



Review

Architectures and strategies for low power optimization in the ESP32 ecosystem: A systematic review of the literature

Arquitecturas y estrategias para la optimización del bajo consumo en el ecosistema ESP32: una revisión sistemática de la literatura

Josué Abraham Emanuel Chambilla Zúñiga 

Universidad Privada de Tacna, Tacna, Perú

Autor de correspondencia: Josué Chambilla Zúñiga, Universidad Privada de Tacna, Tacna, Perú.
Correo electrónico: jaechambilla@upt.pe. ORCID: 0009-0000-5080-841X.

Recibido: 14 de julio del 2026

Aceptado: 30 de septiembre del 2026

Publicado: 2 de octubre del 2026

Abstract. -The objective is to identify the optimal configurations and strategies for low power optimization in the ESP32 series microcontroller ecosystem, addressing the need to balance computing power with energy autonomy in autonomous systems. The methodology employed consisted of a structured analysis based on the PRISMA (Preferred Reporting Items for Systematic Reviews and Meta-Analyses) protocol, selecting twenty-eight technical studies. Key findings reveal that efficiency depends not only on deep sleep states but also on a comprehensive, hardware-sensitive co-design paradigm. The reviewed studies demonstrate that the use of native compiled languages achieves execution speeds up to fifteen times faster than interpreted options, while the application of eight-bit quantization techniques reports reductions of up to seventy-three percent in memory footprint. Furthermore, the hybridization of short- and long-range communication protocols optimizes operational resilience in remote environments. It is concluded that these strategies transform the device from a prototyping tool into a robust scientific instrument for edge artificial intelligence. Practical implications include the development of medical biosensors with months of autonomy and low-cost industrial monitoring networks, democratizing access to advanced digital technologies in critical sectors and regions with limited infrastructure.

Keywords: ESP32; Low power; Embedded systems; Hardware and software co-design; Internet of Things.

Resumen. -El objetivo es identificar las configuraciones y estrategias óptimas para la optimización de bajo consumo en el ecosistema de microcontroladores de la serie ESP32, abordando la necesidad de equilibrar la potencia de cálculo con la autonomía energética en sistemas autónomos. La metodología empleada consistió en un análisis estructurado basado en el protocolo PRISMA (Elementos de informe preferidos para revisiones sistemáticas y metaanálisis), seleccionando veintiocho estudios técnicos. Los hallazgos clave revelan que la eficiencia depende no solo de los estados de bajo consumo, sino también de un paradigma de codiseño integral y sensible al hardware. Los estudios revisados demuestran que el uso de lenguajes compilados nativos alcanza velocidades de ejecución hasta quince veces mayores que las opciones interpretadas, mientras que la aplicación de técnicas de cuantización de ocho bits reporta reducciones de hasta un setenta y tres por ciento en la huella de memoria. Además, la hibridación de protocolos de comunicación de corto y largo alcance optimiza la resiliencia operativa en entornos remotos. Se concluye que estas estrategias transforman el dispositivo de una herramienta de prototipado en un instrumento científico robusto para la inteligencia artificial en el borde. Las implicaciones prácticas incluyen el desarrollo de biosensores médicos con meses de autonomía y redes de monitorización industrial de bajo coste, democratizando el acceso a tecnologías digitales avanzadas en sectores críticos y regiones con infraestructura limitada.

Palabras Clave: ESP32; Bajo consumo; Sistemas embebidos; Codiseño de hardware y software; Internet de las cosas.



1. Introduction

The massive deployment of the Internet of Things (IoT) has transformed sectors such as precision agriculture, healthcare monitoring, and industrial automation, demanding solutions that balance computing power, connectivity, and energy autonomy. In this context, the ESP32 ecosystem has established itself as a leading platform due to its dual-core architecture, native Wi-Fi/Bluetooth support, and superior cost-efficiency compared to traditional microcontrollers and single-board computers like the Raspberry Pi [1]. However, integrating complex tasks at the edge, such as Edge AI, imposes critical design challenges due to hardware constraints, primarily centered on the standard 520 KB SRAM limit [2].

The viability of autonomous sensor nodes depends strictly on optimizing energy consumption, since wireless communication and intensive processing represent the biggest sources of battery drain [3]. Although the ESP32 offers extreme power-saving modes such as deep sleep, capable of reducing consumption to just 10 μ A, achieving autonomy of months or years requires an approach that goes beyond simply managing sleep states [4]. Recent literature suggests that real optimization arises from a hardware-firmware co-design chain, where the choice of specialized variants (such as the ESP32-S3 for AI acceleration) and the use of compiled over interpreted languages are critical to minimizing active CPU time [5].

With the emergence of the TinyML paradigm, the ecosystem has evolved toward local inference execution to preserve privacy and reduce network latency [6]. However, deploying models such as convolutional neural networks (CNNs) or LSTMs on limited hardware necessitates the implementation of aggressive compression strategies, such as INT8 quantization, which allows for massive reductions in memory footprint [2]. Despite the abundance of experimental studies, there is a fragmentation in the literature regarding which are the optimal configurations that balance metrological accuracy with resource cost under the concept of Pareto fronts [7]. Furthermore, existing literature reviews on low power embedded systems and TinyML primarily offer broad, platform-agnostic overviews. These generic surveys often fail to address the low-level hardware-firmware interactions, language execution overheads, and severe memory constraints such as the native 520 KB SRAM limitation inherent to specific microcontroller families. This study extends current knowledge by delivering a systematic, ESP32-centric review. It critically analyzes the interplay between hardware variants (e.g., Xtensa LX6, RISC-V C3, and AI-accelerated S3), native versus interpreted execution environments, and model quantization strategies, thereby providing a targeted framework for continuous edge operation.

