equipo de ingeniería de validación y desarrollo que aseguran la calidad, seguridad y durabilidad de los productos. Desde la fase de diseño de prototipos, el equipo de ingeniería debe aplicar rigurosos procesos de validación para cumplir con estándares (ej. IATF 16949), en donde entran pruebas de durabilidad en climas extremos (85°C/-40°C), vibraciones y corrosión, para simular la vida útil de los componentes. Un ejemplo son los sistemas automotrices no críticos como el control de ventanas, espejos y luces interiores. Para las fases de desarrollo y validación en un nuevo proyecto automotriz, los ingenieros utilizan herramientas especializadas que se adaptan a la complejidad de cada etapa. Estás herramientas suelen de ser de un costo elevado, lo que puede representar una limitación significativa para las pequeñas empresas automotrices. Este factor no sólo restringe el acceso a los dispositivos, sino que también puede generar retrasos en la validación de prototipos e inspecciones de calidad. En el peor de los casos, puede llevar al uso de herramientas inadecuadas, lo que compromete la fiabilidad de los resultados y la calidad de los productos.

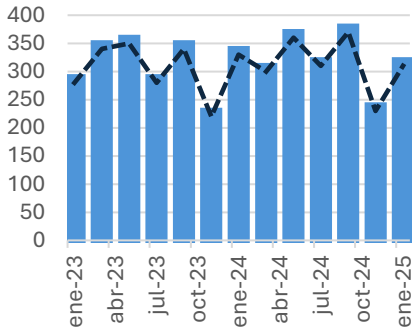


Figura 2(a). Producción trimestral de vehículos en Miles de Unidades en México (2023-2024).

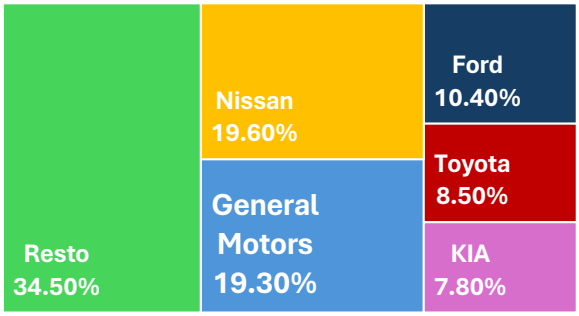


Figura 2(b). Pparticipación por marcas (en porcentaje%).

Para abordar esta problemática, este artículo presenta el diseño y desarrollo de un controlador LIN Bus de bajo costo. El proyecto se basa en el uso de software de código abierto y componentes electrónicos comerciales, con el objetivo de lograr una comunicación bidireccional fiable para la validación de accesorios automotrices.

2. GENERALIDADES DEL PROYECTO

2.1 Definición del problema

La definición del problema de este estudio se centra en los desafíos actuales de la industria automotriz ante la compleja transformación eléctrica y electrónica, donde la integración masiva de unidades de control electrónico exige procesos de validación constantes. El problema reside específicamente en que las herramientas de inspección para el estándar LIN Bus suelen requerir inversiones elevadas y procedimientos técnicos que interrumpen el flujo normal de las líneas de producción. Esto impacta negativamente en la calidad final, ya que la dificultad para realizar pruebas rápidas y frecuentes limita la detección temprana de fallos, comprometiendo así la fiabilidad de los sistemas y el cumplimiento de los estándares industriales. En consecuencia, la falta de dispositivos de diagnóstico accesibles y portátiles genera cuellos de botella y retrasa los cronogramas de proyectos, evidenciando la necesidad urgente de una solución de bajo costo que permita realizar análisis técnicos eficientes sin afectar el flujo operacional de las plantas de fabricación.

2.2 Justificación

La relevancia de este proyecto surge de la necesidad crítica de adaptar los procesos de validación a la actual transformación eléctrica y electrónica de la industria automotriz. Dado que el estándar LIN Bus gestiona sistemas esenciales para la experiencia del usuario, como la iluminación y sensores internos, contar con herramientas de diagnóstico eficientes no es solo una ventaja operativa, sino una necesidad estratégica. La implementación de este controlador de bajo costo se justifica plenamente al ofrecer una alternativa técnica que elimina la dependencia de equipos costosos y complejos, democratizando el acceso a herramientas de alta precisión mediante el uso de componentes electrónicos accesibles y tecnología de comunicación inalámbrica vía Bluetooth®.

Asimismo, el desarrollo de esta ECU portátil tiene un impacto directo en la eficiencia productiva y la calidad del producto final. Al permitir que los ingenieros realicen validaciones de diseño y análisis de señales en tiempo real sin interrumpir las líneas de producción, se eliminan los cuellos de botella que suelen retrasar el lanzamiento de nuevos modelos. Esta agilidad no solo asegura el cumplimiento de los estrictos estándares de la industria y previene paros operativos por fallos no detectados, sino que también optimiza la inversión de capital. En última instancia, este proyecto fortalece la competitividad de los fabricantes al acelerar los ciclos de desarrollo y garantizar la fiabilidad de los sistemas electrónicos mediante un método de inspección ágil, económico y altamente funcional.

2.3 Objetivos

Esta tesis de maestría tiene como alcance evaluar un controlador que utiliza el protocolo automotriz LIN BUS. La primera parte del trabajo presenta los antecedentes y el estado del arte de la red LIN BUS en vehículos. En la segunda parte se desarrolla una investigación en la selección de componentes eléctricos/electrónicos con los que se trabajaran para el desarrollo de un diagrama esquemático por consiguiente un PCB y se programa una pantalla táctil realizando pruebas con accesorios automotrices. Finalmente, se documentarán los resultados obtenidos y los desafíos encontrados en la parte práctica.

2.3.1 Objetivo general

Elaborar un controlador de bajo costo para uso automotriz utilizando protocolo de comunicación LIN BUS para controlar motores o dispositivos de iluminación automotriz que utilicen LIN BUS.

2.3.2 Objetivos específicos

- Revisión del estado del arte de los principales protocolos de comunicación seriales.
- Estudio y selección componentes electrónicos con los que se establecerá comunicación con el protocolo LIN BUS.
- Elaboración de un diagrama eléctrico esquemático del circuito LIN BUS a utilizar.
- Diseño de PCB del controlador.
- Definición y programación de la arquitectura para el establecimiento de comunicación del protocolo de comunicación automotriz LIN BUS, utilizando un lenguaje de programación de libre acceso.
- Obtención de resultados experimentales con al menos 3 productos diferentes que contengan comunicación LIN.
- Realizar la escritura de documento de tesis.

2.4 Limitaciones

Los controladores LIN BUS son dispositivos versátiles que permiten gestionar una amplia gama de funciones en componentes automotrices, incluyendo control, lectura y diagnóstico. En este trabajo de maestría, se propone el desarrollo de un controlador LIN BUS de bajo costo, utilizando software libre y componentes electrónicos comerciales. El objetivo principal es establecer una comunicación bidireccional efectiva entre los componentes automotrices de la empresa, con el fin de enviar y recibir comandos de manera confiable, realizando acciones de encendido o apagado de las luces o motor.

3. ESTADO DEL ARTE

3.1 Evolución de la arquitectura electrónica y arquitecturas de red vehicular

La cantidad de componentes electrónicos en los vehículos ha aumentado drásticamente en los últimos años y se prevé que crezca aún más en el futuro. Los avances técnicos en la tecnología de semiconductores permiten funciones cada vez más complejas gracias a una mayor densidad de integración.

Un factor determinante para el éxito de la industria automotriz ha sido la serie continua de innovaciones que se han ido incorporando a los vehículos. Desde la década de 1970, el objetivo era utilizar nuevas tecnologías para ayudar en el desarrollo de coches seguros, limpios y económicos. La búsqueda de la eficiencia y la ecología estuvo estrechamente ligada a otros beneficios para el cliente, como el confort de conducir. Un claro ejemplo de ello fue una mejora en la seguridad vial gracias a los sistemas electrónicos de control de frenado. En 1978 se introdujo el sistema antibloqueo (ABS), el cual evolucionó constantemente hasta el punto de que hoy se instala de serie en todos los vehículos en Europa. Siguiendo esta misma línea de desarrollo, en 1995 debutaría el programa electrónico de estabilidad (ESP), que integra al propio sistema ABS.

Los desarrollos más recientes en la industria automotriz se enfatizan en la comodidad de los ocupantes mediante sistemas electrónicos integrados de gestión ambiental (térmica, acústica y lumínica) que se adaptan dinámicamente al usuario. Esta personalización no solo optimiza la eficiencia energética a través de un control térmico preciso, sino que mejora el bienestar del conductor y pasajeros al mitigar frecuencias para reducir el estrés auditivo e implementar iluminación adecuada que minimice la fatiga visual.

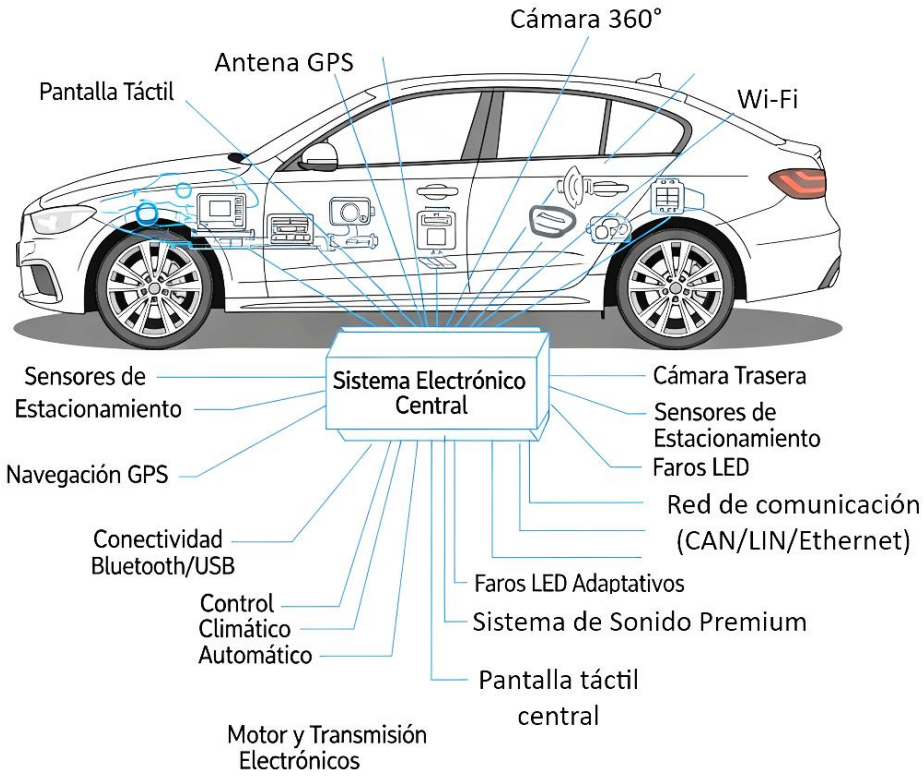


Figura 3. Representación de Funciones dentro del Sistema Electrónico Central del automóvil.

En la actualidad están surgiendo muchas funciones nuevas vinculadas a los sistemas de asistencia al conductor. El objetivo de las armadoras es crear un 'vehículo sensible' que utilice sensores y sistemas electrónicos para detectar e interpretar su entorno. El aprovechamiento de tecnologías de sensores por ultrasonido, radar y vídeo ha permitido desarrollar soluciones que desempeñan un papel fundamental en la asistencia al conductor; por ejemplo, mediante la mejora de la visión nocturna o el control de la distancia de seguridad como se ve en la Figura 4.

Diversos estudios sectoriales indican que los costos de producción automotriz han mantenido un crecimiento moderado, a pesar del ritmo acelerado de la innovación tecnológica. Actualmente, no se proyecta un incremento significativo en el valor de mercado de los sistemas mecánicos e hidráulicos convencionales, incluso ante un eventual aumento en los volúmenes de manufactura [17].

Este fenómeno se explica, primordialmente, por el cambio de paradigma hacia la electrificación de funciones. Este proceso consiste en la sustitución de componentes mecánicos por sistemas eléctricos y electrónicos, lo cual permite simplificar la arquitectura del vehículo y optimizar los procesos de ensamblaje. En la Figura 5 se presenta la evolución de este reemplazo tecnológico.

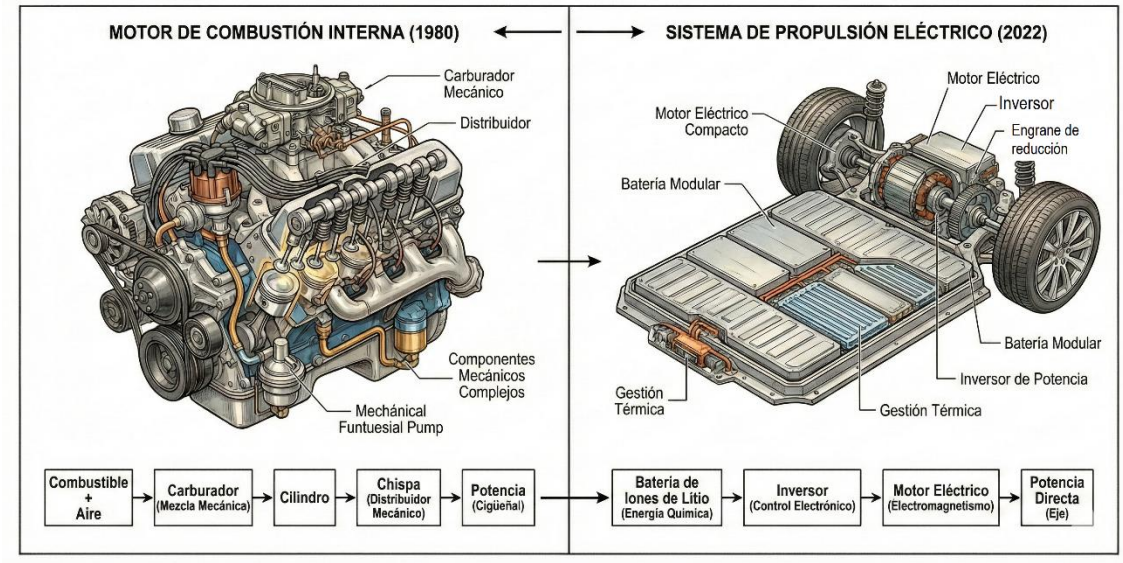


Figura 4. Evolución de introducción eléctrica al motor automotriz (1980 – Actualidad).

Los sistemas de control de frenado ilustran perfectamente esta evolución. Mientras que el esquema convencional dependía casi en su totalidad de elementos mecánicos, la integración del sistema ABS (Antilock Braking System) introdujo una mayor proporción de componentes electrónicos, tales como sensores y unidades de control (ECU). Este auge de la electrónica está intrínsecamente ligado al crecimiento del software. Hoy en día, los costos de desarrollo de software han dejado de ser marginales frente a los de hardware. En este contexto, la ingeniería de software enfrenta dos grandes desafíos derivados de la creciente complejidad sistémica del vehículo: la gestión del volumen de trabajo y el diseño de una arquitectura robusta y claramente estructurada.

En resumen, esta transición del componente mecánico al software define la era actual, pero es solo la etapa más reciente de una larga evolución. Considerando que se han lanzado millones de unidades desde los inicios del automóvil, resulta útil dividir los aproximadamente 125 años de historia en etapas. Para ello, nos basaremos tanto en las características técnicas de los coches como en su impacto cultural. En la tabla 1, se detallan estos hitos históricos, integrando la experiencia técnica con la evolución del uso del automóvil en la sociedad.

Tabla 1. Evolución de la industria automotriz 1880-Actualidad divido en 5 fases.

Periodo	Descripción
Inicios y Estandarización (1880 - 1917)	Esta fase se caracterizó por la experimentación técnica y el surgimiento de numerosas compañías. Con el tiempo, se consideró un diseño estándar: vehículos familiares, grandes y pesados, construidos predominantemente de forma artesanal hasta la llegada de la producción en cadena de Henry Ford.
Adaptación y Configuración Básica (1917 - 1940)	En esta época se estableció la configuración que conocemos hoy: el motor de combustión interna como estándar y la transición hacia el coche familiar pequeño y cerrado (sedán), lo que permitió el uso del vehículo en cualquier clima, dejando atrás los carruajes abiertos.
Confort y Motorización Masiva (1945 - 1973)	Fue la era del automóvil familiar asequible y potente. El enfoque principal fue la comodidad del usuario, introduciendo innovaciones como la dirección hidráulica, el aire acondicionado y transmisiones automáticas.
Conciencia Energética y Automatización (1973 - 2000)	Debido a crisis de petróleo a nivel global, la prioridad pasó a ser la eficiencia de combustible y la reducción de emisiones contaminantes. En esta etapa surgió la electrónica automotriz (sensores, inyección electrónica).
Electrificación y Digitalización (2001 - Actualidad)	Impulsada por la lucha contra el cambio climático, esta era marca el fin de la hegemonía absoluta del petróleo. El coche moderno se define como un "dispositivo inteligente con ruedas", donde el software es tan importante como la mecánica. Los sistemas de propulsión alternativos (híbridos y eléctricos) representan un cambio de paradigma hacia la movilidad sostenible y la conducción autónoma.

Por último, para ilustrar esta transformación eléctrica-electrónica es la comparativa entre el Ford Modelo T y el Mustang Mach-E. El Ford Modelo T, lanzado en 1908, no sólo introdujo la producción en masa y la democratización del transporte personal hasta el final de su fabricación en 1927. Durante ese periodo inicial, la innovación se centraba en la durabilidad mecánica y la eficiencia de los motores de combustión interna, estableciendo una base que dominó el mercado por más de un siglo. No obstante, la transición hacia la modernidad alcanzó un hito disruptivo con la llegada del Mustang Mach-E en 2021, que simboliza el cambio de paradigma hacia la movilidad eléctrica y la digitalización total. Mientras que en 1908 el avance consistía en reemplazar los carruajes de caballos por engranajes y acero, la tecnología actual del Mach-E integra inteligencia artificial, actualizaciones de software inalámbricas y una arquitectura de baterías de última generación, evidenciando que el automóvil ha evolucionado de ser una herramienta de transporte puramente mecánica a convertirse en un ecosistema digital inteligente y sostenible, ver Figura 6.

Esta evolución evidencia que el automóvil ha dejado de ser una herramienta de transporte puramente mecánica para convertirse en un ecosistema digital inteligente y sostenible. El pilar técnico que sustenta esta transformación es la función del sistema electrónico automotriz, cuya arquitectura permite que el vehículo procese información de manera autónoma, descrita a continuación.

Figura 5(a). Automóvil Ford modelo T lanzado en 1908.


Figura 6(b). Automóvil Ford modelo Mach-E lanzado en 2021.

3.1.1 Control y gestión de señales en redes electrónicas modernas

La unidad de control (ECU) actúa como el cerebro de cualquier sistema electrónico automotriz, ya que es el lugar donde se ejecutan todos los algoritmos de control, tanto de lazo abierto como de lazo cerrado. El componente principal de esta unidad es un microcontrolador de rango automotriz (AEC−Q101), el cual contiene una memoria (Flash EPROM) con las instrucciones necesarias para que el sistema funcione (Figura 7).

El proceso comienza cuando la unidad recibe información de los sensores, estos datos de entrada alimentan los algoritmos internos del microcontrolador, los cuales calculan de forma constante qué acciones debe tomar el sistema. Una vez procesada la información, el microcontrolador envía señales eléctricas de activación hacia los actuadores de la etapa de salida.

Los actuadores convierten las señales eléctricas (emitidas por el microcontrolador y amplificadas en los módulos de la etapa de salida) en variables mecánicas. Esto podría ser, por ejemplo, energía mecánica generada por un servomotor (como en un elevalunas eléctrico) o energía térmica generada por una bujía de incandescencia (calentador).

Si bien la unidad de control opera como el cerebro individual de un sistema específico, la complejidad de los vehículos modernos exige que estas unidades no funcionen de manera aislada. En la práctica, múltiples sistemas influyen entre sí de manera mutua para optimizar el rendimiento y la seguridad como se describe a continuación.

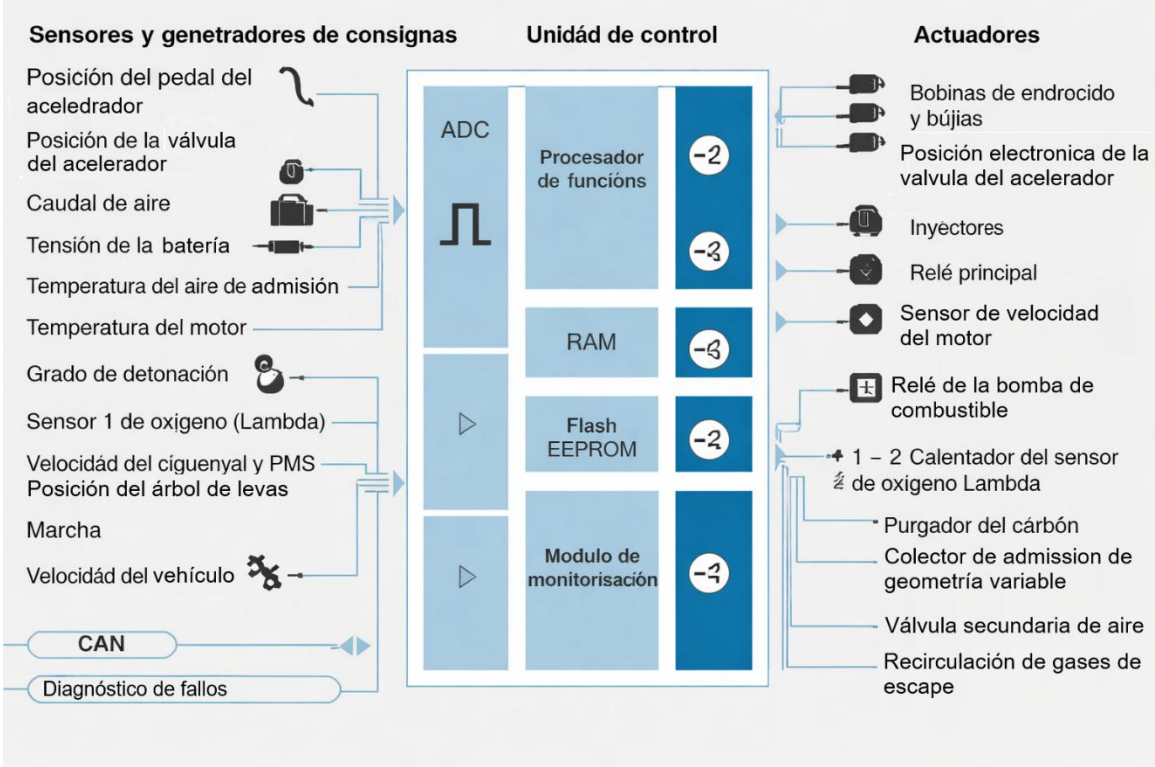


Figura 7. Módulos de función de un sistema electrónico.

3.1.2 Procesamiento de datos en sistemas automotrices actuales

Como se mencionó anteriormente, a la velocidad a la que avanza la tecnología electrónica/digital, el número de sistemas electrónicos en uso aumenta cada vez más. Este crecimiento también continúa en la ingeniería automotriz. Sin embargo, esto también significa que la complejidad del sistema global va en aumento.

Si bien componentes individuales como la gestión del motor han evolucionado significativamente, hoy las innovaciones surgen principalmente de la interacción entre diversos sistemas. Por ello, es fundamental que los componentes operen en red; esto permite que el gran volumen de datos gestionado por un módulo sea aprovechado por el resto del vehículo. Para lograrlo, se emplean distintos protocolos de comunicación adaptados a necesidades específicas de fiabilidad, tolerancia a fallos y costos.

En este contexto, se entiende por **red** a un sistema en el que un grupo de elementos intercambia información a través de un medio de transporte. Si visualizamos estos elementos como "nodos" y sus relaciones de comunicación como "líneas", obtenemos la imagen de una red donde múltiples nodos están interconectados entre sí, ver Figura 8.

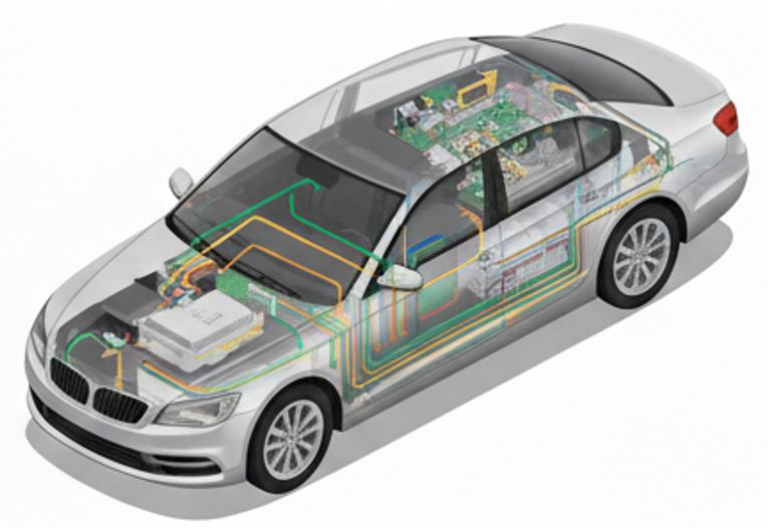


Figura 8. Representación de redes de sistemas electrónicos en vehículos.

En los vehículos motorizados, estas estaciones pueden ser unidades de control complejas, tales como el sistema de gestión del motor, el programa electrónico de estabilidad (ESP), el sistema de control de la transmisión o los módulos de las puertas. Sin embargo, un sensor equipado con un circuito de acondicionamiento que procesa y digitaliza un valor de medición también puede actuar como parte de la red, poniendo sus señales a disposición de los demás componentes. El medio físico a través del cual se realiza este intercambio de datos se conoce como bus o bus de datos.

Esta topología de red también se conoce como bus lineal (ver Figura 9). El elemento central de una topología de bus es un único cable al cual se conectan todos los nodos mediante cables de conexión cortos. Esta topología facilita enormemente la incorporación de otros suscriptores (dispositivos) a la red. La información es transmitida por los suscriptores individuales del bus en forma de los denominados 'mensajes' y se distribuye a través de todo el bus. Los nodos transmiten y reciben estos mensajes. Si un nodo falla, los datos que se esperan de dicho nodo dejan de estar disponibles para los demás integrantes de la red; sin embargo, los nodos restantes pueden seguir intercambiando información entre sí. Por el contrario, una red con topología de bus falla por completo si la línea principal está defectuosa (por ejemplo, debido a una rotura del cable).

Figura 9. Topología Bus Lineal.

3.2 Arquitecturas de comunicación vehicular: Protocolos y herramientas de diagnóstico actuales

Como se vio en la sección 2.1, la arquitectura electrónica vehicular ha pasado de ser un conjunto de sistemas independientes a una red compleja y altamente interconectada. En sus inicios, la comunicación entre sistemas críticos como la inyección y el encendido se resolvía mediante líneas de señal simples; sin embargo, la integración masiva de nuevas funciones electrónicas saturó rápidamente esta capacidad física. Esta saturación se hace evidente al examinar las señales que se procesan en los sistemas actuales. Hoy en día, una misma variable es requerida por múltiples unidades de control: por ejemplo, la velocidad de conducción debe ser evaluada simultáneamente por el Programa Electrónico de Estabilidad (ESP) para el control dinámico, por la gestión del motor para el control de crucero, y por el sistema de sonido para el ajuste automático del volumen.

La preparación de estas variables a partir de señales de sensores requiere una potencia de cálculo significativa, consumiendo valiosos recursos de hardware y software. Por lo tanto, resulta técnica y económicamente aconsejable que estas variables se calculen en una sola unidad de control y se transmitan a las demás a través de una red de comunicación eficiente, evitando la duplicidad de componentes.

Un ejemplo crítico y actual de esta interacción ocurre durante una colisión inminente: al detectar el evento, los sensores de precolisión activan la unidad de control del airbag, la cual envía una orden inmediata a los módulos de las puertas y del techo para cerrar las ventanillas y el techo corredizo, protegiendo así a los ocupantes. Bajo esta misma lógica de integración, funciones como el Control de Crucero Adaptativo (ACC) exigen una coordinación precisa entre el radar, la gestión del motor, el ESP y la transmisión, permitiendo ajustar el par motor y los frenos en tiempo real según las condiciones del tráfico.

Ante este desafío, surge la necesidad de implementar **protocolos de comunicación automotriz**, definidos como lenguajes estandarizados que gestionan el intercambio masivo de datos entre las Unidades de Control Electrónico (ECUs).

Estos protocolos no solo optimizan el espacio y la complejidad del cableado, sino que son fundamentales para la fiabilidad del sistema. Un ejemplo de estos es el protocolo LIN (*Local Interconnect Network*), un estándar de bajo costo para la comunicación de sistemas no críticos como el control de ventanas, espejos y luces interiores. Para las fases de desarrollo y validación en un nuevo proyecto automotriz, los ingenieros utilizan herramientas especializadas que se adaptan a la complejidad de cada etapa.

Tres dispositivos actuales utilizados en este campo son: Baby LIN de la empresa Lipowsky® es ideal para pruebas iniciales y desarrollos básicos, ya que su costo y facilidad de uso lo hacen perfecto para el prototipado. Por otro lado, los módulos de National Instruments® (NI) LIN están orientados a la integración en líneas de producción. Finalmente, CANoe de Vector® representa la solución profesional de la industria, utilizada para simulación avanzada y validación de sistemas y subsistemas con el soporte de diferentes protocolos [1].

Tabla 2. Controladores Automotrices para Sistemas LIN.

