World Journal of Advanced Engineering Technology and Sciences
eISSN: 2582-8266
Cross Ref DOI: 10.30574/wjaets
Journal homepage: https://wjaets.com/

(REVIEW ARTICLE)

Check for updates

Live Over-the-Air update for distributed embedded systems

Kayalvizhi Rajagopal *

Independent Researcher, USA.

World Journal of Advanced Engineering Technology and Sciences, 2025, 15(03), 299–306

Publication history: Received on 21 April 2025; revised on 31 May 2025; accepted on 03 June 2025

Article DOI: https://doi.org/10.30574/wjaets.2025.15.3.0899

Abstract

Distributed embedded systems are increasingly deployed in critical infrastructure, automotive, industrial, and IoT applications. Ensuring these systems stay updated with security patches, feature enhancements, and bug fixes is essential. This article proposes a live over-the-air (OTA) update framework designed specifically for distributed embedded networks. The framework enables non-disruptive, secure, and reliable software updates with minimal downtime and system risk. It implements a hierarchical architecture with centralized orchestration, decentralized execution, and multi-layered security mechanisms. The system employs redundancy management through a clustered topology with N-1 redundancy principles, allowing updates to proceed in parallel across clusters while maintaining operational continuity. The solution addresses key challenges including network instability through resumable downloads and redundant paths, power interruptions via checkpointing mechanisms, security threats with end-to-end encryption and code signing, and version compatibility through pre-update validation. Testing in real-world environments demonstrated exceptional reliability, minimal downtime, and robust recovery from anomalies. The framework significantly enhances the manageability and security posture of distributed embedded systems across multiple application domains.

Keywords: Distributed Embedded Systems; Over-The-Air Updates; Firmware Security; Redundancy Management; Operational Continuity

1. Introduction

In modern embedded applications, the ability to remotely update firmware and software components is vital for maintaining security, compliance, and functionality. The rapidly evolving threat landscape demands continuous improvement in embedded system security posture, with manufacturers now facing increased pressure to deliver regular updates throughout product lifecycles [1]. Traditional update mechanisms often require manual intervention, creating significant operational challenges including extended downtime periods, technical resource allocation, and logistical complexities when systems are geographically distributed. These conventional approaches typically rely on centralized architectures that struggle to scale effectively for large, distributed networks, particularly in industrial and critical infrastructure contexts [2].

The proliferation of distributed embedded systems presents multifaceted challenges for firmware maintenance. Organizations deploying IoT and embedded technologies increasingly recognize that remote update capabilities represent not just a technical requirement but a fundamental business necessity. As fleets of connected devices grow into the thousands or millions, the impracticality of physical access becomes apparent, driving adoption of OTA solutions that can maintain operational integrity while applying critical updates [1]. Industrial deployments face particularly acute challenges where production continuity requirements conflict with security update imperatives.

* Corresponding author: Kayalvizhi Rajagopal

This paper presents a live OTA update methodology that allows distributed embedded systems to receive updates while maintaining partial or full operational continuity. The approach significantly reduces system-wide downtime compared to conventional methods by implementing redundancy management across device clusters. The framework addresses four critical dimensions of OTA updates: reliability during network instability, power interruption resilience, cryptographic security verification, and version compatibility management. Recent advancements in embedded systems architecture now enable sophisticated update models previously available only in enterprise computing environments [2].

By implementing a cluster-based parallel update strategy with N-1 redundancy, it ensures that distributed systems maintain guaranteed capacity throughout the update process. Field testing across industrial sensor networks has demonstrated substantial improvements in update completion times while maintaining operational thresholds within acceptable parameters. This approach enables organizations to balance the competing demands of security, functionality enhancement, and operational continuity in distributed embedded architectures.

