

Article

Design and Performance Analysis of an Embedded CAN-Wi-Fi Gateway for Connected Vehicle Networks

Yuan-Chih Chung ^{1,*}  and Ching-Hung Lee ^{1,2} ¹ Institute of Electrical and Control Engineering, National Yang Ming Chiao Tung University, Hsinchu 300093, Taiwan; chl@nycu.edu.tw² Department of Electrical Engineering, Chung Yuan Christian University, Taoyuan 320314, Taiwan

* Correspondence: timchung.ee05@nycu.edu.tw

Abstract

Controller Area Network (CAN) is widely used in in-vehicle communication systems because of its reliability and robustness. However, conventional CAN networks are primarily designed for wired communication and do not directly support wireless access for local visualization, remote monitoring, or connected-vehicle services. This paper presents an embedded CAN-Wi-Fi gateway architecture for connected-vehicle monitoring applications. The proposed system integrates a CAN interface, an embedded gateway controller, a wireless communication module, a mobile human-machine interface (HMI), and an edge-server-based remote monitoring service. A modular hardware and software framework is developed to support CAN message acquisition, identifier-based filtering, priority classification, queue-based buffering, structured packet generation, wireless transmission, local visualization, and remote data access. To reduce unnecessary wireless traffic and improve the responsiveness of urgent monitoring traffic, the gateway selectively forwards monitoring-related CAN identifiers and applies strict-priority dispatch using separate High-, Medium-, and Low-priority queues. The system is verified using representative Saab P-bus-based vehicle data packets derived from publicly available reverse-engineering information. Controlled wireless experiments achieved 100% packet delivery under offered traffic rates of 20, 50, and 100 frames/s, with application-level throughput increasing from approximately 33.8 to 170.0 kbps and round-trip time (RTT)-based estimated one-way wireless delay ranging from approximately 4.1 to 10.8 ms. Under transient burst contention, priority-based dispatch reduced the mean High-priority warning-traffic queue waiting time from 31.116 ms with single-FIFO forwarding to 2.268 ms, reduced the P95 value from 53.560 to 4.808 ms, and reduced the 25 ms threshold-exceedance ratio from approximately 61.7% to 0%. The results demonstrate the feasibility of the proposed CAN-Wi-Fi gateway for integrated local and remote vehicle monitoring and show that strict-priority dispatch can substantially improve the responsiveness of urgent monitoring traffic under temporary gateway contention.



Academic Editors: Nurul Sarkar, Dionisis Kandris, Djamel Djenouri and Piotr Zwierzykowski

Received: 17 August 2026

Revised: 7 September 2026

Accepted: 14 September 2026

Published: 16 September 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

Keywords: CAN bus; CAN-Wi-Fi gateway; vehicle monitoring; in-vehicle network; wireless communication; edge server; message filtering; connected vehicle

1. Introduction

The rapid advancement of intelligent transportation systems, connected vehicles, and vehicle data services has significantly increased the demand for real-time access to in-vehicle information. Modern vehicles are equipped with numerous electronic control units

(ECUs) that exchange control and status information through in-vehicle communication networks. Among these technologies, the Controller Area Network (CAN) remains one of the most widely adopted communication protocols because of its reliability, robustness, priority-based arbitration, and suitability for distributed embedded control systems [1–4].

Although CAN provides efficient communication among ECUs within a vehicle, its data are typically confined to the local in-vehicle network and are not directly accessible through wireless or Internet-based communication systems. At the same time, connected vehicle services such as local visualization, remote diagnostics, fleet management, maintenance support, and service-oriented vehicle monitoring require a mechanism for transferring selected in-vehicle data to external monitoring platforms. Therefore, an effective gateway architecture is required to bridge CAN-based vehicle networks with wireless communication interfaces and remote monitoring services.

Previous studies have investigated a wide range of automotive gateway and vehicle-monitoring solutions, including CAN-to-CAN, CAN-to-Ethernet, CAN-to-GPRS, CAN-to-FlexRay, wireless CAN frame relaying, telematics gateways, and vehicle data acquisition platforms [5–13]. These studies establish the feasibility of connecting CAN-based vehicle networks with heterogeneous communication systems. However, their principal objectives have generally emphasized protocol conversion, data acquisition, external connectivity, or application-specific monitoring. Comparatively less attention has been given to the internal forwarding behavior of an embedded CAN–wireless gateway when messages with different monitoring importance compete for the limited gateway forwarding capacity. In particular, quantitative baseline comparisons of priority-based dispatch against non-priority FIFO forwarding under transient contention are rarely reported together with local visualization and remote monitoring functionality.

Recent IoT-based vehicle-monitoring studies have further shown that wireless communication, MQTT-style messaging, and server-side monitoring platforms can support vehicle data visualization and remote access services [14–17]. In addition, modern in-vehicle network research has considered CAN FD, SOME/IP, Automotive Ethernet, and 10BASE-T1S for higher-bandwidth and service-oriented vehicle architectures [18–21]. These developments do not eliminate the need to support existing CAN-based systems; rather, they underscore the importance of gateways that can selectively transfer legacy CAN data to wireless, edge-server, and Ethernet-oriented environments.

Our previous work addressed formal timing analysis of CAN communication [22], whereas the present study focuses on the implementation and experimental evaluation of CAN–Wi-Fi gateway forwarding and monitoring behavior.

To address these issues, this paper presents an embedded CAN–Wi-Fi gateway architecture that integrates a CAN interface, an embedded gateway controller, a wireless communication module, a mobile human–machine interface, and an edge-server-based remote monitoring service. The gateway performs identifier-based message selection, priority classification, queue-based buffering, structured packet generation, and scheduled wireless forwarding. A strict-priority dispatch mechanism is implemented using separate High-, Medium-, and Low-priority queues, while FIFO ordering is retained within each class. In addition to functional verification of the local and remote monitoring path, the proposed scheduler is quantitatively evaluated against a single-FIFO baseline under controlled transient burst contention. Representative Saab P-bus-based vehicle data are used to configure the CAN-derived traffic employed in the experiments.

The main contributions of this work are summarized as follows:

1. A modular CAN–Wi-Fi gateway architecture is developed to bridge conventional CAN-based vehicle networks with local mobile and remote server-based monitoring platforms.

2. An integrated hardware-and-software framework is implemented for CAN-message acquisition, identifier-based filtering, priority classification, queue management, structured packet generation, wireless transmission, mobile visualization, and remote monitoring.
3. A strict-priority dispatch mechanism with separate High-, Medium-, and Low-priority queues is developed to provide differentiated forwarding of monitoring traffic while preserving FIFO ordering within each priority class.
4. A controlled baseline experiment compares the proposed priority-based scheduler with single-FIFO forwarding under transient burst contention. For High-priority warning traffic, the proposed scheduler reduces the mean queue waiting time from 31.116 to 2.268 ms, reduces the P95 value from 53.560 to 4.808 ms, and reduces the 25 ms threshold-exceedance ratio from approximately 61.7% to 0%.
5. The complete communication path is verified using representative Saab P-bus-based traffic via both a mobile HMI and an edge-server remote monitoring interface, while controlled wireless experiments evaluate packet delivery, application-level throughput, and RTT-based estimated one-way wireless delay.

The remainder of this paper is organized as follows. Section 2 reviews related work on CAN-based gateways, vehicle monitoring, and in-vehicle communication systems. Section 3 presents the proposed CAN–Wi-Fi gateway architecture, including the hardware configuration, software framework, message dispatch mechanism, packet format, and system operation flow. Section 4 describes the experimental setup and verifies the proposed system through representative vehicle-data transmission, local HMI visualization, and remote monitoring. Section 5 concludes the paper.

2. Related Work and Background

2.1. Controller Area Network (CAN)

CAN is a broadcast, event-triggered serial communication protocol widely used to connect electronic control units (ECUs) in automotive and industrial systems. Classical CAN supports data rates of up to 1 Mbit/s and is designed for the reliable transmission of short, priority-based messages under electrically noisy operating conditions. Rather than using destination addresses, each CAN frame contains an identifier that represents both the message type and its transmission priority. A Classical CAN data frame can carry up to 8 bytes of payload data [3,23].

CAN primarily defines the data-link layer and relevant portions of the physical layer. At the physical layer, differential signaling improves resistance to electromagnetic interference, while the dominant and recessive bus states implement a wired-AND logic function. A dominant bit overrides a recessive bit when multiple nodes transmit simultaneously [4].

When the CAN bus is idle, any node may initiate transmission. If multiple nodes begin transmitting simultaneously, nondestructive bitwise arbitration determines which message gains access to the bus. Each transmitting node monitors the bus while sending its identifier. A node that transmits a recessive bit but detects a dominant bit loses arbitration and immediately stops transmitting. This arbitration loss is a normal part of CAN bus access control and does not indicate a transmission failure or a physical-layer error; the losing node retains its pending frame and retries when the bus becomes available. Consequently, the frame with the lowest numeric identifier has the highest priority and continues to be transmitted without corruption due to the arbitration process [3,4].

From a system-level perspective, CAN communication can be represented as a sequence of message generation, bus-access arbitration, frame transmission, and transmission completion, as illustrated in Figure 1. When multiple ECUs request bus access simultaneously, the identifier-based arbitration mechanism determines which message is transmitted

first, while lower-priority messages that lose arbitration remain pending until the bus becomes available.

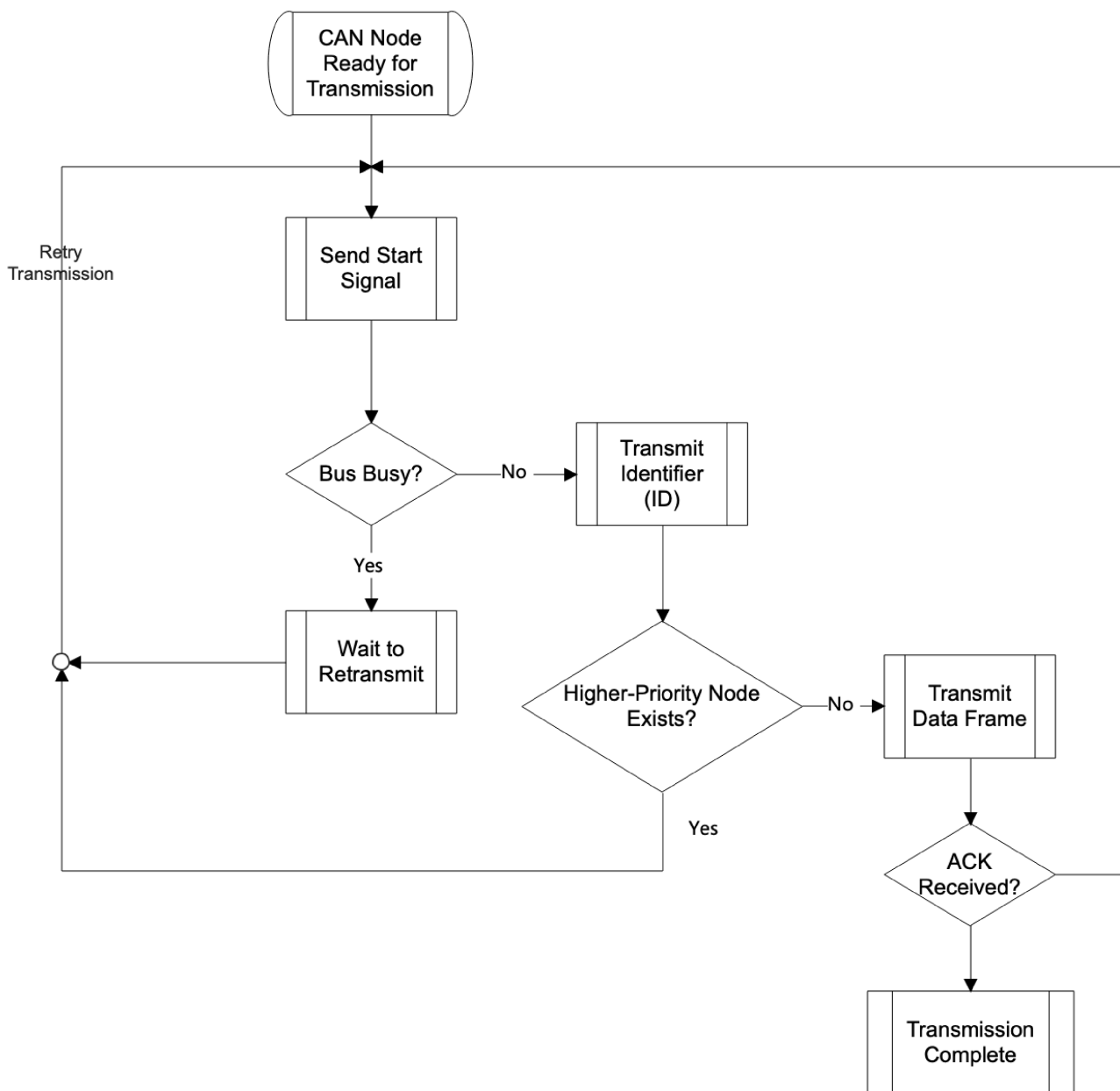


Figure 1. System-level flowchart of the CAN communication mechanism.