Característica	BabyLIN-II	USB-8476	CANoe
Fabricante	Lipowsky Industrie-Elektronik	National Instruments	Vector Informatik
Tipo de dispositivo	Software y hardware para simulación.	Software y hardware para simulación.	Software y hardware para simulación.
Protocolos soportados	LIN	LIN	LIN, CAN, FlexRay, Ethernet
Conector para comunicación PC	USB 2.0	USB 2.0	DB9
Aplicaciones	Simulación, pruebas automotrices.	Producción, simulación, pruebas de validación automotrices.	Desarrollo, simulación, pruebas de validación automotrices.
Compatibilidad de software	Baby-LIN-Tool (propietario)	NI-XNET, LabVIEW (propietario)	CANoe (Vector) (propietario)
Nivel de programación	Bajo (Contiene software desarrollado por fabricante)	Alto (Se necesita desarrollar programación en LabVIEW)	Medio (Contiene software desarrollado por fabricante, se requiere conocimiento previo LIN, CAN)
Precio estimado (2025)	~550 EUR (~10,500 MXN)	~600–800 USD (~11,000–14,500 MXN)	~2,000 EUR (~43,000 MXN)

Dispositivo Baby LIN-II

Baby-LIN® es un dispositivo compacto diseñado para la comunicación y control de redes LIN a través de una conexión USB y una interfaz desarrollada por el mismo fabricante (ver Figura 10). Se utiliza ampliamente en entornos de prueba y validación de sistemas electrónicos automotrices que emplean el protocolo LIN. Este dispositivo permite que el dispositivo PC o laptop actúe como monitor LIN, maestro LIN o esclavo LIN, ofreciendo gran flexibilidad durante el proceso de prueba. Una de sus principales ventajas es la opción de utilizar la memoria interna del dispositivo para funcionar en modo autónomo, lo que significa que puede ejecutar secuencias predefinidas de forma continua sin necesidad de estar conectado a una computadora. Esto resulta útil para pruebas prolongadas o automatizadas. Gracias a su tamaño reducido y versatilidad, el Baby-LIN es una herramienta muy utilizada en laboratorios de prueba, bancos de simulación y entornos de validación automotriz. Su precio aproximado en el año 2025 ronda los €550 euros (≈$12,015 MXN), lo que lo posiciona como una opción competitiva dentro del mercado de herramientas para análisis y control de redes LIN [3].

Figura 10. Diagrama general de sistema Baby-LIN-II.

Dispositivo NI USB 8476

El NI USB-8476® de National Instruments es una interfaz de comunicación LIN que permite a una computadora actuar como nodo maestro o esclavo dentro de una red LIN (ver Figura 11). Se conecta mediante un conector DB9 de 9 pines y no requiere alimentación externa, ya que opera directamente con la energía suministrada por el puerto USB. Esto facilita su uso en entornos de laboratorios, pruebas automotrices, desarrollos de prototipos y también en estaciones o "líneas" de producción, donde se requiere una interfaz confiable, portátil y fácil de integrar con sistemas existentes de uso industrial como los PLC. El dispositivo es compatible con los entornos de desarrollo como LabVIEW, lo que permite configurar, monitorear y registrar tramas LIN de forma sencilla, además de realizar pruebas funcionales o de diagnóstico sobre la red LIN diagnosticada [4].

Figura 11. Diagrama general de sistema NI USB 8476.

Dispositivo CANoe

El dispositivo VN1630A de Vector Informatik es ampliamente utilizado en actividades de desarrollo y validación de sistemas electrónicos automotrices, especialmente en el diseño y prueba de unidades de control electrónico (ver Figura 12). Permite configurar nodos maestros o esclavos en redes LIN, simular tráfico de red y analizar el comportamiento de la comunicación en tiempo real. El VN1630A permite trabajar con hasta con dos canales configurables de manera individual como LIN o CAN. Estos canales se conectan mediante conectores estándar DB9. El dispositivo se conecta a la computadora mediante puerto USB. Este dispositivo se puede utilizar en una amplia gama de aplicaciones, que abarcan desde el análisis básico de buses de comunicación hasta simulaciones avanzadas, así como tareas de diagnóstico, ajuste de parámetros y programación de unidades de control electrónico. Gracias a su compatibilidad con diversas herramientas, es posible ejecutar varios programas en paralelo, incluso en el mismo canal y desde un solo equipo. Es importante mencionar que, para utilizarlo, se requiere una licencia de software para poder operarlo con herramientas como CANalyzer o CANoe (ambos desarrollados por Vector). El precio de una licencia básica de ronda los 2,000 a 3,000 dólares, dependiendo de los módulos incluidos, lo que equivale un costo total aproximado de $40,000 hasta $60,000 pesos mexicanos [8].

Figura 12. Diagrama general de sistema Vector VN1630A.

3.3 Estado actual de las interfaces de comunicación LIN para plataformas de desarrollo abiertas.

La LIN-Bus Breakout Board de la firma británica SK Pang Electronics representa un componente esencial en el estado del arte, actuando como el puente físico entre la lógica de un microcontrolador y la red eléctrica de un vehículo. En el diseño de sistemas electrónicos, existe una barrera crítica: mientras que los procesadores modernos operan con voltajes muy bajos y delicados (típicamente entre 3.3V y 5V), los sistemas de un automóvil funcionan bajo una red de 12V mucho más robusta pero también más "ruidosa" eléctricamente. Esta placa de desarrollo soluciona dicho conflicto integrando un transceptor especializado, el chip TJA1021 [28], que se encarga de traducir las señales digitales de baja potencia en pulsos de alta tensión.

Desde una perspectiva funcional, la importancia de este modelo radica en su versatilidad para adoptar dos roles distintos dentro de una red de datos. Puede configurarse como un "Nodo Maestro", encargado de dirigir y organizar la comunicación en el bus, o como un "Nodo Esclavo", que simplemente responde a las solicitudes de información; esta flexibilidad permite a los investigadores simular desde el comportamiento de un sensor de lluvia hasta el control de espejos eléctricos. Además, la placa de SK Pang no solo es un convertidor de señales, sino que actúa como un escudo de seguridad. El entorno de un motor genera constantes interferencias electromagnéticas y picos de voltaje que podrían destruir fácilmente un microcontrolador convencional. Gracias a su diseño industrial, esta interfaz incorpora mecanismos de protección contra descargas electrostáticas y cortocircuitos, garantizando que la comunicación se mantenga estable y segura durante las pruebas experimentales. Por estas razones, se ha consolidado como una herramienta de referencia que permite validar sistemas de control de bajo costo con el mismo rigor y fiabilidad que exigen los estándares de la industria automotriz global. Con un precio de lista de aproximadamente £16.50 (unos $395.00 MXN), y un costo total de adquisición que ronda los $1,200.00 MXN tras considerar gastos de importación y logística hacia México, se sitúa como una alternativa sumamente competitiva [29].

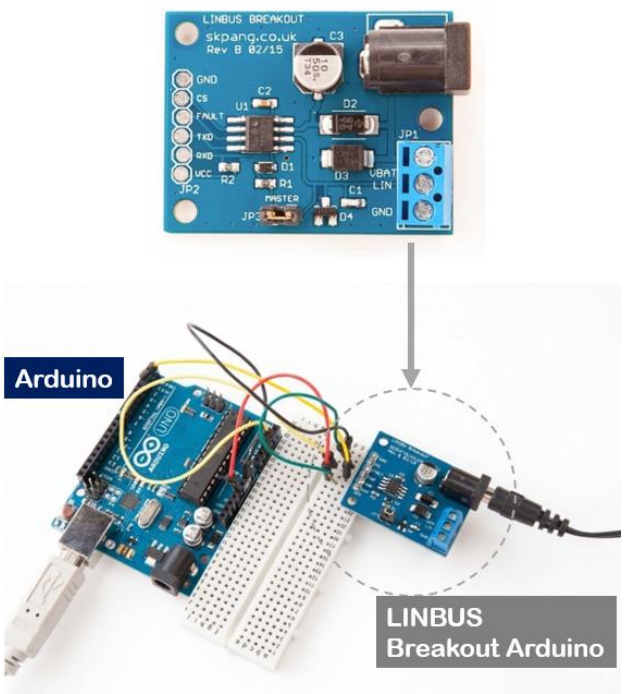


Figura 13. Controlador para Arduino LIN-Bus Breakout Board.

4. MARCO TEÓRICO

4.1 Protocolos de comunicación

El lenguaje empleado en la transmisión de datos se denomina protocolo de comunicación, se trata de un conjunto de reglas determinadas, que deben conocer tanto el emisor como el receptor.

Estos protocolos definen los elementos básicos de la comunicación, como el conjunto de caracteres utilizados, la estructura y secuencia de los mensajes, así como los mecanismos para detectar y corregir errores durante la transmisión. La asignación de códigos binarios a cada carácter permite la representación digital de la información y su transmisión a través de canales de comunicación. La correcta implementación de un protocolo garantiza la fiabilidad y eficiencia en la transmisión de datos [7].

4.2 Interfaces y protocolos de comunicación seriales

Los avances tecnológicos han llevado a la creación de una amplia gama de interfaces de comunicación para sistemas electrónicos, cada una adaptada a distintas necesidades y aplicaciones. Factores como la velocidad de transferencia de datos, la distancia entre dispositivos, la cantidad de elementos conectados al bus y el costo de implementación son determinantes en la elección de una interfaz particular. A lo largo de la historia, se han desarrollado numerosas interfaces de comunicación, destacando las siguientes:

4.2.1 SPI (Serial Peripheral Interface)

La interfaz de comunicación serial periférica (SPI) es un protocolo de comunicación síncrono ampliamente empleado en sistemas electrónicos para interconectar microcontroladores con una variedad de periféricos. Este estándar de comunicación se caracteriza por su simplicidad y eficiencia, lo que lo convierte en una elección popular en aplicaciones donde se requiere una transferencia de datos rápida y confiable a corta distancia. Tiene una configuración maestro/receptor, en el que el primero controla la comunicación y el receptor responde a las órdenes del maestro. Utiliza un número limitado de cuatro señales (MOSI, MISO, SCLK y SS/CS, ver Figura 14). Tiene una velocidad de 10 Mbps [6].

Tabla 2. Señales de protocolo de comunicación SPI.

Tipo de señal	Descripción
MOSI (Master Output/Slave Input)	Maestro envía datos a receptor.
MISO (Master Input/Slave Output)	Receptor envía datos al maestro.
SCLK (Clock)	Reloj proporciona los pulsos que sincronizan la comunicación.
SS/CS (Slave Select/Chip Select)	Selecciona un receptor especifico en un bus SPI con múltiples receptores.

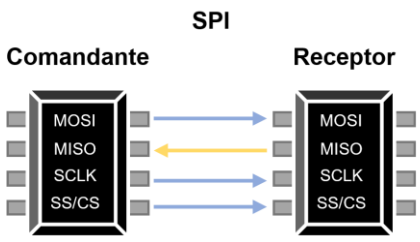


Figura 14. Representación gráfica de comunicación SPI

En la comunicación serial, los bits son mandados uno a uno por un solo cable. La Figura 15 muestra un ejemplo de transmisión de datos, el maestro manda la información un bit a tiempo a través de la línea MISO.

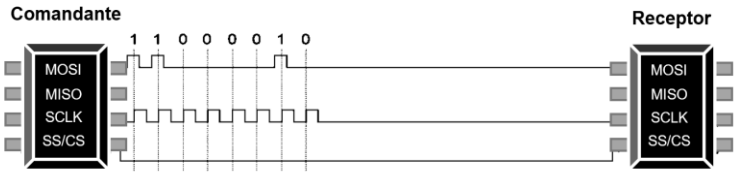


Figura 15. Representación de comunicación serial SPI.

4.2.2 UART (Universal Asynchronous Receiver/Transmitter)

En español se traduce como Transmisor/Receptor Asíncrono Universal. En un enlace UART, basta únicamente dos líneas de señal para transmitir información entre ambos dispositivos. Los datos son enviados desde el pin de transmisión (TX) del UART emisor hacia el pin de recepción (RX) del UART receptor, ver Figura 16.

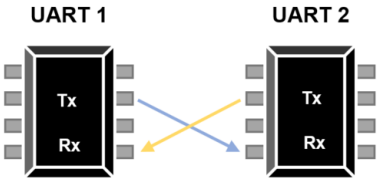


Figura 16. Representación gráfica de comunicación UART.

La comunicación UART es asíncrona, es decir, no requiere una señal de reloj común para sincronizar la transmisión y recepción de datos. En su lugar, el UART transmisor incorpora bits de inicio y parada al flujo de datos, delimitando el comienzo y el final de cada trama. Estos bits sirven como referencia temporal para que el receptor muestre los datos en el momento preciso [6].

Al detectar el bit de inicio, el receptor UART inicia el muestreo de los datos entrantes a una tasa fija conocida como velocidad en baudios. Esta velocidad, medida en bits por segundo, representa la cantidad de bits transmitidos en un segundo. Para una comunicación exitosa, ambos UART deben estar configurados con la misma velocidad, permitiendo una desviación máxima del 10%. La información se transmite en marcos, cada uno compuesto por un bit de inicio, entre 5 y 9 bits de datos, opcionalmente un bit de paridad y uno o dos bits de parada, como se muestra en la Figura 17.

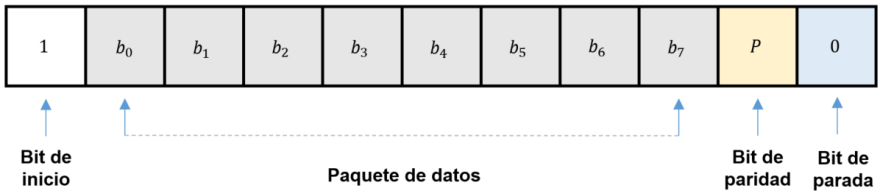


Figura 17. Formato de paquete de datos UART.

- **Bit de inicio (Start Bit):** La línea de transmisión UART permanece en un estado de marca lógica '1' (alto nivel de voltaje) en ausencia de datos. Para indicar el comienzo de un nuevo marco de datos, el transmisor genera un flanco de bajada, transaccionando la línea de un estado alto a uno bajo. Este evento, conocido como bit de inicio, señala al receptor que debe sincronizarse con la señal y comenzar a muestrear los datos entrantes a la frecuencia de baudios preconfigurada.

- **Paquete de datos (Data Frame):** El campo de datos de un marco UART, que contiene la información a transmitir, tiene una longitud variable que depende de si se utiliza o no un bit de paridad. En el caso de emplear paridad, la longitud del campo de datos está comprendida entre 5 y 8 bits. Si no se emplea paridad, la longitud máxima del campo de datos es de 9 bits. Convencionalmente, los datos se transmiten con el bit menos significativo al principio.

- **Paridad (Parity):** La paridad es una técnica de verificación de integridad de datos utilizado en comunicaciones serie. Consiste en un bit adicional, denominado bit de paridad, que se añade al final de un marco ("frame") de datos. El valor de este bit se establece de manera que el número total de bits con valor 1 en el marco sea par (paridad par) o impar (paridad impar). Al recibir un marco, el receptor recalcula la paridad y la compara con el valor recibido. Si ambas paridades coinciden, se asume que los datos se han transmitido sin errores. En caso contrario, se detecta una condición de error, lo que indica que al menos un bit ha sido alterado durante la transmisión.

- **Bit de parada (Stop bits):** El indicio del final de un paquete de datos en una comunicación UART se realiza mediante un período de marca lógica '1' (alto nivel de voltaje) de una duración mínima equivalente a dos bits. Este espacio en blanco, conocido como bit de parada, permite al receptor sincronizarse para la recepción del siguiente marco y detectar cualquier condición de error relacionada con la pérdida de sincronización.

4.2.3 RS232

Es un protocolo de comunicación serial que transmite información de manera serial en una sola dirección. En orden de establecer la comunicación transmisión/recepción son al menos necesarios 3 cables (RX, TX y GND) ver Figura 18.

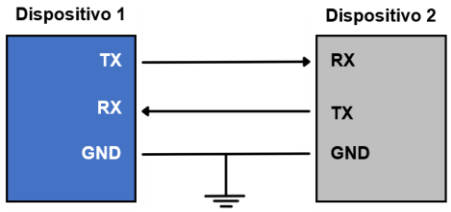


Figura 18. Representación comunicación RS232.

El protocolo es de comunicación asíncrona. Se utilizan bits de "start" y "stop" para informar al receptor cuando recibir la información. La lógica del protocolo funciona mediante la configuración (-12) Volts representan "1" y (+12) Volts representan "0", ver Figura 19.

Figura 19. Representación de niveles de voltaje RS232.

El protocolo RS232 tiene la capacidad de transmitir un máximo de 8 bits. Si el bit de paridad no se utiliza se puede alcanzar un total de 9 bits, y después de alcanzar el límite máximo el transmisor envía el bit de "stop", como referencia ver Figura 16. Este protocolo soporta una velocidad máxima de 20kbps y la máxima longitud de cable es de 20 metros [16].

4.3 Protocolos de comunicación automotrices

Una red de comunicación vehicular es un sistema integrado por diversos dispositivos electrónicos (Unidades de Control Electrónico o ECU) que se encuentran interconectados, ya sea a través de cables o de forma inalámbrica. Esta red facilita el intercambio de información digital entre las distintas ECU, permitiendo así la gestión eficiente de los múltiples sistemas que componen un vehículo (ver Figura 20). Las ECU son los "cerebros" que controlan funciones específicas del vehículo, desde el motor hasta los sistemas de seguridad. A través de señales eléctricas, los ECU intercambian datos en formato digital (codificados en lenguaje binario). Esta comunicación digitalizada permite un procesamiento rápido y preciso de la información, lo que es fundamental para el adecuado funcionamiento del vehículo. El objetivo principal de una red de comunicación vehicular es garantizar la coordinación y el control óptimo de todos los sistemas del vehículo. Al permitir un flujo constante de información entre las distintas ECU, se logra una mayor eficiencia, seguridad y comodidad para el conductor y los pasajeros.

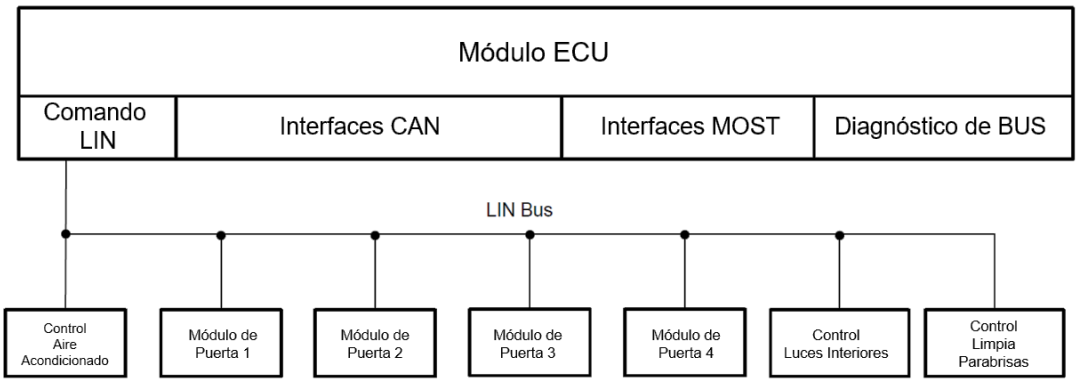


Figura 20. Diagrama de arquitectura de aplicación protocolos automotrices [6].

En los últimos años, la industria automotriz ha experimentado un notable incremento en la integración de componentes electrónicos. Esta tendencia ascendente ha permitido a los fabricantes desarrollar vehículos con mayor seguridad y fiabilidad. La creciente demanda de nuevas funcionalidades ha impulsado el desarrollo de Unidades de Control Electrónico (ECU) cada vez más sofisticadas. Estas ECU, que superan las 100 unidades en algunos modelos de alta gama, requieren una comunicación interconectada eficiente para garantizar el correcto funcionamiento de los diversos sistemas del vehículo. La implementación de protocolos de comunicación ha sido fundamental en este proceso. Al reemplazar sistemas mecánicos tradicionales por módulos electrónicos, se han logrado reducciones significativas en costos, peso total del vehículo y consumo de combustible [15].

Un sistema de bus serie se utilizó por primera vez en un vehículo en 1991, cuando se empleó el bus CAN en el Mercedes-Benz 500E [16]. La demanda de mayor seguridad en la conducción, confort, economía y requisitos legales más estrictos sobre la compatibilidad medioambiental de los vehículos de motor solo puede lograrse con la ayuda de electrónica adicional. Por lo tanto, el número de sistemas electrónicos en los vehículos aumenta constantemente (Ver Figura 21).

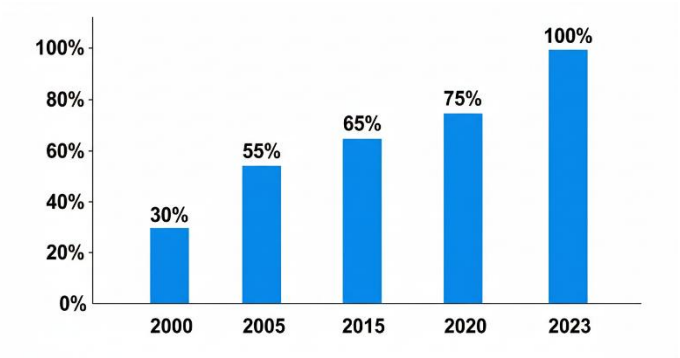


Figura 21. Incremento del uso de protocolos automotrices en vehículos.

La SAE (Society of Automotive Engineers) ha establecido una clasificación para los protocolos de comunicación utilizados en la industria automotriz, dividiéndolos en cuatro categorías principales (A, B, C y D) según sus características y capacidad de transmisión de datos, (ver Tabla 2) [17].

- **Clases A y B (Non safety critical):**
  Estas clases engloban aplicaciones que no son críticas para la seguridad del vehículo y no requieren una respuesta inmediata, como el control de ventanas eléctricas, la iluminación interior o los sensores de lluvia. Debido a que estas funciones demandan una menor cantidad de datos, utilizan protocolos de comunicación menos exigentes, como LIN (Local Interconnected Network).

- **Clase C (Safety critical):**
  Esta categoría incluye sistemas que son cruciales para la seguridad del vehículo y requieren una respuesta en tiempo real, como los sistemas ABS (Antiblock Braking System), ASR (Anti-Slip Regulation) y ESP (Electronic Stability Program). Estos sistemas demandan altas velocidades de transmisión de datos y tiempos de respuesta extremadamente bajos, por lo que utilizan protocolos más robustos como CAN (Controller Area Network).

- **Clase D (Multimedia):**
  Esta categoría está destinada a aplicaciones multimedia, entretenimiento y control en tiempo real que requieren velocidades de transmisión aún mayores (superiores a 1 Mbps). Un ejemplo de protocolo utilizado en esta clase es MOST (Media Oriented System Transport).

Tabla 3. Clasificación SAE de protocolos de comunicación automotrices.

Clase	Tipo	Clasificación	Velocidad de transmisión	Configuración	Aplicaciones (Ejemplos)
A	LIN	No critica	20 kbps	1 cable	Control de ventanas eléctricas, iluminación de interiores.
B/C	CAN	Critica	1 Mbps	2 cables	ABS, ASR, ESP.
D	MOST	Alta velocidad	25 Mbps	Fibra Óptica	Sistemas multimedia, sistemas de control en tiempo real.

Como ya se ha señalado para que un auto moderno funcione bien, sus diferentes partes (como el motor, los frenos y las luces) necesitan hablar entre sí y compartir mucha información rápidamente. El uso de los sistemas de bus presenta las siguientes ventajas en comparación con una solución que utiliza el cableado convencional:

- Reducción de costos con menor peso y espacio de instalación debido a la menor cantidad de cables en el mazo de conductores (arnés).
- Mejor confiabilidad y seguridad operativa gracias a la reducción de conexiones de enchufe.
- Simplificación del ensamblaje del vehículo durante la producción.

- Conexión sencilla de los componentes del sistema a un bus.
- Manejo intuitivo del equipamiento y de sus variantes en un vehículo.

Al seleccionar un sistema de bus, se deben tener en cuenta tanto las limitaciones financieras (por ejemplo, costes de cableado y de componentes) como las técnicas. A continuación, se explican los criterios de selección técnica más importantes.

4.3.1 Tasa de transferencia de datos

Esta variable especifica el volumen de datos que se transmite durante una unidad de tiempo. La unidad más pequeña de datos es el bit, y la tasa de transferencia suele especificarse en bits por segundo (bit/s). Otros nombres alternativos para esta expresión son: tasa de transferencia, velocidad de datos o tasa de bits.

4.3.2 Inmunidad a las interferencias

Idealmente, los datos deberían transferirse sin interferencias. Sin embargo, esto no se puede garantizar en un vehículo motorizado debido a los efectos electromagnéticos. Los requisitos de inmunidad a las interferencias que se establecen dependen de la relevancia para la seguridad de los sistemas electrónicos involucrados. Se exigen requisitos menores a los sistemas de confort y conveniencia que, por ejemplo, al sistema de frenos antibloqueo (ABS).

Para cumplir con estos requisitos, se incorporan mecanismos que detectan errores de transmisión en los protocolos de red. Se puede realizar una comprobación sencilla utilizando el bit de paridad, el cual se calcula en el transmisor y se transmite junto con los datos útiles. Este especifica si el número de unos en el byte transferido es par o impar. Esta información es comprobada por el receptor. Mediante este método se pueden detectar errores simples.

Otro método es la comprobación por suma de comprobación (checksum). Si se transmiten varios bytes de datos, el transmisor calcula una suma de comprobación a partir de los bytes de datos individuales utilizando una fórmula predefinida y transmite este valor. El receptor también calcula la suma de comprobación de los bytes de datos que se han recibido y la compara con la suma de comprobación que se ha recibido. Si se detecta un error de transmisión de datos, los datos recibidos no se utilizan y se solicita una retransmisión. El funcionamiento detallado de estos algoritmos y su implementación técnica se analizan en profundidad en las siguientes secciones.

4.3.3 Capacidad de tiempo real

Un sistema de tiempo real garantiza que sus resultados se calculen dentro de un intervalo de tiempo fijo. La duración de dicho intervalo depende de la aplicación. El sistema antibloqueo de frenos (ABS) debe reaccionar al bloqueo incipiente de una rueda en pocos milisegundos (reducción de la velocidad de la rueda), mientras que tiempos de respuesta de 100 ms son adecuados para accionar el motor del elevalunas eléctrico. Los seres humanos no pueden percibir periodos de retraso inferiores a 100 ms. Se plantean diferentes exigencias al comportamiento en tiempo real según la aplicación:

- Requisito de tiempo real flexible: El sistema generalmente se ajusta al tiempo de respuesta especificado y, si estos tiempos se exceden ocasionalmente, no se producen efectos graves (por ejemplo, tirones en la imagen durante la transmisión de video).

- Requisito de tiempo real estricto: la especificación de tiempo debe cumplirse rigurosamente. Si se excediera el tiempo de respuesta especificado, el resultado calculado no podría utilizarse. Esto puede provocar problemas graves en sistemas críticos para la seguridad.

Por ejemplo, si se excedieran los márgenes de tiempo en el sistema ABS, el bloqueo incipiente de las ruedas no se detectaría lo suficientemente pronto y la presión en el cilindro maestro no se reduciría a tiempo, lo que resultaría en el bloqueo de las ruedas. Los márgenes de tiempo también deben cumplirse estrictamente en muchas funciones del sistema de gestión del motor. Los retrasos en la transmisión de las señales de inyección e encendido podrían provocar vibraciones en el motor e incluso fallos de encendido. Estas reacciones deben evitarse, ya que representan un peligro potencial. Por lo tanto, se deben exigir requisitos de tiempo real estricto a estos sistemas. Sin embargo, esto no significa necesariamente que la transmisión de datos a través de un sistema de bus también deba estar sujeta a estos requisitos de tiempo real estricto. El cumplimiento de los requisitos de tiempo real flexible suele ser suficiente. Si se necesitan señales de otras unidades de control para determinadas funciones (poniendo como caso, una solicitud de reducción de par durante una operación de cambio de marcha), el sistema de bus debe transmitir los datos a una velocidad de transferencia más rápida y con un menor retraso temporal para que el sistema global cumpla con los requisitos de tiempo real especificados.

4.3.4 Número de nodos de red

El número máximo de nodos a integrar varía según las diferentes áreas de operación del vehículo. El número de nodos para los sistemas de confort y conveniencia puede ser elevado debido a la interconexión de servomotores (por ejemplo, el ajuste de asientos) y sensores inteligentes (por ejemplo, sensores de lluvia). Si es necesario, se pueden utilizar varios buses idénticos.

4.4 Aplicaciones de protocolos automotrices en el vehículo

El sistema general del vehículo se puede dividir en cuatro dominios o áreas funcionales desde el punto de vista eléctrico/electrónico: tren motriz (o propulsión), chasis, interior y telemática, ver Figura 22. En los dominios del tren motriz y el chasis, el énfasis se pone principalmente en las aplicaciones en tiempo real. En el dominio del interior, el enfoque principal reside en los aspectos de multiplexado de las redes. Por último, en el dominio de la telemática, se conectan principalmente aplicaciones de multimedia y sistemas de info entretenimiento, ver Figura 22.