This systematic literature review aims to synthesize and critically analyze the low power optimization architectures and strategies documented between 2021 and 2026. Using a standardized protocol based on the PRISMA flowchart, 28 studies providing verifiable technical evidence were selected. The research contributes a technological taxonomy encompassing hardware variants and firmware frameworks, as well as hybrid communication protocols and AI compression techniques. The document is structured in sections detailing the methodology employed, the development of the state of the art, a multi-objective resource analysis, and a discussion of current knowledge gaps.

2. Review Methodology

The methods applied and the information collected are summarized in Table 1, which provides a structured classification of the 56 selected articles by application area and technological axis. This categorization allows for article-by-article traceability and serves as the basis for subsequent comparative analysis.



2.1. Scope, sources and period

The review focused on scientific literature indexed in Scopus and IEEE Xplore, published between 2021 and 2026. The search of these databases ended on July 2, 2026, and focused on hardware architectures, firmware optimization, and the execution of low power artificial intelligence models in the ESP32 ecosystem. As shown in Figure 1, a PRISMA flowchart was used to document the identification, selection, eligibility, and inclusion stages, thus ensuring the transparency and reproducibility of the process. The initial search identified 56 records. After removing duplicates, 50 remained. Of these, 11 were excluded during the title and abstract review, leaving 39 articles for full-text evaluation. Finally, 11 were excluded for not meeting the eligibility criteria, resulting in a final dataset of 28 studies included in the review.

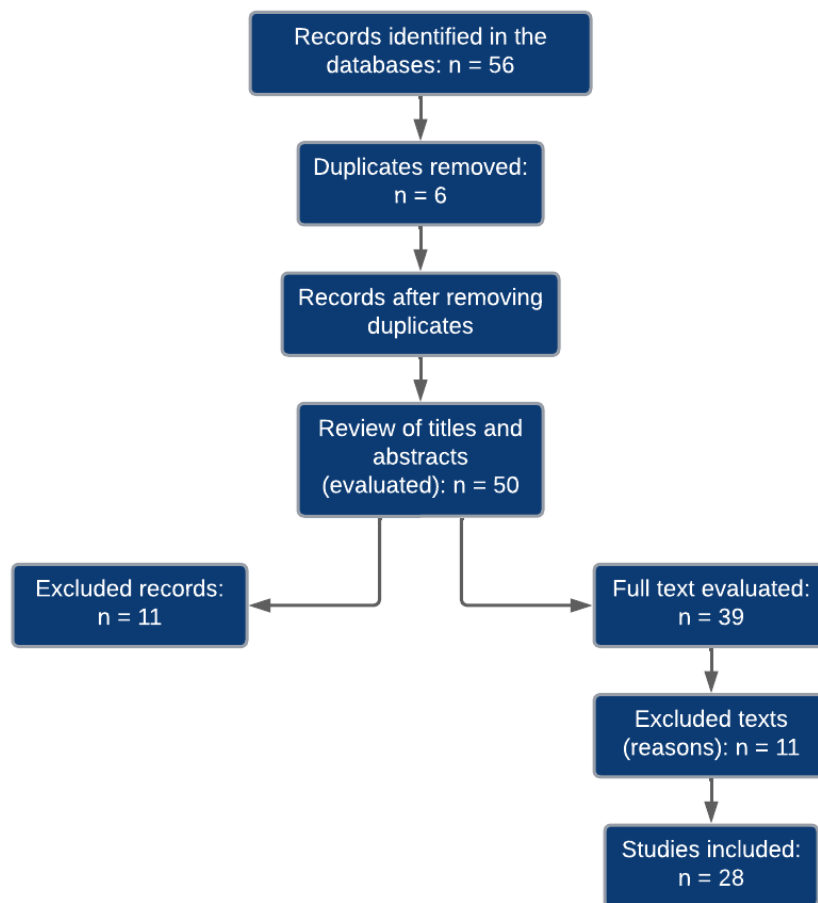


Figure 1. PRISMA diagram of the search and selection process for studies published between 2021 and 2026.

Following the consolidation of the final dataset using the PRISMA protocol, the twenty-eight selected studies were organized into technical categories, as shown in Table 1, based on their technological axes and areas of application. This methodological classification ensures traceability on a case-by-case basis and forms the basis for comparative analysis, allowing for the identification of cross-cutting trends in low consumption optimization and critical gaps. To synthesize the core findings of this systematic literature review, Figure 2 illustrates the cross-cutting trends identified across the analyzed literature, mapping the relationships between ESP32 hardware variants, firmware and TinyML optimization strategies, and key application domains.