In our previous study, this communication procedure was formally represented using a Deterministic and Stochastic Petri Net (DSPN) to analyze CAN arbitration, transmission timing, and priority-dependent message behavior [22]. Petri-net-based approaches have also been applied to analyze and evaluate the real-time performance of CAN networks [24]. The simplified model shown in Figure 2 represents the principal states and transitions associated with message generation, bus availability, arbitration, and frame transmission. The complete DSPN model further incorporates multiple priority queues and inhibitor arcs to represent contention among CAN messages with different identifiers. In the present study, these figures are included only to provide a system-level view of the CAN communication mechanism; the primary focus is on the design and implementation of the gateway that transfers CAN data to an external Wi-Fi network.

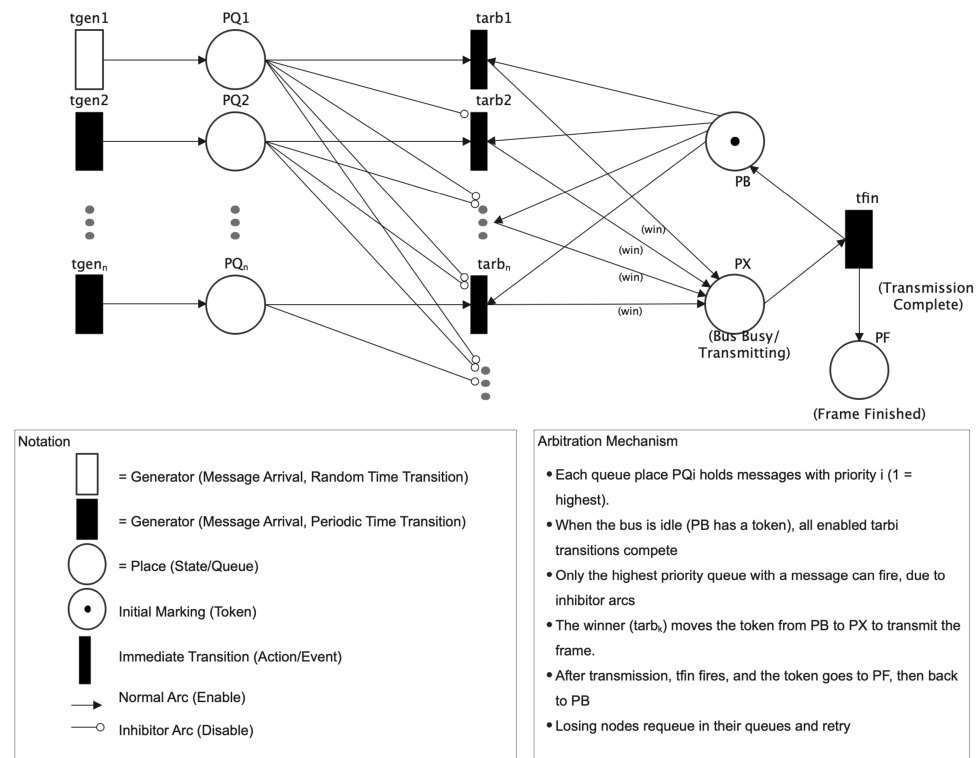


Figure 2. Simplified DSPN model of CAN arbitration and transmission, adapted from [22].

Although CAN provides robust, priority-aware communication between ECUs, a conventional CAN network does not offer a native interface for wireless networks, remote monitoring platforms, or cloud-based applications. Therefore, a gateway is required to acquire CAN frames, interpret or encapsulate their contents, and forward the information through an external communication interface. In the proposed system, CAN messages are received by the embedded gateway, processed based on their identifiers and payload fields, converted into a suitable wireless data format, and transmitted via Wi-Fi to a remote monitoring dashboard. Consequently, CAN serves as the in-vehicle communication interface for the proposed CAN–Wi-Fi gateway.

2.2. Automotive Gateway Technologies

Automotive gateways interconnect heterogeneous communication domains within a vehicle and between the vehicle and external networks. As vehicle electronic architectures incorporate numerous electronic control units (ECUs), sensors, and embedded controllers, gateways perform functions such as message routing, traffic isolation, protocol conversion, filtering, and data exchange between network segments. Depending on the vehicle architecture, they may connect CAN, CAN FD, LIN, FlexRay, and Automotive Ethernet networks or provide access to external diagnostic, wireless, and Internet-based services [25–27].

Automotive gateways can be broadly considered from two functional perspectives. An in-vehicle gateway connects internal communication domains, such as CAN-to-CAN, CAN-to-CAN FD, CAN-to-FlexRay, or CAN-to-Automotive Ethernet networks. Its principal responsibilities include message routing, traffic separation, data-format conversion, and coordination between networks with different bandwidth and timing characteristics. A vehicle-to-external gateway links the in-vehicle network to external equipment or communication infrastructure. Representative implementations include CAN-to-USB gateways for diagnostic and host-computer access, CAN-to-Ethernet gateways for IP-based communication, CAN-to-GPRS gateways for earlier cellular remote-monitoring applications, and CAN-to-Wi-Fi gateways for local or Internet-connected wireless access. More recent

cellular gateways may use LTE, 4G, or 5G communication instead of GPRS. These interfaces enable vehicle information to be accessed by diagnostic tools, remote-monitoring platforms, and cloud-connected applications [18,26,27].

A gateway generally performs more than direct electrical or frame-level conversion. For example, when a CAN message is transferred to an external IP-based network, the gateway receives the CAN frame, examines or filters its identifier, extracts or encapsulates the payload, maps the information to the selected communication format, and forwards it through the external interface. Its end-to-end performance can therefore be affected by message reception, filtering, buffering, scheduling, protocol processing, and transmission through the destination network. Previous CAN–Ethernet gateway studies have similarly evaluated processing delay, buffering, routing, and the effects of protocol conversion on gateway performance [18,26].

In connected-vehicle applications, gateway functionality has extended beyond message forwarding to include data aggregation, remote monitoring, vehicle diagnostics, and communication with external service platforms. These functions require the gateway to preserve the relevant timing and reliability properties of CAN traffic while accommodating the different communication characteristics of wireless and Internet-based networks. Consequently, message-dispatch design and resource management become important when CAN frames arrive concurrently or when the external link cannot immediately transmit all pending information [18,22,26,27].

Figure 3 presents a simplified protocol view of communication between a CAN network and an external network. The CAN side primarily provides physical- and data-link-layer communication, whereas the external side may incorporate Wi-Fi or Ethernet access together with network, transport, and application-layer services. The gateway receives CAN frames through the CAN interface and converts, encapsulates, or maps their information into a format that can be delivered through the external communication stack.

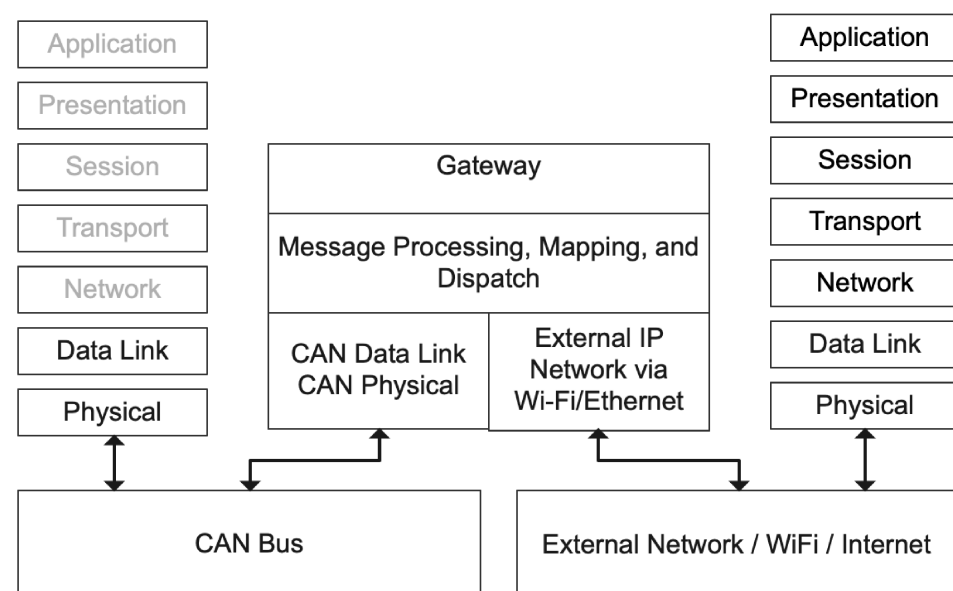


Figure 3. Protocol and service conversion between CAN and external communication networks.

Various gateway implementations have demonstrated communication between CAN and heterogeneous interfaces, including CAN-to-CAN, CAN-to-USB, CAN-to-Ethernet, CAN-to-GPRS, CAN-to-WiFi, and CAN/CAN FD-to-SOME/IP systems [18,26,27]. These studies establish the feasibility of protocol conversion and message forwarding and also show that gateway processing and routing mechanisms can affect end-to-end commu-

nication performance. Nevertheless, the behavior of an embedded CAN–Wi-Fi gateway depends not only on protocol conversion but also on its internal message buffering, dispatch policy, wireless transmission, and shared processing resources.

In this work, the proposed CAN–Wi-Fi gateway serves as a vehicle-to-external gateway, transferring in-vehicle CAN data to a wireless monitoring service. In addition to message acquisition and protocol conversion, the proposed system incorporates message filtering and priority-based dispatch to coordinate CAN reception, message processing, and Wi-Fi transmission.

2.3. Wireless Connectivity for Connected Vehicles

Connected-vehicle systems require communication among in-vehicle networks, mobile devices, roadside infrastructure, and remote data services. Vehicle information is therefore no longer limited to local diagnostic equipment but may be collected, transmitted, stored, and processed through wireless and Internet-based platforms. These communication capabilities support applications such as remote diagnostics, fleet management, predictive maintenance, traffic coordination, and emergency services [10,11,15,16].

Wireless connectivity provides an interface between conventional in-vehicle networks and external monitoring or service platforms. Common alternatives include Wi-Fi, cellular communication, and vehicle-to-everything (V2X) technologies. These technologies serve different operating ranges and application requirements. Cellular networks provide wide-area connectivity through public mobile infrastructure, while V2X communication supports direct or network-assisted interactions among vehicles, roadside infrastructure, pedestrians, and communication networks [10,11,15,28]. Wi-Fi, based on the IEEE 802.11 family of wireless local area network standards, provides a widely available interface for short-range communication between a vehicle gateway and nearby mobile devices, access points, or local monitoring systems [29]. IEEE identifies the 802.11 family as the basis of Wi-Fi wireless local-area networking, while ETSI specifications define cellular and enhanced V2X service scenarios [28].

For CAN-based vehicles, wireless connectivity is commonly provided through a gateway that receives CAN frames, filters or interprets their identifiers, converts or encapsulates their payloads, and forwards the resulting data through a wireless interface. A CAN–Wi-Fi gateway, therefore, provides a practical means of connecting a conventional CAN network to a mobile monitoring platform without modifying the existing ECUs or CAN communication protocol. Vehicle-status information can be accessed locally via a smart device or wireless terminal and subsequently relayed to a remote server when Internet connectivity is available [27]. A published in-vehicle edge-gateway implementation similarly demonstrated the transfer of in-vehicle data through Wi-Fi to an external server for vehicle monitoring and self-diagnosis [30].

However, integrating CAN and wireless communication introduces differences in timing and transmission behavior. CAN uses identifier-based priority arbitration over a controlled wired medium, whereas wireless performance may vary with channel occupancy, interference, signal quality, packet loss, retransmission, and network congestion. Consequently, the gateway must accommodate potential differences between the CAN message arrival rate and the available wireless forwarding capacity. Buffering, scheduling, message dispatch, packet formatting, and transmission-error handling can therefore affect gateway latency and reliability [18,26,27].

In this work, Wi-Fi is adopted as the local wireless communication interface between the in-vehicle CAN network and the mobile monitoring platform. This choice provides direct connectivity to commonly available mobile devices and wireless access points without requiring a dedicated cellular subscription or roadside V2X infrastructure. The proposed

gateway supports real-time visualization of vehicle status and provides an extensible interface for delivering monitored data to remote diagnostic or Internet-based services.

2.4. Research Gap and Positioning of This Work

Previous automotive gateway studies have demonstrated the feasibility of interconnecting CAN networks with heterogeneous communication interfaces. Representative implementations include CAN–FlexRay gateways [6], CAN–industrial-network and Internet gateways [7], CAN–Ethernet gateways [8,9,26], CAN–GPRS gateways [10], wireless relaying of CAN frames [11], CAN gateway modules [12], telematics gateways [13], CAN FD–SOME/IP gateways [18], and lightweight in-vehicle edge gateways [30]. These studies establish the technical feasibility of message forwarding, protocol conversion, data acquisition, and external network access. Their objectives and evaluation criteria, however, differ substantially. Some emphasize protocol conversion or communication bridging, whereas others focus on data reduction, security, diagnostic support, or heterogeneous in-vehicle networking.