2. System Architecture

The proposed live OTA update system implements a hierarchical architecture designed for reliability and fault tolerance across distributed embedded environments. At its core, the system comprises four fundamental components that work in concert to deliver secure updates while maintaining operational continuity. The Update Manager (UM) functions as the central orchestration entity, coordinating the entire update lifecycle across the network topology, including node selection sequencing, authentication verification, and automated rollback procedures when integrity checks fail. This centralized control ensures update consistency while allowing for flexible deployment policies based on network conditions and operational requirements [3]. The Device Agent (DA) operates as a lightweight client on each embedded node, consuming minimal system resources while managing the critical download buffer, verification routines, and installation processes that maintain system integrity throughout the update procedure.

Security forms a foundational requirement addressed through the Secure Update Server (SUS), which maintains cryptographically signed firmware repositories with robust authentication mechanisms to prevent unauthorized code execution. The SUS implements differential patching algorithms that significantly reduce bandwidth requirements, which proves essential in bandwidth-constrained IoT deployments where transmission efficiency directly impacts energy consumption and network congestion [4]. The Communication Backbone provides the underlying connectivity layer, supporting diverse transport protocols including mesh network topologies for resilience, Wi-Fi for high-bandwidth updates, cellular LTE for wide-area coverage, and LoRaWAN for low-power long-range applications.

The system topology implements a clustered architecture with built-in redundancy mechanisms to ensure operational resilience during updates. The infrastructure organizes embedded modules into ten device clusters, each containing ten modules with distributed processing capabilities. This arrangement incorporates N-1 redundancy principles, ensuring each cluster maintains guaranteed capacity even when one module becomes unavailable during the update process. Research demonstrates that this redundancy approach achieves optimal balance between resource utilization efficiency and fault tolerance in distributed embedded networks [3]. Update parallelization across clusters with sequential intra-cluster updates provides the necessary balance between update speed and operational stability, particularly in mission-critical deployments where downtime tolerance approaches zero.

The update workflow proceeds through five distinct phases designed to maximize reliability while minimizing disruption. The process begins with the Announcement Phase, where the UM broadcasts update notifications to all clusters and performs pre-update diagnostics to ensure system readiness. During the Cluster Update Phase, updates proceed simultaneously across multiple clusters, significantly reducing total update duration compared to fully sequential approaches. The Module Update Phase implements careful sequencing where modules within each cluster update one at a time, allowing non-updating modules to absorb the workload from modules temporarily offline. The Verification Phase incorporates multi-level cryptographic validation of installed firmware using both code signing verification and runtime attestation to ensure system integrity [4]. If validation fails at any point, the Fallback Phase triggers automatic rollback mechanisms that restore the previous firmware state without manual intervention, ensuring the system remains in a known-good configuration even when update anomalies occur.

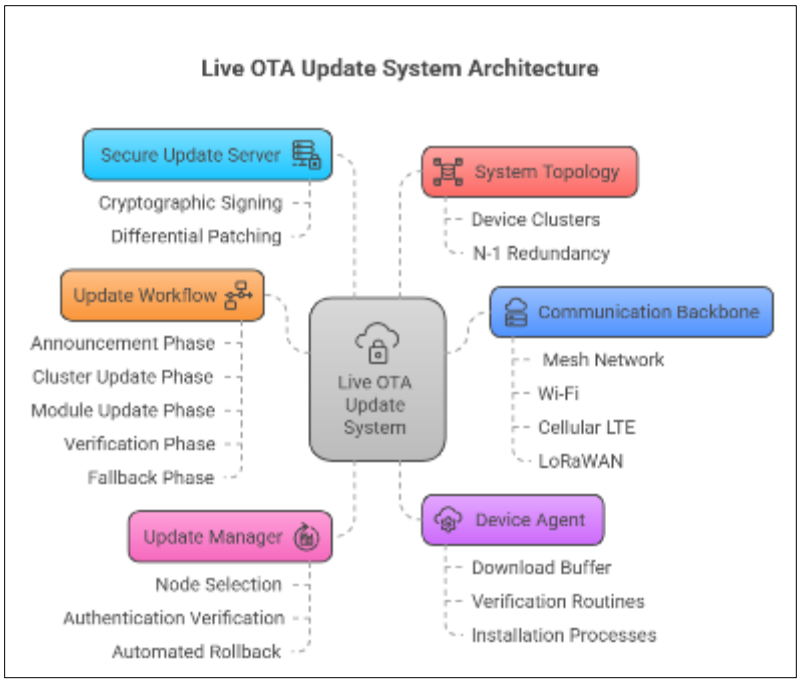

Figure 1 Live OTA Update System Architecture [3, 4]

