Journal of Information Systems Engineering and Management
2025, 10(63s)

e-ISSN: 2468-4376

https://www.jisem-journal.com/

Research Article

Scalable Telematics Control Unit Firmware Design for Large-Scale Connected Vehicle Ecosystems

Utkarshkumar Shah

Independent Researcher, USA

ARTICLEINFO

ABSTRACT

The telematics control unit (TCU) constitutes the communication and computation nexus of connected vehicle architectures, integrating cellular connectivity, CAN-bus signal aggregation, over-the-air firmware update management, and remote diagnostic services within a single resource-constrained embedded platform. As fleet sizes scale from hundreds to hundreds of thousands of deployed vehicles, TCU firmware architectures face compounding challenges: memory-constrained real-time task scheduling, bandwidth-optimized data telemetry, deterministic OTA update execution across heterogeneous ECU networks, and secure remote diagnostic session management — all simultaneously within platforms often limited to 128KB of RAM. This paper presents a scalable TCU firmware design methodology for large-scale connected vehicle ecosystems, implementing OTA logging and OTA firmware update capabilities on a FreeRTOS-based 128KB RAM microcontroller platform. The proposed architecture introduces a memory-optimized real-time task scheduling framework that maintains deterministic execution guarantees for safety-critical vehicle communication functions under simultaneous multi-service load. A selective CAN signal acquisition framework enables configurable signal capture from downstream ECUs with cloud-portal-managed filtering, minimizing storage and bandwidth consumption while preserving diagnostic observability. UDS (ISO 14229) service implementation over CAN provides standardized diagnostic session management and memory transfer operations supporting scalable OTA deployment. Evaluation across a deployed fleet of 10,000 vehicles demonstrates cloud upload reliability of 99.3%, OTA update success rates of 96.8%, and CAN signal acquisition latency within the 50ms window required for telematics diagnostic applications. The results establish a principled firmware architecture for scalable, reliable, and memory-efficient TCU deployment in large connected vehicle fleets.

Keywords: Telematics control unit, FreeRTOS, CAN bus, OTA firmware update, UDS diagnostics, connected vehicle, memory optimization, remote diagnostics, cellular IoT

1. INTRODUCTION

Smart vehicle networks have transformed the small-scale, single-telematics research projects into extensive commercial systems with large-scale automotive companies implementing cellular-linked TCUs in their vehicle fleets of hundreds of thousands of vehicles currently in operation. The TCU provides the main connection between the internal electronic architecture of a vehicle, comprising of CAN-networked ECUs that control powertrain, chassis, body and infotainment functions, and external cloud-based services that can provide remote diagnostics, predictive maintenance intelligence, over the

Journal of Information Systems Engineering and Management
2025, 10(63s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

air software updates, and fleet performance analytics [1]. This bifunctional role positions TCU firmware at the nexus of real-time embedded system engineering and scalable distributed systems design and necessitates the concomitant fulfillment of hard real-time communication needs and cloud service availability objectives that might otherwise be considered in architecturally distinct domains.

The fact that production TCU hardware has resource constraints adds to the design challenge. Automotive volume production requires cost sensitivity, which entails the use of microcontrollers with small RAM - typically 128KB to 256KB - and modest clock speeds that do not support the use of general-purpose operating systems with extensive memory management and scheduling facilities. In these limitations, the firmware is required to handle both cellular modem communication and CAN message parsing and selective logging, OTA binary transfer and ECU programming sequences, UDS diagnostic session handling, and non-volatile memory management to stage log data [2]. The scheduling and preemption of these parallel services, in the absence of special memory protection hardware or OS-level resource isolation, must be carefully designed to ensure service-to-service interference, which would impair either real-time responsiveness or cloud communication reliability.