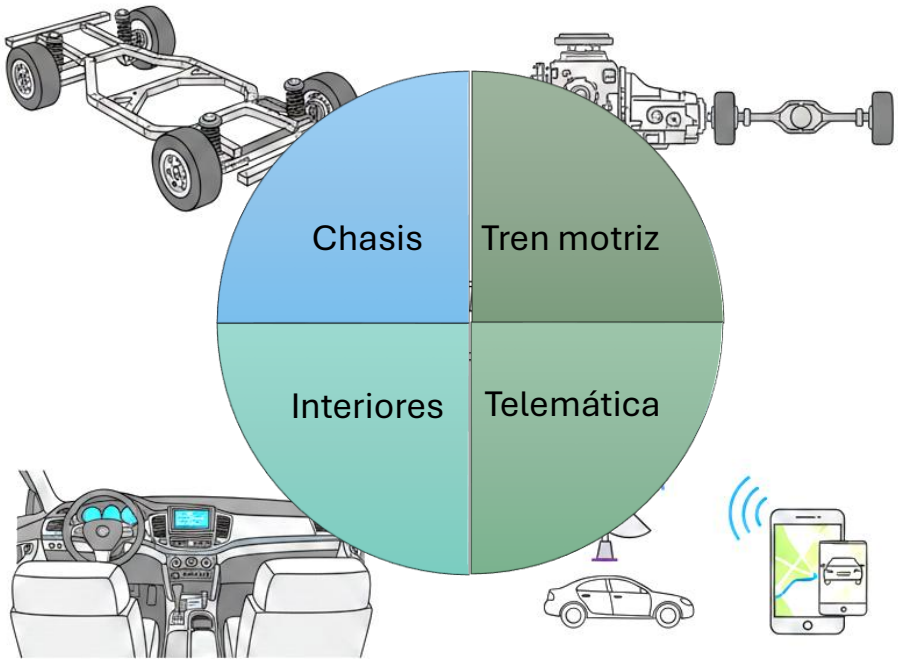


Figura 22. Dominios en el sistema general del vehículo

4.4.1 Aplicaciones en tiempo real

La interconexión de estos sistemas exige un alto nivel de rendimiento al sistema de comunicación. Son típicos los procesos sincronizados con el cigüeñal o procesos dentro de un intervalo de tiempo fijo con ciclos de apenas unos pocos milisegundos. Si los tiempos de respuesta del sistema son adecuados para la tarea en cuestión, se dice que el sistema tiene "capacidad de tiempo real" (por ejemplo, el rápido ajuste del avance del encendido en el sistema electrónico del motor tras una solicitud del control de tracción para reducir el par motor y evitar así que las ruedas patinen).

Los sistemas del tren motriz y del chasis se asignan a la "clase C". Estos requieren altas velocidades de transferencia para garantizar el comportamiento en tiempo real necesario para estas aplicaciones. También exigen una tolerancia a fallos considerable. Estos requisitos se cumplen mediante el bus CAN orientado a eventos, con una velocidad de transferencia de 500 kBaud (CAN de alta velocidad o *high-speed CAN*).

Algunos ejemplos incluyen:

- Sistema de gestión del motor.
- Control de la transmisión.
- Control de dinámica del vehículo: (por ejemplo, el programa electrónico de estabilidad, ESP).
- Sistemas de control del chasis: (por ejemplo, el control activo de la carrocería, ABC).
- Sistemas de asistencia: (por ejemplo, el control de crucero adaptativo, ACC).

4.4.2 Aplicaciones en redes multiplexadas

Las aplicaciones de multiplexado son adecuadas para el control y la regulación de componentes en el área de la electrónica de carrocería, confort y habitabilidad (clase B). Entre estos se incluyen las pantallas, la iluminación, la autorización de acceso con sistemas de alarma antirrobo, la climatización, el ajuste de asientos y espejos, los módulos de puerta (que integran los elevalunas eléctricos y el ajuste de los retrovisores), los limpiaparabrisas y la regulación de los faros.

Los requisitos de velocidad de transmisión para los sistemas de clase B no son tan exigentes como los de la clase C; por este motivo, se puede utilizar el bus CAN de baja velocidad (Low-Speed CAN) con una tasa de 125 kBit/s. Si los requisitos de velocidad descienden por debajo de los 20 kBit/s, suele emplearse con mayor frecuencia el bus LIN, que es más económico. Estas aplicaciones se sitúan principalmente en el ámbito de la mecatrónica, como ocurre con la transmisión de información de interruptores o la activación de actuadores.

4.4.3 Aplicaciones multimedia

La interconexión multimedia en aplicaciones de comunicación móvil integra componentes como sistemas de audio para el automóvil, sistemas de navegación, sistemas de información para el conductor, sistemas de video, comandos de voz, teléfono, internet y cámaras de reversa (ver Figura 23). La conexión en red de estos elementos permite disponer de una pantalla y una unidad de control centralizadas para múltiples aplicaciones. De esta manera, los procedimientos de operación pueden estandarizarse y la información de estado puede sintetizarse, lo que garantiza que se minimicen las distracciones del conductor.

En la interconexión multimedia, es fundamental distinguir entre los datos de control y los datos de audio o video. Para las tareas de control, como la selección de cambio de aplicación por ejemplo teléfono a navegador, bastan velocidades de transferencia de hasta 125 kBit/s, por lo que se puede emplear el bus CAN de baja velocidad. No obstante, la transmisión directa de datos de audio o video requiere velocidades de transferencia extremadamente altas, superiores a los 10 MBit/s; para este fin se utiliza, por ejemplo, el bus MOST [16].

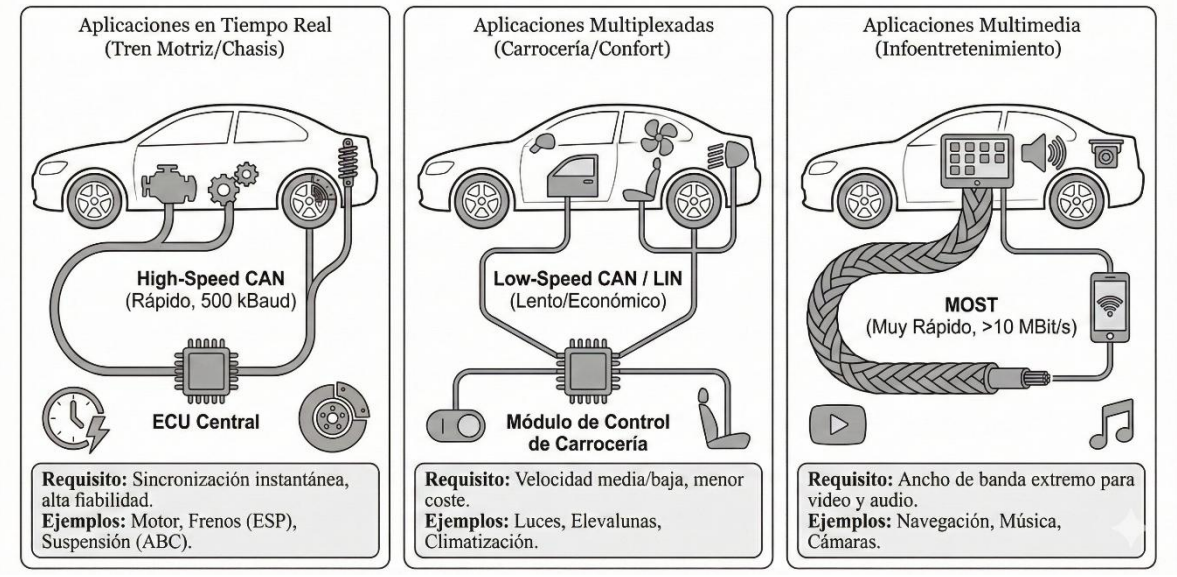


Figura 23. Aplicaciones de protocolos automotrices en el vehículo.

4.4.5 Ciberseguridad

De las aplicaciones mencionadas anteriormente podemos observar que la creciente integración de sistemas electrónicos complejos y la conectividad en la industria automotriz ha expandido significativamente la superficie de ataque en de los vehículos modernos, transformando la seguridad informática en un componente crítico de la ingeniería vehicular. Para mitigar estos riesgos y asegurar la homologación de vehículos, la industria se rige por marcos normativos estrictos, principalmente la norma ISO/SAE 21434, que estandariza la ingeniería de ciberseguridad a la largo del ciclo de la vida del vehículo. En esta sección se aborda la naturaleza de las acciones adversarias diseñadas para comprometer tanto la información del vehículo como sus capacidades operativas de seguridad y protección.

Desde una perspectiva técnica, las amenazas se clasifican fundamentalmente en ataques pasivos y activos. Los ataques pasivos se centran en el reconocimiento y la recopilación de información sin alterar el sistema, utilizando técnicas de escucha en redes internas (como CAN, o Ethernet) para registrar tráfico de datos o interceptar actualizaciones de software mediante tácticas mediante ataques de intermediarios (ver Figura 24), permitiendo al atacante identificar vulnerabilidad para una explotación futura. Por el contrario, los ataques activos implican una interferencia directa con el "objetivo" mediante la modificación, inserción o bloqueo de datos. Entre estos se destacan la suplantación de identidad, donde el atacante simula ser una fuente confiable; los ataques de repetición, que retransmiten datos antiguos para inducir comportamientos no deseados; y la manipulación del firmware o configuraciones de los ECU. Asimismo, se describen los ataques de denegación de servicio, orientados a agotar recursos computacionales o saturar el bus de comunicaciones, y los ataques de canal lateral, que explotan fugas físicas como variaciones de potencia o tiempo para poder manipular material criptográfico sensible.

Para contrarrestar estas vulnerabilidades y dar cumplimiento a los requisitos de la ISO/SAE 21434, es imperativo establecer y validar objetivos de seguridad rigurosos, categorizados en cinco clases fundamentales: integridad, autenticidad, confidencialidad, responsabilidad y disponibilidad. La integridad asegura que los archivos binarios de software y los mensajes de red no sean corrompidos ni manipulados; la autenticidad verifica la legitimidad de la fuente de los datos y la identidad de los clientes de diagnóstico; la confidencialidad protege la privacidad del usuario y la propiedad intelectual del fabricante; la responsabilidad garantiza que las acciones críticas, como el registro de datos de accidentes, sean irrefutables; y la disponibilidad asegura la continuidad operativa del sistema incluso bajo condiciones de ataque. Actualmente la criptografía es una de las herramientas fundamentales para proteger y autenticar la información ayudando a alcanzar estos objetivos de seguridad en los sistemas automotrices [23].

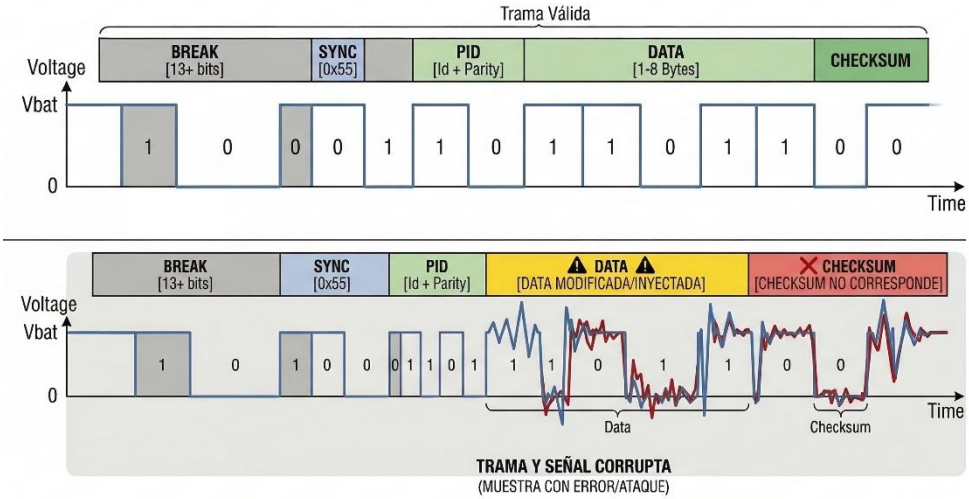


Figura 24. Comparativa detallada de señales físicas LIN: Normal vs. Corrupta.

4.5 Red de Interconexión Local (LIN)

LIN (*Local Interconnect Network*) es un concepto para redes automotrices de bajo costo que complementa el portafolio actual de redes multiplexadas en vehículos. LIN será el factor clave para la implementación de una red vehicular jerárquica, con el fin de lograr una mayor mejora en la calidad y una reducción de costos en los vehículos. La estandarización reducirá la variedad de soluciones multiplexadas de gama baja existentes y recortará los costos de desarrollo, producción, servicio y logística en la electrónica vehicular [10].

Se consideró que un sistema de bus con un protocolo sencillo y un control de secuencia simple permitiría utilizar incluso microcontroladores de baja capacidad sin necesidad de hardware adicional para la interfaz de comunicación. El nombre LIN deriva del hecho de que todas las unidades de control electrónico se encuentran dentro de un espacio de instalación delimitado (por ejemplo, en la puerta). Por lo tanto, el LIN es un subsistema local que sirve de apoyo a la red del vehículo mediante redes CAN de nivel superior.

3.5.1 Historia y aplicaciones

A finales de la década de 1990, con el objetivo de tener un protocolo de categoría secundaria estándar, fabricantes automotrices de la unión europea como BMW, Volkswagen, Audi, Volvo y Mercedes-Benz formaron un consorcio para definir un nuevo estándar de comunicaciones para el sector automotriz. El nuevo bus, llamado LIN, fue inventado para ser utilizado en aplicaciones de conmutación simples como iluminación interior, movimiento de techos solares y espejos, etc. (Ver Tabla 4).

En el año 2002 surgió la primera versión ampliamente adoptada, el LIN 1.3, la cual estableció las bases de la comunicación 'Maestro-Esclavo'. Al basarse en el protocolo UART, permitió el desarrollo sobre hardware ya conocido. Posteriormente, la versión LIN 2.0 introdujo archivos de configuración que facilitaron la interoperabilidad entre diferentes proveedores automotrices, esta actualización hace alusión a que los diseñadores del vehículo pueden integran diferentes componentes de distintos proveedores siempre y cuando se conserve el archivo LIN declarado por el fabricante. Finalmente, con la llegada de LIN 2.1 en 2006, se perfeccionó la gestión de energía de los nodos (modos *sleep* y *wake-up*), una mejora crítica para el ahorro de batería en los vehículos modernos. El estándar más reciente de LIN fue definido en 2010 (LIN 2.2A, por el Consorcio LIN). Posteriormente, fue transcrito a la Organización Internacional de Normalización (ISO) para ser aceptado como ISO 17897 y publicado oficialmente en 2016:

- Especificaciones de protocolo LIN
- Especificación del Lenguaje de Configuración LIN
- Especificación de la Interfaz Físicas de Aplicación LIN

Cada una de estas partes es necesaria para crear el clúster LIN completo de manera consistente en el mercado, permitiendo a cualquier fabricante de automóviles utilizar el esquema de comunicación. La especificación del protocolo describe las capas físicas y de enlace de datos, mientras que el lenguaje de configuración LIN permite que el clúster (armadora) LIN sea descrito en un archivo que es sencillo para cualquier desarrollador [2].

Hoy en día, el protocolo LIN es utilizado en amplias aplicaciones automotrices, como las descritas a continuación en la tabla 4 y Figura 25:

Tabla 4. Ejemplo de aplicaciones LIN BUS.

Sistema Automotriz	Subsistema LIN
Puerta	Espejos laterales, ventanas, cerraduras.
Volante	Control limpiaparabrisas, radio.
Asientos	Control motor de posición.
Tren motriz	Sensores velocidad y presión.
Accesorios	Techos solares, sensor lluvia.

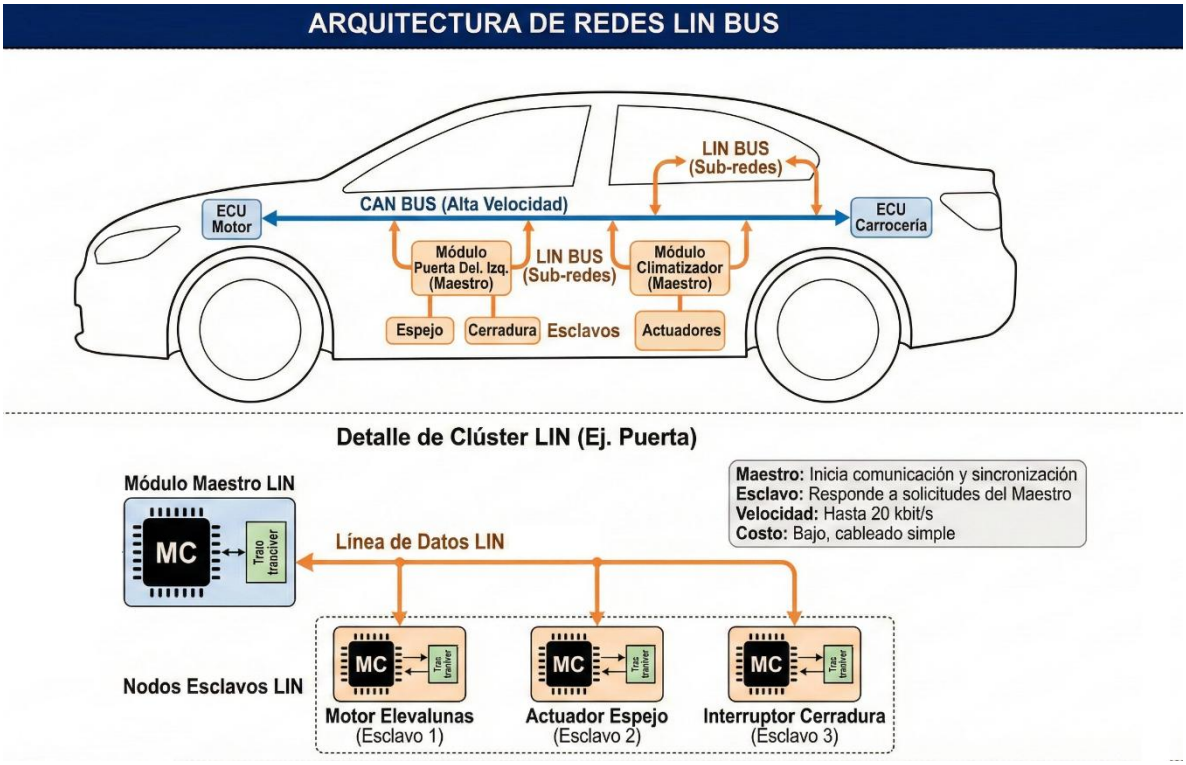


Figura 25. Representación de la comunicación LIN BUS en un automóvil.

El bus LIN es adecuado para tasas de datos bajas de hasta 20 kBit/s y, por lo general, está limitado a un máximo de 16 participantes. La interfaz eléctrica puede implementarse de manera sencilla y económica en los nodos de la red. En cuanto a los nodos, se hace una distinción entre el maestro, que generalmente es una unidad de control electrónica conectada a un sistema de bus de nivel superior, y los esclavos. Estos últimos son actuadores inteligentes, sensores inteligentes o, simplemente, interruptores con hardware adicional para la interfaz del bus LIN.

Los participantes del bus suelen organizarse en una topología de bus lineal y se conectan entre sí mediante una línea monofilar. No obstante, esta topología no está especificada de forma explícita. La comunicación en el bus LIN se realiza de manera síncrona en el tiempo, siendo el maestro quien define la trama temporal. En consecuencia, se produce una respuesta del bus LIN estrictamente determinista.

En resumen, las principales propiedades del bus LIN son:

- Concepto de maestro único con múltiples esclavos.
- Implementación de silicio de bajo costo basada en hardware de interfaz común UART/SCI, un equivalente en software o como una máquina de estados pura.
- Auto sincronización sin necesidad de resonador de cuarzo o cerámica en los nodos esclavos.
- Transmisión de señales determinista (transmisión que ocurre en tiempo exacto y predecible).
- Implementación de un solo hilo (un sólo cable) de bajo costo.
- Velocidad de hasta 20 kbit/s.
- Interacción de aplicaciones basada en señales.
- Re-configurabilidad.
- Soporte de capa de transporte y diagnóstico.

La popularidad del protocolo LIN (Local Interconnect Network) en la arquitectura electrónica automotriz actual no reside en su rendimiento bruto, sino en su optimización para aplicaciones de baja criticidad bajo los siguientes tres ejes fundamentales:

4.5.1.1 Ventajas protocolo LIN BUS

a) Optimización Estructural de Costos

El modelo maestro-esclavo centraliza la inteligencia de la red. Al delegar la gestión del bus y la precisión temporal al nodo maestro, los nodos esclavos pueden operar con microcontroladores de bajas prestaciones y prescindir de cristales de cuarzo externos, utilizando en su lugar osciladores RC internos. Esta arquitectura, sumada al uso de una capa física de un solo cable, minimiza tanto el costo de los semiconductores como el del mazo de cables y conectores.

b) Eficiencia en la Gestión de Datos y Complejidad

LIN está diseñado para servicios donde el ancho de banda no es un factor crítico, operando a velocidades de hasta 20 kbps. Esta limitación técnica es, en realidad, una ventaja de diseño: reduce drásticamente las emisiones electromagnéticas (EMI) y elimina la necesidad de hardware complejo para la detección y resolución de colisiones. Esto lo posiciona como la solución estándar para desarrollo de hardware simple, como controles de climatización, iluminación ambiental y ajuste de asientos.

c) Gestión Energética y Modos de Operación

La eficiencia energética se logra mediante una implementación estricta de estados operativos. El protocolo permite que los nodos entren en un modo de reposo (Sleep Mode) de ultra bajo consumo, del cual pueden salir mediante una señal de despertar (Wake-up) transmitida por cualquier nodo del bus. Esta capacidad es vital en vehículos modernos para prevenir la descarga de la batería cuando el motor está apagado, manteniendo activas solo las funciones de confort necesarias.

Aunque el protocolo LIN es la solución ideal para funciones de confort, sus limitaciones técnicas lo hacen inviable para sistemas críticos de seguridad o de alto flujo de datos. A continuación, se detallan sus principales desventajas desde una perspectiva técnica:

4.5.1.2 Limitaciones protocolo LIN BUS

a) Restricción de Ancho de Banda y Velocidad

La desventaja más evidente es su baja tasa de transferencia, limitada por especificación a un máximo de 20 kbps. Esto lo inhabilita para transmitir señales de sensores complejos (como cámaras o radares) o flujos de

audio/video. En un entorno donde los vehículos modernos requieren procesar gigabits por segundo para la conducción autónoma, LIN queda relegado a funciones puramente mecánicas o de interfaz de usuario simple.

b) Vulnerabilidad a Fallos en el Nodo Maestro

Al ser una red estrictamente jerárquica, el nodo maestro es un punto único de fallo. Si el maestro experimenta un error de software o un daño físico, toda la red queda inoperativa, ya que los esclavos no tienen la capacidad de iniciar una comunicación por sí mismos ni de coordinar el bus. No existe un mecanismo de "elección de líder" o redundancia nativa como en protocolos más robustos.

c) Escalabilidad Limitada y Latencia

Aunque teóricamente puede albergar hasta 16 nodos, el rendimiento de la red se degrada conforme se añaden dispositivos. Debido a su naturaleza determinista basada en una tabla de planificación fija, añadir un nuevo nodo implica repartir el mismo ancho de banda entre más participantes, lo que aumenta el ciclo de actualización de los mensajes. Si la tabla es muy larga, la latencia (el tiempo que tarda un mensaje en volver a ser transmitido) puede volverse inaceptable incluso para funciones de confort.

4.5.2 Arquitectura de red LIN

Una agrupación LIN se define como un conjunto de nodos LIN conectados a través de un cable físico. Existen dos tipos de nodos en cada agrupación: un nodo maestro y hasta 16 nodos esclavos. Este nodo maestro es el encargado de gestionar la comunicación a lo largo del bus hacia cada esclavo, (ver Figura 26).

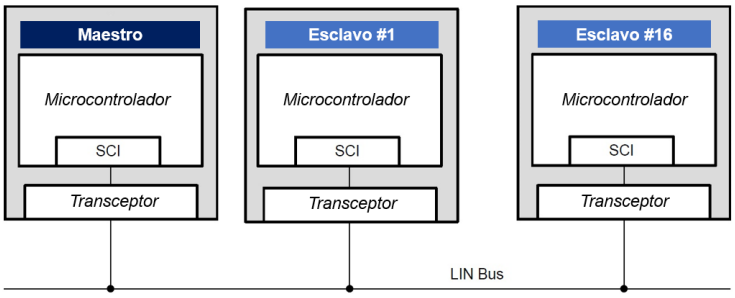


Figura 26. Topología LIN Bus.

En el sistema maestro-esclavo, un nodo de la red actúa como maestro. Este nodo determina la frecuencia de comunicación interrogando a sus nodos subordinados (esclavos). Un esclavo solo responde si el maestro se dirige a él; Sin embargo, algunos protocolos maestro-esclavo permiten que un esclavo contacte con el maestro para transmitir un mensaje (por ejemplo, transmitir información sobre la posición del elevalunas al módulo de la puerta).

La idea detrás de LIN Bus es ser una interfaz de comunicación simple y, al mismo tiempo, económica. Por esta razón, no se utiliza un controlador de comunicación dedicado. En su lugar, se programa un microcontrolador con el protocolo LIN y se utiliza para controlar la comunicación hacia el transceptor a través de la interfaz serie.

La transmisión en el bus LIN solo requiere un cable, y se utiliza una velocidad de comunicación más lenta para manejar adecuadamente cualquier problema de emisiones radiadas (20kbps). Todos los nodos están conectados pasivamente al bus, y se utiliza una resistencia de pull-up para garantizar que el bus esté al nivel de voltaje de alimentación cuando los nodos están en estado apagado.

4.5.2.1 Flujo de trabajo

El protocolo de comunicación LIN no se limita a la simple transmisión de datos binarios (1s y 0s). Ha evolucionado para incluir un flujo de trabajo LIN que define una serie de reglas y procedimientos estandarizados

para la comunicación entre los diferentes dispositivos de una red. Este facilita la implementación y el uso del protocolo. Un elemento clave en la configuración de una red LIN es el Archivo de Descripción LIN (LDF) "*LIN Description File*". Este archivo contiene toda la información necesaria para definir las características específicas de un conjunto LIN en particular. A partir del archivo LDF, se genera automáticamente un conjunto de códigos C o archivos de cabecera mediante herramientas adecuadas. Estos se utilizan como base para implementar las funciones de maestro y esclavo en las unidades de control electrónico conectadas al bus. El LDF es, por lo tanto, un medio para configurar toda la red LIN. Representa una interfaz común entre el fabricante del vehículo y el proveedor de los módulos maestro o esclavo. Desde una perspectiva estratégica, el LDF es el elemento que diferencia a las diversas armadoras automotrices, ya que estas poseen la libertad de definir el uso específico y las propiedades de sus aplicaciones. Entre estas definiciones se incluyen la cantidad de nodos, la descripción detallada de las tramas de mensajes y los identificadores de red, tal como se ilustra en la Figura 27. En última instancia, la utilidad de este archivo trasciende la generación automática del software de comunicación, extendiéndose hacia la provisión de datos críticos para las herramientas de diagnóstico involucradas en el análisis técnico, tanto para la armadora como para sus proveedores externos.

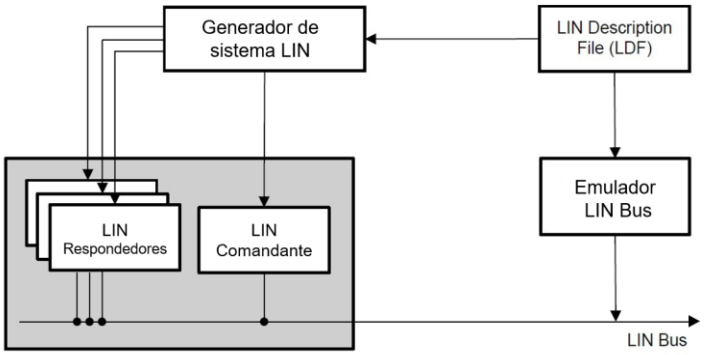


Figura 27. Diagrama de Flujo de trabajo LIN Bus [10].

Esta capacidad de estandarización de datos es, precisamente, la que permite la interoperabilidad de los equipos de diagnóstico modernos utilizados en el servicio postventa. En la práctica, el LDF actúa como el mapa de referencia que los escáneres automotrices consultan para traducir las señales eléctricas del bus en información legible para el técnico mecánico. Dado que cada fabricante personaliza su arquitectura de red en el LDF, las herramientas de diagnóstico dependen de esta estructura para identificar con precisión qué nodo presenta una anomalía o qué trama de mensaje ha sido interrumpida, ver Figura 28.

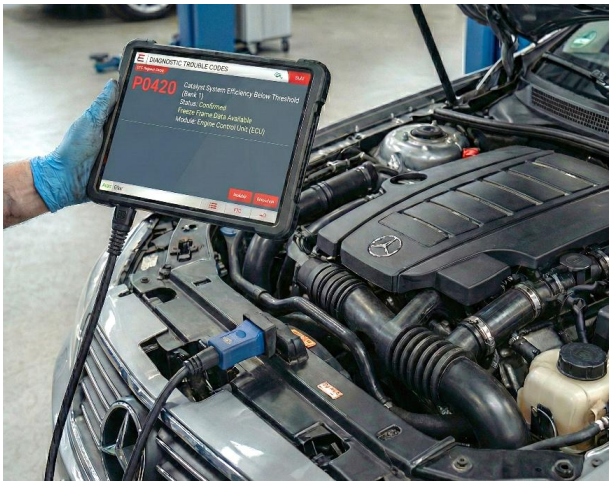


Figura 28. Representación de uso de herramientas de diagnóstico automotriz.

4.5.2.2 Fundamentos del Diagnóstico UDS en Redes LIN

La integración del protocolo UDS (Unified Diagnostic Services) sobre redes LIN (Local Interconnect Network) representa un equilibrio estratégico entre la estandarización del diagnóstico y la optimización de costos en la arquitectura electrónica del vehículo. Mientras que UDS proporciona un conjunto de servicios universales definidos por la norma ISO 14229 [27], LIN actúa como el medio de transporte eficiente para aquellos componentes que, por su naturaleza, no requieren las altas velocidades de una red CAN. Esta combinación permite que incluso los módulos más sencillos, como sensores de iluminación o actuadores de espejos, utilicen el mismo lenguaje de diagnóstico que las unidades de control más complejas, facilitando una gestión de fallos coherente en todo el automóvil.