A CAN–Wi-Fi gateway must coordinate communication domains with substantially different operating characteristics. CAN employs identifier-based priority arbitration and transmits frames over a controlled wired medium, whereas wireless forwarding can be affected by variable access delay, interference, retransmission, packet loss, and temporary congestion. The gateway must therefore perform more than direct protocol conversion; it must also manage the difference between CAN-frame arrivals and available wireless forwarding capacity through filtering, buffering, queue management, packet construction, and dispatch scheduling.

Existing implementations provide useful performance evidence, but their quantitative results generally characterize objectives different from those considered in this work. For example, the CAN FD–SOME/IP gateway in [18] evaluated protocol conversion and security overhead and reported gateway-processing latency below approximately 116 μ s, whereas the CAN–Ethernet gateway with data reduction in [26] reduced CAN bus load by up to 33.68% and achieved an average end-to-end processing time of 336 μ s. The lightweight in-vehicle edge gateway in [30] reported an average 10.83% reduction in message translation and transmission delay. These results demonstrate effective optimization of protocol conversion and heterogeneous in-vehicle communication, but they do not directly quantify the effect of a CAN-to-wireless dispatch policy on urgent monitoring traffic under transient forwarding contention.

Accordingly, the present study focuses on the internal forwarding behavior of an embedded CAN–Wi-Fi monitoring gateway. The proposed architecture jointly incorporates identifier-based filtering, priority classification, separate message queues, strict-priority dispatch, structured packet generation, mobile HMI visualization, and remote server communication. In addition to verifying the complete communication path, the dispatch mechanism is quantitatively compared with a single-FIFO baseline under controlled transient burst contention. This comparison evaluates whether priority-based forwarding provides measurable queueing-delay benefits for urgent monitoring traffic and explicitly identifies the corresponding delay trade-off for lower-priority classes.

Table 1 summarizes the positioning of representative gateway approaches and the proposed work. Because the compared studies employ different communication technologies, hardware platforms, traffic conditions, and delay definitions, their reported numerical values should not be interpreted as direct performance rankings. Instead, the table highlights the principal evaluation objective and the type of quantitative evidence each study provides.

Table 1. Comparison of representative automotive gateway studies and the proposed work.

Study	Gateway/Interface	Primary Focus	Dispatch/Prioritization	Representative Quantitative Evaluation	Local/Remote Monitoring
Xu et al. [10]	CAN-GPRS	Vehicle remote communication	Not reported	Gateway implementation and remote communication demonstrated; directly comparable dispatch-delay metrics not reported	Remote
Johanson et al. [11]	CAN over wireless IP	Remote CAN-frame relay for automotive testing	Not reported	Wireless CAN-frame relaying experimentally demonstrated; gateway dispatch scheduling not reported	Remote
Zuo et al. [18]	CAN/CAN FD-SOME/IP	Protocol conversion and security	Not a primary focus	Gateway-processing latency $\leq$ approximately 116 μ s; CPU and memory overhead also evaluated	Not primary focus
Oh et al. [26]	CAN-Ethernet	CAN data reduction and gateway processing	Not reported	CAN bus load reduced by up to 33.68%; average E2E processing time reduced to 336 μ s	Not primary focus
Jeong et al. [30]	CAN/FlexRay/MOST/Ethernet edge gateway	Lightweight protocol translation and self-diagnosis	Priority multi-queue/event queue	Average message-translation/transmission delay reduced by 10.83%	Partial
Proposed work	CAN-Wi-Fi	Selective vehicle monitoring and differentiated wireless forwarding	Strict High/Medium/Low priority; FIFO within class; single-FIFO baseline	High-priority burst mean queue wait: 31.116 $\rightarrow$ 2.268 ms; P95: 53.560 $\rightarrow$ 4.808 ms; 25 ms threshold exceedance: 61.7% $\rightarrow$ 0%	Integrated local HMI and remote server

Note: Reported numerical results are reproduced from the cited studies and reflect different architectures, traffic conditions, hardware platforms, and performance definitions; they are provided for methodological comparison rather than for direct numerical ranking.

3. System Architecture and Implementation

3.1. Overall System Architecture

Figure 4 illustrates the overall architecture of the proposed CAN–Wi-Fi gateway system. The system is divided into two major domains: the in-vehicle/local system and the remote monitoring system. Similar vehicle gateway architectures have been used to connect in-vehicle data sources with local monitoring devices and remote servers [5,30].

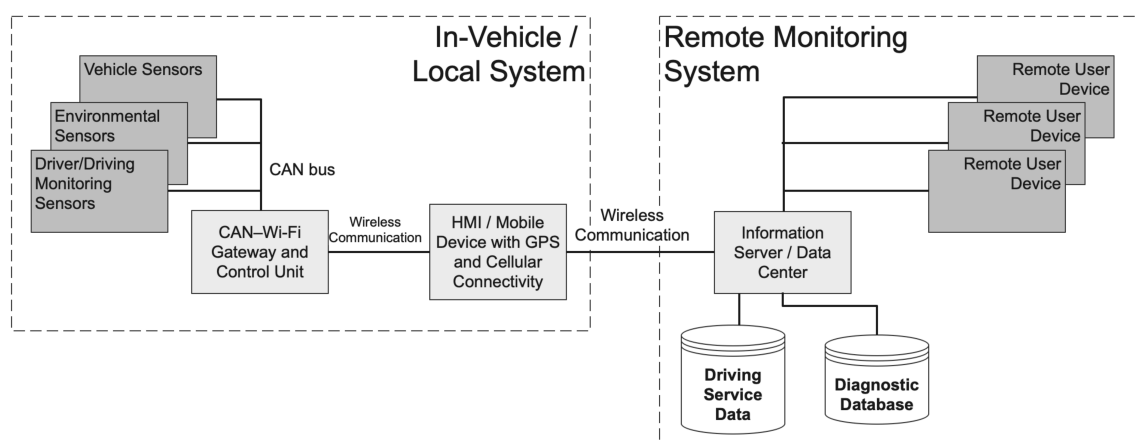


Figure 4. Proposed CAN–Wi-Fi gateway architecture connecting the in-vehicle local system with the remote monitoring system.

The in-vehicle/local system contains vehicle sensors, environmental sensors, and driver/driving monitoring sensors. The acquired information is transmitted through the CAN bus to the CAN–Wi-Fi gateway and control unit. The gateway receives CAN

frames, identifies the required messages, processes or formats the corresponding data, and wirelessly forwards the selected information to an HMI/mobile device.

The HMI/mobile device performs two functions. First, it serves as the local monitoring terminal by presenting vehicle, environmental, and driving-related information through a graphical user interface. Second, it acts as an intermediate communication node between the embedded gateway and the remote monitoring system. The mobile device incorporates GPS and cellular connectivity, enabling the received CAN data to be combined with positioning data and transmitted to the remote server via an external wireless network.

On the remote side, the information server or data center receives and manages the uploaded information. The data may be stored in a driving-service database and a diagnostic database for subsequent retrieval, analysis, and monitoring. Multiple remote-user devices can access the server to obtain vehicle status, location, driving service, or diagnostic information [15–17,30].

Accordingly, the proposed architecture supports both local and remote vehicle monitoring. Local users can view vehicle information directly on the HMI/mobile device, whereas remote users can access stored or transmitted information via the remote monitoring platform. The mobile device, therefore, provides the connection between the short-range Wi-Fi communication used inside or near the vehicle and the cellular or Internet-based communication used for remote access.

3.2. Hardware Configuration

Figure 5 shows the hardware and communication configuration of the implemented CAN-Wi-Fi gateway prototype. The architecture comprises three functional parts: the vehicle-side CAN interface, the local monitoring interface, and the remote monitoring interface.

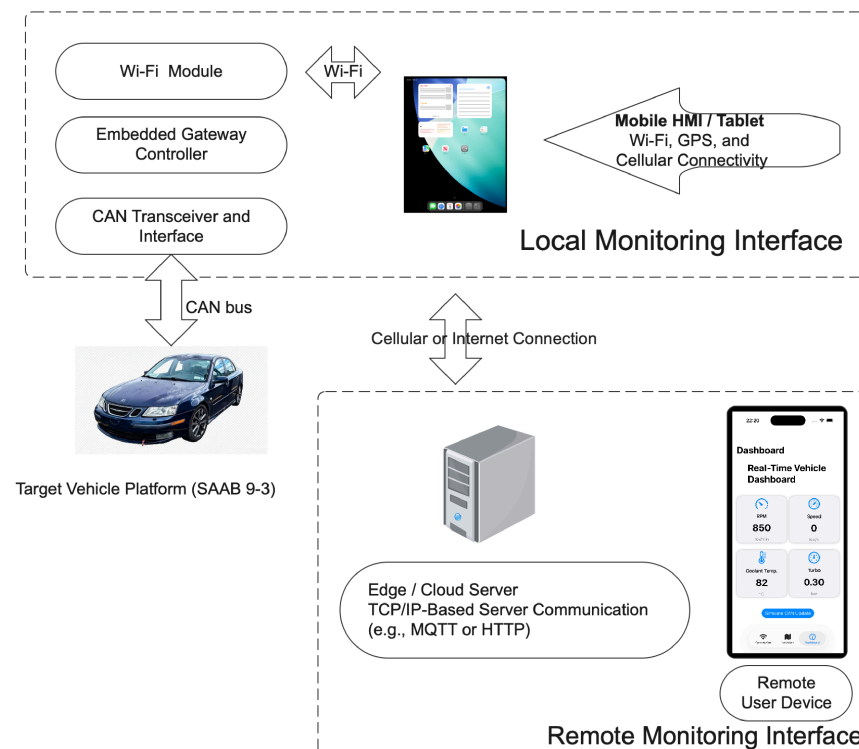


Figure 5. Hardware and communication architecture of the proposed CAN–Wi-Fi gateway prototype, including the target vehicle platform, local mobile interface, cellular connection, edge/cloud server, and remote monitoring device.

On the vehicle side, CAN messages are acquired from the vehicle network or a controlled CAN message source through a CAN transceiver and interface circuit. The embedded gateway controller receives the CAN frames, performs identifier-based message selection, processes the payload data, and converts the selected information into a format suitable for wireless transmission. A Wi-Fi module then transfers the processed data to the mobile HMI or tablet through a short-range wireless link.

The local monitoring interface is implemented on a mobile device or tablet. It displays the received vehicle information on a graphical dashboard and may combine CAN data with timestamps and GPS-based positioning. The mobile device also provides Wi-Fi or cellular connectivity, serving as the communication bridge between the embedded gateway and the remote monitoring interface.

For remote monitoring, selected vehicle and positioning data are forwarded by the mobile device through a cellular or Internet connection to an edge or cloud server. The server communication is implemented over TCP/IP using an application-layer protocol such as MQTT or HTTP. The server stores or processes the received data and provides a service endpoint through which remote user devices can retrieve vehicle-status, location, and diagnostic information [14,15].

In the prototype, an Arduino-compatible microcontroller platform is used as the embedded gateway controller. The CAN interface consists of a CAN controller and a CAN transceiver, while the wireless interface is implemented using a WiShield module. The WiShield communicates with the microcontroller via a Serial Peripheral Interface (SPI) and an interrupt signal, providing IEEE 802.11b wireless connectivity between the gateway and the mobile monitoring device.

The hardware architecture is modular and not restricted to the specific controller, CAN interface devices, or Wi-Fi module used in the prototype. Equivalent embedded controllers with integrated or external CAN peripherals and more recent Wi-Fi modules can be substituted without changing the principal data path or gateway operating concept. This modularity allows the proposed architecture to be adapted to different vehicle networks, communication requirements, and monitoring applications.

3.3. Software Framework

Figure 6 shows the software framework and end-to-end data flow of the proposed CAN–Wi-Fi gateway system. The software architecture is organized into three main components: the gateway firmware, the mobile HMI application, and the remote server software. Together, these components support CAN-message acquisition, local monitoring, wireless data forwarding, remote data storage, and remote user access.

The gateway firmware operates on the in-vehicle embedded controller. The CAN driver initializes the CAN interface, receives CAN frames from the in-vehicle network, and performs basic error handling. The received frames are stored in a CAN message buffer, which supports temporary storage, timestamping, and FIFO-based management. The message-filtering and decoding module then identifies relevant messages according to predefined CAN identifiers, extracts the required signals, and converts the raw payload data into application-level parameters. After signal decoding, the data-formatting module packs the selected information into a structured transmission format suitable for wireless communication. Finally, the Wi-Fi communication task manages data transmission and connection control between the embedded gateway and the mobile HMI.