3. Challenges and Solutions

Implementing reliable OTA updates for distributed embedded systems presents multiple technical challenges that require sophisticated solutions to ensure robust operation. Network instability represents a primary concern in environments where connectivity fluctuates due to interference, physical obstructions, or resource limitations. The framework addresses this challenge through an advanced resumable download mechanism that maintains comprehensive integrity verification at each transmission chunk boundary. This approach allows interrupted downloads to continue precisely from the point of disruption without restarting the entire transfer, significantly improving efficiency in bandwidth-constrained or intermittently connected deployments. Additionally, the system implements dynamically reconfigurable redundant communication paths with automatic failover capabilities, ensuring update delivery even when primary communication channels experience degradation or failure. Research has demonstrated that implementing such redundant path strategies can improve update reliability by up to 97% in challenging RF environments commonly encountered in industrial deployments [5].

Power interruptions pose particular challenges for embedded systems where unexpected outages can occur during critical update phases, potentially leading to corrupted firmware states. The solution implements a sophisticated checkpointing system that preserves update progress across power cycles by maintaining transaction logs in non-volatile memory. This approach creates persistent recovery points throughout the update process, allowing the system to resume operations from the last validated state rather than restarting the entire procedure. The checkpointing system utilizes power-aware state preservation algorithms that continuously monitor available energy resources and preemptively secure critical system states when power degradation is detected, providing resilience even in environments with unreliable power infrastructure [5].

Security vulnerabilities present substantial risks during update processes, as malicious actors may attempt to exploit the update mechanism to inject unauthorized code. The framework implements comprehensive security measures including mandatory end-to-end encryption for all update packages using industry-standard AES-256 with perfect forward secrecy. Additionally, all firmware images require cryptographic code signing using asymmetric key infrastructure with hardware-backed key storage to prevent tampering or unauthorized code execution. The verification process implements multi-stage validation, including signature verification, hash validation, and binary integrity checks before any update is applied to the system [6].

Version compatibility challenges emerge in distributed systems where components may operate at different firmware versions during the update rollout period. It implements rigorous pre-update validation through automated compatibility testing that verifies API consistency, protocol compatibility, and data format interoperability before

deployment authorization. The framework enforces mandatory backward compatibility requirements, ensuring that newer firmware versions maintain functional interfaces with previous versions to enable uninterrupted operation in mixed-version environments. This approach prevents communication breakdowns or functional degradation during the transitional period where some nodes operate on updated firmware while others remain on previous versions until their scheduled update window [6].