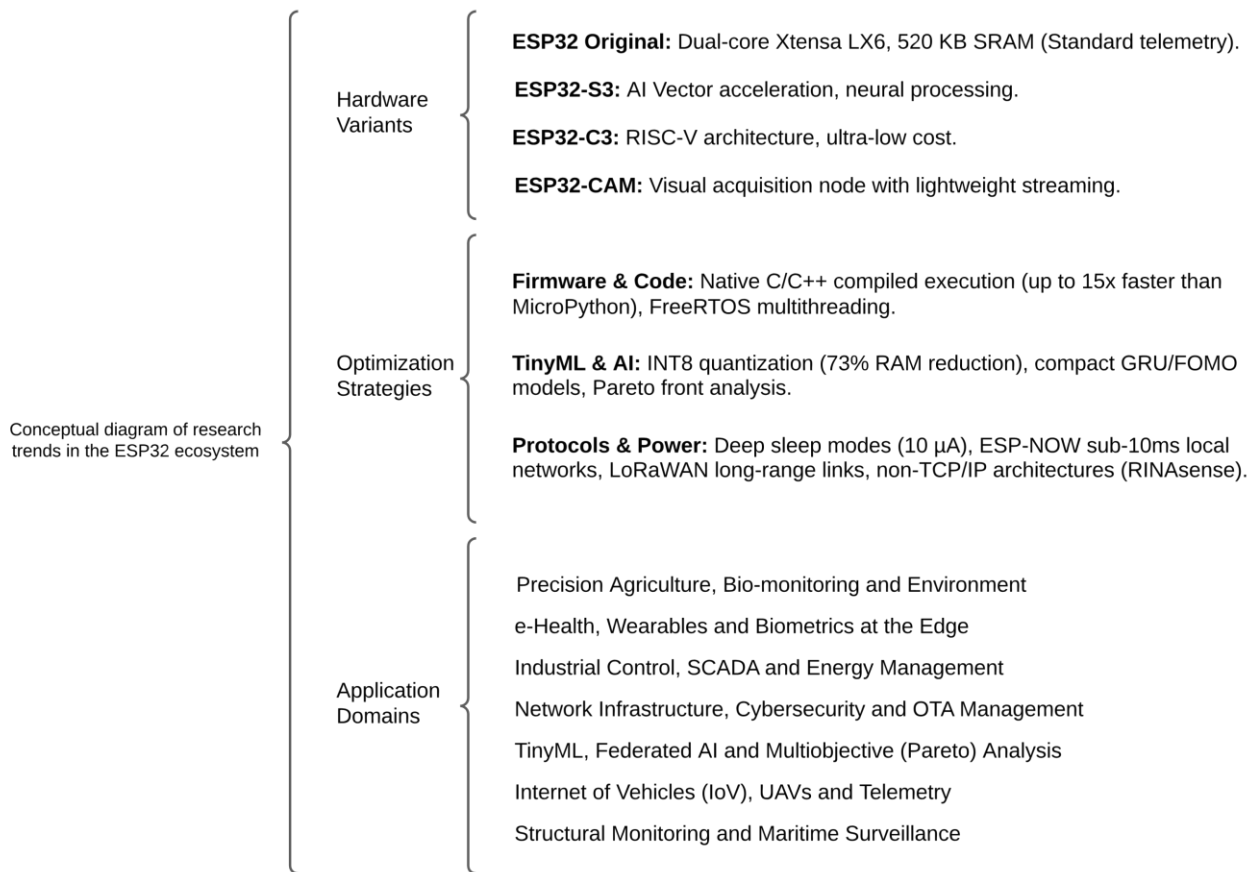


Figure 2. Conceptual diagram of research trends in the ESP32 ecosystem: Hardware variants, optimization strategies, and application domains.

Table 1. Classification of the 28 articles found.

Category	Nº	References
Precision Agriculture, Bio-monitoring and Environment	6	[1], [3], [8], [9], [10], [11], [12]
e-Health, Wearables and Biometrics at the Edge	6	[2], [4], [6], [13], [14], [15]
Industrial Control, SCADA and Energy Management	5	[16], [17], [18], [19], [20]
Network Infrastructure, Cybersecurity and OTA Management	4	[21], [22], [23], [24]
TinyML, Federated AI and Multiobjective (Pareto) Analysis	3	[5], [7], [25]
Internet of Vehicles (IoV), UAVs and Telemetry	2	[26], [27]
Structural Monitoring and Maritime Surveillance	2	[9], [28]



2.2. Search strategy and inclusion/exclusion criteria

The search strategy consisted of applying Boolean combinations to the title, abstract, and keyword fields, restricting the search to the ESP32 microcontroller ecosystem (including its S3 and C3 variants) and to low power or energy-efficient optimization strategies. The focus was specifically on hardware design, software architectures, and frameworks that enable the implementation of power-saving modes such as deep sleep. The representative query was:

("ESP32" OR "ESP-32" OR "ESP32-S3" OR "ESP32-C3") AND ("low power" OR "energy efficiency" OR "power consumption" OR "deep sleep") AND ("architecture" OR "software architecture" OR "hardware design" OR "framework")

The search strategy consisted of applying Boolean combinations to the title, abstract, and keyword fields, restricting itself to the ESP32 microcontroller ecosystem (including the S3 and C3 variants) and to low power or energy-efficient optimization strategies [21]. Filters were applied based on publication period (2021-2026) and document type. Inclusion criteria required technical evidence from telemetry and experimental validation, either under laboratory or field conditions, to ensure the practical relevance of the architectural proposals [2], [8]. Representative examples include works that integrate FreeRTOS-based software architectures for multithreaded task distribution [27], the use of TinyML frameworks such as TensorFlow Lite Micro and Edge Impulse for edge inference [2], and the hardware-firmware design to optimize extreme power-saving modes [13], [20]. Studies were included that explicitly addressed optimization strategies using low power communication protocols such as ESP-NOW, BLE (including the FragCast framework for scalable OTA updates) and LoRaWAN [21], [22].

In addition, studies were required to report validations supported by measurable metrics, such as RMS current in milliamperes (mA) [22], energy consumption per inference cycle in Joules (0.154 J) [6], processing latency in milliseconds (ms) [13] and RAM footprint in kilobytes (KB) [5]. Notable examples include deep sleep optimizations that reduce power consumption to 10 μ A [16], [20], the use of Pareto fronts to balance model accuracy against resource cost [23] and reductions of up to 82% in consumption through co-design of hardware and firmware [24].