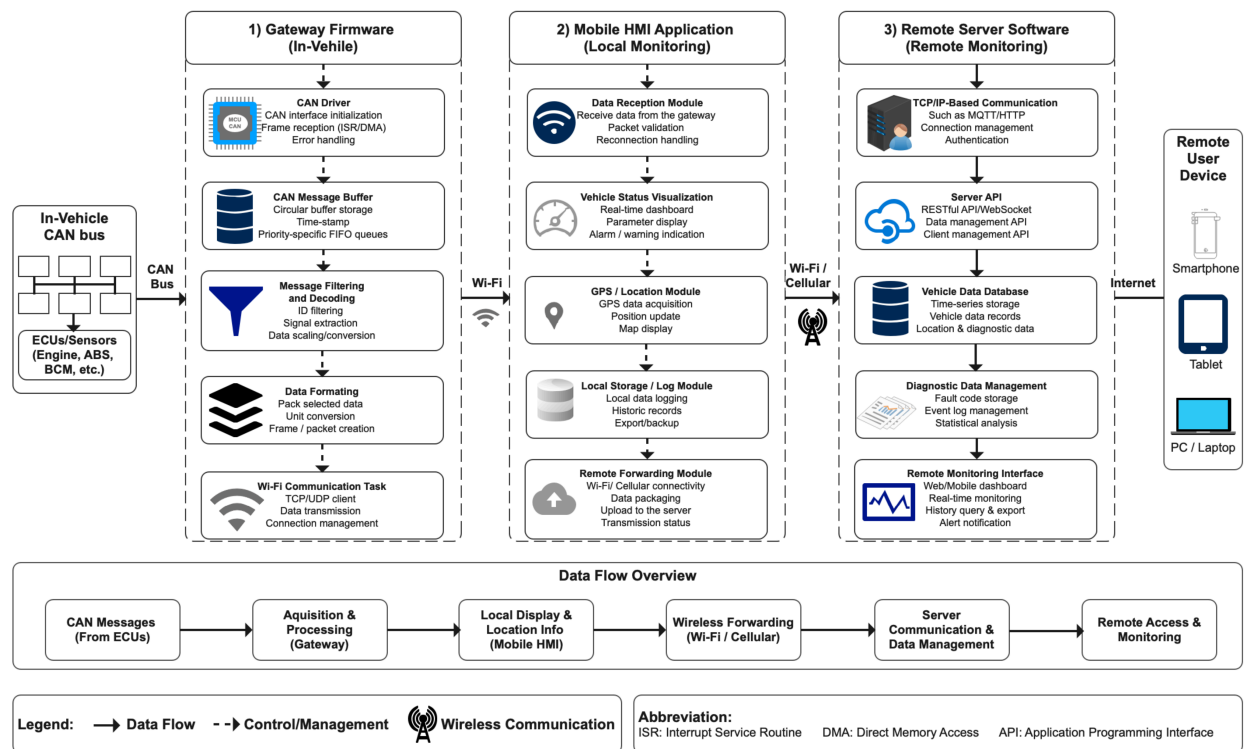


Figure 6. Software framework and data flow.

The mobile HMI application provides the local monitoring functions. Its data-reception module receives packets from the gateway, validates the incoming data, and handles reconnection when communication is interrupted. The vehicle-status visualization module updates the local dashboard, presenting real-time vehicle parameters and warning information. In addition, the GPS/location module acquires positioning data, updates location information, and supports map-based display. A local message-log module maintains the 20 most recently received CAN records for local review. When the number of stored records exceeds this limit, the oldest entries are removed to maintain a bounded recent-message history. The mobile application also includes a remote-forwarding module that relays selected vehicle and location data to the remote server via Wi-Fi or cellular connectivity.

The remote server software manages communication, data storage, and remote access. A TCP/IP-based communication service receives data from the mobile application via an application-layer protocol, such as MQTT or HTTP. The server API provides access interfaces for remote monitoring and data management. Received information is stored in the vehicle-data database, which maintains time-series vehicle records as well as location and diagnostic data. Additional diagnostic data management and remote monitoring functions support data review, historical query, and remote visualization for user devices such as smartphones, tablets, and personal computers [15–17,30].

The modular organization of the software framework allows individual functions to be updated without changing the entire system structure. For example, the wireless communication method, packet format, server interface, or visualization module may be revised independently. This separation of gateway processing, local monitoring, and remote server functions improves maintainability and supports adaptation of the proposed CAN-Wi-Fi gateway platform to different monitoring requirements.

3.4. Message Filtering and Priority-Based Message Dispatch

The proposed CAN-Wi-Fi gateway applies identifier-based filtering and priority-based message dispatch before forwarding selected CAN information to the wireless monitoring

interface. In an in-vehicle network, CAN traffic may contain periodic status information, event-triggered messages, diagnostic data, and other frames that are not required by a particular monitoring application. Indiscriminate forwarding of the complete CAN traffic stream would increase gateway processing and wireless transmission overhead. Therefore, the gateway first selects only CAN identifiers required for monitoring, diagnosis, or user visualization.

Figure 7 illustrates the proposed priority-based message-dispatch mechanism. When a CAN frame is received, the gateway first identifies the message according to its CAN identifier. Only the identifiers required for monitoring, diagnosis, or user display are selected for further processing. Messages that are not relevant to the current application are discarded at this stage to reduce processing and wireless transmission overhead.

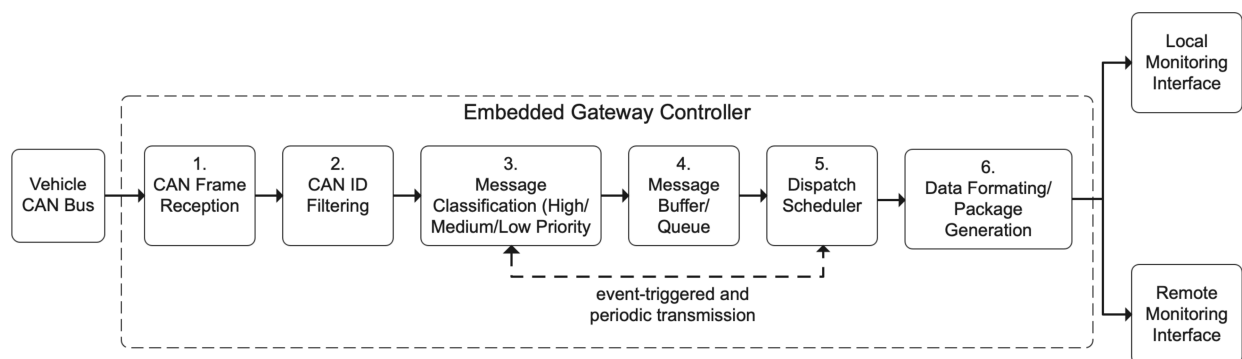


Figure 7. Priority-based message dispatch mechanism in the proposed CAN-Wi-Fi gateway.

After filtering, selected messages are assigned to one of three priority classes according to their functional importance and update requirements. High-priority messages represent warning signals, abnormal events, fault-related information, or other data requiring prompt delivery. Medium-priority messages represent periodically updated operating information used for continuous monitoring, such as vehicle speed and engine-related parameters. Low-priority messages represent noncritical status, logging, or auxiliary information for which longer gateway residence times are acceptable.

The implementation uses three independent queues corresponding to the High-, Medium-, and Low-priority classes, each with a capacity of 256 packets. Message ordering within each queue follows a first-in, first-out (FIFO) policy. The dispatch scheduler applies strict priority between the queues: when a transmission opportunity becomes available, the High-priority queue is examined first, followed by the Medium- and Low-priority queues. Consequently, a newly queued High-priority message may be transmitted before previously queued lower-priority messages, whereas messages belonging to the same priority class retain their arrival order.

When the capacity of a priority queue is exceeded, the oldest packet is discarded to retain more recent monitoring information. The evaluated implementation uses strict priority without an additional aging or weighted-fairness mechanism. Consequently, prolonged overload can increase the waiting time of Medium- and Low-priority traffic. The controlled contention experiment described in Section 4.6 therefore evaluates both the latency benefit for High-priority traffic and the corresponding delay trade-off for lower-priority traffic.

The different update periods used for the message classes reflect their monitoring functions rather than the CAN arbitration process itself. High-priority and rapidly changing signals are assigned shorter generation or forwarding periods, Medium-priority monitoring parameters are updated periodically at moderate rates, and Low-priority or slowly changing information is transmitted less frequently or when relevant state changes occur.

cur. The resulting dispatch architecture therefore combines selective CAN-ID filtering, application-oriented priority classification, FIFO ordering within each class, and strict inter-class scheduling before packet generation and wireless transmission.

By combining selective filtering with priority-based dispatch, the proposed mechanism avoids forwarding CAN identifiers not required by the monitoring application and provides differentiated treatment of selected monitoring information based on functional importance and update requirements. The modular dispatch table can also be extended with additional CAN identifiers or monitoring variables without requiring transmission of the complete CAN traffic stream. The resulting architecture supports local visualization, remote data forwarding, and diagnostic monitoring while allowing urgent and noncritical information to be handled according to different transmission requirements.

3.5. Communication Protocol and Data Packet Format

After CAN messages are filtered, classified, and selected by the dispatch scheduler, the corresponding data are converted into structured packets for wireless transmission. The communication protocol provides a consistent data-exchange mechanism among the embedded CAN–Wi-Fi gateway, the mobile HMI application, and the remote monitoring server. Rather than forwarding the complete CAN traffic stream, the gateway transmits only the CAN identifiers and vehicle parameters required by the monitoring application. This selective packetization reduces unnecessary wireless traffic and simplifies data parsing, visualization, and storage at the receiving side.

The packet contains communication metadata together with the selected CAN-related information. Table 2 presents the proposed packet fields. The packet header identifies the beginning of a valid message, while the device identifier identifies the gateway or monitored vehicle. A monotonically increasing sequence number uniquely identifies each transmitted packet and supports detection of missing, duplicate, or reordered packets during communication and experimental evaluation. The CAN identifier records the source message type, and the timestamp records the relevant reception or transmission time. The priority field indicates the dispatch category assigned by the gateway, and the data-length field specifies the length of the valid payload. The payload may contain either the original CAN data bytes or decoded vehicle parameters, depending on the monitoring application. A checksum or equivalent integrity field is included to detect packet corruption during transmission.

Table 2. Proposed data packet format for wireless transmission.

Figure	Description
Packet Header	Indicates the beginning of a valid packet and may identify the packet type or protocol version.
Device ID	Identifies the gateway, vehicle, or monitored data source.
Sequence Number	A monotonically increasing packet identifier used to detect missing, duplicate, or reordered packets during gateway communication and experimental evaluation.
CAN ID	Stores the identifier of the source CAN frame.
Timestamp	Records the CAN-frame reception time or packet-generation time.
Priority Level	Indicates the assigned High-, Medium-, or Low-priority class.
Data Length	Specifies the number of valid bytes or fields in the payload.
Payload Data	Contains decoded vehicle parameters or the original CAN payload.
GPS Information	Optionally contains latitude, longitude, and related positioning information added by the mobile device.
Checksum	Supports basic packet integrity verification.

For communication between the embedded gateway and the mobile HMI, packets are transmitted through the local Wi-Fi connection using the implemented socket-based protocol. The mobile application receives the packets, validates their structure and integrity, extracts the associated fields, and updates the graphical monitoring interface. The sequence number supports detection of missing, duplicate, or reordered packets, while the timestamp and priority information allow the application to distinguish urgent event-triggered messages from periodically transmitted vehicle-status data.

When remote monitoring is required, the mobile application may extend or repack the received gateway data before forwarding it to the server. For example, GPS coordinates obtained by the mobile device can be combined with the CAN-derived vehicle information. The resulting data are then transmitted through Wi-Fi or cellular connectivity using a TCP/IP-based application protocol, such as HTTP or MQTT, depending on the implemented server interface [14,15]. The remote server validates and stores the received records and provides access via an application programming interface or a remote monitoring interface.

The packet-based design supports scalability because the wireless connection carries only selected CAN data rather than the complete in-vehicle traffic stream. Additional CAN identifiers, decoded parameters, or optional metadata fields can be incorporated without changing the overall communication architecture. The priority and timestamp fields also support differentiated processing of urgent diagnostic events and periodic monitoring data. Consequently, the proposed packet format provides a common basis for local visualization, data logging, remote storage, and diagnostic processing.

3.6. System Operation

The operation of the proposed CAN–Wi-Fi gateway system follows a staged end-to-end data-flow process. Vehicle information generated by electronic control units and associated sensors is first transmitted through the in-vehicle CAN bus. The CAN transceiver and interface circuitry receive the CAN frames and deliver them to the embedded gateway controller. For each received frame, the gateway extracts the CAN identifier and payload information and determines whether the message is required for monitoring, diagnostic processing, or user display.

The dispatch mechanism described in Section 3.4 then filters unrelated CAN frames and classifies the selected messages according to their functional importance and update requirements. Warning signals, abnormal events, and fault-related information are assigned higher priority and are forwarded with minimal additional gateway delay. Normal vehicle status parameters, such as vehicle speed, engine speed, battery voltage, and temperature, are transmitted periodically at predefined update intervals. Other diagnostic, service, or logging information may be assigned medium or low priority according to the requirements of the monitoring application. Within each priority class, message order is maintained using FIFO-based queue management before the dispatch scheduler selects the next message for transmission.

The selected information is subsequently converted into the structured packet format described in Section 3.5. The core packet contains the device identifier, sequence number, source CAN identifier, timestamp, priority level, data length, and payload information required by the monitoring application. Depending on the implementation, an integrity field may also be included. Additional information generated outside the embedded gateway, such as GPS coordinates obtained by the mobile device, can be associated with the received CAN data before remote forwarding.