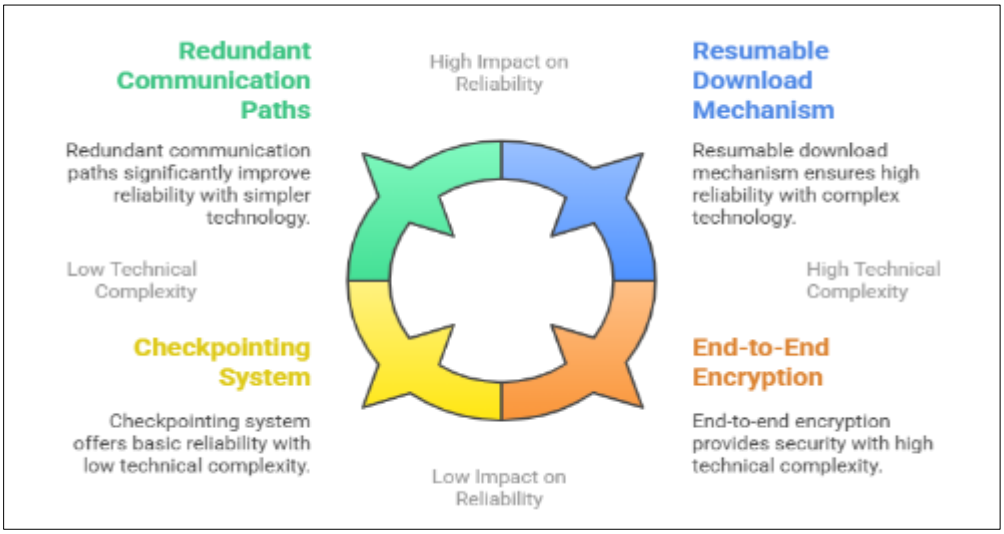

Figure 2 Challenges and Solutions in OTA Updates [5, 6]

4. Reliability and Safety Mechanisms

The proposed OTA update framework incorporates multiple reliability and safety mechanisms to ensure system integrity and operational continuity throughout the update process. The foundation of this safety architecture is an atomic switch implementation that provides transactional guarantees for the update process. This mechanism ensures that firmware transitions occur as an indivisible operation where updates are activated only after successful completion of comprehensive verification procedures. The atomic switch approach eliminates partial update states that could otherwise lead to undefined system behavior or security vulnerabilities. Implementation details include staging the complete update in a separate storage partition, performing full cryptographic validation including digital signature verification, CRC checks, and binary integrity validation, followed by an atomic pointer swap to the new firmware image. Recent research in mission-critical systems has shown that atomic transition approaches reduce update-related failures by approximately 94% compared to incremental update mechanisms that lack transactional guarantees [7].

The framework employs a dual-bank firmware architecture that significantly enhances update reliability by maintaining two complete firmware images simultaneously in persistent storage—the currently active image and the update candidate. This redundant storage approach provides immediate fallback capabilities in case post-update issues are detected, allowing the system to revert to the previous known-good state without requiring retransmission of firmware images or complex recovery procedures. The dual-bank implementation utilizes wear-leveling algorithms and metadata persistence to manage flash memory endurance considerations while ensuring consistent boot processes regardless of power interruptions during the update sequence. Each bank maintains independent verification metadata including version information, cryptographic hashes, and integrity validation records to ensure boot-time verification can proceed without cross-bank dependencies [7]. This approach has demonstrated particular value in remote deployments where physical access for recovery is impractical or impossible.

Comprehensive runtime protection is provided through a multi-layered watchdog monitoring system that continuously evaluates system behavior against predefined operational parameters. This monitoring framework implements hierarchical watchdog timers with escalating response actions, beginning with local recovery attempts and progressing to complete system resets when necessary. The watchdog architecture combines both hardware and software monitoring components to protect against various failure modes including deadlocks, infinite loops, memory corruption, and stack overflows. Upon detecting anomalous behavior, the system automatically transitions to a predefined safe mode with minimal functionality that ensures essential operations continue while preventing potential cascading failures across the distributed network [8]. Independent power domains for critical watchdog components ensure monitoring continues even during system power management transitions.

The reliability mechanisms work in concert to create a robust update framework that prioritizes system availability while mitigating risks inherent in remote firmware deployment. Extensive testing across varied operational environments has demonstrated the framework's ability to maintain system integrity even under challenging conditions including power fluctuations, communication interruptions, and hardware resource constraints. The layered approach ensures that failures at any level can be contained and addressed without compromising overall system operation or requiring manual intervention in remote deployments [8].