The exclusion criteria ruled out studies on hardware unrelated to the ESP32 ecosystem, industrial power mechanical or electrical engineering research without a focus on embedded systems, and purely theoretical proposals lacking verifiable efficiency metrics. The selection process and data extraction were conducted independently by a single researcher using a standardized, pre-defined screening protocol to minimize selection bias. Study quality and methodological rigor were systematically evaluated based on explicit criteria, including the presence of verifiable empirical power consumption metrics, hardware/firmware reproducibility, and experimental validation under controlled laboratory or field conditions.

2.3. Data extraction and quality assessment

To ensure comparability between heterogeneous studies on ESP32 ecosystem optimization, a standardized extraction protocol was defined, encompassing a consistent set of variables systematically collected from each article. These variables included the specific microcontroller variant (original ESP32, S3, or C3) with its SRAM characteristics (520 KB) and clock frequency (240 MHz) [2], [7], the type of software architecture used (FreeRTOS, Arduino IDE or ESP-IDF) [3], [10], [20] and the programming language used, contrasting the efficiency of native C/C++ versus MicroPython [15].



In addition, TinyML frameworks and deployment platforms such as TensorFlow Lite Micro, Edge Impulse, and EON Compiler were documented [1], [9]. The energy dimension was evaluated in terms of current consumption in various states, highlighting deep sleep (10 μ A) and light sleep (0.927 mA) [13], [20], as well as the energy per operating cycle measured in Joules (0.154 J by inference) or electrical charge in mAs [6]. The deployment environment was classified into precision agriculture, industrial monitoring, wearable health, and maritime surveillance [8]. The validation protocols were classified according to laboratory comparisons with high-precision instrumentation (Oti Arc Pro, Power Profiler Kit 2) or extended field trials [20], [23].

Operational performance was measured using inference latency (ms), memory footprint (Peak RAM and Flash in KB) and data throughput (Throughput in Mbps) [24]. Also recorded were hardware implementation costs, generally below \$60 USD, and the use of multi-objective analysis to identify Pareto-front configurations that balance accuracy and resource cost [5], [7], [15]. Table 2 of the review provides the operational definitions of these variables, allowing traceability between lightweight AI models and low power architectures.

Table 2. Variables and operational definitions for extraction and synthesis.

Dimension	Variable	Operational Definition	Unit of Measurement / Category	Examples of Evidence in the Sources
System Architecture	MCU variant	Specific Espressif SoC model used.	ESP32 (Xtensa), S3 (AI Acceleration), C3 (RISC-V)	The use of ESP32-S3 for vision tasks or predictive maintenance with neuron acceleration [10], [14], [23].
	Framework / OS	Real-time development environment and operating system.	ESP-IDF, Arduino IDE, FreeRTOS	Using FreeRTOS for multithreaded management in competition telemetry [5], [27]
	Language	Firmware base programming language.	C, C++, MicroPython	Efficiency comparison where C++ outperforms MicroPython in execution time of critical tasks [1], [15].
Energy Dimension	Resting Current	Electricity consumption in low power states.	Deep Sleep (μ A), Light Sleep (mA)	Extreme optimization of 10 μ A in deep sleep mode for autonomous sensors [16], [20], [24].
	Energy Cost per Cycle	Total energy required to complete a task or operation.	Joules (J), mAs	Complete biometric inference consuming 0.154 J or 29.8 mJ of data transmissions [2], [5].
Quality and Robustness	Precision Metrics	Degree of adjustment and error in predictions or measurements.	RMSE, MAE, R^2 , F1-Score	Calibration of sensors with R^2 between 0.95 and 0.97 in precision agriculture [8], [11], [25].
	RF Link Performance	Reliability and strength of wireless communication.	RSSI (dBm), PDR (%), SNR	LoRa and ESP-NOW transmissions with a PDR > 95% in real-world deployments [4], [20].
Optimization Strategy	Technical Method	Technique applied to improve system efficiency.	Pareto Front, INT8, Quantization	Pareto fronts are used to balance accuracy and latency, and INT8 quantization is used to reduce RAM usage [5], [9], [15].



These variables articulate five key comparative dimensions across the studies: system architecture (MCU variant, framework, and language), computational efficiency (inference latency and RAM/Flash memory footprint), energy feasibility (deep sleep consumption, projected autonomy, and energy per cycle), quality and robustness (metrological accuracy using R^2 , RMSE, and link reliability using PDR), and optimization strategies (Pareto analysis and quantization). Taken together, this framework supports the identification of cross-cutting trends and methodological gaps in the ESP32 ecosystem literature.

3. Development

3.1. Hardware Architectures and MCU Variants

1. *Original ESP32 (WROOM/WROVER Series)*: Based on the 240 MHz dual-core Xtensa LX6 processor, it remains the standard for telemetry and general monitoring applications [2]. It stands out for its 520 KB SRAM, which, although superior to other microcontrollers in its class, acts as a critical bottleneck when attempting to deploy complex AI models [2], [7]. Its main advantage for low power consumption is the Deep Sleep mode, capable of reducing the current to 10 μ A while maintaining the ability to wake up via external interruptions [4].
2. *ESP32-C3 (RISC-V Architecture)*: Less powerful than previous versions, but selected in studies that prioritize ultra-low cost and simplicity [15]. Its use is mainly reported in sensor nodes that do not require intensive processing but do require robust Wi-Fi/BLE connectivity [22].
3. *ESP32-CAM*: A specific, widely documented implementation for visual monitoring where research suggests that, due to memory limitations, it is more efficient to use it as a lightweight acquisition node that streams images for centralized processing, rather than performing heavy computer vision locally [4], [8].