For local monitoring, the formatted packets are transmitted via Wi-Fi to the mobile HMI or tablet. The mobile application receives and validates the packets, extracts the

corresponding vehicle parameters, and updates the graphical dashboard. In this manner, the mobile device provides local visualization of the selected CAN information and may also combine the vehicle data with positioning and timestamp information.

When remote monitoring is required, the mobile device acts as the communication intermediary between the embedded gateway and the remote server. Selected vehicle and location information is forwarded through Wi-Fi or cellular connectivity to an edge or cloud server. The server receives and stores data and provides access via an application programming interface or a monitoring service, allowing remote user devices to retrieve current or historical vehicle information.

The staged operating procedure separates CAN acquisition, message filtering and dispatch, packet generation, local visualization, remote forwarding, and server-side data management into distinct functional modules. This modular organization allows individual components, such as the wireless protocol, packet format, visualization software, or remote server interface, to be modified without redesigning the complete system. The selective filtering and priority-based dispatch mechanism also reduces unnecessary wireless transmissions and supports the timely delivery of important monitoring and diagnostic information.

Based on the architecture and operating procedure presented in this section, Section 4 describes the experimental platform and evaluates the functional and communication performance of the implemented CAN–Wi-Fi gateway system.

4. Experimental Results and Performance Evaluation

4.1. Experimental Setup

To evaluate the proposed CAN–Wi-Fi gateway, two complementary experimental configurations were used. The first configuration implemented the vehicle-side gateway architecture using an Arduino-compatible embedded controller, CAN interface circuitry, and a Wi-Fi communication module. This configuration was used to verify the functional processing path, including CAN-message acquisition, identifier-based filtering, message classification, packet generation, wireless transmission, local HMI visualization, and remote data forwarding. The embedded prototype reflects the hardware architecture described in Section 3 and demonstrates the feasibility of integrating CAN and wireless monitoring functions within a modular gateway platform.

The second configuration was established to evaluate controlled, repeatable communication performance. In this configuration, an iPad-based gateway emulator reproduced the packet generation, filtering, priority classification, buffering, and dispatch functions of the gateway software, while a MacBook-based UDP server received the transmitted packets and returned acknowledgment (ACK) messages. This software-controlled platform was used to evaluate packet delivery, throughput, round-trip time (RTT), estimated one-way wireless delay, and priority-dependent queueing behavior under reproducible traffic conditions. Because the communication-performance measurements were obtained from this configuration, the reported quantitative delay and scheduling results should be interpreted as controlled implementation-level measurements rather than direct timing measurements of the Arduino-based embedded hardware.

For the local and remote monitoring verification, representative CAN-derived vehicle information was generated using the Saab P-bus message definitions and transmission characteristics described in Section 4.3. Selected messages were processed according to the filtering and dispatch mechanism described in Section 3.4 and converted into the packet format described in Section 3.5. The resulting packets were transmitted to the monitoring interfaces for decoding, visualization, and server-side processing.

The two configurations therefore served different but complementary purposes. The embedded prototype verified the architectural feasibility and functional organization of the proposed gateway, whereas the iPad/MacBook platform provided a controlled environment for quantitative communication and dispatch experiments. This separation avoids attributing the measured communication delays or queueing statistics to hardware components that were not used in those measurements.

The principal components of the embedded prototype and the controlled communication-performance platform are summarized in Tables 3 and 4, respectively.

Table 3. Embedded prototype configuration used for functional verification.

Component	Prototype Implementation	Function
CAN message source	Controlled CAN-message source based on Saab 9-3 P-bus definitions	Provides CAN frames for gateway evaluation
CAN interface	CAN controller and transceiver module	Receives CAN frames and provides the electrical interface to the CAN bus
Embedded controller	Arduino-based controller	Processes CAN frames and executes filtering and dispatch functions
Wireless module	Wi-Fi communication module	Transfers processed data to the mobile HMI
Local HMI	Mobile/tablet device	Displays selected vehicle information and supports local monitoring
Remote server	PC-based server application	Receives, stores, and manages forwarded monitoring data
Communication method	TCP/IP socket-based communication	Provides data exchange between monitoring clients and server

Table 4. Controlled communication-performance platform used for quantitative evaluation.

Component	Implementation	Function
Gateway emulator	iPad	Generates, classifies, buffers, and dispatches controlled CAN-like traffic
Wireless link	Local Wi-Fi network	Provides UDP packet transmission
Receiver/server	MacBook Pro	Receives packets, returns ACKs, and records performance data
Protocol	UDP with ACK	Supports RTT and delivery measurements
Timing source	iPad monotonic clock	Measures RTT without inter-device clock-synchronization error
Stage 1 scheduler	Priority or FIFO	Enables controlled baseline comparison

4.2. Performance Metrics

To evaluate the communication and dispatch performance of the proposed CAN–Wi-Fi gateway system, two groups of metrics were considered. Communication-level performance was characterized using transmission delay, packet delivery ratio (PDR), packet loss rate (PLR), and throughput. For the controlled priority-dispatch comparison, queue waiting time and its statistical distribution were additionally evaluated to quantify the effect of the scheduling policy under transient contention.

For Communication-level performance, five metrics were considered: end-to-end transmission delay, average transmission delay, packet delivery ratio, packet loss rate, and throughput. These metrics characterize the timeliness, reliability, and effective data-transfer capability of the gateway during local and remote monitoring experiments.

The end-to-end transmission delay of the i th packet is defined as

$$D_i = t_{r,i} - t_{s,i}, \quad (1)$$

where D_i is the end-to-end delay of the i th packet, $t_{s,i}$ is the packet transmission time at the sender, and $t_{r,i}$ is the reception time of the same packet at the corresponding monitoring endpoint. For local monitoring, the endpoint is the mobile HMI; for remote monitoring, it is the remote server or the user-side monitoring interface, depending on the measurement configuration.

The average transmission delay is calculated as

$$\bar{D} = \frac{1}{N} \sum_{i=1}^N D_i, \quad (2)$$

where N is the total number of successfully received packets. The average delay provides an overall measure of the monitoring system's responsiveness under the tested communication condition.

The packet delivery ratio (PDR) is used to evaluate the reliability of wireless transmission and is defined as

$$PDR = \frac{N_r}{N_t} \times 100\%, \quad (3)$$

where N_t is the number of transmitted packets and N_r is the number of successfully received packets. Based on the packet delivery ratio, the packet loss rate (PLR) can be calculated as

$$PLR = \left(1 - \frac{N_r}{N_t}\right) \times 100\%. \quad (4)$$

A lower packet loss rate indicates more reliable data delivery over the evaluated communication path.

Throughput represents the amount of successfully received data per unit time and is defined as

$$T = \frac{B_r}{\Delta t}, \quad (5)$$

where B_r is the total number of successfully received bits during the measurement interval and Δt is the corresponding measurement duration. Throughput, therefore, reflects the gateway's effective data-transfer capability under the selected communication conditions.

For the controlled dispatch experiment described in Section 4.6, the queue waiting time of a packet i is defined as

$$T_{q,i} = t_{d,i} - t_{a,i}, \quad (6)$$

where $t_{a,i}$ is the time at which the packet i is inserted into the corresponding dispatch queue and $t_{d,i}$ is the time at which the packet is selected for wireless transmission. Both

timestamps are obtained from the gateway emulator using the same local time source; therefore, the queue-wait measurement does not require clock synchronization between different devices.

In addition to the mean queue waiting time, the standard deviation, median, 95th-percentile (P95), and maximum waiting time were calculated across repeated runs.

The mean queue waiting time:

$$\bar{T}_q = \frac{1}{N} \sum_{i=1}^N T_{q,i}. \quad (7)$$

For the standard deviation:

$$\sigma_q = \sqrt{\frac{1}{N} \sum_{i=1}^N (T_{q,i} - \bar{T}_q)^2}. \quad (8)$$

The median queue waiting time, $\tilde{T}_q$, is the 50th percentile of the observed queue-wait samples. The 95th-percentile queue waiting time, $T_{q,95}$, represents the value below which 95% of the observed queue waiting times fall, while $T_{q,max}$ denotes the maximum observed queue waiting time. The median represents the typical queueing behavior, whereas P95 and maximum delay characterize the upper-tail response under transient contention.

$$T_{q,max} = \max_{1 \leq i \leq N} (T_{q,i}). \quad (9)$$

For High-priority traffic, a 25 ms queue-residence threshold was defined according to the configured High-priority update period. The threshold-exceedance ratio is calculated as

$$R_{25} = \frac{N_{q>25}}{N_H} \times 100\%, \quad (10)$$

where $N_{q>25}$ is the number of High-priority packets whose queue waiting time exceeds 25 ms and N_H is the total number of High-priority packets evaluated. This metric compares the ability of the priority-based and FIFO dispatch policies to serve urgent traffic within a single High-priority update interval.

$$D_{OW,est} = \frac{RTT}{2}. \quad (11)$$

The performance metrics used in this study are defined in Equations (1)–(11). For measurements based on the iPad/MacBook Pro platform, the round-trip time (RTT) was measured using the iPad monotonic clock from packet transmission to reception of the corresponding ACK. The estimated one-way wireless delay, $D_{OW,est}$, is calculated as shown in Equation (11). This approximation assumes approximately symmetric forward and acknowledgment paths and therefore does not represent a directly measured one-way delay. Possible path asymmetry and contention can cause the actual forward and reverse delays to differ; accordingly, the $RTT/2$ values are used only as communication-delay estimates.

The performance metrics used for gateway evaluation are summarized in Table 5.

Table 5. Performance metrics used for gateway evaluation.

Metric	Symbol	Purpose
End-to-end delay	D_i	Measures transmission delay from sender to monitoring endpoint

Table 5. Cont.

Metric	Symbol	Purpose
Average delay	$\bar{D}$	Evaluates overall system responsiveness
Packet delivery ratio	PDR	Evaluates successful packet reception rate
Packet loss rate	PLR	Evaluates unsuccessful packet delivery rate
Throughput	T	Measures effective received data rate
Queue waiting time	$T_{q,i}$	Measures packet residence time before dispatch
Average queue waiting time	$\bar{T}_q$	Measures average scheduler-induced waiting time
Queue-wait standard deviation	σ_q	Characterizes variability in queue waiting time
Median queue waiting time	$\tilde{T}_q$	Represents typical queueing behavior
95th-percentile queue waiting time (P95)	$T_{q,95}$	Characterizes upper-tail queueing delay
Maximum queue waiting time	$T_{q,max}$	Reports the largest observed queue waiting time
25 ms threshold-exceedance ratio	R_{25}	Measures the proportion of High-priority packets waiting longer than one configured update interval
Estimated one-way wireless delay	$D_{OW,est}$	Estimates one-way wireless communication delay from RTT

4.3. Communication and Functional Verification

The communication and functional behavior of the proposed CAN–Wi-Fi gateway was evaluated using controlled CAN message inputs based on publicly documented Saab 9-3 powertrain bus (P-bus) definitions. Publicly available reverse-engineering documentation reports that the P-bus operates at 500 kbit/s and that the documented messages use 8-byte data fields [31,32]. The same source provides representative CAN identifiers, signal definitions, and transmission intervals, which were used as reference information for constructing the controlled gateway inputs.

For the present evaluation, representative Saab P-bus identifiers, signal descriptions, and reported transmission intervals were used to construct controlled CAN message inputs. The generated traffic was not intended to reproduce the complete temporal behavior of an operating vehicle or a recorded P-bus trace. Instead, it provided realistic message identifiers, signal categories, and update characteristics for verifying the filtering, classification, packetization, wireless forwarding, and monitoring functions described in Section 3. Table 6 summarizes the representative P-bus messages and their use in the gateway verification.

During the experiment, the controlled CAN source generated the selected identifiers and corresponding payload data according to the representative message patterns summarized in Table 6. The gateway rejected identifiers not required by the monitoring application, classified the selected messages according to the policy described in Section 3.4, and placed them in the corresponding priority queues. The selected information was then converted into the structure packet format described in Section 3.5, including the sequence number and associated communication metadata, before being transmitted via the Wi-Fi interface to the monitoring application.

High-update-rate signals, such as engine-related information and vehicle speed, were used to evaluate continuous dashboard updates and the gateway’s ability to process frequently arriving frames. Lower-update-rate or event-related signals, including coolant temperature, gearbox status, headlight status, and climate-control information, were used to verify selective forwarding and differentiated dispatch behavior. The test, therefore, exercised a mixture of dynamic monitoring traffic and relatively infrequent status infor-

mation without requiring the entire P-bus traffic stream to be transmitted through the wireless interface.

Table 6. Representative Saab P-bus messages and reported transmission intervals used for controlled gateway communication verification [32].