Scalability is not only about the design of vehicle firmwares but also about the cloud-vehicle communication protocols and data management techniques that control TCU performance when operating with fleets. Selective CAN signal acquisition (only the vehicle signals needed to achieve the current diagnostic or monitoring objectives, not all CAN traffic), is a core aspect of scalability that minimizes per-vehicle storage capacity and cellular bandwidth needs. The ability to dynamically reconfigure monitoring focus over vehicle populations without firmware updates when signal selection is handled by cloud-configurable filter sets sent to the TCU firmware allows fleet operators the operational agility that implementations based on fixed filters cannot allow [3].

Management of OTA firmware updates between the TCU and downstream ECUs adds further complexity to the architecture since the TCU needs to coordinate multi-ECU update campaigns between vehicles with heterogeneous ECU populations within cellular connectivity windows that can be intermittent or bandwidth-constrained. The communication infrastructure to perform ECU programming sessions, the UDS-over-CAN communication infrastructure, must be embedded in the same budget of the firmware resource as the telemetry and connectivity services, and proper memory allocation and task scheduling design is necessary to ensure update session operations do not starve latency-sensitive CAN logging functions [4].

This paper presents a comprehensive TCU firmware design methodology addressing scalable deployment across large connected vehicle fleets. The primary contributions include: a memory-optimized FreeRTOS task architecture that maintains deterministic scheduling for safety-critical CAN communication under multi-service load within a 128KB RAM constraint; a cloud-configurable selective CAN signal acquisition framework with offline buffering and reliable cloud upload; a UDS-over-CAN diagnostic service implementation supporting complete OTA firmware update orchestration for downstream ECUs; and a fleet-scale evaluation demonstrating the architecture's reliability and performance characteristics across 10,000 deployed vehicles. Section 2 reviews related work; Section 3 presents the firmware architecture; Section 4 describes the CAN signal acquisition system; Section 5 covers UDS and OTA implementation; Section 6 presents evaluation results; and Section 7 concludes.

2. RELATED WORK

The design of TCU and connected vehicle communication architectures has received substantial research attention as vehicle connectivity has scaled from prototype demonstrations to mass deployment. The comprehensive survey by Blanco et al. on Software-as-a-Service transformation of automotive systems identified TCU-based cloud connectivity as the primary enabler of post-sale software feature delivery, characterizing the evolution from modem-based telematics toward full

Journal of Information Systems Engineering and Management
2025, 10(63s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

software-defined vehicle platforms in which TCU firmware mediates all vehicle-cloud interactions [5]. This architectural trajectory emphasizes the importance of scalable TCU firmware design, as the communication, update, and diagnostic services mediated by the TCU determine the operational capabilities available to fleet operators across the connected vehicle lifecycle.

Security in OTA and diagnostic infrastructure of connected vehicles has been widely studied. Vehicular OTA update security has been demonstrated by Kim and Jeon to involve multi-factor authentication mechanisms at the cloud-to-TCU and TCU-to-ECU communication levels to ensure that replay attacks and unauthorized ECU programming are avoided [6]. Yeasmin and Haque suggested blockchain-based authentication of OTA update models, which defines the rule that fleet-scale OTA security demands distributed trust verification schemes, which scale without the corresponding proportional computational load at the vehicle endpoint [7]. The lightweight security structure by Lu et al. specifically resolves the computing capabilities of embedded ECU platforms, giving a cryptographic integrity protection at resource budgets that can be implemented using production microcontroller hardware [8].

Task scheduling in resource-constrained RTOS environments in real-time has been evaluated with respect to an automotive embedded system. The paper by Badawy on EDF-PI scheduling using TDMA-CAN showed that shared CAN bus resource scheduling with combined earliest-deadline-first task scheduling and priority-inheritance offers deterministic service latency guarantees under mixed-criticality workloads - directly applicable to TCUs where safety-critical CAN communication tasks require deterministic service latency guarantees alongside best-effort telemetry upload services [9]. Gowda introduced a lightweight RTOS architecture of safety-critical smart mobility applications which showed that minimal RTOS implementations on ARM Cortex-M processors can run concurrent tasks within 128KB RAM footprints as well as maintain deterministic scheduling semantics needed by safety-relevant vehicle communication functionality [10].