3.2. Software Strategies and Frameworks

1. *Compiled vs. Interpreted Languages (C/C++ vs. MicroPython)*: There is a critical performance gap between using native C/C++ (via ESP-IDF or Arduino IDE) and high-level languages like MicroPython. While MicroPython facilitates rapid prototyping and memory management through garbage collection, C++ implementations have proven to be up to 15 times faster in real-time processing tasks [15]. Manual memory control in C allows for low level optimizations that minimize computational overhead, which is vital in devices with only 520 KB of SRAM [11].
2. *FreeRTOS and Multithreaded Task Management*: The ESP32's dual-core architecture is leveraged by the FreeRTOS real-time operating system. This framework enables deterministic programming where critical tasks (such as sensor sampling or AI inference) are isolated from network operations, preventing system crashes and optimizing CPU duty cycle [3], [20], [23]. Studies in electric vehicle telemetry demonstrate that the use of queues and traffic lights in FreeRTOS guarantees over 95% accuracy in data acquisition under stress conditions [27].
3. *Specialized Energy Management Modules*: Advanced strategies propose the development of firmware based on hardware-software co-design [25]. A notable example is the use of custom Battery Manager modules that coordinate deep sleep states with hardware interrupts, achieving 82% reductions in energy consumption in environmental monitoring applications [24].



4. *Protocols and Scalability in the Cloud*: Integration with IoT platforms such as Blynk, Ubidots, Node-RED and MQTT enables efficient and scalable remote monitoring [19][20]. For fleet maintenance, the FragCast framework introduces a scalable methodology for OTA (Over-The-Air) updates over BLE, overcoming the latency limitations of traditional sequential updates [21].

3.3. Optimization in TinyML and AI at the Edge

1. *Compact Models and Specific Architectures*: A transition towards architectures designed from the ground up for the edge is evident. The use of Gated Recurrent Units (GRUs) has become established compared to traditional LSTMs, achieving a 33% reduction in parameters and up to a six-fold improvement in inference speed for Human Activity Recognition (HAR) and Wi-Fi Signal Processing (CSI) tasks [14]. In computer vision, the FOMO (Faster Objects, More Objects) architecture is preferred over YOLO in microcontrollers due to its minimal memory footprint and low latency, ideal for people counting or pest detection [12].

2. *Quantization Techniques (INT8)*: Post-training quantization (PTQ) is the most common compression strategy in the sources [11]. Converting float32 weights to INT8 has resulted in drastic reductions in RAM usage (up to 73%) and significant improvements in latency without critically degrading accuracy [14]. For example, in EEG seizure detection tasks, quantization allowed the inference time to be reduced from 83.55 ms to just 15 ms [2].

3. *Energy Efficiency and Autonomy*: Low power consumption is achieved through an activation hierarchy. Studies show that using lightweight presence detection filters as a "wake-up mechanism" allows the ESP32 to remain in power-saving modes most of the time. In multimodal biometric systems, an energy cost of only 0.15 J per inference cycle has been validated, projecting an autonomy of up to 46 months with standard 600 mAh batteries under real-world usage cycles [6].

3.4. Low Power Communication Protocols

1. *ESP-NOW for Ultra Low Power Local Area Networks*: Espressif's proprietary peer-to-peer protocol stands out for its ability to perform instant data exchanges without requiring a Wi-Fi connection, drastically reducing wake-up times and, consequently, power consumption. Studies report sub-10 ms latencies in packet delivery, outperforming other local solutions in efficiency [8].

2. *LoRa and LoRaWAN for Long Range*: For deployments requiring coverage of several kilometers (up to 4.5 km in open field), LoRa is consolidating its position as the preferred technology due to its high noise immunity and its operation in sub-GHz bands (868/915 MHz), which offer better penetration than Wi-Fi [12], [20].

3. *Optimization in the Application Layer (MQTT and CoAP)*: Efficiency is also managed through lightweight messaging protocols. MQTT is the standard for real-time telemetry due to its low overhead, while CoAP is used for event-driven messaging on extremely limited networks [1].

In order to consolidate these technical findings in a structured way, Table 3 presents the evidence synthesis and optimization matrix, integrating in a multidimensional manner the variables of computational efficiency (latency and memory), energy viability (consumption at rest and per cycle) and robustness of the system.

**Table 3.** Evidence synthesis and optimization matrix in the ESP32 ecosystem.

Reference	MCU variant	Optimization Strategy	Computational Efficiency	Energy Dimension	Quality and Robustness
[23]	ESP32-S3	INT8 quantification in predictive maintenance	Latency: 42.3 ms	CPU: 53% during inference	High accuracy in faults
[6]	ESP32	TinyML and interrupt-triggered architecture	Multimodal biometric processing	10 μ A (Deep Sleep); 0.154 J per cycle	Autonomy: 46 months
[8]	ESP32 / ESP8266	ESP-NOW / LoRa hybrid network	ESP-NOW latency: 4.4 ms	Quiescent current: 0.01 A	PDR > 95%; calibration with R2 0.95-0.97
[13]	ESP32-S3	CNN-LSTM models (float32)	Latency: 113.6ms; RAM: 199.5KB	Inference: 86.5 mA; Light sleep: 0.927 mA	Accuracy: 99.3% in fall detection
[3]	ESP32	TX event energy analysis	Environmental data transmission	Consumption: 29.8 mJ per transmission event	Stability in 100 experimental samples
[7]	ESP32-S3	Multi-objective analysis (Pareto)	Comparison of MLR, MLP and CNN	Resource cost-based optimization	Identifying optimal accuracy-latency configurations
[18]	ESP32	WiMapper Framework (Kernel-based)	Latency: 0.144ms; Peak RAM: 3.06KB	Algorithmic footprint: 188.5 KB	100% Accuracy in Rogue AP Detection
[2]	ESP32	Automated Pipeline for EEG	Latency: 15 ms (seizure) to 437 ms (emotion)	RAM optimized using EON Compiler	92.6% Accuracy in 3-class classification
[24]	ESP32	Hardware-firmware co-design (RINA sense)	Native RINA architecture	82% reduction in total consumption	Average RTT latency of 2.05 ms
[15]	ESP32	Language comparison (C vs MP)	C++ is 15 times faster than MicroPython	RAM D1 memory usage: 6.5 KB (C++)	Validation in real-time HAR systems
[25]	ESP32	FedGen-Sensor (Generative AI)	Collaborative data reconstruction	INT8 drastically reduces the model size	RMSE and MAE as loyalty metrics