CAN ID	Reported Signal Type	Reported Interval	Representative Update Characteristic	Use in Gateway Verification
0x1A0	Engine information including engine status, RPM-related data, MAP, throttle, and MAF	10ms	High frequency	High-frequency engine-status monitoring
0x2F0	Vehicle speed	20 ms	High frequency	Vehicle-speed display and driving-status monitoring
0x280	Pedal and reverse-gear	1 s or event-triggered	Periodic or event-related	Driving-state monitoring
0x370	Mileage/traveled distance	100 ms	Medium frequency	Distance and status logging
0x3A0	Vehicle speed-related information	55 ms	High frequency	Alternative speed-related monitoring
0x3B0	Headlight status	1 s or event-triggered	Low frequency or event-related	Body/status monitoring
0x3E0	Automatic gearbox status	1 s or event-triggered	Low frequency or event-related	Gear-state monitoring
0x530	Automatic climate control information	1 s	Low frequency	Comfort-system monitoring
0x5C0	Coolant temperature and ambient air pressure	1 s	Low frequency	Temperature and environment monitoring

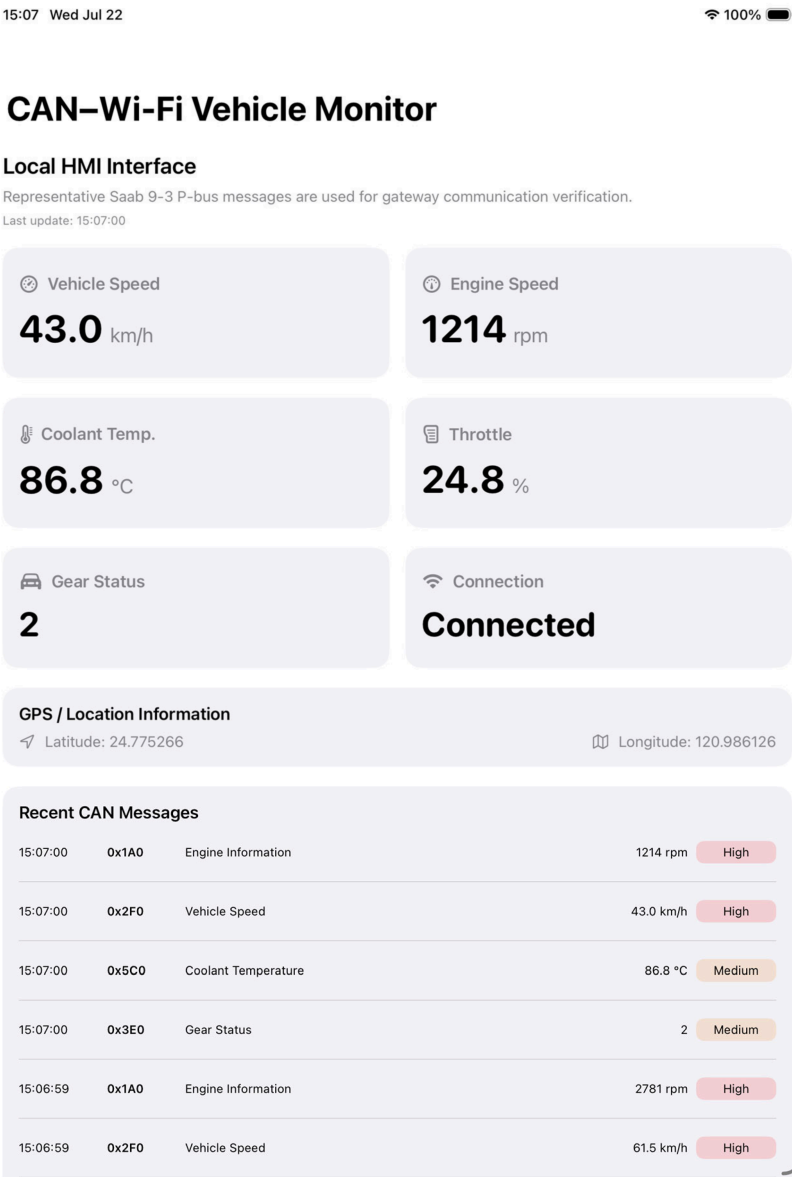
The communication test confirmed the complete processing path from CAN-frame reception to wireless packet delivery and dashboard update. Specifically, the gateway successfully performed CAN-ID filtering, priority classification, FIFO-based buffering within each priority class, packet generation, wireless transmission, and HMI data reconstruction. These results demonstrate the functional operation of the proposed message-dispatch and communication architecture using the controlled Saab P-bus-based message set.

Quantitative communication and dispatch performance was evaluated using the metrics defined in Section 4.2, with the corresponding results presented in Sections 4.5 and 4.6.

4.4. Local and Remote Monitoring Verification

To verify the local and remote monitoring functions of the proposed CAN–Wi-Fi gateway system, representative Saab P-bus-based vehicle data were generated in accordance with the message definitions summarized in Table 6. The generated packets followed the communication format described in Section 3.5 and included the sequence number, timestamp, CAN identifier, decoded signal value, and dispatch priority. The purpose of this experiment was to confirm that selected CAN-derived information could be formatted, transmitted, received, parsed, and presented correctly by both the local and remote monitoring software.

For local monitoring, a mobile HMI interface was implemented to display representative vehicle parameters, as shown in Figure 8. The interface displays vehicle speed, engine speed, coolant temperature, throttle position, gear status, GPS location information, and communication status. A recent-message table additionally displays the reception time, CAN identifier, signal description, decoded value, and priority level of selected messages. High-priority engine- and speed-related messages, and medium-priority temperature and gear information, are therefore directly distinguishable at the monitoring interface.



GPS / Location Information

Latitude: 24.775266

Longitude: 120.986126

Recent CAN Messages

15:07:00	0x1A0	Engine Information	1214 rpm	High
15:07:00	0x2F0	Vehicle Speed	43.0 km/h	High
15:07:00	0x5C0	Coolant Temperature	86.8 °C	Medium
15:07:00	0x3E0	Gear Status	2	Medium
15:06:59	0x1A0	Engine Information	2781 rpm	High
15:06:59	0x2F0	Vehicle Speed	61.5 km/h	High

Figure 8. Local HMI interface displaying representative data from a Saab P-bus-based CAN–Wi-Fi gateway.

The local HMI results verify that the received gateway packets can be decoded and converted into a user-readable representation suitable for in-vehicle or maintenance-side monitoring. The timestamped message records also directly indicate that multiple CAN identifiers can be processed and displayed according to the priority classification defined in Section 3.4.

For remote monitoring, a lightweight edge-server service was implemented to receive formatted vehicle data packets via an HTTP-based application programming interface. Representative packets containing engine information, vehicle speed, mileage, gearbox status, headlight status, climate-control status, and coolant temperature were transmitted to the server and displayed on a web-based dashboard.

Figure 9 shows the remote monitoring interface. The dashboard displays the server’s operating status, the gateway device identifier, the total number of received packets, the latest update time, and a timestamped table of recently received vehicle data packets. Each record includes the CAN identifier, signal type, decoded value, and assigned priority. This confirms that the remote service can receive, interpret, store, manage, and present the selected CAN-derived monitoring information.

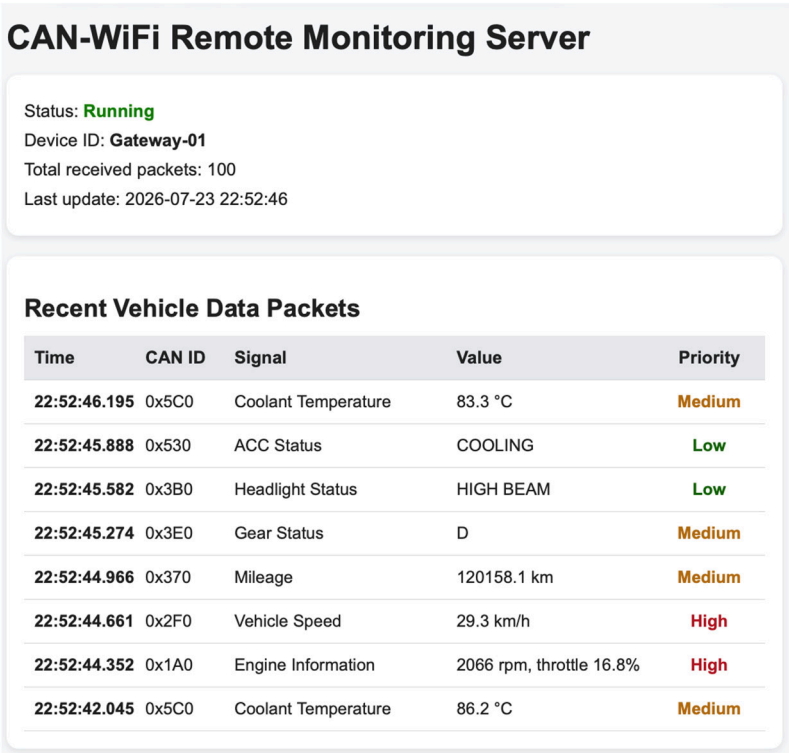


Figure 9. Remote monitoring dashboard displaying representative Saab P-bus-based vehicle data packets received by the edge-server service.

The combined local and remote verification demonstrates the complete monitoring path from representative CAN message generation to local visualization and remote server presentation. The experiment confirms packet reception, message parsing, display of decoded values, priority indication, timestamp handling, and HTTP-based remote forwarding. Because the current verification uses controlled P-bus-based packet generation rather than a complete real-vehicle P-bus log, these results are intended to validate system functionality and end-to-end data flow, rather than to provide final vehicle-level statistical performance measurements.

4.5. Controlled Communication Performance Evaluation

To quantitatively evaluate the wireless communication performance of the proposed CAN–Wi-Fi gateway, controlled experiments were conducted using the iPad/MacBook Pro platform described in Section 4.1. The experimental setup consisted of an iPad (9th generation), an iPhone 12 Pro, and a 16-inch MacBook Pro (Apple Inc., Cupertino, CA, USA). The experimental applications were developed using Xcode 26.5 (Apple Inc., Cupertino, CA, USA), while server-side data processing was performed using Python 3.9.6 (Python Software Foundation, Wilmington, DE, USA). The MacBook Pro ran macOS Tahoe 26.6.2, the iPad ran iPadOS 26.6.2, and the iPhone 12 Pro operated under iOS 26.6.2. Representative Saab P-bus-based message characteristics described in Section 4.3 were used to configure the generated traffic. The objective of this experiment was to evaluate packet-delivery reliability, application-level throughput, and RTT-based estimated one-way wireless delay under repeatable traffic conditions.

Three offered traffic conditions were considered: 20, 50, and 100 frames/s. These conditions represent light, representative, and increased communication loads, respectively. The increased-load condition was introduced as a controlled communication-stress case and does not reflect the nominal traffic characteristics of the Saab P-bus.

For each traffic condition, packets were transmitted from the iPad-based gateway emulator to the MacBook Pro server through the local Wi-Fi network using UDP. The server returned an acknowledgment (ACK) for each successfully received packet. The transmitted and acknowledged packet counts were used to determine PDR and PLR, while the successfully received data volume over the measurement interval was used to calculate application-level throughput. RTT was measured using the iPad monotonic clock, and the estimated one-way wireless delay was calculated according to Equation (11).

The experimental results are summarized in Table 7.

Table 7. Controlled communication-performance results under different traffic conditions.

Traffic Condition	Offered Traffic Rate	Received Rate	Throughput	PDR	Est. One-Way Delay
Light load	20 fps	20.035 fps	33.811 kbps	100.0%	4.121 ms
Representative load	50 fps	50.185 fps	84.888 kbps	100.0%	10.548 ms
Increased load	100 fps	100.315 fps	169.950 kbps	100.0%	10.773 ms

As shown in Table 7, the measured receiving rates closely followed the configured offered traffic rates, and all three conditions achieved 100% packet delivery in the evaluated local Wi-Fi environment. Application-level throughput increased from approximately 33.8 kbps at 20 frames/s to approximately 170.0 kbps at 100 frames/s. The RTT-based estimated one-way wireless delay ranged from approximately 4.1 to 10.8 ms. These results characterize the communication behavior of the controlled iPad/MacBook Pro implementation and should not be interpreted as direct timing measurements of the Arduino-based embedded prototype.

A second controlled communication test was conducted to examine wireless forwarding under heterogeneous message-generation rates. High-, Medium-, and Low-class traffic was generated with periods of 25, 50, and 1500 ms, corresponding to nominal rates of 40, 20, and approximately 0.667 frames/s, respectively. These periods were selected to represent different monitoring-update requirements rather than CAN arbitration priorities. The 50 ms period for the Medium-priority class corresponds to a 20 Hz update rate and was selected as a representative refresh interval for continuously varying dashboard information. This setting is intended as an experimental monitoring interval rather than a safety-critical timing requirement or a physiological limit of human visual perception.

All three classes achieved 100% packet delivery. The RTT-based estimated one-way wireless delays were approximately 3.4 ms for all classes, as summarized in Table 8. The similarity of these values indicates that, under this noncontentious communication condition, the different generation rates did not produce substantial class-dependent wireless delay.

Table 8. Controlled communication performance under heterogeneous message-generation rates.