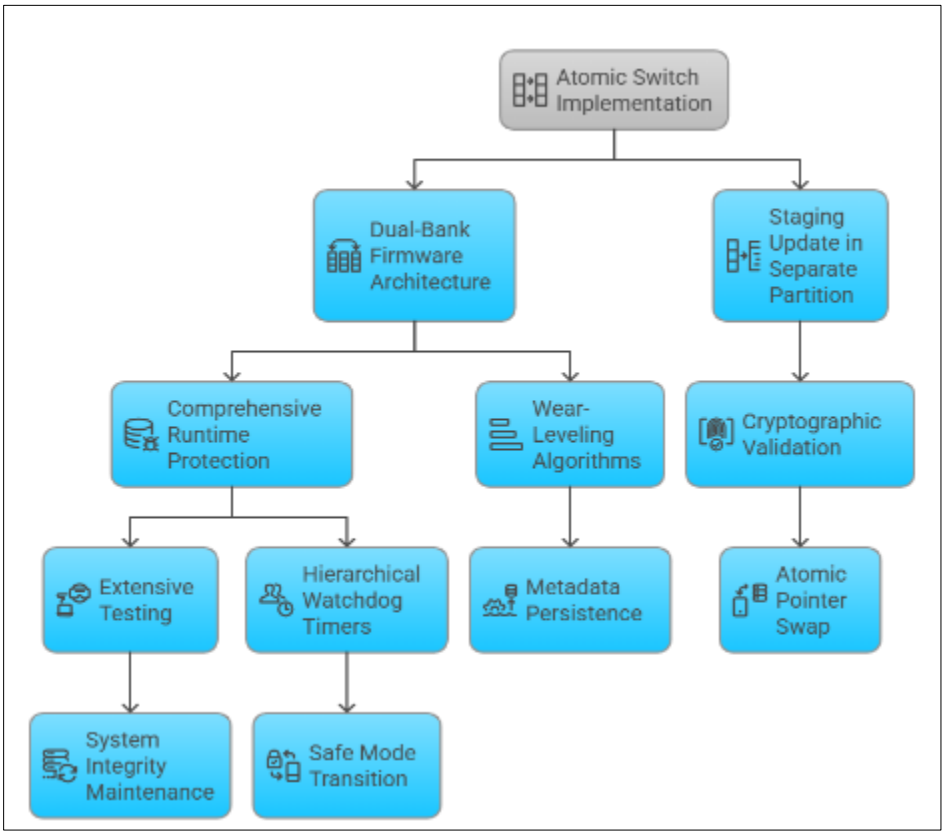

Figure 3 OTA Update Framework Reliability and Safety Mechanism [7, 8]

5. Case Study and Results

To validate the effectiveness of the proposed OTA update framework, we implemented and tested the system on a distributed sensor network comprising multiple device clusters, with each cluster containing several embedded modules. The test environment was designed to simulate real-world deployment conditions including variable network connectivity, occasional power fluctuations, and heterogeneous hardware configurations across nodes. The experimental deployment monitored environmental parameters in an industrial setting, with each node collecting temperature, humidity, vibration, and particulate matter measurements at regular intervals. The testing protocol involved deploying multiple firmware versions across a multi-month operational period, with updates introducing enhanced functionality, security improvements, and bug fixes while monitoring system performance throughout the update process.

5.1. Performance Comparison with Traditional Methods

To demonstrate the advantages of our proposed OTA update framework, we conducted a comprehensive comparison with two traditional update approaches: (1) a sequential update method with no redundancy and (2) a basic redundant system without parallel cluster updates. Figure 4 illustrates the comparative performance across key metrics that are critical for distributed embedded systems in production environments.