[21]	ESP32-S3	FragCast Framework (BLE OTA)	Constant discovery latency (N=100)	Efficiency measured in mAs (electrical charge)	Proven scalability in massive IoT fleets
[19]	ESP32	Edge filtering and dual data architecture (SD/Wi-Fi)	Synchronous sampling every 60s; median filters	Average consumption: 0.50 W (12 Wh/day)	MAE: 0.29 °C; 0.0027 L/min
[17]	ESP32	Non-invasive polyphasic monitoring via LoRaWAN	Atmel M90E32AS Meter Control	LoRa communication up to 5 km	Error < 1% in voltage; 7% in current
[9]	ESP32-CAM	Visual monitoring of deformation	Centralized processing.	Node consumption: 1 W	Absolute error: 2.63 $\mu\epsilon$
[4]	ESP32	Patient safety management	Dynamic profile sampling	Deep Sleep: 10 μA	TX/RX error checking
[22]	ESP32-S3	Wireless platform selection	LASSO regression for energy	RMS current measurement	Improvement in P95 metrics
[26]	ESP32-CAM	UAV data collection	Remote server processing	Active consumption: 1,360 W	End-to-end latency: 11,032 ms
[16]	ESP32-WROOM	Smart Solar Cold Box	MPPT management and telemetry	Deep Sleep: 10 μA	Autonomy: 36-42h (without sun)
[5]	ESP32	Federated anomaly detection	Inference: 3–4 ms	RAM: 12.7 KB (2.4% SRAM)	F-Score stable in 24h
[10]	ESP32-S3	Pest detection by sound	Composite scaling (EfficientNet)	Efficiency score: 4.88 (GRU)	Precision-resources balance
[14]	ESP32	STAR Framework for HAR (Wi-Fi CSI)	Latency: 312 to 427 ms	RAM reduced 73% (INT8)	Parameters reduced 33% (GRU)
[11]	ESP32	Soil moisture monitoring	PTQ INT8 Quantization	CPU cycle optimization	Calibration: R2 0.95-0.97
[27]	ESP32	Real-time IoV telemetry	Average latency < 200 ms	12% improvement in energy efficiency	PDR > 95% in competition
[12]	ESP32-CAM	Visual insect traps	Local inference (FOMO)	Total cycle: 5595 mA (63s)	Standby: 6 mA
[1]	ESP32	Hybrid smart greenhouse	LSTM model for harvests	Average consumption: 240 mA	Accuracy: 88%
[28]	ESP32-S3	Acoustic maritime surveillance	Two-node system (Edge/Wait)	ESP32-S3: 0.69 W continuous	Wake-up sensitive autonomy
[20]	ESP32-WROOM	SCADA for gas pipelines	Monitoring via Blynk IoT	Projected minimum consumption	Minimum voltage error (Voc 13V)



4. Discussion

The analyzed evidence demonstrates that low power optimization in the ESP32 ecosystem has evolved from simple standby states to a hardware-firmware co-design chain. Key findings indicate that the successful deployment of long-duration AI and telemetry relies on three key strategies: hardware specialization (using S3 and C3 variants), language efficiency (C++'s superiority over MicroPython), and aggressive model compression using INT8 quantization. Together, these strategies have enabled the ESP32 to transition from a prototyping platform to a robust component for extreme-edge AI with projected multi-year autonomy.

Unlike traditional approaches focused solely on radio energy savings, the studies analyzed highlight that firmware efficiency is the primary driver of energy conservation. The performance gap where C++ is 15 times faster than MicroPython explains why systems with longer battery life prefer compiled languages to minimize CPU duty cycle. Furthermore, conventional networking theory is challenged by architectures like RINAsense, which, by moving away from the traditional TCP/IP model and using recursive structures, achieves an 82% reduction in total power consumption. This suggests that much of the ESP32's energy expenditure is not inherent to the silicon itself, but rather to the overhead of standard protocols.

In the field of artificial intelligence, the results shift the notion that accuracy is the sole metric for success, validating the use of Pareto Fronts to identify optimal configurations. While the literature previously prioritized complex models like CNNs, current evidence shows that INT8 quantization techniques can reduce the memory footprint by 73%, enabling advanced models like CNN-LSTM to operate within the 520 KB of native SRAM without resorting to external PSRAM. This multi-objective optimization allows critical tasks, such as detecting industrial failures or seizures, to run with sub-second latencies (as low as 15 ms) while maintaining power consumption in the microampere range.

A methodological limitation of this review is its reliance on a single-reviewer screening process, which was mitigated through strict protocol adherence. Furthermore, a critical gap identified across the literature is the scarcity of field-validated studies addressing model domain shifts and on-device Explainable AI (XAI). Most state-of-the-art architectures demonstrate high accuracy under laboratory setups, but lack mechanisms for dynamic in-situ self-calibration when deployed in unpredictable real-world environments.

5. Conclusions

Low power optimization is not an isolated software event, but rather the result of a comprehensive "hardware-aware" co-design paradigm. The viability of long-term autonomous systems depends on a mandatory synergy between specialized hardware variants (such as the ESP32-S3 for AI acceleration), the use of native compiled languages like C++ (which reduces execution times by up to 15 times compared to MicroPython), and the application of aggressive compression techniques such as INT8 quantization, which allows for a 73% reduction in RAM footprint.