Para que esta comunicación sea posible, el estándar LIN reserva dos identificadores de cuadro específicos que funcionan como los canales exclusivos para el diagnóstico. El primero es el "*Master Request Frame"* ($0x3C$), el cual es utilizado únicamente por el nodo maestro para enviar solicitudes o comandos a los esclavos. El segundo es el "*Slave Response Frame"* ($0x3D$), destinado a que los nodos esclavos respondan con la información solicitada o confirmen la ejecución de una tarea. Es importante destacar que, para que el mensaje llegue al componente correcto entre todos los que puedan estar conectados al bus, se utiliza un identificador adicional llamado NAD (Node Address), el cual funciona como la dirección postal específica de cada nodo esclavo dentro de la red.

En la práctica, esta estructura garantiza que las herramientas de diagnóstico externas puedan interactuar con cualquier nodo de la red de manera transparente, lo que simplifica enormemente las tareas de mantenimiento y calibración en el taller sin necesidad de elevar la complejidad del hardware en los periféricos (ver Tabla 5). Implementar este estándar bajo la capa física de LIN ofrece una ventaja competitiva en el diseño automotriz, ya que permite la reutilización de software y herramientas de prueba en diferentes plataformas. Al adoptar un enfoque unificado, los ingenieros pueden diagnosticar estados de salud, leer códigos de error y actualizar parámetros de configuración en dispositivos de bajo costo con la misma precisión que en sistemas de seguridad crítica.

Tabla 5. Principales comandos de diagnóstico UDS e identificadores de red en nodos LIN [27].

Elemento / Servicio	Identificador (Hex)	Descripción Funcional
Master Request Frame	**0x3C**	Identificador fijo del bus para enviar peticiones desde el Maestro.
Slave Response Frame	**0x3D**	Identificador fijo del bus para las respuestas de los Esclavos.
Diagnostic Session Control	**0x10**	Establece el nivel de acceso para funciones de mantenimiento o programación.
ECU Reset	**0x11**	Reinicia el nodo para aplicar cambios de configuración o recuperar el estado inicial.
Read DTC Information	**0x19**	Solicita el reporte de códigos de error para identificar problemas técnicos.
Read Data By Identifier	**0x22**	Obtiene información en tiempo real, como valores de sensores o telemetría.
Tester Present	**0x3E**	Mantiene la sesión activa para evitar que el sistema se cierre por inactividad.

4.5.2.3 Principios de comunicación serial

Las unidades de control poseen en el chip una interfaz sencilla (UART, Receptor/Transmisor Asíncrono Universal) a través de la cual pueden comunicarse con el mundo exterior (por ejemplo, con un PC). Las características esenciales de una transmisión de datos se leen a través de esta interfaz.

Si no se están intercambiando datos, el nivel del bus es de 12 V (tensión de funcionamiento del microcontrolador). Cuando se transmite el bit de inicio (nivel dominante), se notifica a la otra estación conectada al bus (receptor) que está comenzando una transferencia de datos. La duración del bit de inicio determina un tiempo de bit que representa la base para toda la transferencia de datos. Cada bit de datos posterior tiene la misma duración. El recíproco de este tiempo corresponde a la tasa de transferencia de datos; es decir, el número de bits que pueden transmitirse en un segundo en un flujo de datos continuo. Todas las estaciones participantes deben estar configuradas con la misma tasa de transferencia de datos. Tras recibir el bit de inicio, comienza la transmisión de una palabra de datos de 8 bits (1 byte) empezando por el bit menos significativo (LSB, *Low Significant Bit*). El receptor, que se ha sincronizado con el bit de inicio, muestrea el bus de datos entre cada bit de datos y, por lo tanto, recompone el byte de datos transferido. Al octavo bit de datos le sigue el bit de paridad. Este bit indica si el número de unos transmitidos es par o impar. Por lo tanto, permite al receptor realizar una comprobación sencilla de posibles errores de transmisión. La secuencia se completa con el bit de parada, el cual se coloca en el bus con un nivel dominante.

Adicionalmente, la comunicación SCI (Interfaz de Comunicación Serial) es la interfaz principal que se utiliza entre los transceptores LIN y los microcontroladores que se comunican con ellos. El microcontrolador transmite tramas de bits, comenzando con el bit de inicio dominante. Esto sincroniza a todos los receptores en el bus, seguido por el bit menos significativo al más significativo, y luego un bit de parada. Esto constituye una trama SCI, y un mensaje LIN está compuesto por múltiples tramas SCI.

La integración del formato hexadecimal en el estudio del protocolo LIN es fundamental para comprender la transición entre la capa física de impulsos eléctricos y la interpretación lógica de los datos. Aunque el sistema transmite físicamente niveles de tensión de 12 V para el estado recesivo y 0 V para el dominante, la arquitectura de 8 bits que mencionas permite que cada byte sea representado de forma compacta mediante dos dígitos hexadecimales. Esta notación, que abarca desde 00 hasta FF, es la herramienta estándar que utilizan los desarrolladores para simplificar la lectura de las tramas SCI que fluyen a través de la interfaz UART del microcontrolador.

En el contexto de la transmisión de datos, cuando el receptor recompone el byte tras el bit de inicio y el muestreo de los 8 bits posteriores, el valor resultante se procesa matemáticamente en base 16. Esto resulta particularmente eficiente dado que cada dígito hexadecimal corresponde exactamente a un conjunto de 4 bits. Por lo tanto, una secuencia binaria compleja se traduce inmediatamente en un código hexadecimal que facilita la identificación de errores y la programación de identificadores de trama. Los identificadores de mensaje en el bus LIN, por ejemplo, suelen definirse en un rango que va desde 0x00 hasta 0x3F, permitiendo una organización lógica y jerárquica de las unidades de control conectadas.

Asimismo, la relación entre la comunicación SCI y el formato hexadecimal se manifiesta en la estructura completa del mensaje LIN. Dado que un mensaje está compuesto por múltiples tramas SCI —incluyendo el campo de sincronización, el identificador, los datos y el checksum—, el uso del sistema hexadecimal permite que toda esta cadena de información sea visualizada y analizada de manera coherente en las herramientas de diagnóstico conectadas al PC. Mientras que el hardware se ocupa de la sincronización de bits y la paridad, el software de aplicación gestiona estos valores como constantes hexadecimales, optimizando así el uso de la memoria del microcontrolador y garantizando que todas las estaciones participantes interpreten la carga útil del mensaje bajo un mismo estándar numérico.

4.5.2.4 Principios de comunicación Maestro – Esclavo

Como se revisó anteriormente, en el protocolo de comunicación LIN, hay un nodo maestro y hasta 16 nodos esclavos. El nodo maestro controla toda la comunicación en el bus y contiene tanto la tarea de maestro como la de los esclavos que deben ser entregadas, ver Figura 29. Los nodos esclavos no pueden comunicarse entre sí, contienen solo la tarea de esclavo y solo pueden responder al maestro si el mensaje está dirigido a ellos. El maestro envía una solicitud a un esclavo designado como encabezado (comienzo de la trama), y el esclavo contesta al maestro, con una trama de respuesta. También hay un caso en el que el maestro envía al esclavo el encabezado y la trama de respuesta, y el esclavo solo escucha, pero sin respuesta. Ambas situaciones garantizan un tráfico de bus predecible pero definido, impidiendo colisiones en su mayor parte porque el maestro siempre está iniciando la comunicación. Esta naturaleza predecible permite la programación de mensajes. El único problema con el sistema maestro-esclavo es que el maestro controla toda la comunicación, y si el maestro falla, toda la programación falla. En esquemas donde todos los nodos pueden actuar como maestro y esclavo, esto no sucede y esta cualidad es en última instancia lo que impide que LIN se utilice en aplicaciones relacionadas con la seguridad (eso, y la baja velocidad de los mensajes) [16].

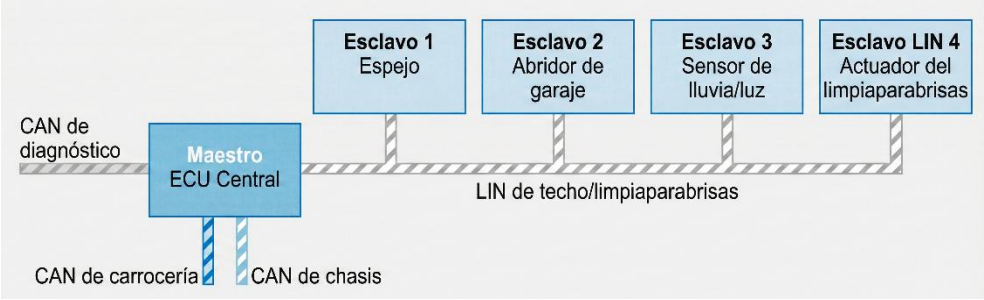


Figura 29. Representación comunicación LIN BUS maestro-esclavos [16].

4.5.2.5 Formato de mensaje LIN

Cada mensaje LIN tiene una estructura específica: la primera parte es el requerimiento del maestro (ver Figura 30) y la segunda parte es el mensaje resultante del esclavo (ver Figura 31). El primer requerimiento siempre es transmitido por la tarea del maestro, y se divide en el corte de sincronización, el campo de sincronización y el identificador protegido (PID). El corte de sincronización y el campo de sincronización se utilizan para que todos los esclavos en el bus LIN se sincronicen con el tiempo del maestro (sin necesidad de ningún cristal u oscilador), y el PID es lo que define qué esclavo es el que responde, reciba o ignore el encabezado del mensaje que se envía. El encabezado en total consta de al menos 13 bits para el corte SYNC, 1 bit delimitador, 10 bits del campo SYNC (1 bit de inicio, 8 bits para sincronización y 1 bit de parada) y 10 bits de identificador (1 bit de inicio, 6 bits para el identificador, 2 bits para paridad y 1 bit de parada).

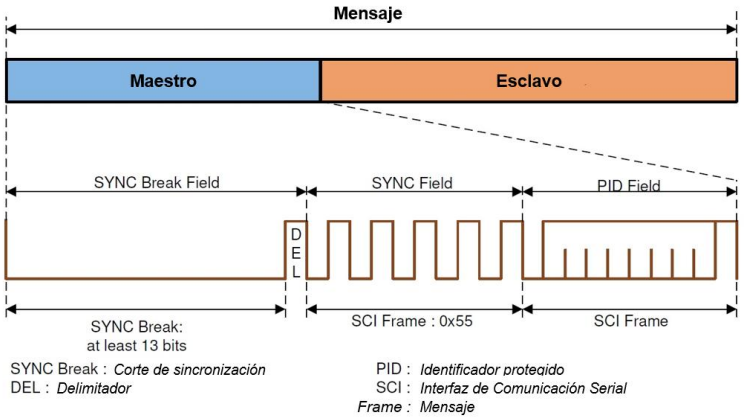


Figura 30. Representación de mensaje comandante LIN [10].

La parte de Datos (respuesta) del mensaje es enviada por la tarea del respondedor, que puede ser enviada por el nodo comandante o el nodo respondedor dependiendo de la "instrucción" del PID. La Respuesta se divide en bytes de datos (hasta 8) y una suma de comprobación (checksum). La suma de comprobación es un esquema de protección para los bytes de datos, verifica que el mensaje enviado sea el que se pretendía y que no se introdujeron errores durante la transmisión. Cualquier nodo puede recibir la trama de respuesta, pero qué nodo realmente la usa depende del LDF (Archivo de Descripción de LIN). La respuesta en total consta de 10 bits para cada byte de datos (1 bit de inicio, 8 bits para datos, 1 bit de parada), para hasta 8 bytes de datos, y 10 bits para la suma de comprobación (1 bit de inicio, 8 bits para la solución de la suma de comprobación, 1 bit de parada).

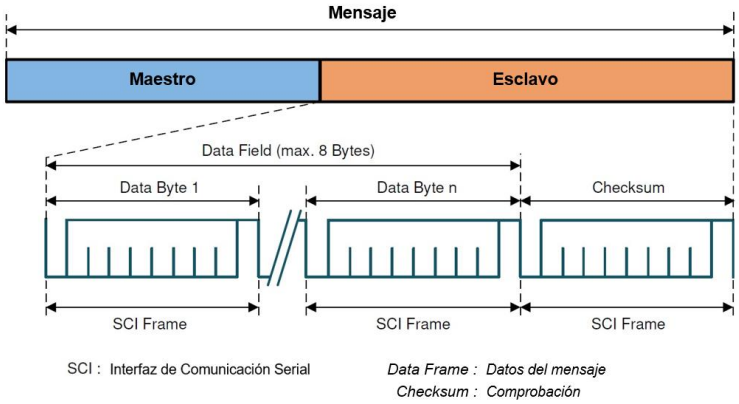


Figura 31. Representación de mensaje respondedor LIN [10].

A continuación, se detallan las partes fundamentales de una trama (*frame*) de LIN Bus.

Sincronización

Para garantizar una transferencia de datos consistente entre el maestro y los esclavos, se lleva a cabo una sincronización al comienzo de cada trama. En primer lugar, el inicio de una trama se marca de forma inequívoca mediante la pausa de sincronización (SynchBreak). El SynchBreak consta de al menos 13 niveles dominantes consecutivos y un nivel recesivo.

Al final de la pausa de sincronización, el maestro envía el campo de sincronización (SynchField), que consiste en la secuencia de bits 01010101 (“55” en hexadecimal). Los esclavos pueden entonces ajustarse a la base de tiempo del maestro y, de este modo, sincronizarse.

El método de sincronización descrito permite una especificación flexible de la temporización de los abonados del bus. El reloj del maestro no debe desviarse más de un ±0.5 % del valor nominal. Se permite que el reloj de los esclavos se desvíe hasta un 15 % antes de la sincronización, siempre que esta reduzca la desviación a un máximo del 2 % antes del final del mensaje. De esta manera, los esclavos pueden construirse sin un costoso oscilador de cuarzo, por ejemplo, utilizando un simple circuito RC.

Identificador

El tercer byte de la cabecera se utiliza como identificador LIN. Al igual que en el bus CAN, se utiliza un método de direccionamiento basado en el contenido; por lo tanto, el identificador proporciona información sobre el contenido de un mensaje (por ejemplo, el régimen del motor). Basándose en esta información, todos los nodos conectados al bus deciden si desean recibir y procesar el mensaje o simplemente ignorarlo. Este proceso se conoce como filtrado de aceptación.

Dentro del campo del identificador, seis de los ocho bits disponibles definen el identificador propiamente dicho, cuyas permutaciones permiten un total de 64 variantes posibles expresadas en formato hexadecimal desde el 0x00 hasta el 0x3F. Estas direcciones se categorizan según su función específica dentro del bus: los identificadores comprendidos entre 0x00 y 0x3B se destinan a la transmisión ordinaria de señales entre nodos. Por su parte, el valor 0x3C se utiliza para la solicitud del maestro en tareas de diagnóstico y comandos, mientras que el 0x3D corresponde a la respuesta del esclavo ante dicha solicitud. Los valores finales, 0x3E y 0x3F, se reservan para comunicaciones específicas del fabricante y para futuras extensiones del protocolo, respectivamente.

La capacidad de carga de estos 64 mensajes posibles se distribuye de manera que 32 de ellos (del 0x00 al 0x1F) pueden contener únicamente dos bytes de datos, 16 de ellos (del 0x20 al 0x2F) transportan cuatro bytes, y los 16 restantes (del 0x30 al 0x3F) gestionan ocho bytes cada uno. Para asegurar la integridad de esta información, los dos bits finales del campo del identificador integran sumas de comprobación o bits de paridad, los cuales validan que el identificador no haya sufrido alteraciones durante la transmisión y previenen asignaciones de mensajes incorrectas en la red.

Campo de datos

Una vez que se ha transmitido la cabecera enviada por el nodo maestro, llega el momento de que comience la transferencia de los datos reales. Los esclavos saben, a partir del identificador transmitido, si están siendo direccionados o no y, en caso afirmativo, responden con su contestación en el campo de datos. Se pueden empaquetar varias señales en una sola trama. En este caso, cada señal tiene precisamente un generador, es decir, siempre es escrita por el mismo nodo de la red.

Durante la transferencia de datos de los bytes, siempre se emite primero el bit menos significativo (LSB). Cada byte (8 bits) va precedido de un bit de arranque y seguido de un bit de parada, lo que significa que cada byte implica la transmisión de diez bits. El propósito de los bits de arranque y parada es resincronizar los nodos y, de este modo, evitar errores de transmisión. La respuesta de datos de los esclavos se verifica mediante una suma de comprobación (checksum).

4.5.2.6 Tipos de tramas de mensaje LIN

El protocolo LIN 2.1 clasifica el tráfico en cuatro categorías funcionales:

- Trama Incondicional: Es el funcionamiento o “mecanismo” estándar. El Maestro envía la cabecera correspondiente a una ID (identificador) y el nodo productor asociado rellena el bus con datos. No existen condiciones; la transmisión ocurre siempre que la tabla de planificación lo dicte.
- Trama Activada por Eventos: Permite que múltiples esclavos respondan a un mismo ID de cabecera solo si sus datos han cambiado, sin embargo, pueden llegar a surgir “Colisiones” si dos esclavos responden simultáneamente, el Maestro detecta la corrupción de datos y cambia automáticamente a una resolución de eventos de colisión para interrogar a los nodos individualmente.
- Trama Esporádica: Comparte un "espacio" de tiempo con otras tramas esporádicas. El Maestro solo genera la cabecera si tiene datos actualizados que enviar a un esclavo. Si no hay cambios, el slot queda vacío, optimizando el ciclo del bus.
- Trama de Diagnóstico: Utiliza identificadores reservados para la capa de transporte (ISO 15765-2 simplificada), entre los más utilizados se encuentran:
    - 0x3C: *Master Request* (Maestro → Esclavo).
    - 0x3D: *Slave Response* (Esclavo → Maestro).

4.5.2.7 Planificación de mensaje LIN

El protocolo es totalmente determinista, lo que significa que tiene la capacidad de garantizar que un evento o una transmisión de datos ocurrirá en un tiempo exacto y predecible. El Maestro cicla a través de una "Tabla de Planificación" (*Schedule Table*) predefinida en el archivo de descripción (LDF, mencionado en el subtema 3.4.2.1).

- Esto garantiza que cada señal tenga un tiempo máximo de latencia asegurado.
- El Maestro puede cambiar entre diferentes tablas de planificación según el modo de operación del vehículo (ej. "Modo Normal", "Modo Diagnóstico", "Modo Sleep").

4.5.2.8 Gestión de Estado y Sleep/Wake-up

El ciclo de vida de la red LIN se gestiona mediante comandos específicos y condiciones físicas:

- Comando Go-to-Sleep: El Maestro transmite una trama de diagnóstico (ID 0x3C) con el primer byte de datos igual a 0x00. Esto obliga a todos los esclavos a entrar en modo de bajo consumo.
- Señal de Wake-up: Cualquier nodo puede solicitar el despertar forzando el bus a estado dominante durante un periodo de $250\mu s$ a $5ms$.Tras detectar este pulso, el Maestro tiene un tiempo limitado para reiniciar la transmisión de la cabecera y restablecer el cronograma de comunicación. Si el Maestro no responde, el nodo puede reintentar el pulso de despertar hasta 3 veces.

4.5.2.9 Manejo de Errores

El protocolo LIN 2.1 delega la responsabilidad de la detección de errores a cada nodo receptor y transmisor, sin mecanismos de retransmisión automática por hardware.

- Monitoreo de Bits (Bit Error): El transmisor compara el valor enviado con el valor real en el bus. Si difieren (ej. envía recesivo, pero lee dominante), detecta un error de bit y detiene la transmisión inmediatamente.
- Error de Checksum: El receptor recalcula la suma de verificación; si no coincide con el byte final recibido, la trama se descarta.
- Error de Identificador (Parity Error): Si la paridad del PID es incorrecta, el esclavo no escucha el resto de la trama.
- Error de Trama (Framing Error): Se detecta si no se encuentran los bits de parada (recesivos) en el lugar esperado de cada byte.

4.5.5 Gestión de la red LIN BUS

Como se mencionó anteriormente los nodos de una red LIN pueden ser forzados al modo de reposo para minimizar la corriente sin carga de todo el sistema eléctrico y electrónico del vehículo. El modo de reposo se puede lograr de dos maneras:

- El maestro envía la instrucción de ir a reposo (*go-to-sleep*) con el identificador reservado 60.
- Los esclavos entran en modo de reposo por iniciativa propia si no se ha producido ninguna transferencia de datos en el bus durante un periodo de tiempo relativamente largo (4 segundos).

Tanto el maestro como los esclavos pueden activar la red. Para ello, deben enviar la señal de activación (*wake-up*), la cual consiste en el byte de datos 128. Tras una pausa de entre 4 y 64 tiempos de bit (delimitador de activación), todos los nodos deben haberse inicializado y estar listos para responder al maestro.

De manera ilustrativa, se utilizará el ejemplo de la gestión del control del aire acondicionado:

El bus LIN se utiliza habitualmente en el control del sistema de aire acondicionado. La unidad de mando y visualización funciona como el maestro del bus. Aquí es donde se almacena el software para los algoritmos de regulación y control. Una de sus tareas consiste en ajustar la velocidad del soplador de aire fresco. Las variables de control para esto son la temperatura real en el habitáculo y la temperatura deseada establecida por el conductor. La unidad de control electrónico recibe la temperatura interior de un sensor situado en un lugar adecuado del habitáculo. La temperatura deseada se ajusta en la unidad de mando, por ejemplo, mediante un anillo sensor, (ver Figura 32). A partir de las variables de entrada, la unidad de control electrónico calcula que la velocidad del soplador debe aumentarse a un valor de 200 rpm, por ejemplo. La unidad de control electrónico transmite un mensaje que contiene la instrucción del maestro al bus LIN en un intervalo de tiempo definido. El identificador en este ejemplo sería "ajustar la velocidad del soplador". En este sub-bus, esta instrucción corresponde al identificador 25, por ejemplo. Tras el encabezamiento que contiene este identificador, el maestro transmite un valor numérico en el campo de datos que equivale al valor físico de 200 rpm. Cada esclavo posee una lista de identificadores que son relevantes para ese nodo. El soplador de aire fresco es el único esclavo que responde al identificador "ajustar velocidad del soplador" y ejecuta la petición del maestro. A velocidades inferiores al ralentí (régimen mínimo de revoluciones por minuto), el soplador de aire fresco debe desconectarse. A esta velocidad tan baja, la fuerte carga podría provocar que el motor se cale. El bus LIN está acoplado al bus CAN a través de las interconexiones (“*gateway*”) y, por tanto, recibe la velocidad actual del motor de forma continua. Si la velocidad cae por debajo del umbral de velocidad del motor especificado, el maestro LIN envía el mensaje con el identificador "ajustar velocidad del soplador" cuyo campo de datos contiene el valor 0. El soplador de aire fresco se apaga como respuesta.

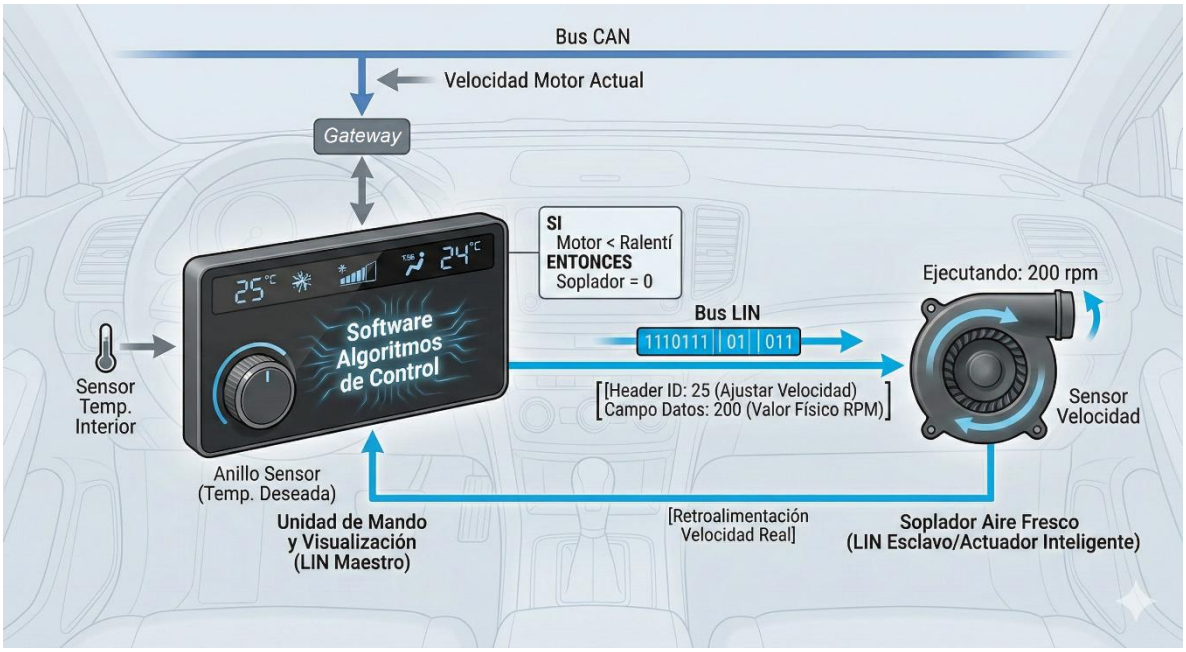


Figura 32. Topología y Flujo de Datos del Bus LIN en el Sistema de Aire Acondicionado.

Los actuadores inteligentes pueden enviar información actualizada sobre su estado de funcionamiento a la unidad de activación. El soplador de aire fresco registra la velocidad mediante un sensor y la devuelve a través del bus LIN como un valor numérico. Debido al método de acceso maestro-esclavo, el valor solo se incluye en un mensaje con respuesta del esclavo iniciado por el maestro. La señal de retroalimentación con la velocidad actual del motor permite controlar y, por lo tanto, mantener con precisión el valor deseado.

4.5.6 Requisitos físicos

Los requisitos físicos LIN se basan en el estándar ISO 9141. El cuál describe una interfaz de comunicación de bus bidireccional, polarizada a la tensión de la batería del vehículo a través de una resistencia y un diodo (solo en el nodo maestro), y está conectada al transceptor de cada nodo en el agrupamiento LIN (ver Figura 33).

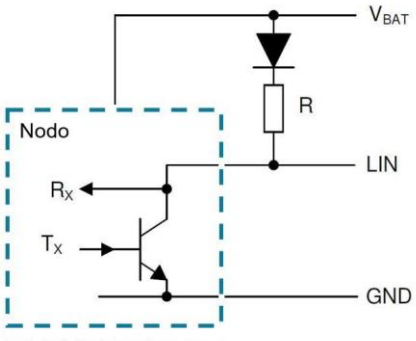


Figura 33. Esquema simplificado de controlador LIN [10].

El transceptor es lo que facilita la comunicación entre el bus y la red. El transceptor LIN convierte la lógica de bits del microcontrolador en niveles de voltaje más altos, para su transmisión a través del bus y viceversa. Las señales TXD (transmisión) y RXD (recepción) del transceptor LIN, facilitan la comunicación hacia y desde el bus a través de la conversión de voltaje que ocurre cuando las señales pasan a través del transceptor. La señal TXD está conectada al microcontrolador, donde se envía el mensaje y luego se transmite en el bus LIN. La señal RXD monitorea el bus y convierte los mensajes en el bus LIN a niveles de voltaje que el microcontrolador puede interpretar, y así responder a la comunicación que ocurre en el bus. Los niveles de voltaje típicos para TXD y RXD son los típicos de la mayoría de los niveles de microcontroladores: 3.3 V y 5 V. El bus LIN y los transceptores LIN generalmente operan con voltajes que oscilan entre 9 V y 18 V, pero algunos pueden llegar hasta 30 V (dependiendo de la aplicación). Un vehículo típico tiene un sistema de batería de 12 V, pero algunos vehículos más grandes llegan hasta 24 V.

3.5.6.1 Señalización del Bus de estado "dominante y recesivo"

Considerando los niveles de voltaje mencionados en la sección 3.4.5, debido a que el esquema de comunicación es de un solo extremo y los umbrales no se pueden establecer por una diferencia de voltaje, estos umbrales se basan en un porcentaje del voltaje de la batería en el sistema. Estos umbrales determinan cuándo se considera que el bit es "recesivo" o "dominante". Recesivo y dominante son diferentes formas de decir alto y bajo (respectivamente) en el bus. La convención de nomenclatura proviene del concepto de cómo el bus y los transceptores interactúan entre sí y cómo se generan las señales dentro del IC (circuito integrado). Desde un nivel alto, un agrupamiento LIN es equivalente a un circuito de drenaje abierto, lo que significa que el bus requiere una resistencia pull-up y todos los nodos están conectados pasivamente al bus a través de los transceptores. Cuando se le solicita, los transceptores controlan los niveles de voltaje en el bus. La resistencia pull-up asegura que los niveles de voltaje en dicho bus alcancen o estén cerca del nivel de voltaje de la batería cuando el control TXD del transceptor está en estado apagado. Una vez que el transceptor se activa y el transistor de control TXD se enciende, el bus se lleva a un nivel bajo, casi a tierra, y se sobrescribe el estado alto. Por lo tanto, el bus se mantiene en alto cuando el transceptor está apagado o en un estado pasivo, de ahí el término "recesivo". Cuando el TXD del transceptor comienza a conducir y se activa, el bus se lleva a bajo, de ahí el término "dominante".

4.5.6.2 Valores de resistencia "Pull-up"

Los valores de la resistencia pull-up son diferentes para los nodos comandantes y los nodos respondedores. El nodo comandante requiere una resistencia pull-up y un diodo externo de acuerdo con la especificación LIN. El valor típico que se ve es de 1 kΩ (otros valores típicos son 600 Ω y 500 Ω) en serie con un diodo, para protección contra polaridad inversa, conectado al voltaje de la batería. El valor típico de pull-up de los respondedores LIN es de 30 kΩ, y en todos los transceptores LIN modernos, esto está integrado dentro del IC (circuito integrado), por lo que no es necesario un pull-up externo en la configuración de estos.