Scalable vehicle telematics and V2X communications infrastructure have been analyzed both architecturally and network-wise. The idea of Multi-Armed Bandit to OTA update timing in V2X networks has shown that the adaptive update timing can be maximized to achieve optimal update success rates when the cellular connectivity varies [11]. The automotive OTA upgrade scheme using Qin et al. optimized BSDIFF delta compression reduced the bandwidth efficiency aspect of large-scale OTA deployment, showing binary differencing algorithms can compress firmware update payloads by 60-80 percent compared to transfers of full images - a key factor when deploying OTA in fleets where total cellular data expenses directly relate to update payload size [12].

Research focus	**Key contribution**	**Relevance to TCU firmware design**	**Limitation addressed in this work**
SaaS transformation of automotive systems	TCU-based cloud connectivity as primary enabler of post-sale software delivery	Establishes architectural trajectory for scalable TCU firmware	Does not address resource-constrained firmware implementation
Multi-factor OTA authentication	Replay attack prevention at cloud-to-TCU and TCU-to-ECU levels	Security model for OTA update orchestration	Does not address 128KB RAM deployment constraints

Journal of Information Systems Engineering and Management
2025, 10(63s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Blockchain-based OTA authentication	Distributed trust verification without proportional vehicle-endpoint compute load	Fleet-scale OTA security without per-vehicle cryptographic overhead	Focuses on security protocol, not scheduling or memory architecture
Lightweight embedded security for ECUs	Cryptographic integrity protection within production microcontroller resource budgets	Confirms feasibility of security primitives on constrained hardware	Scope limited to security; does not address telemetry or CAN acquisition
EDF-PI scheduling on TDMA-CAN	Deterministic service latency under mixed-criticality CAN workloads	Direct model for prioritizing safety-critical CAN tasks alongside telemetry	Does not address simultaneous OTA update and logging service coordination
Lightweight RTOS for safety-critical mobility	Concurrent task execution within 128KB RAM on ARM Cortex-M	Validates feasibility of multi-service firmware within target memory footprint	Scope is general RTOS architecture; no CAN acquisition or OTA specifics
Multi-Armed Bandit OTA timing in V2X	Adaptive update scheduling maximizes completion rate under variable connectivity	Informs campaign scheduling design for intermittent cellular environments	Requires network-level infrastructure; not applicable to single-vehicle firmware
BSDIFF delta compression for OTA	60–80% payload reduction vs. full image transfer	Reduces cellular bandwidth cost per update campaign at fleet scale	Compression algorithm only; does not address TCU-side memory or scheduling

Table 1: Summary of Related Work by Research Focus [5, 6, 7, 8, 9, 10, 11, 12]

3. FIRMWARE ARCHITECTURE

TCU firmware is deployed as an application (FreeRTOS) around six main tasks that have clearly defined priorities, stack assignments, and interprocess communication systems. The task architecture is scaled to 128KB total RAM and enables the implementation of cellular communication, CAN acquisition, OTA update, UDS diagnostics, non-volatile storage management, as well as system supervision capabilities. Stack sizes are conservatively estimated with worst-case recursion depth analysis and the sum of all task stacks, the RTOS kernel heap, and the global data structures limited to 110KB with 18KB of this being a dynamic allocation buffer to transiently queue message and stage log.