These findings transform the perception of the ESP32, elevating it to a scientific and commercial instrument capable of balancing precision with energy efficiency through the use of Pareto principles. In real world practice, these results have a direct impact on critical sectors. In healthcare, they enable the development of wearable biosensors for detecting seizures or falls with complete privacy through local processing and projected autonomy of up to 46 months. In precision agriculture, the validation of hybrid protocols such as ESP-NOW/LoRa facilitates the deployment of low-cost monitoring networks with operational resilience exceeding 95%. Similarly, in the industrial sector, a roadmap is



established for implementing sub-50 ms deterministic local diagnostics, eliminating dependence on constant cloud connectivity and reducing operating costs.

Ultimately, the primary contribution of this systematic literature review lies in providing researchers and system architects with a pragmatic, structured roadmap. By synthesizing a technical taxonomy, identifying critical operational variables, and establishing a multi-objective evidence matrix, this work bridges the gap between theoretical optimization and field execution. Addressing technical challenges such as Explainable AI (XAI) directly at the microcontroller level and in-situ self-calibration algorithms will mitigate the domain shift observed when transferring models from the laboratory to the real field, ensuring that the ESP32 ecosystem becomes the technological heart of a new generation of intelligent, resilient, and affordable sensors.

6. Declaration of Artificial Intelligence:

The artificial intelligence tools Gemini (Google) and Scopus AI (Elsevier) were used during the literature review stage of this study. Specifically, they were implemented to optimize search strings, summarize indexed search results, and identify emerging subtopics within the text corpus. The validity of the outputs and literature citations provided by the AI was manually verified by the authors against indexed databases to ensure the reliability and accuracy of the sources. The aspects of the study modified or organized by the tool were strictly limited to the literature classification matrix and the structure of the literature background; the AI tool did not participate in drafting, interpreting, or concluding any part of this manuscript.

References

- [1] V. V et al., «A Hybrid IoT and Machine Learning Framework for Smart Greenhouse Automation in Sustainable Agriculture», *Int. Res. J. Multidiscip. Technovation*, pp. 58-69, jul. 2025, doi: 10.54392/irjmt2545.
- [2] C. A. Popa, I. Dogaru, y R. Dogaru, «An End-to-End Automated Pipeline for EEG Classification on TinyML Platforms: From Signal to On-Device Inference», *IEEE Access*, vol. 14, pp. 8918-8937, 2026, doi: 10.1109/ACCESS.2026.3651653.
- [3] J. Tepmanee, «A Measurement-Based Study of Data Transmission Energy in Long-Term IoT Environmental», *Eng. Technol. Horiz.*, vol. 43, n.º 2, jun. 2026, doi: 10.55003/ETH.430207.
- [4] E. Arrieta Rodriguez et al., «A Platform for Inpatient Safety Management Based on IoT Technology», *Inventions*, vol. 7, n.º 4, p. 116, dic. 2022, doi: 10.3390/inventions7040116.
- [5] V. Ieropoulos, E. Anthi, T. Spyridopoulos, P. Burnap, P. Carnelli, y A. Khan, «Federated Detection at the Edge: Collaborative Anomaly Detection for Resource-Limited IoT», *IEEE Internet Things J.*, vol. 13, n.º 11, pp. 24205-24218, jun. 2026, doi: 10.1109/JIOT.2026.3674644.
- [6] J. A. De La Torre, F. Rincón, S. Escolar, A. Caruso, J. Caba, y J. Barba, «Extreme Edge Computing for Secure and Private Multimodal Biometric Identification in Intelligent IoT Systems», *Sensors*, vol. 26, n.º 12, p. 3756, jun. 2026, doi: 10.3390/s26123756.
- [7] P. Rojas-Carrasco y M. Guinaldo, «Multi-Objective Evaluation of Lightweight AI Models on Low-Cost Edge Devices Using Pareto Fronts», *Appl. Sci.*, vol. 16, n.º 8, p. 3679, abr. 2026, doi: 10.3390/app16083679.
- [8] A. Abdelli et al., «Field Validated Hybrid ESP-NOW and Long Range IoT Monitoring System for Energy Autonomous Precision Agriculture», *IEEE Access*, vol. 13, pp. 207511-207526, 2025, doi: 10.1109/ACCESS.2025.3641487.
- [9] A. K. Anim, W. Li, X. Zhao, J. Ma, R. Liu, y D. Sun, «Design and Laboratory Validation of a Low-Cost Vision-Based Strain Monitoring System Using ESP32-CAM with Centralized Processing», *Buildings*, vol. 16, n.º 9, p. 1681, abr. 2026, doi: 10.3390/buildings16091681.