4.5.6.3 Valores de umbral

El transmisor y el receptor tienen diferentes niveles que cumplen con los requisitos de nivel de voltaje recesivo y dominante. Para los pulsos dominantes (bajo), el transmisor debe llevar el nivel de voltaje hasta el 20% del nivel de voltaje de la batería, mientras que el receptor interpretará un bit dominante cuando el nivel de voltaje alcance el 40% en su extremo (ver Figura 34). Para los pulsos recesivos (alto), el transmisor debe llevar el voltaje al 80% del voltaje de la batería, mientras que el receptor interpreta un bit recesivo cuando el nivel de voltaje alcanza el 60% en el bus (ver Figura 35).

La diferencia en los niveles entre el receptor y el transmisor se debe a las diferencias en el voltaje de alimentación externo y el voltaje real del bus LIN. Las caídas de voltaje que pueden ocurrir en el cableado, los desplazamientos de tierra o simplemente los cambios causados por los componentes de filtrado a lo largo del bus son las principales causas de la desviación del suministro externo frente al nivel del bus.

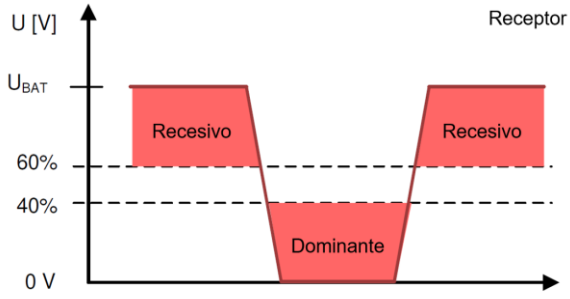


Figura 34. Umbrales de Señal del Bus para Receptores.

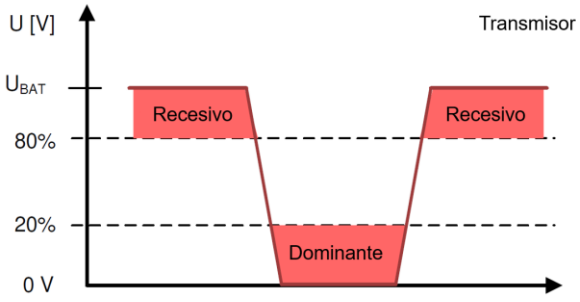


Figura 35. Umbrales de Señal del Bus para Transmisores.

4.5.6.4 Tolerancia de Velocidad de Bits y Requisitos de Sincronización

La velocidad de transmisión de bits para LIN varía de 1 a 20 kilobits por segundo, con una tolerancia de velocidad de bits de ±14%. Este valor del 14% proviene del hecho de que se utilizan osciladores integrados de bajo costo y, con la calibración interna, se puede lograr una precisión mejor que ±14%. Esta precisión permite la detección de una interrupción en el flujo de mensajes y, con la calibración de tiempo utilizando el campo de sincronización, se asegura la recepción y transmisión de mensajes. Los cambios de temperatura y la deriva de voltaje pueden causar cambios en la velocidad de bits, y el oscilador integrado tiene en cuenta estas variaciones al medir la velocidad de bits y generar el resto de la trama del mensaje (después del campo de sincronización).

4.5.6.5 Sincronización y Muestro de Bits

A excepción de un caso de uso especial, todos los tiempos de bit utilizan la temporización de bits del nodo comandante como referencia. El byte de sincronización consta de '0x55' (8 bits de 1 y 0 alternados, comenzando con 0), que es esencialmente una señal de reloj a una frecuencia determinada. Los flancos descendentes del patrón se utilizan para las sincronizaciones que se combinan con el bit de inicio y parada (10 bits en total), lo que permite 4 flancos descendentes en total para la sincronización de los nodos respondedores. Esto también permite una medición precisa del ancho de bit (T_{BIT}).

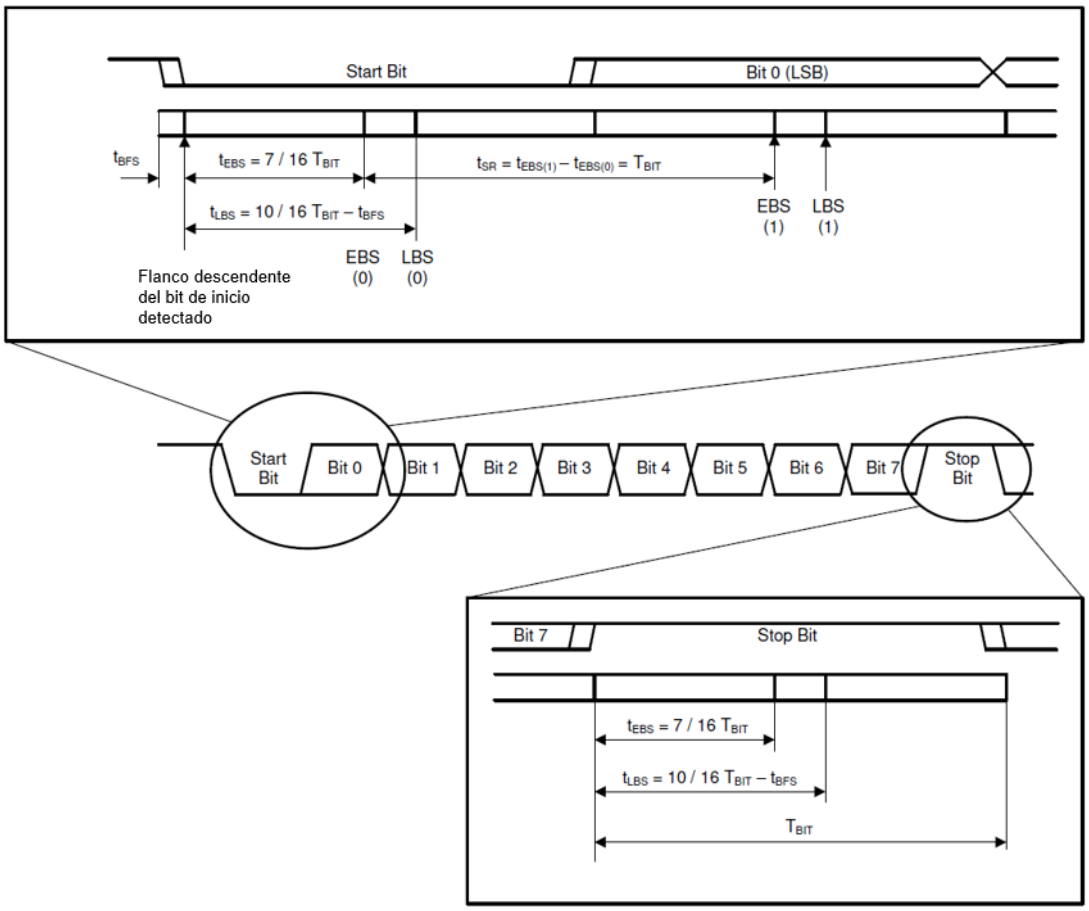


Figura 36. Ilustración de muestreo de Bits [10].

Después de que ocurre la sincronización de los nodos respondedores, cada bit debe ser muestreado con precisión para asegurar la correcta interpretación del mensaje por el agrupamiento LIN (ver Figura 36). Cada bit debe ser muestreado entre la muestra de bit más temprana (t_{EBS}) y la muestra de bit más tardía (t_{LBS}); t_{LBS} depende de t_{BFS} a través de la Ecuación 1:

$$t_{LBS} = \frac{10}{16} T_{BIT} - t_{BFS} \quad (1)$$

T_{EBS} se define como mínimo 7/16 de T_{BIT}. El resto de los bits después del primer bit se muestrean con una velocidad de muestreo (tSR). Estos se basan en la muestra de bit más temprana del bit anterior (n – 1) y la muestra de bit más temprana del bit actual (n), y están dados por la Ecuación 2:

$$t_{SR} = t_{EBS(n)} - t_{EBS(n-1)} = T_{BIT} \quad (2)$$

4.5.6.6 Ciclo de trabajo

Para asegurar que los mensajes enviados se interpreten correctamente, el bus LIN debe cumplir con los niveles de voltaje correctos según el suministro de la batería y estos voltajes deben cumplirse dentro del tiempo de muestreo de bits correcto del receptor.

La Figura 35 hace referencia al transceptor de red de interconexión local (LIN) modeloTLIN1029-Q1 *Texas Instrument* con tiempo de espera de estado dominante. Define la temporización del bus como un requisito para el muestreo adecuado de cada bit. Esta definición se establece de manera que cuando se diseñan los

transceptores, el ciclo de trabajo no se distorsione al propagarse de TXD a LIN y de LIN a RXD. Debido a que no se envía una señal de reloj con los mensajes y la sincronización se basa en el campo de sincronización, si hay demasiada variación del ciclo de trabajo, el reloj del comandante o el reloj del respondedor también pueden variar. Esto afecta la temporización para el resto de ese ciclo de energía.

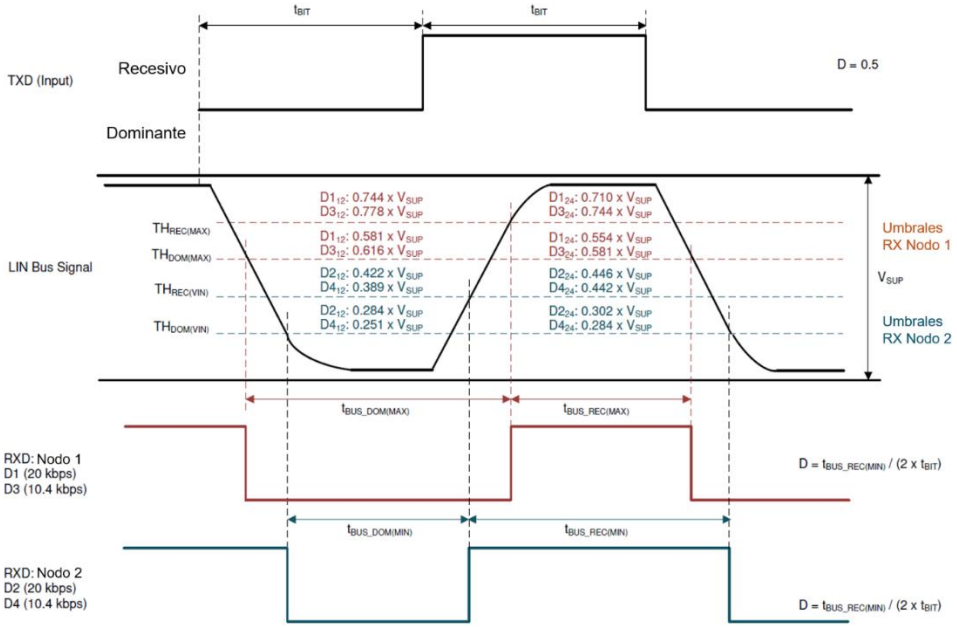


Figura 37. Requisito del Ciclo de Trabajo del Bus [10].

4.6.6 Distancia y Nodos en el Bus

La especificación LIN define el número máximo de nodos que se pueden conectar a un bus LIN: 1 nodo maestro y 16 nodos esclavos, la longitud máxima del mazo de cables: 40 m. Debido al nivel de criticidad, no hay preocupación por tener demasiados nodos en el bus ni por que el cable sea demasiado largo, a diferencia de otras interfaces de comunicación. La capacitancia en el bus aún debe estar dentro de un rango razonable para una comunicación adecuada, y esto puede verse afectado por la longitud del cable, los nodos o el filtrado del bus.

Dado que la longitud del cable y la cantidad de nodos están limitadas por definición, el único parámetro para tener en cuenta es cualquier capacitancia adicional; una buena guía es mantenerse por debajo o cerca de 10 nF de capacitancia total del bus a una velocidad de comunicación de 20 kbps. La Figura 38, la Figura 39 y la Figura 40 muestran el efecto de tener la cantidad nominal de capacitancia, y cómo se ve tener demasiada capacitancia. En el caso de demasiada capacitancia, los flancos ascendentes no pueden alcanzar el nivel de voltaje completo a tiempo para el siguiente bit y, por lo tanto, los bits no se interpretan correctamente, como ilustra la forma de onda RXD.

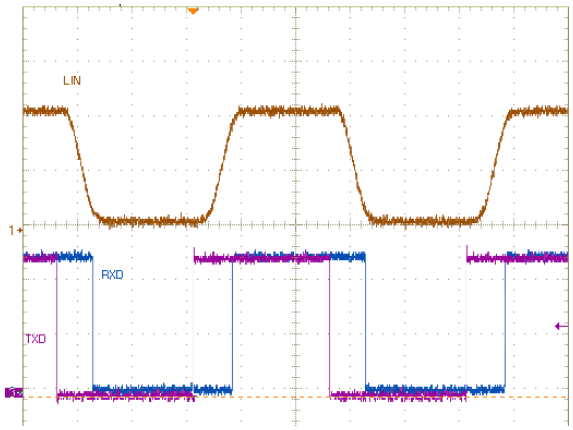


Figura 38. Bus LIN con 220 pF, mensaje a 20 kbps. [10]

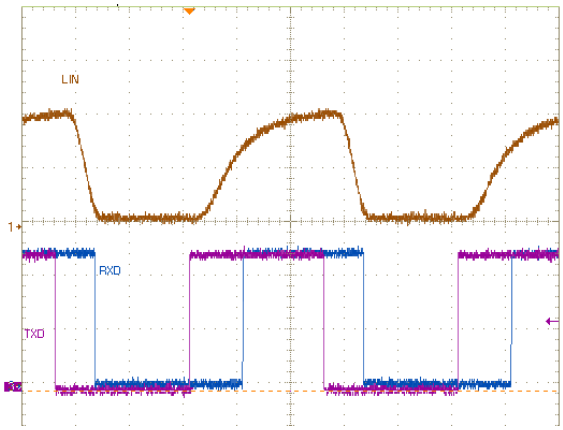


Figura 39. Bus LIN con 10 nF, mensaje a 20 kbps. [10]

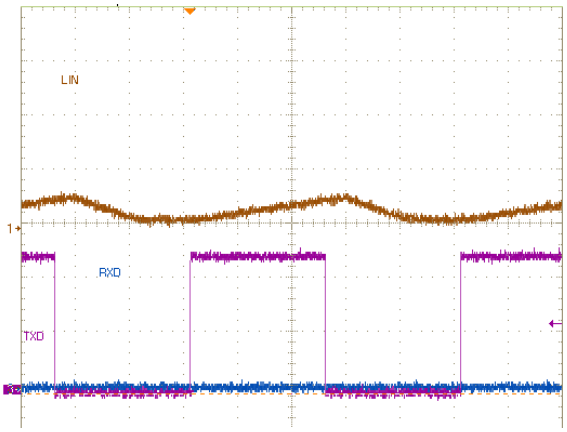


Figura 40. Bus LIN con 220 nF, mensaje a 20 kbps. [10]

4.7 Capacidades Operativas de LIN

El análisis del protocolo LIN confirma su posición como una solución de baja complejidad y alta rentabilidad para el control de funciones secundarias. Al operar bajo una arquitectura de *Maestro-Esclavo* sobre una línea unifilar no blindada, LIN reduce drásticamente el peso del cableado y los costos de implementación en comparación con protocolos más robustos como CAN. Aunque su tasa de transferencia está limitada a 20 kBit/s y su alcance a 40 metros, estas especificaciones son óptimas para nodos que no requieren un ancho de banda crítico, permitiendo una sincronización precisa de hasta 16 nodos sin necesidad de resonadores de cuarzo externos, lo que simplifica el diseño de hardware en aplicaciones periféricas.

5. Metodología

El presente proyecto está diseñado para dar una clara y completa metodología de cómo realizar un controlador LIN BUS, dividido en 5 etapas:

1. Investigación y Análisis:
    - Revisión bibliográfica: Estudio profundo de los protocolos de comunicación automotriz, con énfasis en LIN.
    - Selección de componentes electrónicos: Elección de componentes ideales basado en sus características técnicas y compatibilidad con LIN.
2. Diseño:
    - Diagrama esquemático: Creación del plano eléctrico del circuito LIN BUS.
    - Diseño de propuesta de PCB: Desarrollo de la placa de circuito impreso para alojar los componentes electrónicos compatibles LIN BUS.
3. Desarrollo:
    - Arquitectura de software: Definición de la estructura del programa para la comunicación LIN.
    - Programación: Codificación de las funciones necesarias en el microcontrolador/pantalla táctil utilizando un lenguaje de programación de libre acceso.
4. Implementación y Pruebas:
    - Integración: Conexión del circuito diseñado a dispositivos con protocolo LIN.
    - Pruebas: Verificación del correcto funcionamiento del sistema.
5. Resultados:
    - Redacción resultados: Sección que detalle los resultados obtenidos, así como cualquier comparación que pueda surgir entre el control de los dispositivos.

5.1 Fundamentos para el Desarrollo de un Sistema de Control Automotriz de Bajo Costo mediante Redes de Comunicación LIN Bus

La fase de implementación de este proyecto se fundamenta en el diseño y construcción de una Unidad de Control Electrónica (ECU) de bajo costo, con la finalidad de tener acceso a redes de comunicación vehicular sin comprometer la fiabilidad técnica. La relevancia de este enfoque económico radica en la capacidad de replicar arquitecturas automotrices complejas de manera eficiente, demostrando que es factible alcanzar estándares industriales de comunicación mediante el uso optimizado de recursos electrónicos accesibles en el mercado local. Bajo esta premisa la ruta metodológica comienza con la selección estratégica de un microcontrolador de alto rendimiento que, actuando como el cerebro del sistema, permite la ejecución de algoritmos de control tanto de lazo abierto como de lazo cerrado mediante instrucciones almacenadas en su memoria. A diferencia de las soluciones comerciales cerradas, este enfoque permite una arquitectura flexible donde el procesamiento de señales se optimiza para minimizar la latencia, transformando los datos de entrada en acciones precisas sobre la etapa de salida.

La arquitectura del hardware se consolida mediante la configuración del microcontrolador como un nodo Maestro, cuya responsabilidad metodológica es la gestión íntegra de la red. Este nodo no solo gestiona el cronograma de comunicación y las tramas de datos, sino que coordina la sincronización de los nodos Esclavos dentro de la topología LIN Bus. Para validar la eficiencia de esta configuración económica, el despliegue físico incluye módulos de accesorios automotrices reales, tales como un sistema de iluminación y un actuador motorizado. Estos componentes permiten observar la conversión de señales eléctricas en variables mecánicas como el torque de un motor asegurando que la reducción de costos en los componentes no afecte la integridad operativa del bus de datos en condiciones de uso real.

Finalmente, la integración del software se aborda mediante un ecosistema de desarrollo de código abierto que elimina gastos derivados de licencias propietarias, utilizando específicamente el lenguaje C++ y la plataforma Android Studio®. Esta metodología de desarrollo culmina en la creación de una interfaz de control móvil que utiliza el protocolo Bluetooth® como puente de comunicación inalámbrica.

Esta capa de interacción permite al usuario final supervisar el desempeño del controlador y gestionar los periféricos de forma remota, consolidando una infraestructura técnica que es, a la vez, práctica, eficiente y escalable para futuras aplicaciones en el sector automotriz, tal como se detalla en los diagramas de flujo y esquemáticos de la Figura 41.

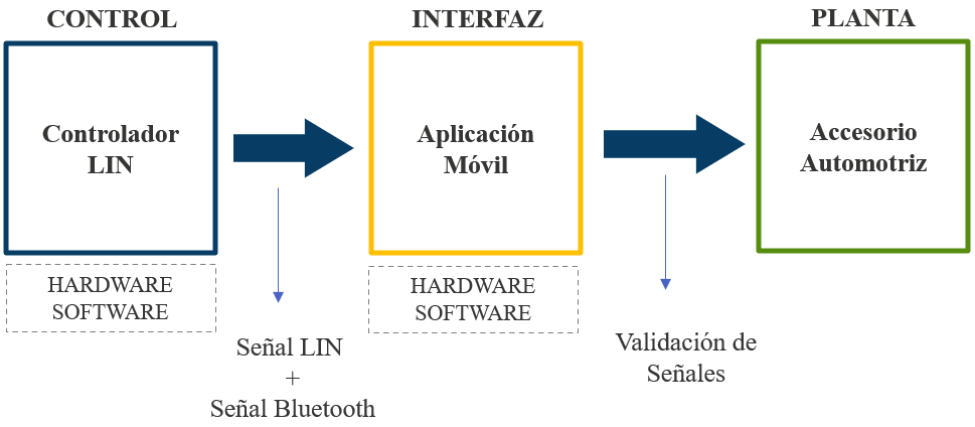


Figura 41. Diagrama de bloques de controlador LIN BUS.

5.2 Hardware para el Desarrollo de un Sistema de Control Automotriz de Bajo Costo mediante Redes de Comunicación LIN Bus

La determinación del hardware principal constituye un punto crítico en el desarrollo del controlador, ya que de esta elección depende la estabilidad de la red y la viabilidad económica del prototipo. En esta fase de evaluación se analizaron dos plataformas de procesamiento con enfoques distintos: el ecosistema Raspberry Pi® en conjunto con la tarjeta de expansión PiCAN FD, y el microcontrolador Teensy 4.1. La primera opción, basada en Raspberry Pi, ofrece una capacidad de procesamiento superior al operar bajo un sistema operativo completo, lo que facilita la implementación de herramientas de diagnóstico complejas y una integración nativa con protocolos de red de alta velocidad como CAN FD. Sin embargo, su implementación conlleva una elevación considerable en el presupuesto del proyecto, debido tanto al costo unitario de la placa base como a la necesidad de adquirir módulos adicionales para la interfaz física de comunicación, lo que además incrementa la complejidad del diseño y el consumo energético global de la unidad.

Frente a esta alternativa, se evaluó el microcontrolador Teensy 4.1 como el núcleo de procesamiento para la ECU de bajo costo. Este dispositivo destaca por integrar un procesador ARM Cortex-M7 de alto rendimiento capaz de operar a 600 MHz, superando la velocidad de la mayoría de los microcontroladores en su rango de precio. Desde una perspectiva metodológica, el Teensy 4.1 representa la opción más eficiente para la gestión del protocolo LIN Bus, ya que dispone de múltiples puertos serie (UART) por hardware que pueden configurarse con precisión para manejar las tramas de datos y los intervalos de sincronización requeridos por el estándar automotriz. Su arquitectura permite un control de tiempo real estricto, eliminando las latencias inherentes a los sistemas operativos multihilo, lo cual es fundamental para asegurar que la comunicación entre el nodo Maestro y los Esclavos sea determinista.

La comparativa económica refuerza la selección del Teensy 4.1, pues su costo en el mercado es significativamente inferior en comparación con el conjunto Raspberry Pi® y PiCAN FD, permitiendo cumplir con el objetivo de accesibilidad financiera planteado inicialmente. Al integrar esta plataforma, el proyecto no solo logra una reducción drástica en la inversión de componentes, sino que también obtiene una unidad de control más compacta y eficiente energéticamente. Esta decisión técnica asegura que la arquitectura resultante sea capaz de ejecutar algoritmos de control de lazo cerrado y gestionar la interfaz Bluetooth® con fluidez, garantizando que el ahorro de costos no se traduzca en una degradación del desempeño o la seguridad en la transmisión de datos. Bajo este criterio de selección, ambas opciones tecnológicas y sus implicaciones técnicas se revisarán a detalle a continuación, con el fin de justificar plenamente la arquitectura final adoptada para el sistema de control.

5.2.1 Microcontrolador Raspberry Pi® PiCAN FD

En la fase inicial de investigación, se evaluó la viabilidad de utilizar la tarjeta Raspberry Pi en conjunto con la placa de expansión PiCAN FD (ver Figura 42) como núcleo del sistema. Esta solución destaca por integrar una interfaz de comunicación completa, donde el chip Microchip MCP2518FD gestiona las tramas CAN y un microcontrolador dsPIC33 dedicado actúa como puente para el bus LIN [22]. A través de una investigación documental, se identificó que este ecosistema permite una interacción simplificada mediante comandos de texto ASCII vía UART, facilitando el desarrollo de herramientas de diagnóstico con interfaces gráficas en Python3.

Sin embargo, al profundizar en el diseño de una unidad autónoma, se observó que la incorporación de una pantalla táctil para eliminar la dependencia de periféricos externos representaba un incremento crítico en el presupuesto total del hardware. Este costo adicional, sumado al valor de la placa base y los módulos de expansión, alejaba el prototipo de la meta de bajo costo, convirtiéndolo en una solución financieramente menos competitiva para un desarrollo de nivel académico o de prototipado inicial.

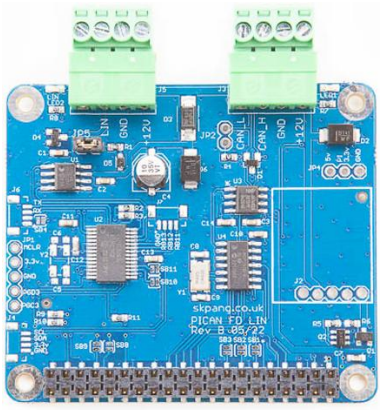


Figura 42. Microcontrolador PiCAN FD.

A la barrera económica se añade una limitación operativa fundamental relacionada con la naturaleza de los protocolos vehiculares. Aunque las herramientas asociadas a la PiCAN FD facilitan la manipulación de redes mediante archivos de descripción de LIN (LDF), la investigación determinó que, en el entorno profesional, estos archivos suelen ser propiedad intelectual confidencial de las empresas automotrices y no están disponibles para el público general. Por lo tanto, basar el desarrollo en herramientas que dependen de la carga de un LDF carecería de utilidad práctica para un controlador diseñado desde cero. En su lugar, el proyecto requiere una arquitectura que permita la interpretación y generación manual de tramas de datos sin depender de archivos propietarios, permitiendo así una mayor flexibilidad en la ingeniería inversa y el control de periféricos genéricos.

Tabla 6. Selección de componentes LIN BUS con accesorios Raspberry Pi®.

Hardware			**Software**		
Microcontrolador	**Interfaz**	**Chip LIN BUS**	**Programación Microcontrolador**	**Diseño PCB**	**Programación Interfaz**
Raspberry Pi	Pantalla Táctil	PiCAN LIN BUS Microchip MCP2518FD	Python	KiCAD	Android Studio
$1,950.00 MXN	$ 1,700.00 MXN	$1,350.00 MXN	$0 MXN	$0 MXN	$0 MXN
$ 5,000 MXN / $250 USD					

Raspberry Pi	Pantalla táctil para Raspberry	PiCAN FD with LIN Bus for Raspberry Pi
		

Finalmente, el análisis técnico concluyó que el uso de la placa PiCAN FD contrapone el objetivo primordial de la investigación: el desarrollo propio de la interfaz de comunicación. Debido a que este módulo entrega una solución pre-desarrollada (ver Figura 43) donde la lógica de red está encapsulada en el firmware del dsPIC33, se pierde la oportunidad de programar y gestionar el protocolo desde el nivel de bits. Por consiguiente, la combinación de un elevado costo por componentes periféricos detallados en la Tabla 6, la inaccesibilidad de los archivos LDF en el mundo real y la falta de autoría en la lógica del bus, llevaron al descarte definitivo de este ecosistema en favor de una plataforma más versátil y económica.

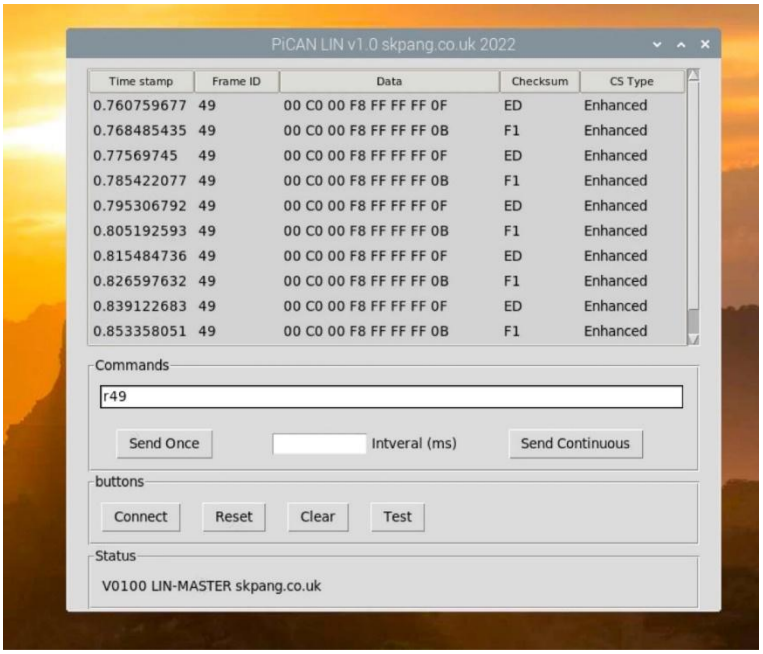


Figura 43. Interfaz de comunicación LIN con el uso de tarjeta Raspberry® con PiCAN. [22].

5.2.2 Microcontrolador Teensy 4.1

Como contraparte a las limitaciones observadas en la fase de investigación previa, se seleccionó el microcontrolador Teensy 4.1 (Figura 44) como el componente central para el desarrollo de la unidad de control. Esta plataforma destaca por integrar un procesador ARM Cortex-M7 operando a una frecuencia de 600 MHz, lo que proporciona una capacidad de procesamiento en tiempo real significativamente superior a los microcontroladores convencionales en su rango de precio. Esta potencia es fundamental para garantizar que la ejecución de los algoritmos de control y la gestión de las tramas del bus de datos se realicen con una latencia mínima, asegurando un comportamiento determinista en la red vehicular. La arquitectura del Teensy 4.1 permite un acceso total a los registros del sistema, lo que facilita el desarrollo de una pila de protocolos propia, cumpliendo así con el objetivo primordial de diseñar la lógica de comunicación desde el nivel físico y de enlace sin depender de soluciones pre-fabricadas [19].