Traffic Class	Generation Period	Nominal Rate	Received/ACK Frames	Est. One-Way Delay
High	25 ms	40 fps	8784	3.391 ms
Medium	50 ms	20 fps	4392	3.377 ms
Low	1500 ms	0.667 fps	147	3.478 ms

The results in Tables 7 and 8 demonstrate stable packet delivery and consistent application-level wireless communication performance under the evaluated controlled traffic conditions. However, the heterogeneous-rate experiment does not by itself demonstrate the latency benefit of the priority-based dispatch scheduler because the offered load does

not create sufficient queue contention. Therefore, a separate baseline comparison under transient overload is presented in Section 4.6 to directly compare priority-based dispatch with single-FIFO forwarding.

4.6. Baseline Comparison Under Burst Contention

To directly evaluate the effect of the proposed priority-based dispatch mechanism, a controlled baseline comparison was conducted using the iPad/MacBook Pro platform described in Section 4.1. Two dispatch policies were compared under identical traffic conditions: (1) the proposed strict-priority scheduler with separate High-, Medium-, and Low-priority queues and FIFO ordering within each queue; and (2) a single-FIFO baseline in which all packets were forwarded in arrival order without priority differentiation. Identifier filtering remained enabled in both cases so that the comparison isolated the effect of the dispatch policy.

In the priority-based implementation, each priority queue had a capacity of 256 packets. When a queue reached capacity, the oldest packet was discarded to retain more recent monitoring information. The strict-priority scheduler always served High-priority traffic before Medium-priority traffic and Medium-priority traffic before Low-priority traffic. No aging or weighted-fairness mechanism was applied. For the single-FIFO baseline, a queue capacity of 768 packets was used, corresponding to the aggregate capacity of the three priority queues, thereby avoiding confounding the comparison with different buffer sizes.

The background traffic consisted of High-, Medium-, and Low-class packets generated with periods of 25, 50, and 1500 ms, corresponding to nominal rates of 40, 20, and approximately 0.667 frames/s, respectively. To create transient contention, an additional High-priority warning burst of 160 frames/s was generated for 0.5 s every 5 s. During each burst, the aggregate offered rate increased to approximately 220.7 frames/s. The burst scenario represents temporally clustered warning messages arriving at the gateway dispatch layer and is not intended to reproduce physical CAN bus arbitration.

The dispatcher was configured with a 5 ms service interval, corresponding to a maximum dispatch rate of 200 packets/s. Because the burst offered rate exceeded this service capacity, packets accumulated temporarily in the dispatch queue, thereby creating a controlled contention condition suitable for evaluating scheduling behavior.

Each policy was evaluated in ten 30 s runs under the same traffic configuration. The primary metric was queue waiting time, measured from packet insertion into the gateway queue to selection for wireless transmission using the local monotonic clock. High-priority queue-wait threshold exceedance was evaluated using the 25 ms criterion defined in Section 4.2.

Under transient contention, the strict-priority scheduler substantially reduced the queue waiting time of High-priority warning traffic compared with single-FIFO forwarding. The mean High-priority burst queue wait decreased from approximately 31.1 ms to 2.27 ms, while the P95 delay decreased from approximately 53.6 ms to 4.81 ms. The maximum observed queue wait decreased from approximately 64.6 ms to 7.41 ms. Most importantly, the 25 ms threshold-exceedance ratio was reduced from approximately 61.7% under FIFO to 0% under priority-based dispatch.

The latency improvement for High-priority traffic was achieved at the expense of longer waiting times for lower-priority traffic. Under the same contention condition, the mean queue waiting time of Medium-priority packets increased from approximately 9.91 ms with FIFO to 39.63 ms with strict-priority dispatch, while the corresponding Low-priority mean increased from approximately 17.19 ms to 74.97 ms. This behavior is expected for a strict-priority scheduler without aging or weighted fairness, and it demonstrates that the mechanism provides service differentiation rather than a uniform reduction in delay.

across all traffic classes. Under the single-FIFO policy, traffic-class labels do not affect dispatch order; therefore, the observed differences in class-specific mean queue waiting time arise from their different generation patterns and temporal alignment with the warning bursts rather than from class-dependent scheduling.

These results show that, under transient overload, the proposed priority-based dispatch mechanism can protect urgent monitoring traffic from queueing delay more effectively than single-FIFO forwarding. The results do not indicate superior overall latency for every traffic class; instead, they demonstrate an explicit scheduling trade-off in which High-priority responsiveness improves at the cost of increased delay for lower-priority traffic.

To further illustrate the effect of the dispatch policy under transient burst contention, Figures 10 and 11 provide statistical and time-domain comparisons of High-priority queue waiting time. Figure 10 summarizes the mean and 95th-percentile (P95) values across ten repeated runs, while Figure 11 shows the temporal evolution of High-priority queue waiting time during a representative run, with the warning-burst intervals indicated.

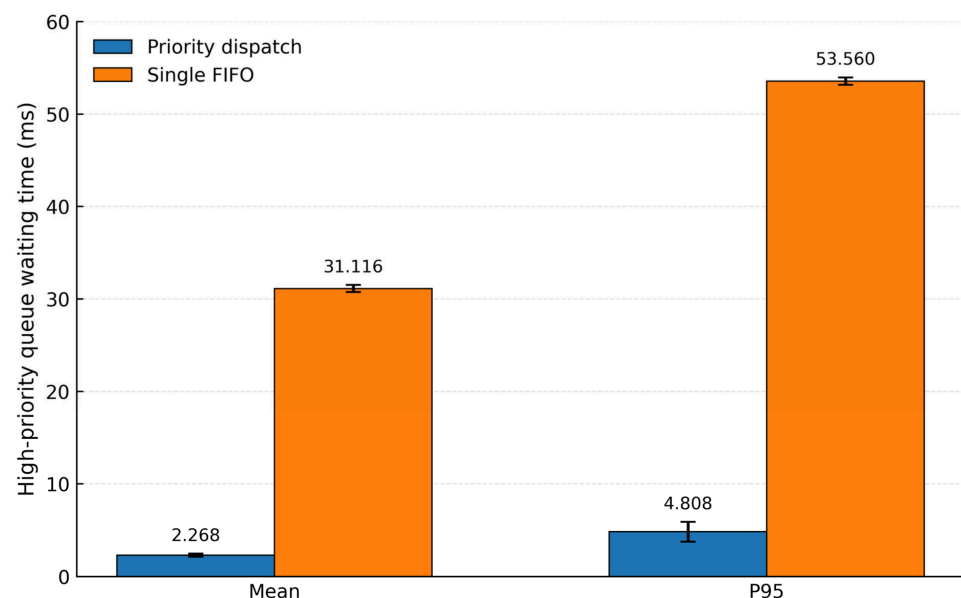


Figure 10. Comparison of mean and P95 High-priority queue waiting times under transient burst contention for priority-based dispatch and single-FIFO forwarding. Error bars represent the standard deviation across ten repeated runs.

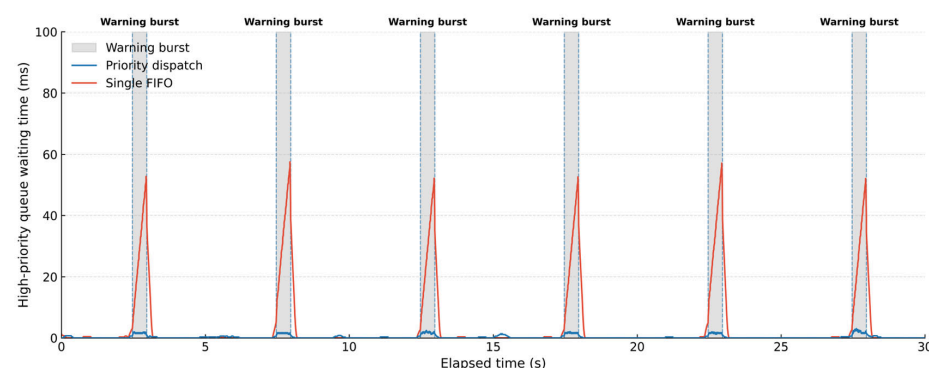


Figure 11. High-priority queue waiting time versus elapsed time during representative matched runs of priority-based dispatch and single-FIFO forwarding under transient burst contention. Shaded regions indicate warning-burst intervals identified from the experimental packet logs. The displayed traces use an 11-sample rolling mean for visualization.

As shown in Figure 10, priority-based dispatch substantially reduced both the mean and P95 High-priority queue waiting times compared with single-FIFO forwarding. The mean value decreased from 31.116 ms to 2.268 ms, while the P95 value decreased from 53.560 ms to 4.808 ms. These results confirm that the proposed scheduler reduces both typical and upper-tail queueing delay for urgent traffic under transient burst contention.

Figure 11 illustrates the time-domain response of the two dispatch policies during a representative matched run. Outside the warning-burst intervals, the High-priority queue waiting time remained relatively low under both policies. During each burst, however, the queue waiting time under single-FIFO forwarding increased sharply as newly generated High-priority packets waited behind previously queued traffic. In contrast, the priority-based scheduler maintained substantially lower waiting times in the High-priority queue by serving requests from that queue first. This temporal behavior is consistent with the statistical results in Table 9 and Figure 10, including the substantially lower mean and P95 queue waiting times and the elimination of 25 ms threshold exceedances under priority-based dispatch.

Table 9. High-priority queueing performance under transient burst contention.

Metric	Priority Dispatch	Single FIFO
Mean queue waiting time	2.268 ± 0.165 ms	31.116 ± 0.382 ms
Median queue waiting time	2.576 ± 0.036 ms	31.717 ± 0.745 ms
P95 queue waiting time	4.808 ± 1.069 ms	53.560 ± 0.397 ms
Maximum queue waiting time	7.413 ± 0.850 ms	64.570 ± 0.310 ms
25 ms threshold-exceedance ratio	0.000%	61.720 ± 0.763%

Note: Values are reported as the mean ± standard deviation across ten repeated runs and correspond to High-priority warning-burst packets.

Taken together, Tables 9 and 10 and Figures 10 and 11 demonstrate that strict-priority dispatch provides effective protection for urgent monitoring traffic during transient contention, while introducing the expected increase in queueing delay for lower-priority classes.

Table 10. Mean queue waiting time by traffic class under transient burst contention.

Traffic Class	Priority Dispatch	Single FIFO	Scheduling Effect
High	0.786 ms	12.116 ms	Reduced under priority dispatch
Medium	39.626 ms	9.907 ms	Increased under priority dispatch
Low	74.972 ms	17.192 ms	Increased under priority dispatch

Note: Values represent the mean queue waiting time averaged across ten repeated runs for each dispatch policy.

4.7. Discussion

The experimental verification demonstrates the functional feasibility of the proposed CAN–Wi-Fi gateway architecture for local and remote vehicle monitoring. The implemented data path integrates CAN-message acquisition, identifier-based filtering, priority-based message dispatch, packet formatting, Wi-Fi transmission, local HMI visualization, and HTTP-based remote monitoring. Using representative Saab P-bus message definitions and transmission characteristics, the experiments confirmed that selected CAN-derived vehicle information can be generated, processed, transmitted, parsed, and displayed through both local and remote monitoring interfaces.

An important characteristic of the proposed architecture is its modular organization. CAN reception, message buffering and dispatch, packet generation, wireless communication, local visualization, and remote data management are implemented as separate functional components. This separation allows individual hardware or software elements to be replaced or upgraded without fundamentally changing the overall communication architecture. For example, the embedded controller, CAN interface devices, Wi-Fi module, packet format, or server communication protocol can be replaced with functionally equivalent components according to the requirements of a particular monitoring application.

The identifier-based filtering mechanism reduces unnecessary wireless forwarding by selecting only CAN messages required by the monitoring application. The priority-based dispatcher provides an additional level of service differentiation when gateway transmission capacity becomes constrained. Under the transient burst-contention experiment in Section 4.6, the strict-priority scheduler substantially reduced the queue waiting time of High-priority warning traffic compared with single-FIFO forwarding. The mean High-priority burst queue waiting time decreased from 31.116 ms to 2.268 ms, while the P95 value decreased from 53.560 ms to 4.808 ms. The 25 ms threshold-exceedance ratio was reduced from approximately 61.7% under FIFO to 0% with priority-based dispatch. These results provide direct evidence that the proposed dispatch policy can substantially reduce queueing delay for urgent monitoring traffic during short-term overload rather than merely assigning different nominal update rates to different message classes.

The improvement in High-priority responsiveness is accompanied by an explicit scheduling trade-off. Under strict-priority dispatch, the mean queue waiting times of Medium- and Low-priority traffic increased relative to the single-FIFO baseline. This behavior is expected because the evaluated scheduler always serves higher-priority queues first and does not employ aging, weighted fairness, or other starvation-prevention mechanisms. Accordingly, the proposed scheduler should not be interpreted as minimizing delay uniformly across all traffic classes. Instead, it deliberately redistributes service toward urgent messages when contention occurs. For monitoring applications in which prolonged high-priority loading is possible, weighted scheduling, aging, or bounded-service mechanisms could be considered to limit excessive waiting by lower-priority traffic.

The burst-contention experiment represents simultaneous or temporally clustered warning traffic at the gateway dispatch layer rather than physical CAN bus arbitration. CAN arbitration occurs before the gateway receives a successfully transmitted frame and follows the nondestructive identifier-based mechanism described in Section 2.1. The experiment in Section 4.6 therefore evaluates competition for gateway forwarding capacity after CAN-frame acquisition. This distinction is important because the measured queue waiting times characterize the gateway scheduler and the wireless forwarding path, not the CAN bus-access delay.