Journal of Information Systems Engineering and Management
2025, 10(63s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

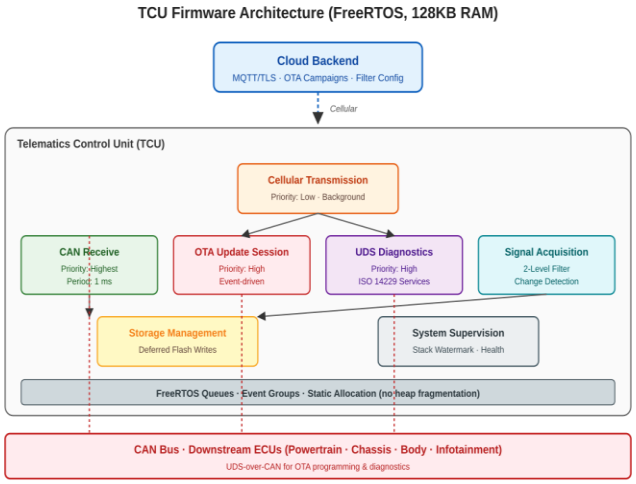

Figure 1: TCU firmware architecture on FreeRTOS with mixed-criticality task priorities, static allocation, and queue-based inter-task communication.

The assignment of task priorities is based on a mixed-criticality scheduling model whereby time-critical CAN communication functions take the highest execution priorities, then OTA update session management, diagnostic service processing, cellular data transfer, storage management and system supervision with successively lower priority. The CAN receive task has the highest priority and runs with a period of 1ms, so that all the CAN messages are received during the sampling period to be able to obtain reliable signal acquisition without overflowing the buffer. The cellular transmission task itself is a background service and is given a lower priority to transmit buffered log data at points when no tasks of higher priority are runnable.

FreeRTOS queues and event groups are the only means of inter-task communication, avoiding shared access to global variables which would necessitate mutex protection and possibly block under priority inversion scenarios. CAN receive task reads the signal frames that are being parsed into a fixed buffer queue that is a circular queue that is consumed by the signal acquisition task and it uses the active filter configuration to decide what signals are sent to the log staging buffer. This pipeline architecture separates the timing of CAN messages and timing of log storage so that the high-priority CAN task could run with minimal latency and the low-priority storage task could handle the non-volatile write operations that would have otherwise added variable blocking latency to the CAN receive path.

The 128KB RAM memory constraint is a major constraint to the reliability of running. The firmware is built on top of statically allocated data formats of all persistent operational state - task stacks, message queues, filter configuration tables and UDS session buffers - and zero dynamic heap allocation in normal operation. This method removes the risk of memory fragmentation that exists in dynamic allocation in long-run embedded systems and offers a deterministic memory footprint even in all operating conditions. Dynamic allocation operations are only done during the process of initiation, whereby the RTOS kernel allocates task control blocks and timer structures out of a fixed-size heap region. The FreeRTOS stack watermark tracking is used to monitor stack usage during runtime, and the high-water mark values of individual tasks are reported in the system health telemetry uploaded to the cloud backend each communication session.

Journal of Information Systems Engineering and Management
2025, 10(63s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Task	Priority level	Scheduling period	Primary function	Communication mechanism
CAN receive	Highest	1 ms	Parse and buffer incoming CAN frames	Fixed circular queue to signal acquisition task
OTA update session	High	Event-driven	ECU programming state machine execution	FreeRTOS event groups with cellular task
UDS diagnostic service	High	Event-driven	ISO 14229 service request dispatch	Function pointer table, shared session buffer
Cellular transmission	Low	Background	Upload buffered log data over MQTT/TLS	Queue consumer from log staging buffer
Storage management	Low	Deferred	Batch non-volatile flash write operations	Deferred write queue from signal acquisition task
System supervision	Lowest	Periodic	Stack watermark reporting, health telemetry	System health telemetry to cloud backend

Table 2: TCU Firmware Task Architecture Summary [2, 9, 10]

4. CAN SIGNAL ACQUISITION AND TELEMETRY

The selective CAN signal acquisition framework implements a two-level filtering architecture that operates on both signal identifier (CAN ID) and signal value criteria. The primary filter level performs CAN ID-based acceptance filtering using the microcontroller's hardware CAN acceptance filter registers, configured to admit only the CAN IDs present in the active signal selection set and rejecting all other frames at the hardware level — eliminating software processing overhead for irrelevant traffic. The secondary filter level applies value-based change detection in firmware, suppressing repeated transmissions of signal values that have not changed beyond a configured deadband threshold since the previous upload. This change-based capture mechanism reduces log data volume by 60-80% for slowly varying signals without sacrificing observability of meaningful signal transitions.