La ventaja técnica crítica del Teensy 4.1 reside en su infraestructura de comunicación, la cual dispone de múltiples puertos serie (UART) por hardware con buffers de alta velocidad. Esta característica es esencial para la implementación del protocolo LIN Bus, ya que permite generar con precisión la señal de "Break" y los intervalos de sincronización requeridos por el estándar, tareas que serían complejas de ejecutar en sistemas con menos recursos o capas de software adicionales. Al no contar con una interfaz LIN pre-configurada, el dispositivo obliga al desarrollo de funciones específicas para la construcción de cada trama, lo que otorga una flexibilidad absoluta ante la ausencia de archivos de descripción de red (LDF) comerciales. De esta manera, el controlador se vuelve capaz de realizar ingeniería inversa y gestionar periféricos mediante la manipulación directa de identificadores y sumas de comprobación (checksums), permitiendo la comunicación con módulos automotrices genéricos sin las restricciones de confidencialidad de la industria.

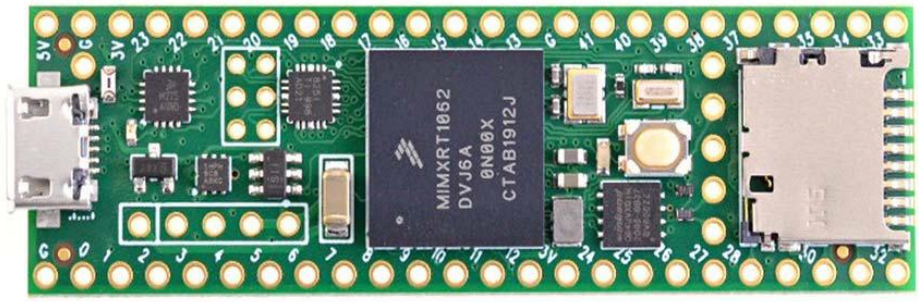


Figura 44. Placa de desarrollo Teensy 4.0 [19].

Desde una perspectiva económica, la elección del Teensy 4.1 consolida el objetivo de bajo costo del proyecto de manera contundente. Mientras que el ecosistema analizado anteriormente requería una inversión significativa en placas base, módulos de expansión y pantallas táctiles, el costo unitario de este microcontrolador representa apenas una fracción de ese presupuesto. Su formato compacto y su capacidad para ser programado en entornos de código abierto mediante C++ eliminan gastos adicionales de licenciamiento y hardware periférico complejo. En consecuencia, el Teensy 4.1 no solo ofrece un rendimiento técnico excepcional para la orquestación del nodo Maestro, sino que garantiza la viabilidad financiera del prototipo, permitiendo la creación de una herramienta de diagnóstico y control potente, autónoma y accesible para el desarrollo tecnológico local, tal como se detalla en las especificaciones de la Figura 45.

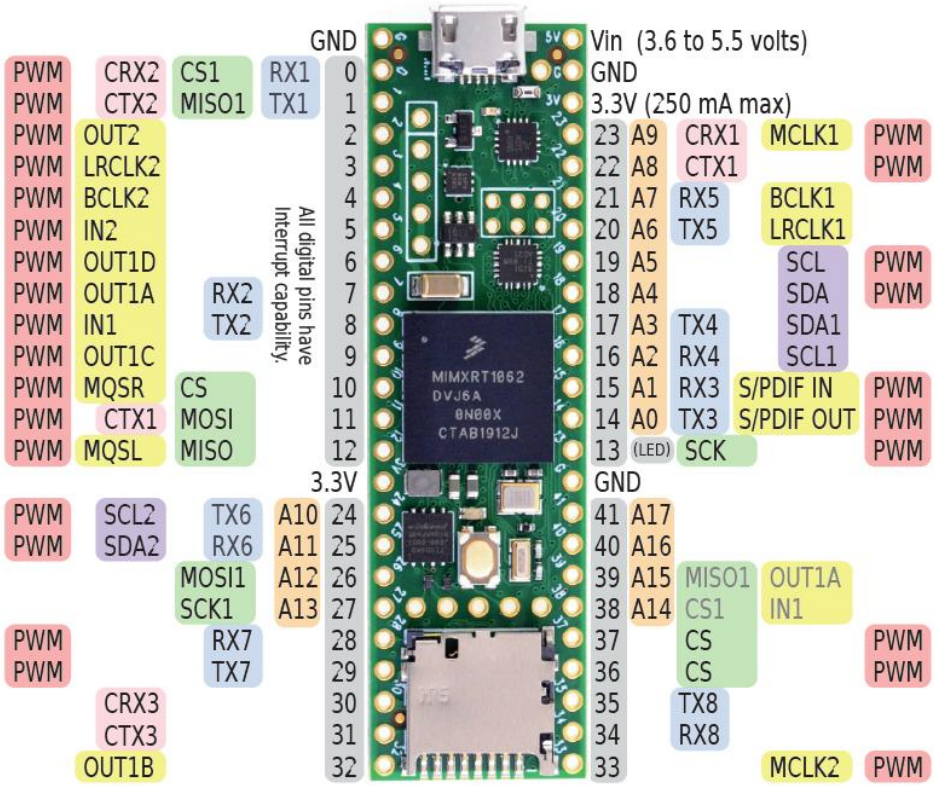


Figura 45.Puertos GPIO (Entrada/Salida) de Microcontrolador Teensy 4.1.[19]

Para complementar la arquitectura de hardware, la tarjeta Teensy 4.1 ofrece una expansión significativa en la capacidad de interconexión y control periférico a través de su robusta matriz de puertos de entrada y salida de propósito general (GPIO). Esta versión dispone de un total de 55 pines I/O, todos ellos con capacidad de interrupción, lo que permite una respuesta inmediata ante eventos externos críticos, como señales de sensores o paradas de emergencia. De estos, 35 pines pueden configurarse como entradas analógicas con una resolución de hasta 12 bits, y la gran mayoría soporta modulación por ancho de pulso (PWM), facilitando el control preciso de la intensidad lumínica o la velocidad del actuador motorizado en los nodos esclavos. Una ventaja técnica determinante para este proyecto es la disponibilidad de 8 puertos seriales (UART) por hardware, los cuales operan de forma independiente.Complementariamente, el manejo de la información se optimiza mediante el uso de buffers FIFO (First-In, First-Out) en sus puertos seriales. Estos buffers funcionan como una "sala de espera" inteligente para los datos: almacenan la información entrante en el orden exacto en que llega, permitiendo que el microcontrolador la procese conforme sus ciclos de trabajo lo permitan, sin riesgo de que los datos nuevos sobrescriban o eliminen a los anteriores. Esta gestión es vital para el éxito del proyecto, ya que asegura que ninguna instrucción del bus LIN o de la interfaz Bluetooth® se pierda mientras el procesador ejecuta algoritmos de control complejos, manteniendo la fluidez y la integridad de la red de bajo costo.

No obstante, dado que el Teensy 4.1 opera con una lógica de 3.3 V y el entorno automotriz utiliza niveles de 12 V, se debe considerar el diseño de una etapa de protección y acondicionamiento de señal indispensable para evitar daños irreversibles en el microcontrolador. Para el canal de comunicación principal, se implementan transceptores especializados (como el MCP2003), que actúan como una barrera física y un traductor de niveles, aislando los pines sensibles del microcontrolador de los picos de voltaje y el ruido eléctrico del bus vehicular. Esta infraestructura de protección asegura que la alta velocidad y eficiencia del Teensy 4.1 se mantengan operativas bajo las rigurosas condiciones eléctricas del vehículo, garantizando la durabilidad del hardware sin incrementar significativamente los costos de producción.

5.2.3 Circuito Integrado (IC) LIN MCP2003

Para complementar la arquitectura de hardware, es indispensable abordar el componente encargado de la interfaz física entre la lógica digital y el entorno eléctrico del vehículo. El transceptor Microchip MCP2003 (Figura 46), específicamente diseñado bajo el estándar LIN J2602, se seleccionó como el puente de comunicación bidireccional debido a su capacidad para traducir las señales de bajo voltaje del microcontrolador a los niveles de tensión de la batería automotriz. Este dispositivo de 8 pines actúa como una barrera de protección y acondicionamiento, permitiendo que el Teensy 4.1 envíe y reciba datos en una línea de bus de 12 V sin riesgo de daños por sobretensión o ruido electromagnético. Su arquitectura interna está optimizada para soportar las especificaciones de las revisiones LIN 2.0, 2.1 y 2.2, lo que garantiza una compatibilidad total con la mayoría de los módulos de accesorios modernos y asegura que el proyecto cumpla con los estándares industriales de comunicación serie de un solo cable, ver Figura 47, [20].

Figura 46. Circuito Integrado (IC) LIN MCP2003 [20].

Una de las características más valiosas del MCP2003 para este proyecto es su robustez ante las condiciones adversas del entorno vehicular. El integrado cuenta con mecanismos de protección contra cortocircuitos a batería o a tierra, además de un sistema de apagado térmico que desactiva el transmisor si la temperatura del chip excede los límites de seguridad debido a una falla externa. Asimismo, presenta una alta inmunidad a la interferencia electromagnética (EMI) y una excelente resistencia a las descargas electrostáticas (ESD), eliminando la necesidad de implementar componentes de filtrado externos costosos y complejos. Estas funciones de seguridad permiten que la etapa de potencia sea compacta y eficiente, manteniendo la integridad de los datos incluso en presencia de picos de voltaje transitorios comunes en el encendido de motores o actuadores inductivo.

Desde la perspectiva del bajo costo, el MCP2003 es una solución sumamente competitiva que refuerza la viabilidad financiera de la ECU desarrollada. Al ser un componente de amplia disponibilidad y bajo precio unitario en comparación con transceptores de marcas propietarias o módulos pre-ensamblados, permite reducir el costo total de la lista de materiales sin sacrificar el rendimiento técnico. Su bajo consumo de corriente en modo de espera (*standby*) es otro factor ideal, ya que minimiza el drenaje de energía de la batería cuando el sistema no está en operación activa. En conclusión, la integración de este transceptor no solo resuelve la disparidad de voltajes entre el microcontrolador y el bus de datos, sino que proporciona una capa de confiabilidad industrial a un precio accesible, consolidando un diseño profesional, seguro y alineado con los objetivos de economía de escala del proyecto.

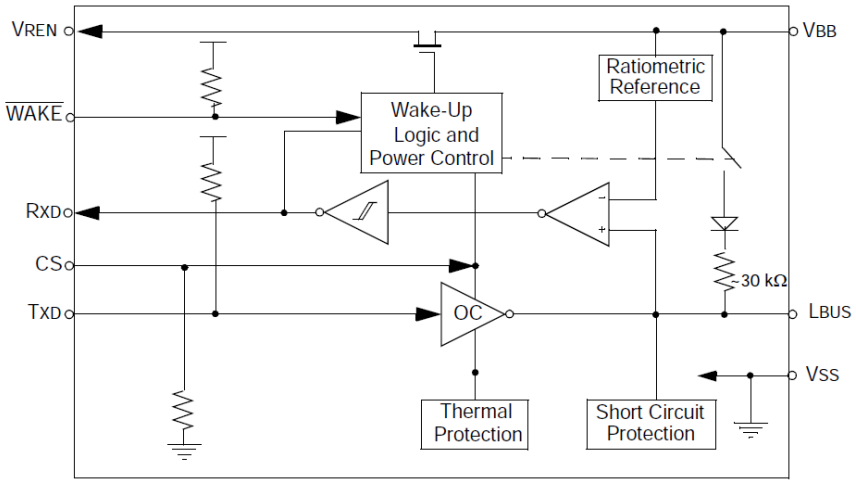


Figura 47. Diagrama de bloques interno de Circuito Integrado (IC) LIN MCP2003 [20].

5.2.4 Circuito Integrado ESP32C3

La capacidad de interacción remota del sistema se materializa a través del ESP32-C3, un sistema en chip (SoC). A diferencia de los módulos Bluetooth convencionales que actúan como simples puentes seriales, este componente funciona como un nodo de procesamiento secundario dedicado exclusivamente a la gestión de comunicaciones inalámbricas. Su soporte nativo para Bluetooth 5.0 resulta fundamental, ya que garantiza un enlace de alta fidelidad con dispositivos móviles, permitiendo que la aplicación en Android Studio® reciba telemetría del bus LIN en tiempo real sin las latencias ni el elevado consumo de energía de las versiones de Bluetooth clásicas.

En cuanto a la eficiencia presupuestaria, el ESP32-C3 redefine el concepto de bajo costo al integrar en un solo encapsulado diminuto un procesador potente, conectividad Wi-Fi y Bluetooth de última generación, ver Figura 48. En lugar de adquirir módulos de comunicación limitados y costosos, la adopción de este IC permite reducir la complejidad del esquema eléctrico. Esta decisión técnica no solo optimiza la inversión en materiales, sino que proyecta al prototipo hacia un estándar de conectividad industrial moderna, demostrando que es posible construir una herramienta de diagnóstico profesional utilizando silicio de vanguardia a una fracción del costo de las soluciones automotrices cerradas [21].

Figura 48. Circuito Integrado XIAO- ESP32-C3 [21].

5.2.5 Costo de Hardware para el Desarrollo de un Sistema de Control Automotriz de Bajo Costo mediante Redes de Comunicación LIN Bus

El análisis de viabilidad económica del proyecto confirma una ventaja competitiva excepcional al contrastar la arquitectura desarrollada frente a otras soluciones de prototipado y herramientas de grado industrial. El presupuesto de hardware se optimizó mediante la selección estratégica de componentes de alto rendimiento y bajo costo, donde el microcontrolador Teensy 4.1 encabeza la inversión con un valor de $446.62 MXN, seguido por la pasarela inalámbrica ESP32C3 de apenas $27.00 MXN y el transceptor LIN MCP2003 con un costo de $121.81 MXN. Con un total de $717.24.00 MXN esta estructura financiera permite consolidar un sistema funcional por una fracción del capital requerido por otras plataformas evaluadas durante la fase de investigación; por ejemplo, la implementación basada en el ecosistema Raspberry Pi® habría demandado una inversión de $1,950.00 MXN por la placa base, $1,350.00 MXN por el módulo PiCAN FD y un costo adicional de $1,700.00 MXN por la pantalla táctil, totalizando un presupuesto que quintuplica la solución adoptada en este proyecto.

Más allá del ahorro en componentes físicos, el proyecto maximiza su rentabilidad mediante el uso de un ecosistema de software de código abierto, eliminando por completo los gastos de licenciamiento al emplear EasyEDA para el diseño de circuitos, Android Studio® para la interfaz móvil y el lenguaje C++ para la lógica de control. Esta decisión no solo reduce el costo operativo a $0 MXN en herramientas de desarrollo, sino que garantiza una arquitectura libre de restricciones propietarias. Al elevar la comparativa hacia el mercado industrial actual, los beneficios son aún más evidentes: herramientas comerciales de diagnóstico como el BabyLin, ampliamente utilizado en la industria automotriz, poseen costos que rondan los $12,000.00 MXN, (ver Tabla 7).

En conclusión, el desarrollo de esta ECU de bajo costo representa un ahorro superior al 90% en comparación con las herramientas industriales profesionales y una reducción del 85% respecto a soluciones de prototipado con Raspberry Pi. Este equilibrio entre el alto desempeño técnico del Teensy 4.1 y la economía de escala de los componentes complementarios demuestra que es posible democratizar el acceso a herramientas de diagnóstico vehicular de alta fidelidad, cumpliendo con los estándares de comunicación exigidos por la red LIN sin comprometer la estabilidad financiera del proyecto, lo que valida la metodología de diseño desde la perspectiva de la ingeniería económica y técnica.

Tabla 7. Costo total de Hardware & Software para el controlador LIN BUS de bajo coste.

Hardware			**Software**		
Microcontrolador Principal	**Interfaz**	**LIN BUS µC**	**Programación µC**	**Elaboración de esquemáticos y PCB.**	**Programación de Interfaz**
~~*Raspberry Pi*~~	~~*Pantalla Táctil*~~	~~*PiCAN with LIN BUS Microchip MCP2518FD*~~	~~*Python*~~	~~*KiCAD*~~	*Android Studio*
~~*$1,950.00 MXN*~~	~~*$ 1,700.00 MXN*~~	~~*$1,350.00 MXN*~~	~~*$0 Código de acceso libre*~~	~~*$0 Código de acceso libre*~~	*$0 Código de acceso libre*
~~**$ 5,000 MXN / $250 USD**~~					
Teensy 4.0	*Dispositivo Android + Chip (ESP32C3)*	*Chip LIN MCP2003A*	*C++*	*EasyEDA*	*Android Studio*
$ 446.62 MXN	*$ 27 MXN*	*$ 121.81 MXN (x2)*	*0 USD*	*0 USD*	*0 USD*
$ 717.24 MXN / $35.33 USD					

5.3 Diagrama esquemático para el Desarrollo de un Sistema de Control Automotriz de Bajo Costo mediante Redes de Comunicación LIN Bus

La integración física del sistema se consolida en el diseño esquemático presentado en la Figura 49, el cual articula las etapas de procesamiento, comunicación inalámbrica y acondicionamiento de potencia en una arquitectura unificada de bajo costo. El núcleo de la unidad reside en el microcontrolador Teensy 4.1 (U1.1), el cual actúa como el gestor principal de la red y se vincula con el primer transceptor MCP2003A-E/P (U2) a través de sus puertos seriales RX7 y TX7 (pines 20 y 21). Un aspecto fundamental en esta sección es la inclusión de la resistencia R2 de conectada entre la línea de datos LBUS y la línea de +12V. Esta configuración establece formalmente al nodo como el Maestro de la topología LIN, otorgándole la capacidad de dictar el cronograma de comunicación y generar las tramas de sincronización necesarias para el resto de la red. De forma complementaria, la interacción inalámbrica se gestiona mediante el módulo XIAO ESP32C3 (U3), el cual establece un puente de datos con el procesador central empleando una interfaz UART secundaria vinculada a los pines 25 y 26 del Teensy. El diseño incorpora estratégicamente un segundo transceptor MCP2003A-E/P (U5) conectado directamente al ESP32C3, lo que permite que este último desempeñe el rol de nodo Esclavo dentro de la misma placa de desarrollo. Esta duplicidad de transceptores, junto con la resistencia R1 de asociada al segundo nodo, facilita la validación empírica de la comunicación sin depender de módulos externos, permitiendo monitorizar la integridad de las señales y la latencia del protocolo en un entorno de lazo cerrado controlado.

La estabilidad eléctrica del prototipo frente a los transitorios comunes en entornos automotrices se garantiza mediante una etapa de regulación liderada por el componente L78L05ABZ (U4). Este regulador lineal de voltaje convierte la entrada de proveniente del vehículo en una salida estable de para alimentar la lógica digital del ESP32C3 y los circuitos internos de los transceptores. Para mitigar el ruido electromagnético y asegurar un flujo de corriente libre de rizado, se implementan capacitores de filtrado C1 y C2 de en la entrada y salida del regulador, respectivamente. Finalmente, el acceso a la red física se centraliza en el conector (U6), el cual proporcionará una interfaz mecánica robusta para la línea de datos LIN, la alimentación principal y la tierra común (GND), consolidando así un hardware compacto, protegido y técnicamente alineado con los requerimientos del bus de datos vehicular.

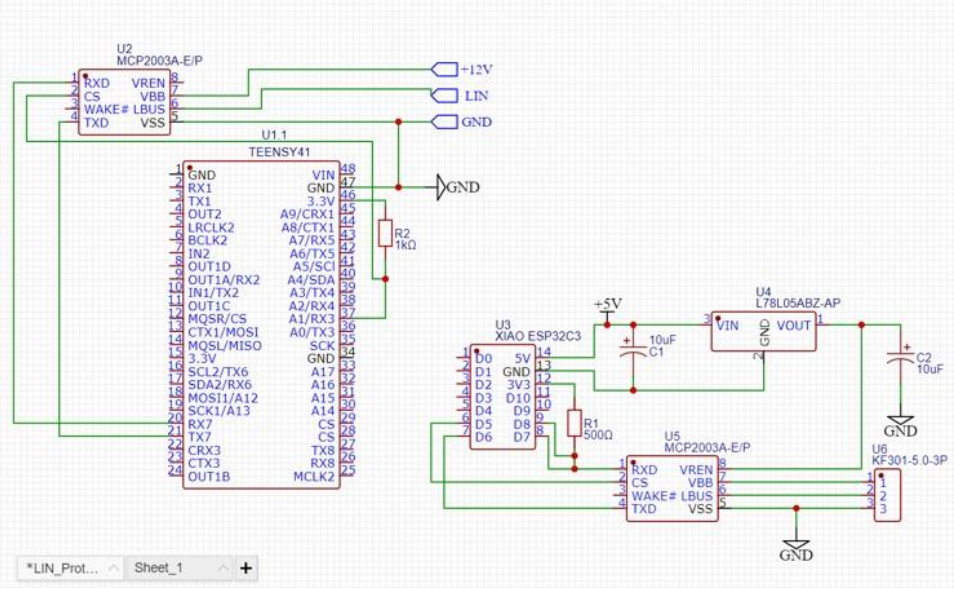


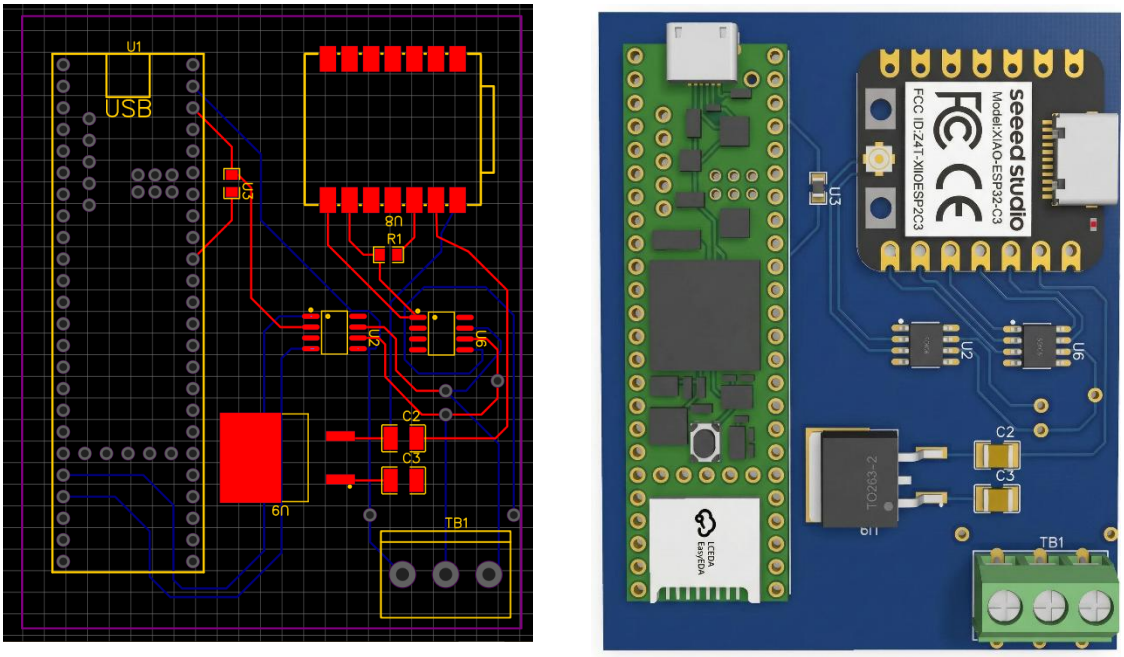
Figura 49. Diagrama esquemático.

5.3.1 Propuesta de Diseño de la Placa de Circuito Impreso (PCB)

La integración física de los elementos sobre la placa de circuito impreso se ha estructurado como propuesta para consolidar la interoperabilidad entre el Teensy 4.1 y el ESP32C3 dentro del bus de datos (ver Fig. 50). En el diseño del PCB, la disposición del Teensy 4.1 responde a la necesidad de mantener trazos de señal cortos hacia su transceptor MCP2003, optimizando así la integridad de las tramas LIN antes de su conversión al nivel físico de 12V. De manera paralela, el ESP32C3 se ha ubicado estratégicamente para permitir un ruteo directo hacia el segundo chip MCP2003, garantizando que ambos nodos operen de forma independiente pero coordinada, minimizando las inductancias parásitas que podrían degradar la comunicación en entornos de alta frecuencia.

La gestión de energía dentro de la PCB se centraliza en el regulador de voltaje L78L05, el cual transforma la entrada principal de 12V en el carril de 5V necesario para alimentar el microcontrolador Teensy 4.1 como al módulo ESP32. Esta sección del hardware incluye una red de filtrado compuesta por los condensadores C1 y C2, posicionados inmediatamente adyacentes a las terminales de entrada y salida del regulador para mitigar rizos de tensión y transitorios provenientes del suministro automotriz. El aprovechamiento del área de la placa es definido por sus dimensiones de 10 cm de alto por 7 cm de ancho.

Finalmente, la interfaz externa se resuelve mediante el bloque de terminales de 3 pines, que unifica los puntos de conexión para la alimentación, la referencia de tierra y la línea física del bus. Esta organización permite que las señales de potencia y datos entren y salgan de la placa de forma ordenada, reduciendo el riesgo de errores en el cableado durante las fases de experimentación. La cohesión entre el Teensy 4.1, el ESP32-C3 y las etapas de comunicación con el circuito integrado LIN MCP2003 bajo este diseño físico no solo materializa el esquema electrónico, sino que proporciona una plataforma robusta y compacta para la validación de los protocolos de comunicación objeto de este estudio.



(a) (b)

Figura 50. (a) Vista de dos capas de las pistas y terminales de la PCB. (b) Representación tridimensional del diseño propuesto final de la PCB.

5.3.2 Normativas y Criterios Técnicos para el Diseño de Circuitos Impresos en el Sector Automotriz

Para el desarrollo de hardware en el sector automotriz, es necesario seguir un marco normativo riguroso que garantice la integridad operativa de los sistemas electrónicos ante condiciones ambientales extremas. La base de esta fiabilidad se sustenta en los estándares del Automotive Electronics Council (AEC), particularmente la norma AEC-Q100, que define los protocolos de pruebas de estrés para circuitos integrados destinados a soportar rangos térmicos críticos de entre -40°C y +150°C, así como vibraciones y estrés mecánico. Esta validación a nivel de componente se complementa con las directrices de seguridad funcional dictadas por la norma ISO 26262, la cual establece una metodología estructurada para la gestión de riesgos mediante los Niveles de Integridad de Seguridad Automotriz (ASIL), asegurando que el diseño de la PCB minimice sistemáticamente la probabilidad de fallos críticos durante todo su ciclo de vida.

En cuanto a la fabricación y el ensamblaje, la calidad del producto final se rige por los estándares de la IPC, donde aplicaciones de alta exigencia como la comunicación vehicular demandan el cumplimiento de la Clase 3 y la normativa IPC-6012. La normativa IPC-6012 establece tres niveles de calidad para las placas rígidas [24]. Mientras que la Clase 1 es para electrónica de consumo básico y la Clase 2 para equipos de cómputo o industriales generales, la Clase 3 (*High Reliability Electronic Products*) es el estándar más exigente. Se reserva para sistemas donde el funcionamiento ininterrumpido es crítico y el tiempo de inactividad no es aceptable. En una PCB de Clase 3, los criterios de inspección son mucho más estrictos; por ejemplo, se exigen depósitos de cobre más gruesos en las paredes de las perforaciones (vías) y tolerancias casi nulas en defectos visuales o estructurales, garantizando que la placa no falle bajo vibraciones constantes o cambios bruscos de temperatura. Estos criterios aseguran una durabilidad superior en las uniones de soldadura y la estructura del laminado, superando los requisitos comerciales convencionales (ver Figura 51(a)). Asimismo, la implementación de un sistema de gestión de calidad bajo la norma IATF 16949 es fundamental para optimizar la eficiencia de los procesos y reducir defectos en la cadena de suministro, integrando los requisitos específicos de la industria con los principios de mejora continua de la norma ISO 9001 [25].

Como medida final de robustez y en cumplimiento con las exigencias de durabilidad para electrónica de Clase 3, se debe de contemplar la aplicación de un recubrimiento conformado (ver Figura 51(b)). Este tratamiento consiste en una película polimérica delgada que se adapta a la topología de la placa, actuando como una barrera aislante frente a factores externos como la humedad, el polvo y la presencia de contaminantes químicos típicos del entorno vehicular. La implementación de este recubrimiento es fundamental para prevenir la corrosión, asegurando que el sistema mantenga su integridad dieléctrica y funcional durante periodos prolongados de exposición a condiciones ambientales severas [26].

(a) (b)

Figura 51. (a) Ilustración de condiciones (Clase 3) para unión de soldadura en almohadilla de PCB [24]. (b) Ejemplo de aplicación de recubrimiento conformado para la protección integral de la PCB y sus componentes [26].

5.4 Software para el Desarrollo de un Sistema de Control Automotriz de Bajo Costo mediante Redes de Comunicación LIN Bus

El desarrollo del software para este proyecto se articula bajo un ecosistema de herramientas de código abierto, lo que permite una integración técnica profunda sin incurrir en gastos de licenciamiento. La arquitectura lógica se divide en dos pilares fundamentales: la programación de bajo nivel en el microcontrolador central, y el despliegue de la interfaz de usuario. Todo el control del hardware se fundamenta en el lenguaje C++, utilizando el entorno de Teensyduino para el Teensy 4.1, lo que otorga un control absoluto sobre los registros del procesador. A diferencia de las soluciones comerciales, el software diseñado no depende de archivos de descripción de red (LDF) propietarios; en su lugar, se implementó una lógica de manipulación de bits desde cero que permite generar manualmente el intervalo de Break, el byte de sincronización y el cálculo de Checksums protegidos, garantizando una compatibilidad universal con periféricos LIN genéricos mediante ingeniería inversa de tramas.