The local and remote monitoring experiments further demonstrate the extensibility of the proposed data path. The mobile HMI converts the received gateway packets into user-readable vehicle information and presents current operating parameters, together with timestamp, location information, connection status, and recent CAN message records. On the remote side, the edge-server service receives formatted vehicle data packets through an HTTP-based interface and provides browser-based monitoring of the selected vehicle data. The structured packet format, including the sequence number, CAN identifier, timestamp, priority information, and payload data, provides a common representation for communication between the gateway and the monitoring software.

The present evaluation should nevertheless be interpreted within its experimental scope. The representative CAN traffic was constructed from publicly documented Saab P-bus identifiers, decoded signal information, and reported transmission intervals rather

than from a complete recorded vehicle P-bus trace. The quantitative communication measurements were obtained using the controlled iPad/MacBook Pro platform rather than directly from the Arduino-based embedded prototype. In addition, the RTT-based estimated one-way wireless delay values assume approximately symmetric forward and acknowledgment paths and should therefore be interpreted as estimates rather than as direct one-way measurements. The burst-contention experiments provide controlled implementation-level evidence of scheduler behavior, but they do not reproduce all timing variations, message interactions, CAN error conditions, wireless interference, or traffic patterns that may occur in an operating vehicle.

Overall, the results indicate that the proposed gateway architecture can integrate selective CAN-data forwarding, differentiated dispatch, local visualization, and remote monitoring within a modular communication framework. The experimental evidence further shows that priority-based dispatch provides a measurable latency advantage for urgent monitoring traffic during transient contention, while introducing increased queueing delay for lower-priority classes. These findings define both the benefit and the practical trade-off of the proposed scheduling mechanism.

5. Conclusions

This paper presented the design and performance evaluation of an embedded CAN–Wi-Fi gateway architecture for connected-vehicle monitoring applications. The proposed system integrates CAN message acquisition, identifier-based filtering, priority-based dispatch, packet formatting, wireless transmission, local HMI visualization, and edge-server-based remote monitoring within a modular communication framework. Representative Saab P-bus message definitions and transmission characteristics were used to construct controlled CAN-derived traffic for functional and quantitative evaluation.

The experimental results confirmed reliable wireless delivery and successful end-to-end monitoring through both local and remote interfaces. Under controlled communication tests, packet delivery remained at 100% across the evaluated traffic conditions, while the RTT-based estimated one-way wireless delay ranged from approximately 4.1 to 10.8 ms. Under heterogeneous message-generation rates, the High-, Medium-, and Low-priority traffic classes also achieved 100% delivery, with estimated one-way wireless delays of approximately 3.4 ms.

The baseline comparison under transient burst contention further demonstrated the practical benefit of the proposed dispatch mechanism. For High-priority warning traffic, strict-priority dispatch reduced the mean queue waiting time from 31.116 ms with single-FIFO forwarding to 2.268 ms, while the P95 value decreased from 53.560 ms to 4.808 ms. The 25 ms threshold-exceedance ratio was reduced from approximately 61.7% to 0%. These results show that priority-based dispatch can substantially improve the responsiveness of urgent monitoring traffic when gateway forwarding capacity becomes temporarily constrained.

The improvement is accompanied by the expected strict-priority scheduling trade-off. Medium- and Low-priority traffic experienced increased queue waiting times because the evaluated scheduler does not employ aging or weighted fairness. The proposed mechanism should therefore be interpreted as providing differentiated service rather than reducing latency uniformly across all message classes.

The present evaluation was conducted using controlled Saab P-bus-based traffic and an iPad/MacBook Pro communication-performance platform, while the embedded Arduino-based prototype was used primarily for functional verification. Accordingly, the reported measurements characterize the implemented gateway communication and dispatch mechanisms under controlled conditions rather than complete vehicle-level timing

behavior. Nevertheless, the results demonstrate that the proposed architecture provides a practical and modular basis for selective CAN-data forwarding, responsive handling of urgent monitoring traffic, and integrated local and remote vehicle monitoring.

Author Contributions: Conceptualization, C.-H.L. and Y.-C.C.; methodology, C.-H.L. and Y.-C.C.; software, Y.-C.C.; validation, Y.-C.C.; investigation, C.-H.L. and Y.-C.C.; writing—original draft preparation, Y.-C.C.; writing—review and editing, C.-H.L.; visualization, C.-H.L. and Y.-C.C.; supervision, C.-H.L.; project administration, C.-H.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Lawrenz, W. *CAN System Engineering: From Theory to Practical Applications*; Springer: New York, NY, USA, 1997.
2. ISO 11898-1:2015; Road Vehicles—Controller Area Network (CAN)—Part 1: Data Link Layer and Physical Signaling. International Organization for Standardization: Geneva, Switzerland, 2015.
3. ISO 11898-1:2024; Road Vehicles—Controller Area Network (CAN)—Part 1: Data Link Layer and Physical Coding Sublayer. International Organization for Standardization: Geneva, Switzerland, 2024.
4. ISO 11898-2:2024; Road Vehicles—Controller Area Network (CAN)—Part 2: High-Speed Physical Medium Attachment (PMA) Sublayer. International Organization for Standardization: Geneva, Switzerland, 2024.
5. Jenkins, W.; Lewis, R.; Lazarou, G.; Picone, J.; Rowland, Z. Real-Time Vehicle Performance Monitoring Using Wireless Networking. In Proceedings of the 3rd International Conference on Communications, Internet, and Information Technology, St. Thomas, VI, USA, 22–24 November 2004.
6. Zhao, R.; Qin, G.H.; Liu, J.Q. Gateway System for CAN and FlexRay in Automotive ECU Networks. In Proceedings of the 2010 International Conference on Information Networking and Automation (ICINA), Kunming, China, 16–18 October 2010. [CrossRef]
7. Vazquez, L.; Rojas, L.; Uribe, E.; Fuertes, J.; Sudria, A. Intelligent CAN-ISA Gateway Development to Industrial Networks and Internet Coupling. In Proceedings of the 2000 IEEE International Symposium on Industrial Electronics (ISIE 2000), Cholula, Mexico, 4–8 December 2000. [CrossRef]
8. Wang, Y.; Wang, H. Design and Implementation of a Data Acquisition System for Vehicle Network Based on Gateway of Ethernet_CAN. In Proceedings of the 2012 31st Chinese Control Conference (CCC), Hefei, China, 25–27 July 2012.
9. Hu, N.; Li, L. The Research and Design of the CAN Bus to Ethernet Gateway Based on SSL. In Proceedings of the 2009 Second International Workshop on Computer Science and Engineering (WCSE '09), Qingdao, China, 28–30 October 2009. [CrossRef]
10. Xu, Y.; Yang, Q.L.; You, Z.H. Design of CAN/GPRS Gateway for Vehicle Remote Communications. In Proceedings of the 2009 International Conference on Measuring Technology and Mechatronics Automation, Zhangjiajie, China, 11–12 April 2009. [CrossRef]
11. Johanson, M.; Karlsson, L.; Risch, T. Relaying Controller Area Network Frames over Wireless Internetworks for Automotive Testing Applications. In Proceedings of the 2009 Fourth International Conference on Systems and Networks Communications (ICSNC '09), Porto, Portugal, 20–25 September 2009. [CrossRef]
12. Taube, J.; Hartwich, F.; Beikirch, H. Comparison of CAN Gateway Modules for Automotive and Industrial Control Applications. In Proceedings of the 10th International CAN Conference (iCC 2005), CAN in Automation, Rome, Italy, 8–10 March 2005. Available online: https://stage.can-cia.org/fileadmin/cia/documents/proceedings/2005_taub.pdf (accessed on 2 August 2026).
13. Chang, P.L.; Guo, Y.X.; Jeng, M.D. Telematics Gateway and Power Saving Method for Electric Vehicles. In Proceedings of the 2011 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Anchorage, AK, USA, 9–12 October 2011. [CrossRef]
14. OASIS. MQTT Version 5.0; OASIS Open: Burlington, MA, USA, 2019. Available online: <https://docs.oasis-open.org/mqtt/mqtt/v5.0/os/mqtt-v5.0-os.html> (accessed on 2 August 2026).
15. Sukmandhani, A.A.; Zarlis, M. Monitoring Applications for Vehicle-Based on the Internet of Things (IoT) Using the MQTT Protocol. *Procedia Comput. Sci.* **2023**, *227*, 73–82. [CrossRef]
16. Karrouchi, M.; Nasri, I.; Snoussi, H.; Atmane, I.; Messaoudi, A.; Kassmi, K. Intelligent Driver Monitoring System: An Internet of Things-Based System for Tracking and Identifying the Driving Behavior. *Comput. Stand. Interfaces* **2023**, *84*, 103704. [CrossRef]

17. Kristiani, E.; Yu, T.-H.; Yang, C.-T. On Construction of Real-Time Monitoring System for Sport Cruiser Motorcycles Using NB-IoT and Multi-Sensors. *Sensors* **2024**, *24*, 7484. [[CrossRef](#)] [[PubMed](#)]
18. Zuo, Z.; Yang, S.; Ma, B.; Zou, B.; Cao, Y.; Li, Q.; Zhou, S.; Li, J. Design of a CANFD to SOME/IP Gateway Considering Security for In-Vehicle Networks. *Sensors* **2021**, *21*, 7917. [[CrossRef](#)] [[PubMed](#)]
19. Min, J.; Park, Y. Application of 10BASE-T1S Ethernet in Delay-Sensitive Vehicular Networks. In Proceedings of the 2023 IEEE Vehicular Networking Conference (VNC), Istanbul, Türkiye, 20–22 November 2023. [[CrossRef](#)]
20. Min, J.; Park, Y. Fault Recovery of 10BASE-T1S Automotive Ethernet with Bus/Ring Hybrid Topology. *J. Commun. Netw.* **2025**, *27*, 1–9. [[CrossRef](#)]
21. Walrand, J.; Turner, M.; Myers, R. An Architecture for In-Vehicle Networks. *IEEE Trans. Veh. Technol.* **2021**, *70*, 6335–6342. [[CrossRef](#)]
22. Chung, Y.-C.; Lee, C.-H. A Cross-Validated DSPN and Worst-Case Response-Time Framework for Timing Analysis of Automotive CAN Networks. *Electronics* **2026**, *15*, 2486. [[CrossRef](#)]
23. Robert Bosch GmbH. *CAN Specification, Version 2.0*; Robert Bosch GmbH: Stuttgart, Germany, 1991.
24. Ding, L.; Shen, Y.; Zhou, Y. Real-Time Performance Analysis and Evaluation of CAN Bus with an Extended Petri Net Model. In *Proceedings of the 2009 IEEE Instrumentation and Measurement Technology Conference, Singapore, 5–7 May 2009*; IEEE: New York, NY, USA, 2009. [[CrossRef](#)]
25. AUTOSAR. *Layered Software Architecture; AUTOSAR Classic Platform, Release R25-11*; AUTOSAR: Munich, Germany, 2025. Available online: https://www.autosar.org/fileadmin/standards/R25-11/CP/AUTOSAR_CP_EXP_LayeredSoftwareArchitecture.pdf (accessed on 2 August 2026).
26. Oh, S.B.; Lee, K.; Kim, H. Performance Enhancement of CAN/Ethernet Automotive Gateway with a CAN Data Reduction Algorithm. *Electronics* **2023**, *12*, 2777. [[CrossRef](#)]
27. Seo, S.; Kim, J.; Hwang, S.; Kwon, K.; Jeon, J. A Reliable Gateway for In-Vehicle Networks Based on LIN, CAN, and FlexRay. In *Proceedings of the International Conference on Computer and Information Technology (CIT 2011)*, Yichang, China, 16–18 August 2011.
28. ETSI EN 303 798 V2.1.1; Intelligent Transport Systems (ITS); LTE-V2X and NR-V2X Access Layer Specification for Intelligent Transport Systems Operating in the 5 GHz Frequency Band. ETSI: Sophia Antipolis, France, 2024. Available online: https://www.etsi.org/deliver/etsi_en/303700_303799/303798/02.01.01_60/en_303798v020101p.pdf (accessed on 2 August 2026).
29. IEEE Std 802.11-2024; IEEE Standard for Information Technology—Telecommunications and Information Exchange between Systems: Local and Metropolitan Area Networks—Specific Requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. IEEE: Piscataway, NJ, USA, 2025.
30. Jeong, Y.; Son, S.; Jeong, E.; Lee, B. A Design of a Lightweight In-Vehicle Edge Gateway for the Self-Diagnosis of an Autonomous Vehicle. *Appl. Sci.* **2018**, *8*, 1594. [[CrossRef](#)]
31. SAAB WIS (Workshop Information System) Online. Available online: <https://saabwisonline.com> (accessed on 2 August 2026).
32. Karlsson, J. SAAB P-Bus CANMessage Information. Available online: <https://roffe.nu/tomi/p-bus.html> (accessed on 2 August 2026).

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.