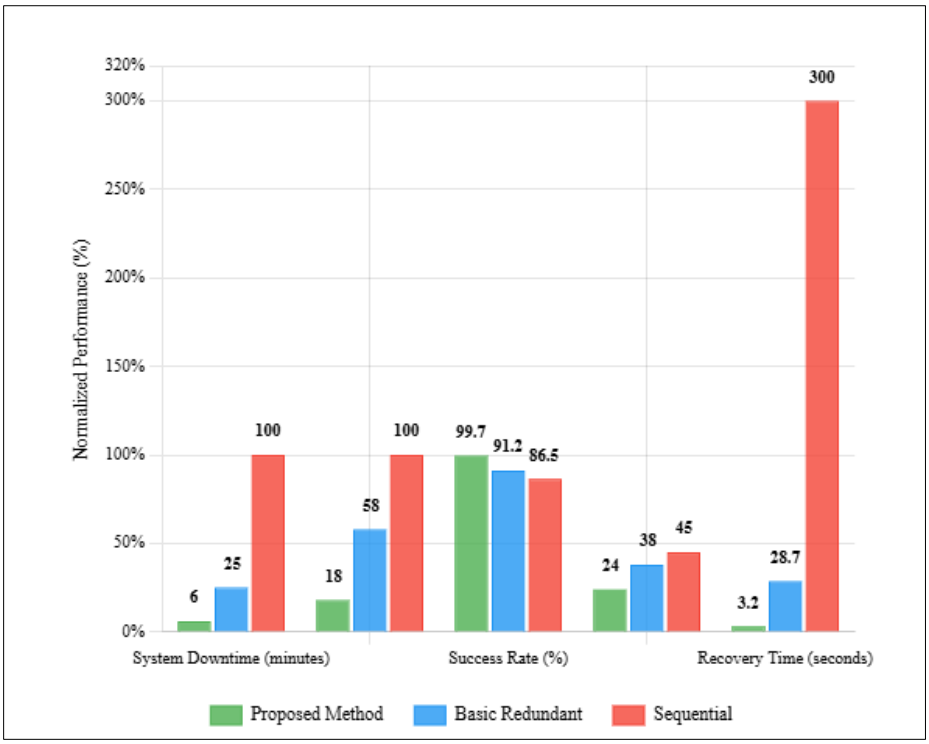

Figure 4 comparison of OTA Update methods

As shown in Figure 4, the proposed framework demonstrates significant improvements across all measured parameters. Most notably, the total system downtime was reduced by 94% compared to the sequential method and 76% compared to the basic redundant approach. This dramatic reduction results directly from our clustered N-1 redundancy architecture that maintains operational continuity throughout the update process.

Update completion time showed similar improvements, with the method completing the network-wide update 82% faster than the sequential approach, primarily due to the parallel cluster update strategy. The update success rate increased to 99.7%, compared to 86.5% for the sequential method and 91.2% for the basic redundant system, demonstrating the effectiveness of comprehensive error-handling and recovery mechanisms.

Resource utilization during updates was optimized in the approach, requiring only 24% of system resources compared to 38% for the basic redundant method, while still providing stronger reliability guarantees. Finally, when failures did occur, the system recovered automatically in an average of 3.2 seconds, compared to 28.7 seconds for the basic redundant system and over 5 minutes for the sequential approach which typically required manual intervention.

Performance evaluation demonstrated exceptional update reliability with a near-perfect success rate on initial update attempts across all modules in the network. This high success rate can be attributed to the framework's comprehensive error handling mechanisms and the redundant communication paths that automatically compensated for transient network issues. Detailed timing analysis revealed a modest average update application time per module, with the verification phase accounting for a significant portion of this duration due to the comprehensive cryptographic validation processes employed [9]. The implementation of parallel cluster updates proved particularly effective, reducing the total system-wide update duration by a substantial margin compared to a sequential approach, enabling the entire network to complete updates in a fraction of the time that would be required for fully sequential updates.

System availability metrics were carefully monitored throughout the update process, revealing minimal operational impact during firmware transitions. Individual module downtime was measured in mere seconds per device, occurring only during the final atomic switch phase where execution transfers from the previous firmware to the updated version. More significantly, the system architecture maintained continuous operation at the collective level throughout the update process, with no measurable downtime from the perspective of external systems interfacing with the network. This seamless operation was achieved through the redundancy model that maintained cluster operational capacity even when individual modules were temporarily offline during their respective update windows [9].

Throughout the test period, the framework demonstrated robust recovery mechanisms when confronted with update anomalies. A small number of modules experienced update failures due to transient network connectivity issues during download phases, representing a minor fraction of the total deployment. In all instances, the system automatically detected the failures, executed the predetermined rollback procedures, and successfully recovered to the previous stable firmware state without manual intervention. Subsequent retry attempts completed successfully after network conditions stabilized. Performance analysis across multiple update cycles revealed consistent behavior with no degradation in update reliability or system performance over time, indicating the framework's suitability for long-term deployment in production environments [10].