La gestión de la conectividad inalámbrica se ejecuta de forma independiente en el ESP32-C3, el cual opera como un coprocesador de comunicaciones. La comunicación entre ambos microcontroladores se sincroniza a través de un protocolo de paso de mensajes por UART, diseñado específicamente para aprovechar las entradas y salidas del hardware. Esta división de tareas por software asegura que el procesamiento de las señales críticas del bus vehicular de 12V en el Teensy no se vea interrumpido por las demandas de procesamiento de la conexión inalámbrica, manteniendo un determinismo temporal estricto en la red de control.

Finalmente, el ecosistema se completa con una aplicación móvil nativa desarrollada en Android Studio®. Esta interfaz actúa como el centro de mando del usuario, transformando las interacciones táctiles en comandos binarios que viajan vía Bluetooth hacia el controlador. El software de la aplicación fue diseñado bajo una arquitectura orientada a objetos. Al utilizar exclusivamente herramientas gratuitas y lenguajes estándar, el costo de desarrollo de software se mantiene en $0 MXN, contrastando radicalmente con el alto valor de los controladores de diagnóstico industriales. Este enfoque no solo valida la viabilidad económica del proyecto, sino que proporciona una plataforma flexible y escalable donde el usuario puede modificar la lógica de control para adaptarse a nuevos accesorios automotrices sin las restricciones de los sistemas cerrados del mercado.

4.4.3 Implementación del Protocolo en C++

El desarrollo del software en C++ asume el control total del envío de señal LIN BUS. Mediante interrupciones de hardware, se estableció un temporizador de precisión que activa el envío de tramas cada 20 ms. La generación de la señal de "Break" se realiza mediante una reconfiguración dinámica del puerto serial, desactivando la UART para forzar un nivel bajo manual durante 14 bits, calculado bajo la ecuación:

$$breakTimeDelay = \left(\frac{14.0}{LIN\ BAUDRATE}\right) x\ 1{,}000.000 \quad (3)$$

Posteriormente, el software calcula la paridad del identificador (PID) y el checksum mejorado de los datos. La función de lectura implementa un mecanismo de vigilancia que detecta el fin de trama tras 200 ciclos de inactividad, validando la respuesta de los esclavos en tiempo real. Este enfoque permite una flexibilidad total, permitiendo al usuario modificar los parámetros de la red para adaptarse a nuevos periféricos sin las restricciones de los sistemas cerrados del mercado.

A continuación, se describe en resumen el código utilizado para generar la comunicación LIN:

```
/* ------------------------------------------------------------------------
---
   PROYECTO: TESIS MAESTRIA CONTROLADOR LIN BUS DE BAJO COSTO
   SISTEMA: ESP32-C3 / TEENSY 4.1
   DESCRIPCIÓN: Implementación de Nodo Maestro y Gestión de Protocolo
   ------------------------------------------------------------------------
--- */

#define LINCS D5
#define LINTX D6
#define RXin  D8
#define LINBAUDRATE  19200
#define UARTBAUDRATE 115200

// Objeto para el segundo puerto serial (usando UART0 para el bus LIN)
HardwareSerial Serial2(0);

// --- Variables Globales ---
uint16_t breakTimeDelay;
bool online = true;
volatile bool timeToSendFrame = false;
uint8_t messageToSend = 0;

// Configuración de Timer
hw_timer_t * timer = NULL;
portMUX_TYPE timerMux = portMUX_INITIALIZER_UNLOCKED;

// --- Datos de Prueba (Frames) ---
uint8_t frameData1[] = {0x0c, 0x18, 0xc0, 0xf8, 0xc0, 0xff, 0x00, 0x00,
0x00};
uint8_t frameData2[] = {0x3c, 0x33, 0x06, 0xc7, 0xff, 0xff, 0xff, 0xff,
0xff};
uint8_t frameData3[] = {0x3c, 0x33, 0x06, 0xb2, 0x00, 0x61, 0x00, 0x01,
0x70};
uint16_t blinkCounter = 0;

// Variables para lectura
uint16_t linMessageLength;
uint8_t linData[200];

// --- Prototipos de Funciones ---
uint8_t idParity(uint8_t pid);
volatile byte dataChecksum(volatile byte* message, int length, uint16_t
sum);
enum genResult { GEN_OK, GEN_ERROR };
genResult generateBreakSyncSignal();
void sendFrame(uint8_t *message, uint8_t length, uint8_t iCsum);
void readLinMessage();
void sendTestMessage1();
void sendTestMessage2();
void sendTestMessage3();

// --- Interrupción del Timer ---
void ARDUINO_ISR_ATTR onTimer() {
  portENTER_CRITICAL_ISR(&timerMux);
```

```
  timeToSendFrame = true;
  portEXIT_CRITICAL_ISR(&timerMux);
}

// --- SETUP ---
void setup() {
  pinMode(LINCS, OUTPUT);
  pinMode(RXin, INPUT);
  digitalWrite(LINCS, LOW);

  // Configuración del puerto serie de depuración
  Serial.begin(UARTBAUDRATE);
  while (!Serial);

  // Configuración del puerto LIN (Serial2)
  // Nota: Se usan RXin (D8) y LINTX (D6)
  Serial2.begin(LINBAUDRATE, SERIAL_8N1, RXin, LINTX);

  // Timer: corre cada 20ms
  timer = timerBegin(0, 80, true);
  timerAttachInterrupt(timer, &onTimer, true);
  timerAlarmWrite(timer, 20000, true);
  timerAlarmEnable(timer);

  delay(50);
  digitalWrite(LINCS, HIGH);

  // Cálculos iniciales
  breakTimeDelay = (uint16_t)((14.0 / LINBAUDRATE) * 1000000);

  Serial.println("-----------------------------------------------");
  Serial.println("* LIN Master Transmitter ESP32C3 Ready");
  Serial.print("* Calculated Break Time Delay (usec) = ");
  Serial.println(breakTimeDelay);
  Serial.println("-----------------------------------------------");
}

// --- LOOP PRINCIPAL ---
void loop() {
  // Escuchar comandos por consola
  if (Serial.available() > 0) {
    char cmd = Serial.read();
    if (cmd == 'f') online = false;
    if (cmd == 'o') online = true;
  }

  if (online && timeToSendFrame) {
    timeToSendFrame = false;

    switch (messageToSend) {
      case 0: sendTestMessage1(); break;
      case 1: sendTestMessage2(); break;
      case 2: sendTestMessage3(); break;
    }

    messageToSend++;
    if (messageToSend > 2) messageToSend = 0;
  }
```

```
}

// --- Funciones de Protocolo LIN ---

uint8_t idParity(uint8_t pid) {
  uint8_t p0 = ((pid >> 0) + (pid >> 1) + (pid >> 2) + (pid >> 4)) & 1;
  uint8_t p1 = ~((pid >> 1) + (pid >> 3) + (pid >> 4) + (pid >> 5)) & 1;
  return (pid | (p0 | (p1 << 1)) << 6);
}

volatile byte dataChecksum(volatile byte* message, int length, uint16_t
sum) {
  for (int i = 0; i < length; i++) {
    sum += message[i];
    if (sum >= 256) sum -= 255;
  }
  return ((byte)~sum);
}

genResult generateBreakSyncSignal() {
  Serial2.flush();
  while (Serial2.available() > 0) Serial2.read();

  Serial2.end();
  pinMode(LINTX, OUTPUT);
  digitalWrite(LINTX, HIGH);
  delay(1);
  digitalWrite(LINTX, LOW);
  delayMicroseconds(breakTimeDelay);
  digitalWrite(LINTX, HIGH);

  // Reiniciar Serial2 tras el Break
  Serial2.begin(LINBAUDRATE, SERIAL_8N1, RXin, LINTX);
  delayMicroseconds(100);
  Serial2.write(0x55); // Sync Byte

  // Verificar Sync
  uint8_t timeout = 0;
  while ((Serial2.available() == 0) && (timeout < 200)) {
    delayMicroseconds(10);
    timeout++;
  }

  return (Serial2.read() == 0x55) ? GEN_OK : GEN_ERROR;
}

void sendFrame(uint8_t *message, uint8_t length, uint8_t iCsum) {
  byte csum = dataChecksum(&message[1], 8, (uint16_t)iCsum);
  generateBreakSyncSignal();

  Serial.print("T,");
  for (uint8_t i = 0; i < length; i++) {
    uint8_t data = (i == 0) ? idParity(message[0]) : message[i];
    Serial2.write(data);
    Serial.print(data, HEX);
    Serial.print(",");
  }
  Serial2.write(csum);
```

```
  Serial.print(csum, HEX);
  Serial.println("");
}

void readLinMessage() {
  bool waitForHigh = false;
  linMessageLength = 0;
  int nl = 0;

  Serial.print("R,");
  while (!waitForHigh) {
    if (Serial2.available() > 0) {
      linData[linMessageLength] = Serial2.read();
      Serial.print(linData[linMessageLength], HEX);
      Serial.print(",");
      linMessageLength++;
    }

    if (digitalRead(RXin) == 0) nl = 0;
    else nl++;

    if (nl > 200) { // Fin de frame por tiempo en alto
      waitForHigh = true;
      uint16_t checksum = 0;
      for (uint16_t x = 1; x < linMessageLength - 1; x++) {
        checksum += linData[x];
        if (checksum >= 256) checksum -= 255;
      }
      uint8_t csum = (uint8_t)~checksum;
      Serial.print(",CS:");
      Serial.print(csum, HEX);
      Serial.println(csum == linData[linMessageLength - 1] ? " OK" : "
ERR");
    } else {
      delayMicroseconds(10);
    }
  }
}

// --- Generación de Mensajes Específicos ---

void sendTestMessage1() {
  blinkCounter++;
  if (blinkCounter > 50) {
    blinkCounter = 0;
    frameData1[8] = (frameData1[8] == 0x0f) ? 0x04 : 0x0f;
  }
  frameData1[7] = frameData1[8];
  frameData1[6] = frameData1[8];
  sendFrame(&frameData1[0], 9, idParity(frameData1[0]));
}

void sendTestMessage2() {
  sendFrame(&frameData2[0], 9, 0);
  generateBreakSyncSignal();
  while (Serial2.available() > 0) Serial2.read();
  Serial2.write(0x7d);
  readLinMessage();
```

```
}

void sendTestMessage3() {
  sendFrame(&frameData3[0], 9, 0);
}
```

En conclusión, al haber eliminado la dependencia de librerías propietarias y archivos LDF confidenciales, el sistema adquiere una versatilidad superior, permitiendo realizar ingeniería inversa y control de periféricos genéricos de forma segura. El costo de desarrollo de software se mantiene en $0 contrastando con las costosas licencias industriales, lo que consolida a este prototipo como una plataforma de diagnóstico y control potente, escalable y financieramente accesible para el desarrollo tecnológico local.

5.4.4 Implementación de Android Studio

La aplicación, desarrollada en Android Studio® utilizando Java, gestiona la interacción entre el usuario y el hardware. Para asegurar escalabilidad y profesionalismo, se diseñó con una arquitectura modular. Esto significa que el sistema se construyó a partir de ventanas independientes, llamados módulos, cada uno con una función específica. Esta estructura encapsula las funciones de búsqueda, vinculación y transferencia de datos inalámbricos, lo que facilita la reutilización del código y desacopla la interfaz de usuario de la complejidad técnica de la pila de protocolos de Bluetooth, [18].

La aplicación opera mediante un conjunto de rutinas y subrutinas vinculadas a un punto central llamado “Main Activity”. Este programa realiza una verificación inicial para confirmar si el Bluetooth del celular está encendido, si la aplicación tiene los permisos de administrador para activarlo y si puede leer el listado de dispositivos Bluetooth, ver diagrama de flujo Fig. 52:

1. ¿Está el Bluetooth del celular encendido?
2. ¿La aplicación cuenta con los permisos de administrador para activar el bluetooth?
3. ¿La aplicación puede leer el listado de dispositivos bluetooth?

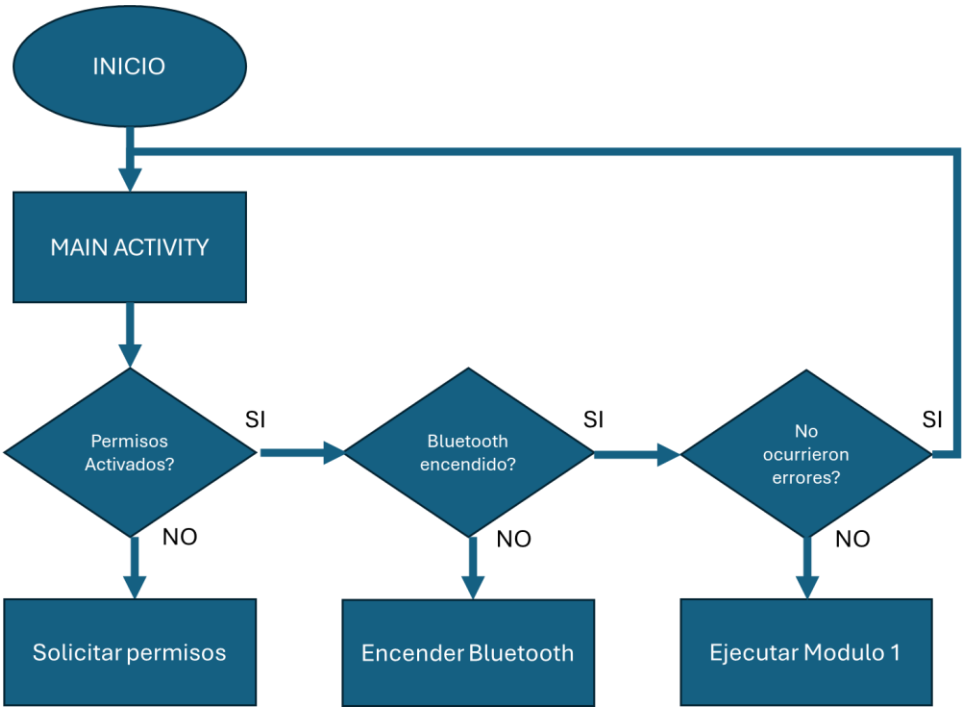


Figura 52. Diagrama de Flujo de aplicación, ventana principal.

La imagen 53 muestra una vista preliminar de la aplicación (interfaz) de la comunicación del usuario:

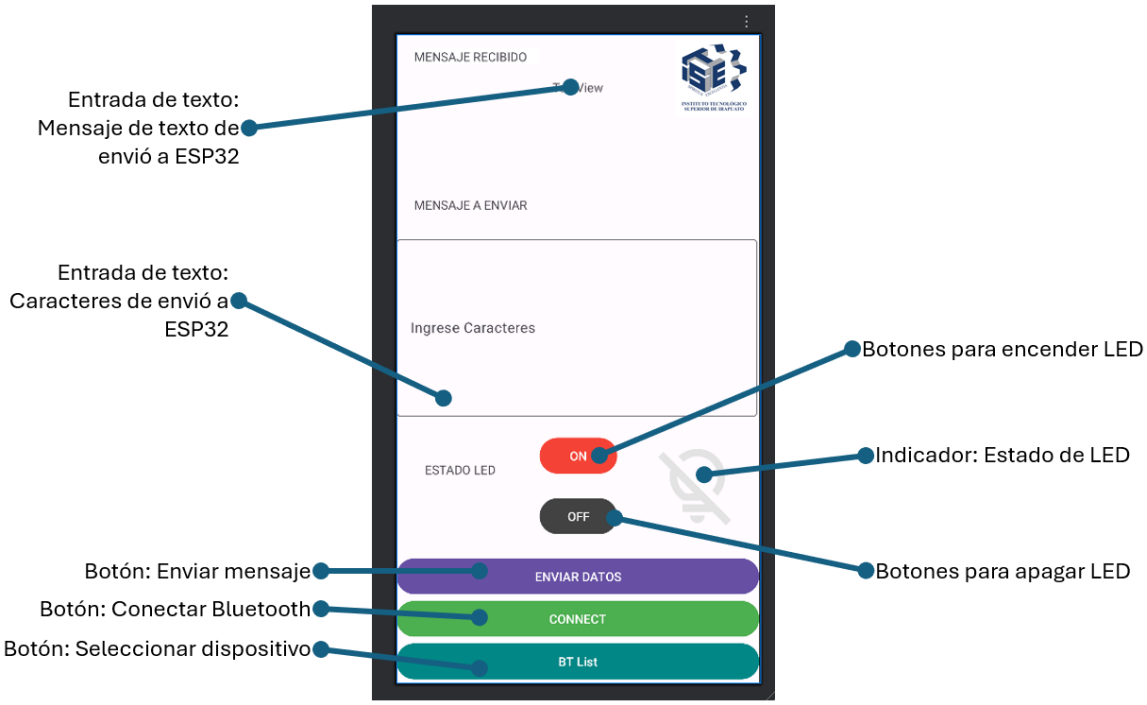


Figura 53. Ventana principal de la aplicación.

Un aspecto crítico en la fase de desarrollo es la gestión de la seguridad y los privilegios de red dentro del archivo AndroidManifest.xml. Debido a las restricciones de privacidad en las versiones modernas de Android (API 31 y superiores), el software debe declarar explícitamente permisos de BLUETOOTH_SCAN y BLUETOOTH_CONNECT. Estos permisos son esenciales para que la aplicación pueda identificar periféricos cercanos y establecer un enlace seguro sin que el sistema operativo bloquee la comunicación. Complementariamente, se incluyeron los permisos de localización fina y gruesa, requeridos por el adaptador inalámbrico para el proceso de descubrimiento de nuevos dispositivos, asegurando que el controlador pueda detectar el módulo ESP32-C3 en cualquier entorno de prueba.

La interfaz de usuario se organiza en torno a una actividad base que contiene varios fragmentos, lo que permite optimizar el uso de la memoria y asegurar una navegación fluida. La ventana principal para la vinculación de dispositivos Bluetooth se denomina "Lista de componentes Bluetooth". Esta ventana utiliza una vista personalizada para mostrar los dispositivos disponibles, clasificándolos en dos categorías principales: dispositivos previamente vinculados y dispositivos detectados en tiempo real. Para el descubrimiento de hardware nuevo, se ha implementado un componente del sistema llamado "Lista desplegable". Este componente escucha activamente las notificaciones del adaptador Bluetooth y actualiza visualmente la lista cada vez que se detecta una nueva dirección. Esta metodología permite al usuario seleccionar el controlador específico del bus LIN de forma intuitiva y rápida, ver diagrama de flujo Fig. 54.

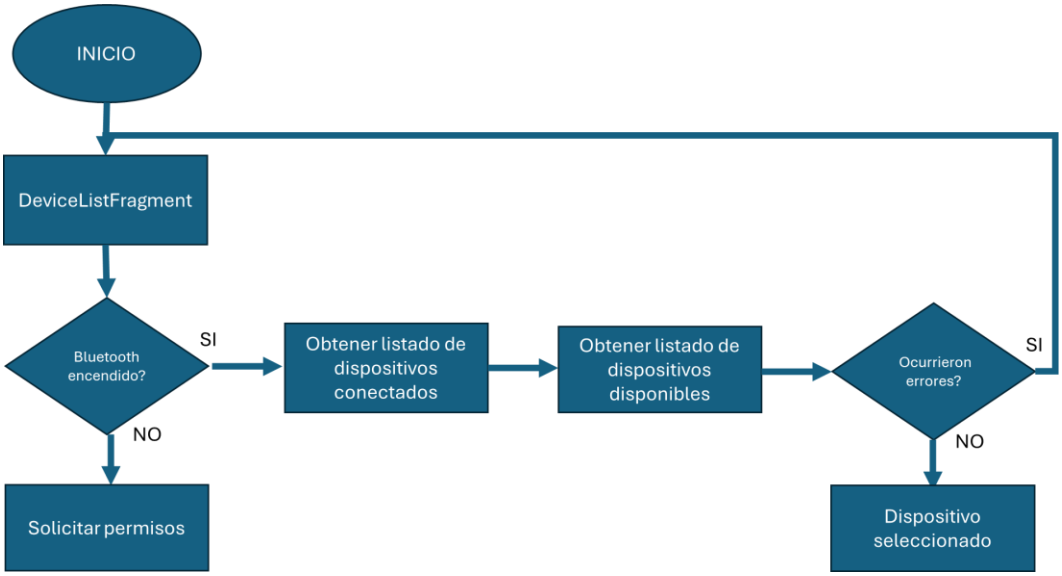


Figura 54. Diagrama de Flujo de aplicación, ventana lista de componentes disponibles o vinculados vía Bluetooth.

La imagen 55 muestra una vista preliminar de la lista de componentes disponibles o vinculados vía Bluetooth:

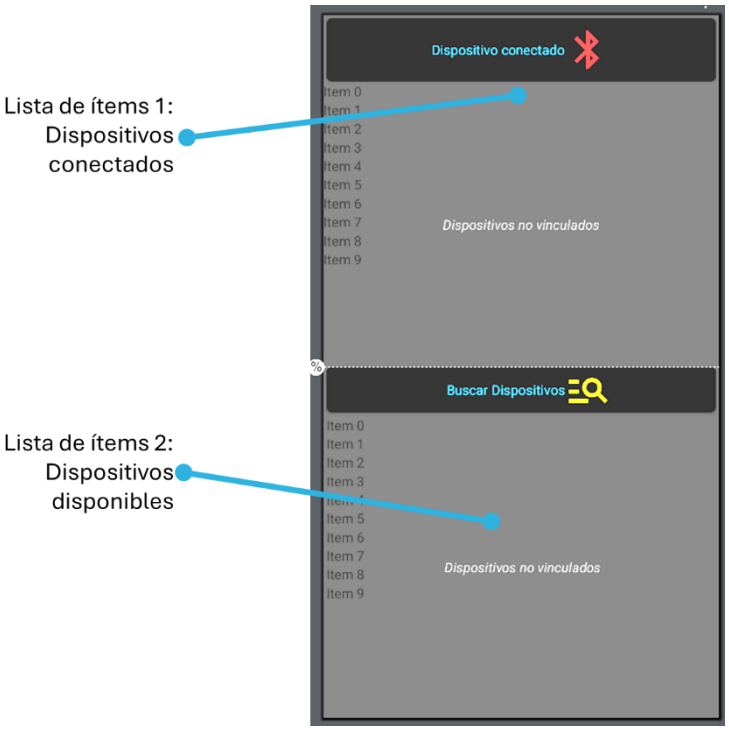


Figura 55. Ventana de lista de componentes conectados vía Bluetooth.

La lógica de comunicación bidireccional se basa en la creación de hilos de ejecución secundarios para evitar el congelamiento de la pantalla táctil durante la transferencia de información. A través de flujos de entrada y salida de datos, la aplicación móvil es capaz de enviar instrucciones de control hacia el circuito integrado ESP32 y mantener la comunicación. Este flujo constante de información se sincroniza con la interfaz mediante objetos de observación, permitiendo que los indicadores visuales en el teléfono reflejen en tiempo real el estado de los actuadores conectados al bus vehicular, ver diagrama de flujo Fig. 56.

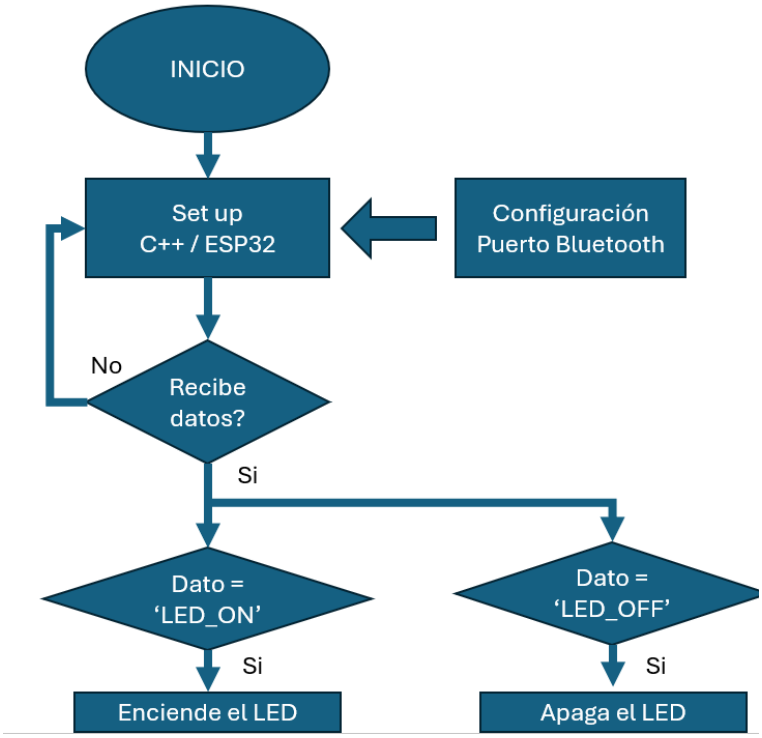


Figura 56. Diagrama de Flujo de aplicación, conexión Bluetooth-ESP32.

En la Figura 57 se ilustra la arquitectura de desarrollo de la aplicación a nivel general:

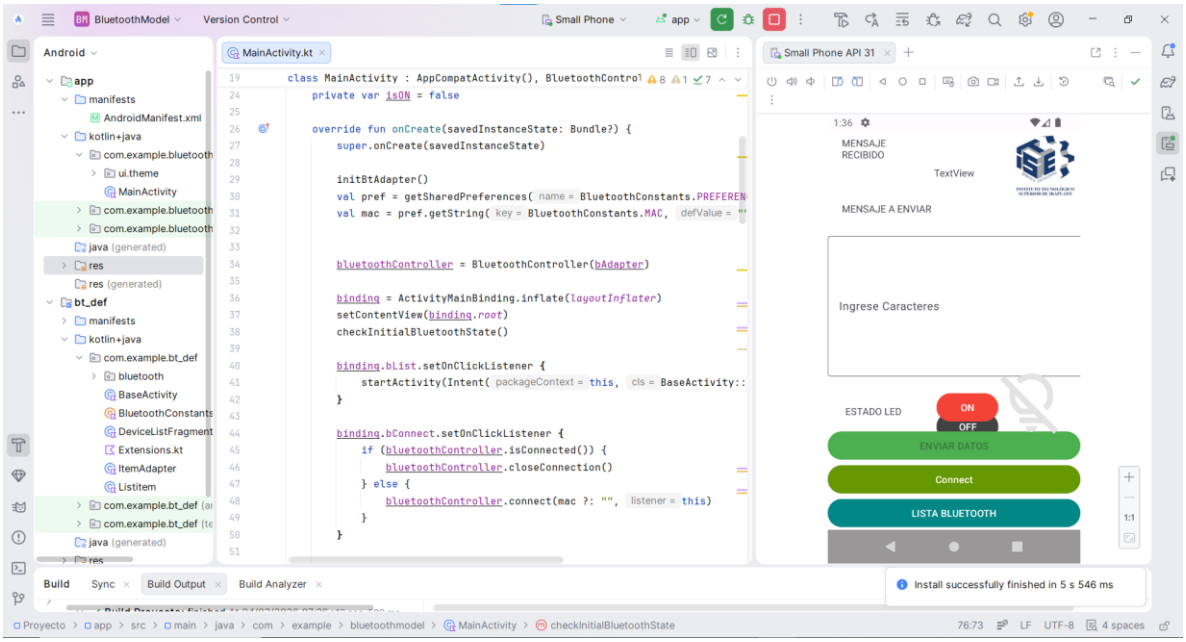


Figura 57. Desarrollo aplicación en Android Studio.

6. RESULTADOS

6.1 Resultados Comunicación LIN

Siguiendo la configuración eléctrica del esquemático del capítulo 4.3, la Figura 58 muestra la configuración y montaje de integración del equipo (osciloscopio, computadora) acoplado al hardware y software desarrollado para obtener la señal esperada del Bus LIN. El comando utilizado es un comando de prueba que cumple con todas las especificaciones LIN, ver Tabla 8.

Tabla 8. Ejemplo de comando LIN Bus para comprobar comunicación LIN.

Sync	ID	Data Length	Data Position	Checksum
0x55	0xA	0x01	0x64	0xAB

Donde:

- **0x55**: Es el byte de sincronización del Bus LIN.
- **0xA:** Este es el byte de identificación de la ventana.
- **0x01**: Este byte indica que solo se enviará una instrucción.
- **0x64**: Representa el posicionamiento del motor para la activación de la ventana. El valor decimal de 0x64 es 100, que en porcentaje indica el 100%. Para cerrar la ventana, este byte debe cambiarse a 0x00, lo que representa el 0% de la posición original del motor.
- **0xAB**: Este es el byte de Checksum, que se usa para verificar que la instrucción se ha enviado y recibido correctamente. Su valor no debe exceder de 256 (el valor máximo para un byte de 8 bits).

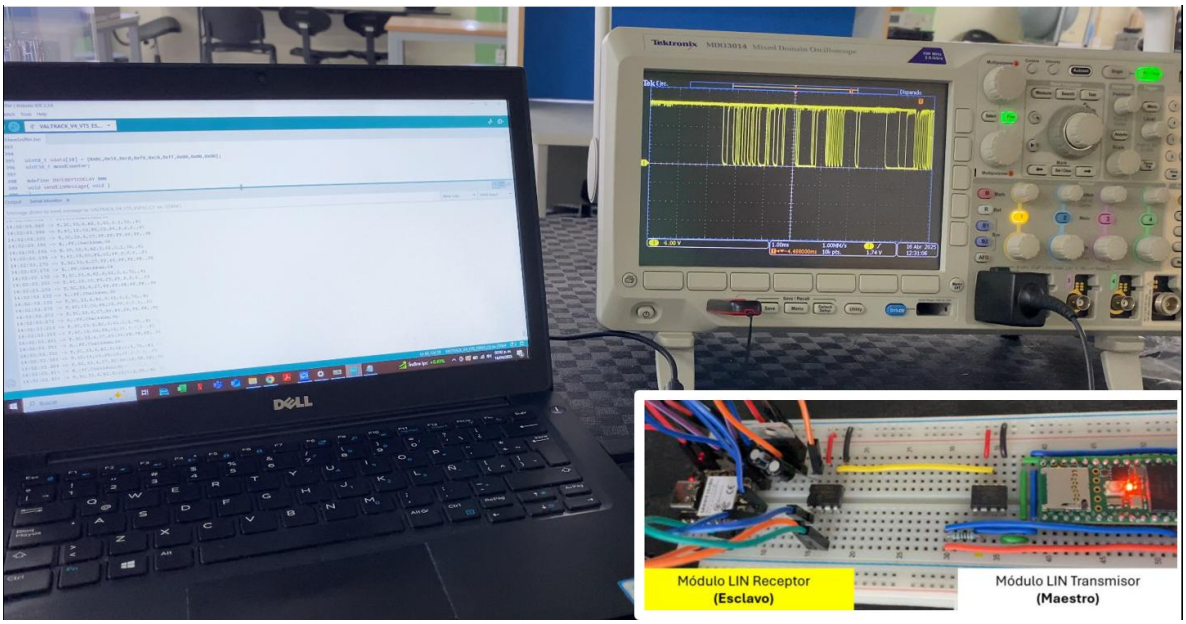


Figura 58. Montaje y Configuración del Sistema prototipo de Comunicación LIN Bus.