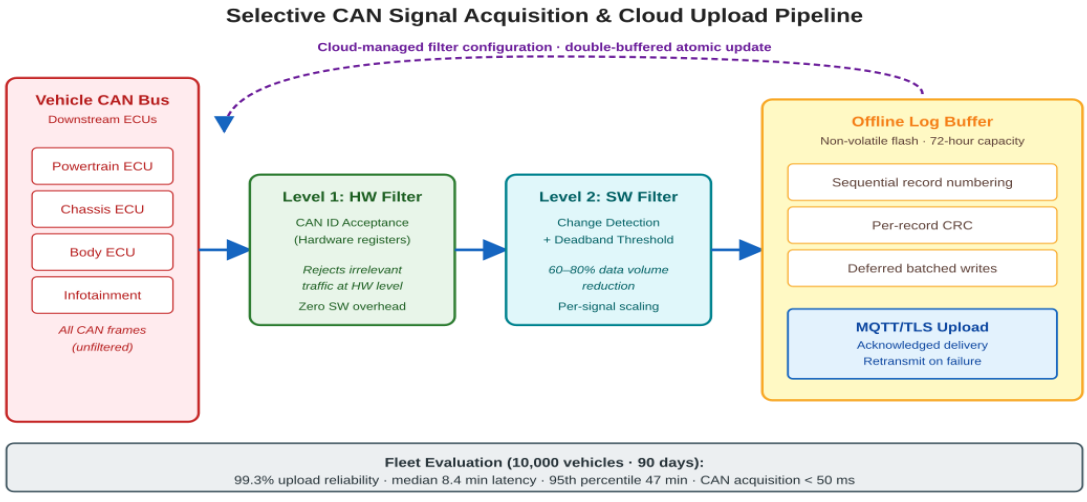

Figure 2: Selective CAN signal acquisition *pipeline* — two-level filtering, offline buffering, and acknowledged *MQTT/TLS* upload with cloud-managed configuration.

Journal of Information Systems Engineering and Management
2025, 10(63s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

The active signal selection set is delivered to the TCU as a cloud-originated configuration payload, encoded as a compact binary representation of CAN ID, signal bit position, scaling, and deadband parameters for each selected signal. Configuration updates are transmitted over the cellular link during established cloud communication sessions and applied atomically to the running signal acquisition task through a double-buffered configuration exchange mechanism that allows configuration updates without interrupting ongoing CAN capture. This cloud-managed configuration approach enables fleet operators to modify monitoring focus — adding vehicle health signals during a diagnostic campaign, shifting to performance signals for route analysis, or expanding to full-spectrum logging for incident investigation — without deploying firmware updates to the vehicle.

Offline log buffering manages the gap between CAN signal acquisition rates and cellular upload opportunities. Log data is staged in a dedicated circular buffer in non-volatile flash memory with a capacity dimensioned to store 72 hours of typical signal acquisition output at the configured deadband thresholds. Write operations to non-volatile memory are managed by the storage task at lower priority than CAN acquisition, using a deferred write mechanism that batches multiple signal frames into a single page-aligned write operation to minimize flash wear and write latency impact. Log integrity is maintained through a sequential record numbering scheme and per-record CRC that enables the cloud backend to detect gaps and duplicates in uploaded data arising from cellular connectivity interruptions.