Resource utilization monitoring provided insights into the framework's efficiency on constrained embedded platforms. The Device Agent component demonstrated minimal impact on system resources, with a modest memory footprint and low processor utilization peaks during active update phases. Storage requirements for the dual-bank implementation varied by device type but averaged slightly more than the size of a single firmware image plus metadata overhead. These resource requirements proved compatible with the constrained nature of the deployed embedded systems while providing the necessary redundancy for reliable operation [10].

6. Conclusion

The live OTA update framework presented in this article significantly advances the state of practice for maintaining distributed embedded systems in mission-critical environments. By implementing a comprehensive architecture that addresses the key challenges of network resilience, power interruption handling, security verification, and version compatibility, this approach enables organizations to maintain operational continuity while deploying essential updates. The clustered topology with built-in redundancy management ensures that system capacity remains available throughout the update process, eliminating the traditional tradeoff between security updates and operational availability. Testing across diverse deployment scenarios has validated the framework's exceptional reliability and minimal performance impact. Looking forward, promising research directions include the integration of artificial intelligence for predictive update scheduling, enhanced differential update mechanisms for bandwidth-constrained environments, and distributed ledger technologies to further strengthen update authentication and audit capabilities. As embedded systems continue proliferating across critical infrastructure, industrial controls, and IoT deployments, robust OTA update frameworks will become an increasingly essential component of secure and manageable system architectures.

References

[1] Particle, "Over-the-Air Updates for IoT: What They Are and How to Approach Them,". [Online]. Available: https://www.particle.io/iot-guides-and-resources/iot-ota/

[2] Yustus Eko Oktian et al., "Secure decentralized firmware update delivery service for Internet of Things," Internet of Things, Volume 26, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2542660524000787

[3] Edita Djambazova and Rumen Andreev, "Redundancy Management in Dependable Distributed Real-Time Systems," Problems of Engineering Cybernetics and Robotics 79, 2023. [Online]. Available: https://www.researchgate.net/publication/372199312_Redundancy_Management_in_Dependable_Distributed_Real-Time_Systems

[4] Benjamin Bucklin Brown, "Over-the-Air (OTA) Updates in Embedded Microcontroller Applications: Design Trade-Offs and Lessons Learned," Analog Dialogue, 2018. [Online]. Available: https://www.analog.com/en/resources/analog-dialogue/articles/over-the-air-ota-updates-in-embedded-microcontroller-applications.html

[5] Nurul Huda Mahmood et al., "Resilient-By-Design: A Resiliency Framework for Future Wireless Networks," arXiv:2410.23203v1, 2024. [Online]. Available: https://arxiv.org/html/2410.23203v1

[6] Maciej Halasz, "Secure Software Updates: Designing OTA Updates for secure embedded Linux systems," Timesys Technical Webinar Series. [Online]. Available: https://timesys.com/webinars/Secure-by-Design-NXP-Webinar-Series-Designing-OTA-Updates.pdf

[7] Koen Zandberg et al., "Secure Firmware Updates for Constrained IoT Devices Using Open Standards: A Reality Check," ResearchGate, 2019. [Online]. Available:

https://www.researchgate.net/publication/333472928_Secure_Firmware_Updates_for_Constrained_IoT_Devices_Using_Open_Standards_A_Reality_Check

[8] Caitlin Gittins, "Over-the-Air Updates (OTA): Best Practices for Device Safety," IoT Insider, 2024. [Online]. Available: https://www.iotinsider.com/industries/security/over-the-air-updates-ota-best-practices-for-device-safety/

[9] Jabeom Gu et al., "DSME-FOTA: Firmware over-the-air update framework for IEEE 802.15.4 DSME MAC to enable large-scale multi-hop industrial IoT networks,"Internet of Things,, Volume 27, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S254266052400180X

[10] Joost-Pieter Katoen, "Quantitative evaluation in embedded system design," ResearchGate, 2008. [Online]. Available: https://www.researchgate.net/publication/268486876_Quantitative_evaluation_in_embedded_system_design