El análisis de la señal capturada en el osciloscopio (Figura 59) confirma el cumplimiento de las especificaciones del protocolo LIN: los niveles de tensión del bus se mantienen entre 0 y 12V, es una señal periódica repetitiva y se establece una velocidad de 20 kbps para cumplir con el estándar. Además, la estructura de la trama muestra un tiempo de sincronización de respuesta, lo que demuestra que la transmisión cumple con la temporización, los niveles de voltaje y el formato del protocolo LIN, confirmando su correcta implementación.

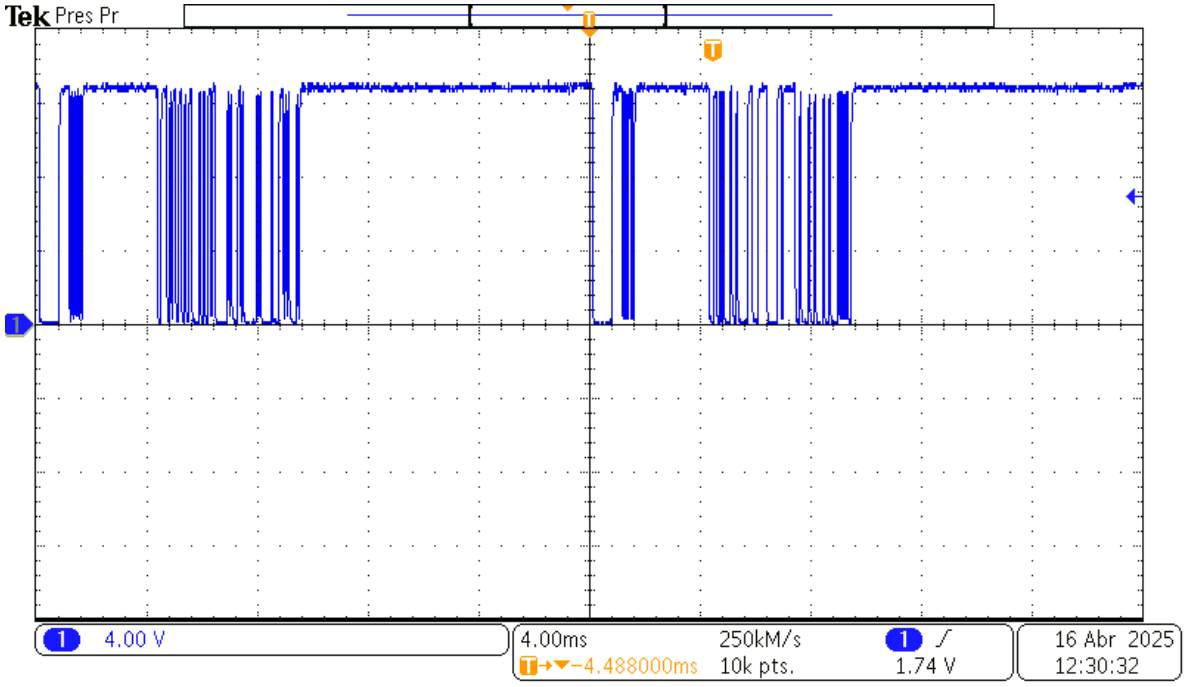


Figura 59. Resultado toma del osciloscopio de señal experimental LIN BUS.

6.2 Resultados Comunicación LIN-Bluetooth

Para el establecimiento del enlace inalámbrico, se desplegó la aplicación desarrollada en el terminal móvil OPPO CPH2205 (ver Fig. 60), el cual funge como la interfaz principal de control y supervisión del sistema. El proceso de comunicación se articula mediante una arquitectura de puente entre el protocolo de red local interconectada (LIN) y el módulo de radiofrecuencia Bluetooth, gestionado a través del microcontrolador ESP32C3 (ver Fig. 61). Tras la carga del software en el dispositivo terminal, se procedió a la sincronización de los nodos, permitiendo la transferencia de datos en tiempo real entre la capa de aplicación Android y la red de control automotriz.

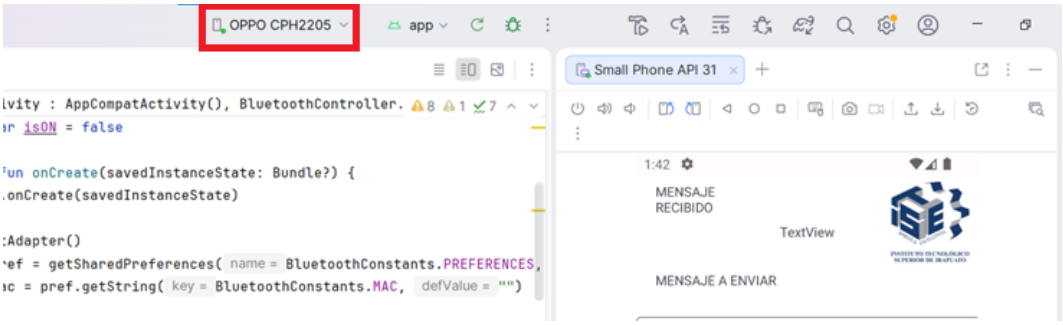


Figura 60. Sincronización Android Studio a celular OPPO CPH2205

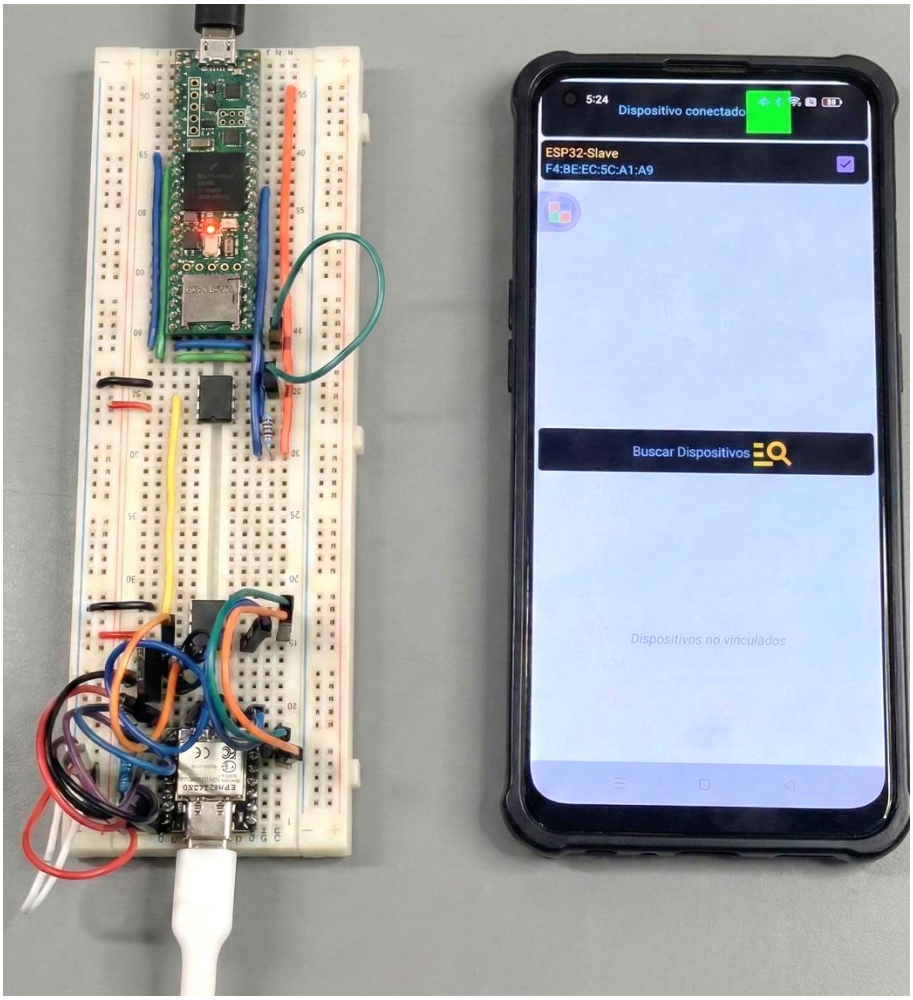
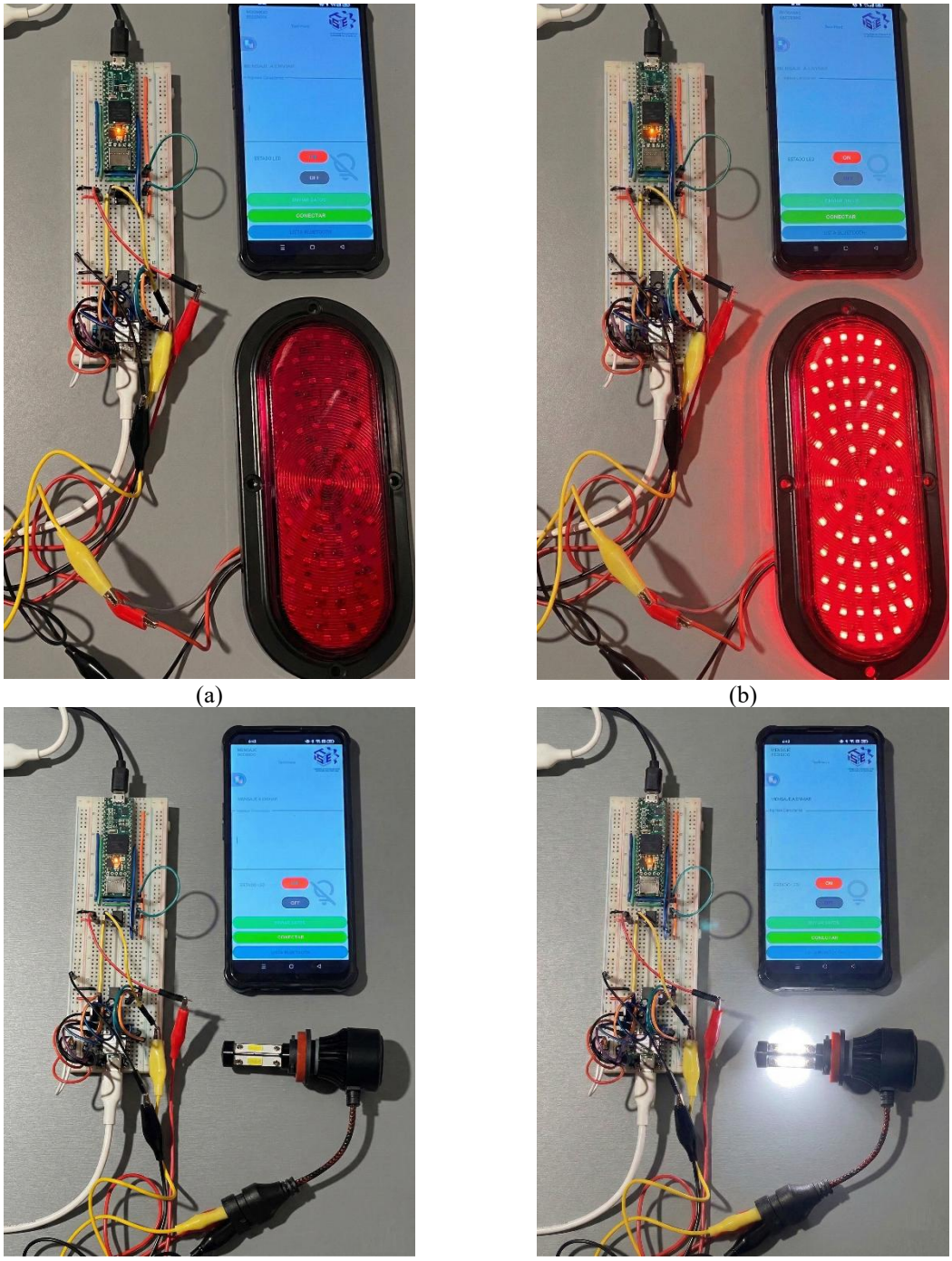


Figura 61. Conexión Bluetooth establecida entre ESP32C3, Teensy 4.1 y celular OPPO CPH2205

6.3 Resultados Prueba accesorios automotrices

A continuación, se presentan los resultados de la integración de accesorios automotrices al bus de datos. Los dispositivos bajo prueba, seleccionado, fueron:

- **Calavera LED (roja):** Utilizada para pruebas de modulación de intensidad.
- **Luz antiniebla LED (blanca):** Empleada para validar la estabilidad de la conexión bajo carga.
- **Espejo lateral Suzuki Grand Vitara (2007):** Utilizado para verificar el control de motor de comunicación LIN.



(a) (b)

(c) (d)

Figura 62. Resultados prueba conexión LIN-Bluetooth con Calavera LED, y luz antiniebla LED.

En la figura 62 se muestran los resultados obtenidos descritos a detalle a continuación.

- Fig. 62 (a): CALAVERA LED con comando apagado (0x00).
- Fig. 62 (b): CALAVERA LED con comando encendido (0x64).
- Fig. 62 (c): LUZ antiniebla LED con comando apagado (0x00).
- Fig. 62 (d): LUZ antiniebla LED con comando encendido (0x64).

Durante las pruebas efectuadas con el lateral de un Suzuki Grand Vitara 2007 (ver Figura 63), no fue posible concretar la activación del motor. Debido a que el componente es una unidad original (Ensambladora Suzuki®) y no se dispone del archivo LDF (LIN Description File) del fabricante, existe una restricción técnica para identificar los ID (identificadores) y los comandos de control específicos. Sin esta documentación, el protocolo de comunicación permanece cerrado, impidiendo el accionamiento del sistema.

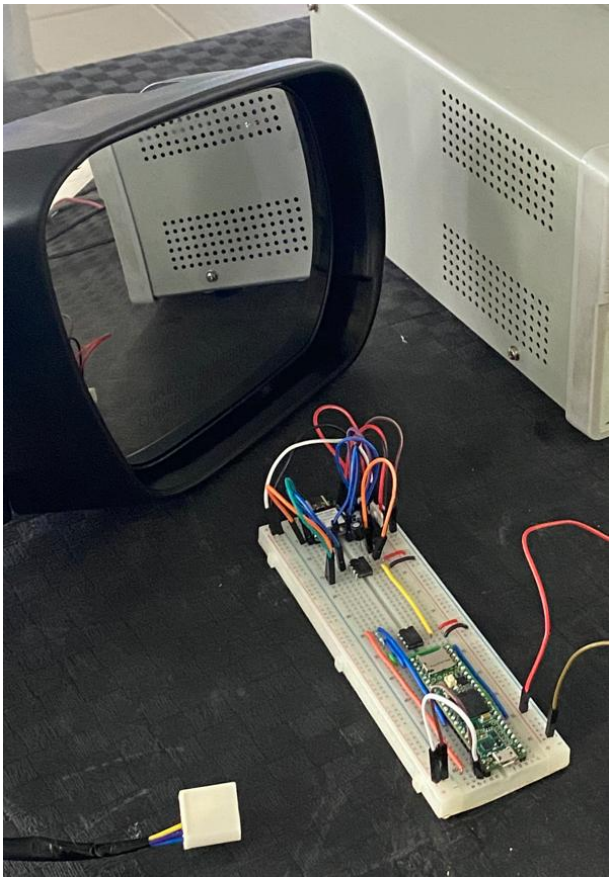


Figura 63. Resultados prueba conexión LIN-Bluetooth con Espejo lateral Suzuki Grand Vitara (2007).

El análisis de la capa física del bus de comunicación permite identificar fenómenos de interferencia electromagnética que comprometen la calidad de los datos transmitidos. En la Figura 64, se observa el comportamiento de la señal LIN al integrar el espejo lateral Suzuki al sistema de control. La traza muestra una degradación notable en el estado de la señal, la captura revela la presencia de ruido de alta frecuencia y transitorios de conmutación que se manifiestan como un engrosamiento de la línea base superior. Esta distorsión sugiere un acoplamiento de ruido provocado por las cargas inductivas de los actuadores del espejo o por una deficiencia en el filtrado de la fuente de alimentación compartida, lo que reduce significativamente el margen de inmunidad al ruido del sistema.

A pesar de que los flancos de subida y bajada conservan una definición suficiente para ser detectados por un componente o “accesorio”, la funcionalidad del componente se vio limitada durante las pruebas experimentales. La presencia de tramas de datos en la Figura 64 confirma que existe un intento de comunicación a nivel eléctrico; no obstante, la activación de las funciones de plegado o ajuste del espejo no pudo concretarse debido a la ausencia del archivo de descripción de LIN (LDF). Este archivo es estrictamente necesario para definir los identificadores de trama específicos del fabricante y los algoritmos de suma de comprobación (*checksum*). Sin esta base de datos, el nodo maestro es incapaz de estructurar mensajes que el esclavo Suzuki reconozca como válidos, resultando en una comunicación eléctricamente ruidosa pero lógicamente inerte.

La captura analizada sirve como evidencia de que la integridad de la señal es una condición necesaria, pero no suficiente, para la interoperabilidad de sistemas embebidos bajo estándares industriales cuando no se dispone de la capa de descripción de red completa.

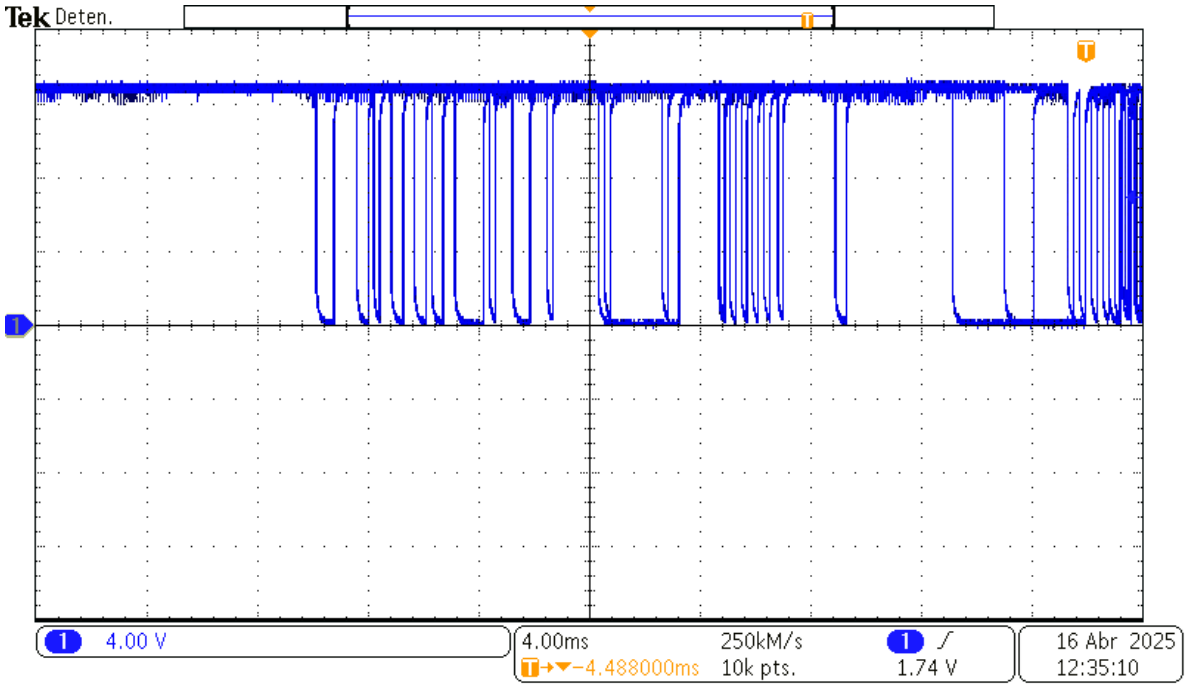


Figura 64. Análisis de la integridad de la señal en el bus LIN y presencia de ruido en el nivel de 12V tras la integración del espejo lateral Suzuki.

7. CONCLUSIÓN

La metodología propuesta en este proyecto ha demostrado una alta eficacia al lograr la integración de un ecosistema de control completo, que abarca desde la interfaz de usuario en Android Studio hasta la capa física del Bus LIN gestionada por el microcontrolador ESP32C3.

La validación técnica del sistema fue exitosa en dos vertientes críticas: primero, mediante la instrumentación con osciloscopio, se corroboró la integridad de las tramas de datos y la concordancia de la señal con el estándar LIN; segundo, se alcanzó una operatividad funcional plena al controlar nodos de iluminación LED genéricos, demostrando que la lógica de programación y el diseño del hardware son correctos para la comunicación en red.

No obstante, la experimentación con el lateral original del Suzuki Grand Vitara 2007 reveló una limitación técnica propia de la industria automotriz: la dependencia de protocolos cerrados. Si bien el sistema emitió las señales correctamente, la ausencia del archivo LDF (LIN Description File) de la compañía impidió la activación del motor, subrayando que, en componentes OEM (Original Equipment Manufacturer), la posesión de los identificadores y comandos específicos es tan vital como el hardware mismo. Esta distinción entre el éxito con componentes genéricos y la restricción con los originales aporta un valor diagnóstico fundamental al proyecto.

Finalmente, este desarrollo destaca por su alta competitividad económica. Con una inversión total de $767.24 MXN, la solución propuesta se posiciona como una alternativa disruptiva frente a herramientas profesionales del mercado. Al comparar este costo con dispositivos de alta gama como el Baby LIN (valorado en aproximadamente $11,925 MXN o €550), se evidencia un ahorro superior al 93%. Esta democratización tecnológica permite que pequeñas empresas y laboratorios con recursos limitados accedan a herramientas de validación temprana. En última instancia, la implementación de este controlador de bajo costo representa un paso estratégico para optimizar los ciclos de diseño y evitar sanciones económicas por retrasos en la calidad o en los plazos de entrega de la serie automotriz.

8. TRABAJOS FUTUROS

Como trabajo futuro, se plantea la evolución de este prototipo hacia la manufactura de un circuito impreso o PCB dedicado visto en la sección 4.3.1 del presente trabajo, con el objetivo de mejorar la robustez del sistema, reducir el ruido electromagnético y compactar el hardware para su implementación en entornos vehiculares reales. Asimismo, resultaría ideal establecer vínculos estratégicos con proveedores o fabricantes del sector para colaborar en la validación de componentes no genéricos. El acceso formal a archivos LDF bajo acuerdos de cooperación permitiría corroborar el desempeño del controlador frente a una gama más amplia de actuadores originales, consolidando así una herramienta de diagnóstico integral que combine la accesibilidad económica con la precisión técnica requerida por la industria automotriz contemporánea.

REFERENCIAS BIBLIOGRÁFICAS

[1] T. Sareh, «Master of Science Thesis in the program Networks and Distributed Systems,» Chalmers University of Technology, Göteborg, Sweden, 2014.

[2] F. Páez y H. Kaschel, «Design and Testing of a Computer Security Layer for the LIN BUS,» *Sensors,* vol. 6901, nº https://doi.org/10.3390/s22186901, p. 22, 2022.

[3] L. I. E. GmbH, « Lipowsky Industrie Elektronik GmbH,» Managing Director Andreas Lipowsky, [En línea]. Available: https://lipowsky.com/products/Baby-Lin-II. [Último acceso: 13 Julio 2025].

[4] N. I. CORP, «USB-8476,» 2025 NATIONAL INSTRUMENTS CORP. TODOS LOS DERECHOS RESERVADOS, [En línea]. Available: https://www.ni.com/es-mx/support/model.usb-8476.html?srsltid=AfmBOorriINShCKZJtAi7Syfq0tMy6djeQ6OdWa-Uo4E27GrcKQ_BYmW. [Último acceso: 13 Julio 2025].

[5] J. Marte Michelen, «Protocolo de Comunicación (USART, I2C & SPI),» PONTIFICIA UNIVERSIDAD CATÓLICA MADRE Y MAESTRE, República Dominicana, 2020.

[6] E. Hackett, «LIN Protocol and Physical Layer Requirements,» Texas Instruments Incorporated, Dallas, Texas, 2022.

[7] J. Artal, J. Caraballo y R. Dufo, «Protocolo CAN/LIN-Bus. Implementación de una red de comunicación serie de bajo coste.,» Universidad de Zaragoza. Departamento de Ingeniería Eléctrica. , Zaragoza, España. , 2014.

[8] V. I. GmbH, «VN1600 - Interfaces de red con USB y Ethernet para CAN / CAN FD / CAN XL, LIN, K-Line, J1708 e IO,» Vector Informatik GmbH, [En línea]. Available: https://www-vector-com.translate.goog/es/es/productos/products-a-z/hardware/network-interfaces/vn16xx/?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es&_x_tr_pto=tc.

[9] L. G. Sánchez Vela, M. J. Molano Clemente, M. D. J. Fabela Gallegos, M. Martínez Madrid, J. R. Hernández Jiménez, D. Vázquez Vega y O. Flores Centeno, «Revisión documental del protocolo CAN como herramienta de comunicación y aplicación en vehículos,» INSTITUTO MEXICANO DEL TRANSPORTE. Publicación Técnica No. 474, Sanfandila, Qro, 2016.

[10] © LIN Consortium, 2006., «Protocol Specification,» de *LIN Specification Package Revision 2.1*, LIN is a registered Trademark ®. All rights reserved., 2006, pp. 24-56.

[11] S. Youyi, *Desarrollo de una pasarela multired LIN para aplicaciones en la industria automotriz,* Barcelona, España.: UPC Escola Politècnica Superior d'Enginyeria de Vilanova i la Geltrú (EPSEVG), 2022.

[12] J. Quiroz y R. M. K. Rubio, *Análisis Sectorial: Industria Automotriz,* Grupo Financiero MONEX®, 2025.

[13] ©DR 2025, Instituto Nacional de Estadística y Geografía, «Conociendo la industria automotriz,» INEGI, Aguascalientes, 2025.

[14] C. M. Garcia Remigio, M. A. Cardenete, P. Campoy Muñoz y F. Venegas Martínez, «Valoración del impacto de la industria automotriz en la economía mexicana: una aproximación mediante matrices de contabilidad social,» *EL TRIMESTRE ECONÓMICO,* vol. LXXXVII, nº 346, pp. 437-461, 2020.

[15] A. Emadi, «Transportation 2.0,» *IEEE Power and Energy Magazine,* vol. 9, nº doi: 10.1109/MPR.2011.941320., pp. 18-19, 2011.

[16] © Robert Bosch GmbH, Bosch Automotive Electrics and Automotive Electronics, Plochingen, Germany: DOI 10.1007/978-3-658-01784-2_1, © Springer Fachmedien Wiesbaden 2014, 2014.

[17] G. MOM, The Evolution of Automotive Technology, Warrendale, Pennsylvania, USA: SAE International, 2023.

[18] N. Ru., «Android + ESP32 & Arduino [Serie de videos].,» YouTube, 2022. [En línea]. Available: https://www.youtube.com/playlist?list=PLmjT2NFTgg1c-CQiAdezZpi9RaFmN1n7I.

[19] A. [. d. Datos], *PJRC Teensy 4.1 Development Board, PRODUCT ID: 4622,* 2025.

[20] ©Microchip Technology Inc., *MCP2003/4 - LIN J2602 Transceiver,* © Microchip Technology Incorporated, Printed in the U.S.A. ISBN: 978-1-60932-080-5, 2025.

[21] Espressif Systems, *ESP32-C3 Series Datasheet v1.2,* Shanghai: Espressif Systems (Shanghai) Co., Ltd. All rights reserved., 2022.

[22] Copperhill Technologies, PiCAN FD with LIN Bus for Raspberry Pi, https://copperhilltech.com/pican-fd-with-lin-bus-for-raspberry-pi/: copperhilltech.com. [Online], 2025.

[23] D. A. M. Nasser, *Automotive Cybersecurity Engineering Handbook: The Automotive Engineer's Roadmap to Cyber-resilient Vehicles.,* Birmingham, UK.: 1st Edition. Packt Publishing. ISBN 978-1-80107-653-1, 2023.

[24] IPC, *Qualification and Performance Specification for Rigid Printed Boards, Automotive Series Addendum*, IPC-6012DA, May 2016.

[25] IATF, *Quality management system requirements for automotive production and relevant service parts organizations*, IATF 16949:2016, Oct. 2016.

[26] IPC, *Guidelines for Design, Selection and Application of Conformal Coatings*, IPC-HDBK-830B, Oct. 2013.

[27] ISO, *Road vehicles — Unified diagnostic services (UDS) — Part 1: Application layer*, ISO 14229-1:2020, Oct. 2020.

[28] NXP Semiconductors, "TJA1021: LIN 2.1/SAE J2602 transceiver," TJA1021 Datasheet, Rev. 5, Nov. 2017. [Online]. Available: https://www.nxp.com/docs/en/data-sheet/TJA1021.pdf

[29] SK Pang Electronics, "LIN-Bus Breakout Board User Guide," LINBUS-BRK, Rev. 1.0, 2026. [Online]. Available: https://www.skpang.co.uk/products/lin-bus-breakout-board