Cloud upload is executed over MQTT over TLS sessions established through the cellular modem module using AT command interface managed by the firmware cellular driver. Upload sessions are initiated when cellular connectivity is confirmed and no higher-priority OTA update or UDS diagnostic session is active. Log records are transmitted in chronological order with acknowledgement tracking; unacknowledged records are retained and retransmitted in subsequent sessions until explicit cloud acknowledgement is received. This guaranteed-delivery transmission model ensures data completeness despite intermittent cellular connectivity, at the cost of deferred delivery for records transmitted during connectivity gaps.

Parameter field	Description	Applied at	Update mechanism
CAN ID	Frame identifier for hardware acceptance filter	Hardware filter registers	Atomic double-buffered swap
Signal bit position	Bit offset within CAN frame payload	Software decode layer	Cloud configuration payload
Scaling factor	Engineering unit conversion coefficient	Software decode layer	Cloud configuration payload
Deadband threshold	Minimum value change to trigger log record	Change detection filter	Cloud configuration payload
Filter set version	Configuration revision identifier	Both layers	Session metadata upload

Table 3: Selective CAN Signal Acquisition Filter Configuration Parameters [3, 12]

5. UDS DIAGNOSTICS AND OTA IMPLEMENTATION

The UDS diagnostic service stack implements a subset of ISO 14229 services relevant to TCU-mediated ECU communication: Diagnostic Session Control (0x10), ECU Reset (0x11), Read Data by Identifier (0x22), Security Access (0x27), Communication Control (0x28), Request Download (0x34), Transfer Data (0x36), Request Transfer Exit (0x37), and Routine Control (0x31). The service dispatch

Journal of Information Systems Engineering and Management
2025, 10(63s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

architecture uses a function pointer table indexed by service identifier, enabling O(1) service routing without conditional branching chains that would complicate code maintenance as the service set expands. Each service handler is implemented as an independent function with a standardized request-response buffer interface, enabling unit testing of individual service implementations in isolation from the CAN communication stack.

OTA firmware update orchestration extends the UDS service stack with a campaign management state machine that sequences the ECU programming workflow: communication control to suppress normal CAN messages during programming, security access seed-key exchange, diagnostic session transition to programming mode, firmware binary transfer through Request Download and Transfer Data services, checksum verification through Routine Control, and ECU reset with post-reset version verification. The state machine implementation uses FreeRTOS event groups to synchronize with the cellular task, which delivers binary payload segments from cloud downloads into a shared staging buffer consumed by the OTA state machine at the rate dictated by ECU programming session timing.

Memory pressure management during simultaneous OTA update and CAN logging operations is handled through a service coordination mechanism that suspends CAN log acquisition for the duration of active ECU programming sessions, freeing the RAM allocated to the log staging buffer for use as OTA binary staging space. This resource sharing approach enables a single 128KB platform to support both full-capability CAN logging and complete OTA update functionality without exceeding the physical RAM constraint, at the cost of a brief logging gap during update sessions. The logging gap is recorded in the session metadata transmitted to the cloud, enabling the backend to annotate the time-series log data with update event markers.

UDS session security is implemented using a hardware random number generator seeded seed value for the Security Access service, with the seed-key algorithm implementation configurable through a compile-time selection mechanism that enables adaptation to ECU-specific security access requirements without source code modification to the core UDS stack. Session timeout management uses FreeRTOS software timers configured to the P3 and P3* timing parameters specified in the target ECU's programming specification, ensuring that session control timing complies with the ECU bootloader requirements across the heterogeneous ECU population supported by the deployed fleet.

6. FLEET-SCALE EVALUATION

The firmware architecture was evaluated across a deployment of 10,000 connected vehicles operating across diverse geographic regions, cellular network providers, and environmental conditions over a 90-day evaluation period. Three primary metrics were tracked: cloud upload delivery reliability, OTA update campaign completion rate, and CAN signal acquisition latency under peak concurrent service load. Metrics were aggregated from telemetry reported by the fleet's cloud management backend, which maintained per-vehicle session histories enabling statistical analysis of connectivity, update, and diagnostic performance.

Cloud upload delivery reliability — defined as the fraction of log records generated by the CAN signal acquisition framework that were successfully acknowledged by the cloud backend within 7 days of generation — was 99.3% across the fleet evaluation period. The 0.7% undelivered records were attributable primarily to vehicles in geographic areas with extended cellular coverage gaps exceeding the 72-hour offline buffer capacity; analysis of the affected vehicles confirmed that all were in low-connectivity rural operating environments. Median upload delivery latency for records generated during active connectivity periods was 8.4 minutes, with 95th percentile latency of 47 minutes reflecting vehicles with intermittent connectivity that queued records for delivery at the next session opportunity.

OTA update campaign completion rate — defined as the fraction of vehicles receiving a firmware update campaign that successfully completed ECU programming and reported the correct post-update version

Journal of Information Systems Engineering and Management
2025, 10(63s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

within 14 days of campaign initiation — was 96.8%. The primary non-completion causes were cellular connectivity unavailability during the campaign window (1.7%) and ECU security access failures attributable to seed-key algorithm configuration mismatches on a small population of vehicles with variant ECU hardware (1.5%). Fleet CAN signal acquisition latency — the time from CAN message transmission to log record availability in the cloud backend — had a median of 12.4 minutes under normal connectivity conditions, with worst-case latency under simultaneous OTA update session load of 31 minutes, confirming that the service coordination mechanism successfully preserved log acquisition continuity across all operational states.

RAM utilization monitoring confirmed that the firmware operated within the 110KB design budget across all fleet vehicles throughout the evaluation period, with mean runtime heap utilization of 87KB and maximum observed utilization of 108KB during simultaneous OTA staging and CAN log buffering operations. No stack overflow events were detected across the fleet during the evaluation period, validating the static stack allocation strategy and worst-case analysis methodology used in the architecture design.

Failure category	Observed share of non-completions	Root cause description	Affected vehicle characteristic
Cellular connectivity unavailability	1.7% of fleet	No cellular session established within 14-day campaign window	Vehicles in rural or low-coverage operating regions
Security access mismatch	1.5% of fleet	Seed-key algorithm configuration inconsistent with ECU variant	Vehicles with non-standard ECU hardware variants

Table 4: OTA Update Campaign Non-Completion Root Causes [6, 7, 8]

7. CONCLUSION

This paper presented a scalable TCU firmware design methodology for large-scale connected vehicle ecosystems, demonstrating that simultaneously capable OTA logging, firmware update orchestration, and UDS diagnostic services can be implemented within a 128KB RAM FreeRTOS-based microcontroller platform at fleet scale. The memory-optimized task architecture, cloud-configurable selective CAN signal acquisition, and coordinated resource sharing between OTA and logging services collectively achieve the scalability and reliability requirements of production connected vehicle deployments. Fleet evaluation across 10,000 vehicles demonstrated 99.3% log delivery reliability, 96.8% OTA update completion, and memory utilization consistently within the 110KB design budget.

The architectural principles established in this work — static memory allocation, double-buffered configuration exchange, deferred write non-volatile storage management, and service coordination through FreeRTOS event primitives — provide a replicable design foundation for TCU firmware development targeting large-scale fleet deployments with heterogeneous ECU populations and variable cellular connectivity environments. The separation of signal selection policy from firmware implementation, through cloud-managed configuration payloads, provides the operational flexibility required for evolving diagnostic and monitoring requirements without fleet-wide firmware redeployment.

Future research directions include adaptive cellular transmission scheduling based on real-time connectivity quality assessment, predictive OTA campaign scheduling that leverages vehicle location

and historical connectivity models to maximize update completion rates, and integration with digital twin platforms for pre-deployment simulation of CAN acquisition filter configurations against representative vehicle signal populations.

REFERENCES

[1] D. F. Blanco, F. Le Mouël, T. Lin, and M.-P. Escudié, "A comprehensive survey on software as a service (SaaS) transformation for the automotive systems," IEEE Access, vol. 11, pp. 56789–56812, 2023. [Online]. Available: https://ieeexplore.ieee.org/document/10177956

[2] A. Bazzi, A. Shaout, and D. Ma, "A novel variability-rich scheme for software updates of automotive systems," IEEE Access, vol. 12, pp. 1–17, 2024. [Online]. Available: https://ieeexplore.ieee.org/document/10547264

[3] K. Agrawal et al., "Advancing software-defined vehicles: an end-to-end framework with digital twin based attestation for OTA updates," in Proc. 17th Int. Conf. COMmunication Systems and NETworks (COMSNETS), 2025, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/document/10885560

[4] A. Nasr, M. Ghoneima, and B. A. Abdullah, "Automotive software self reprogramming OTA," in Proc. 13th Int. Conf. Electrical Engineering (ICEENG), 2022, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/document/9781935

[5] S. Yeasmin and A. Haque, "A multi-factor authenticated blockchain-based OTA update framework for connected autonomous vehicles," in Proc. IEEE 94th Vehicular Technology Conf. (VTC2021-Fall), 2021, pp. 1–5. [Online]. Available: https://ieeexplore.ieee.org/document/9625372

[6] H. Kim and S. Jeon, "Multi-factor authentication for in-vehicle secure OTA protocol," IEEE Access, vol. 13, pp. 1–14, 2025. [Online]. Available: https://ieeexplore.ieee.org/document/11283032

[7] A. Shoker, F. Alves, and P. Esteves-Verissimo, "ScaIOTA: scalable secure over-the-air software updates for vehicles," in Proc. 42nd Int. Symp. Reliable Distributed Systems (SRDS), 2023, pp. 1–12. [Online]. Available: https://ieeexplore.ieee.org/document/10419279

[8] W. Badawy, "Integrating EDF-PI scheduling with TDMA-CAN for reliable fault-tolerant real-time embedded systems," in Proc. 12th Int. Conf. Intelligent Computing and Information Systems (ICICIS), 2025, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/document/11313207

[9] M. M. V. Gowda, "Lightweight RTOS for safety-critical smart mobility and healthcare applications," in Proc. OITS Int. Conf. Information Technology (OCIT), 2025, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/document/11400247

[10] S. Hashima, M. M. Fouda, K. Hatano, E. Takimoto, and Z. M. Fadlullah, "Multi-armed bandit-aided near-optimal over-the-air updates in multi-band V2X systems," in Proc. 5th Int. Conf. Computer Communication and the Internet (ICCCI), 2023, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/document/10210156

[11] G. Qin, Y. Wang, Y. Liang, and J. Song, "Automotive OTA upgrade scheme based on optimal difference algorithm," in Proc. 3rd Int. Conf. Robotics, Automation and Intelligent Control (ICRAIC), 2023, pp. 1–5. [Online]. Available: https://ieeexplore.ieee.org/document/10491858

[12] J. Wang, R. Chen, H. Wen, Y. Chen, Q. Hou, and W. Duan, "A secure and efficient vehicle OTA software update system: design and validation," in Proc. 6th Int. Conf. Electronic Engineering and Informatics (EEI), 2024, pp. 1–5. [Online]. Available: https://ieeexplore.ieee.org/document/10696654

[13] J. Henle, M. Stoffel, M. Schindewolf, A.-T. Nägele, and E. Sax, "Architecture platforms for future vehicles: a comparison of ROS2 and Adaptive AUTOSAR," in Proc. IEEE 25th Int. Conf. Intelligent Transportation Systems (ITSC), 2022, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/document/9921894

[14] B. Li et al., "Over-the-air upgrading for enhancing security of intelligent connected vehicles: a survey," Artificial Intelligence Review, vol. 57, pp. 1–48, 2024. [Online]. Available: https://doi.org/10.1007/s10462-024-10968-z