- [10] M. Kazeneza, A. S. Bosman, D. K. Amenyedzi, D. Hanyurwimfura, E. Ndashimye, y A. Vodacek, «Balancing Complexity and Performance of Machine Learning Models for Avian Pests Sound Detection in Agricultural Environments», IEEE Access, vol. 13, pp. 105813-105827, 2025, doi: 10.1109/ACCESS.2025.3580620.
- [11] V.-K. Nguyen, V.-K. Tran, y C.-N. Nguyen, «Fruit Fly Detection Based on Wingbeat Sound Using Embedded Artificial Intelligence», Int. J. Technol., vol. 17, n.º 3, p. 1012, may 2026, doi: 10.14716/ijtech.v17i3.8284.
- [12] I. Saradopoulos, I. Potamitis, S. Ntalampiras, A. I. Konstantaras, y E. N. Antonidakis, «Edge Computing for Vision-Based, Urban-Insects Traps in the Context of Smart Cities», Sensors, vol. 22, n.º 5, p. 2006, mar. 2022, doi: 10.3390/s22052006.
- [13] L.-C. Duan, N.-N. Minh, T.-C. Dung, y T.-T.-T. Linh, «Energy-Efficient TinyML Approach for Wearable Fall Detection on Edge Devices Using Spatial-Temporal Deep Learning», Int. J. Technol., vol. 17, n.º 3, p. 919, may 2026, doi: 10.14716/ijtech.v17i3.8445.
- [14] K. Liu, Q. Zhao, R. Wang, Y. Lin, J. Yu, y S. J. Fong, «STAR: A Privacy-Preserving, Energy-Efficient Edge AI Framework for Human Activity Recognition via Wi-Fi CSI in Mobile and Pervasive Computing Environments», Sensors, vol. 26, n.º 12, p. 3692, jun. 2026, doi: 10.3390/s26123692.
- [15] H. Najeh, C. Lohr, y B. Leduc, «Real-Time Human Activity Recognition on Embedded Equipment: A Comparative Study», Appl. Sci., vol. 14, n.º 6, p. 2377, mar. 2024, doi: 10.3390/app14062377.
- [16] A. A. Gilani, S. I. Butt, S. H. I. Jaffery, A. Abid, y S. Ali, «Design, development and evaluation of solar powered DC cold box for sustainable supply chain of food and medicine», Int. J. Sustain. Eng., vol. 19, n.º 1, p. 2619276, dic. 2026, doi: 10.1080/19397038.2026.2619276.
- [17] N. E. Guevara, Y. H. Bolaños, J. P. Diago, y J. M. Segura, «Development of a low-cost IoT system based on LoRaWAN for monitoring variables related to electrical energy consumption in low voltage networks», HardwareX, vol. 12, p. e00330, oct. 2022, doi: 10.1016/j.ohx.2022.e00330.
- [18] A. H. Kumar et al., «WiMapper: A Lightweight Kernel-Based Framework for Rogue Access Point Detection on Edge Devices», IEEE Access, vol. 14, pp. 57806-57827, 2026, doi: 10.1109/ACCESS.2026.3682631.
- [19] J. Paguay, A. Cuenca-Sánchez, y P. Proaño, «Smart IoT-Enabled Embedded Platform for Real-Time Energy Efficiency Assessment in Heat Pumps», en The XXXIII Conference on Electrical and Electronic Engineering, MDPI, nov. 2025, p. 14. doi: 10.3390/engproc2025115014.
- [20] M. Waqas y M. Jamil, «Smart IoT SCADA System for Hybrid Power Monitoring in Remote Natural Gas Pipeline Control Stations», Electronics, vol. 13, n.º 16, p. 3235, ago. 2024, doi: 10.3390/electronics13163235.
- [21] E. Djafsia Souboura, D. Wohwe Sambo, P. Dayang, M. V. Komguez, y L. Clavier, «FragCast: A Scalable BLE-Based OTA Framework for Constrained IoT Devices», IET Wirel. Sens. Syst., vol. 16, n.º 1, p. e70031, ene. 2026, doi: 10.1049/wss2.70031.
- [22] K. Gac, J. Gorski, G. Gora, y J. Iwaniec, «Energy-Oriented Wireless Communication Platform Selection System in the Internet of Things», Sensors, vol. 26, n.º 10, p. 3158, may 2026, doi: 10.3390/s26103158.
- [23] G. Kumar et al., «Design of a Lightweight Edge-AI System for Predictive Maintenance on ESP32-S3», Appl. Sci., vol. 16, n.º 11, p. 5287, may 2026, doi: 10.3390/app16115287.
- [24] D. Sarabia-Jácome, E. Grasa, y M. Catalán, «RINAsense: An open-source, low-power IoT sensor node for Recursive InterNetwork Architecture (RINA) Experimentation», HardwareX, vol. 24, p. e00719, dic. 2025, doi: 10.1016/j.ohx.2025.e00719.
- [25] J. Pei, M. Feng, P. Muthukumar, P. V. Krishna, y M. Dai, «Generative AI-Native Edge Sensors: Collaborative Data Reconstruction via Lightweight Federated Learning», IEEE J. Sel. Areas Sens., vol. 3, pp. 180-190, 2026, doi: 10.1109/JSAS.2026.3680380.
- [26] J. García-González, C. Q. Gómez Muñoz, D. Gachet Páez, y J. Sánchez-Soriano, «Automated Road Data Collection Systems Using UAVs: Comparative Evaluation of Architectures Based on



Arduino Portenta H7 and ESP32-CAM», Electronics, vol. 14, n.º 21, p. 4165, oct. 2025, doi: 10.3390/electronics14214165.

[27] A. Pérez-González, A. F. Villa-Salazar, I. N. Gomez-Miranda, J. D. Velásquez-Gómez, A. F. Romero-Maya, y Á. Jaramillo-Duque, «An IoV-Based Real-Time Telemetry and Monitoring System for Electric Racing Vehicles: Design, Implementation, and Field Validation», Vehicles, vol. 7, n.º 4, p. 128, nov. 2025, doi: 10.3390/vehicles7040128.

[28] H. M. Vasconcelos et al., «A Low-Power Architecture for Passive Acoustic Autonomous Maritime Surveillance», J. Mar. Sci. Eng., vol. 14, n.º 9, p. 815, abr. 2026, doi: 10.3390/jmse14090815.

Derechos de Autor (c) 2026 Josué Abraham Emanuel Chambilla Zúñiga



Este texto está protegido por una licencia [Creative Commons 4.0](https://creativecommons.org/licenses/by/4.0/).

Usted es libre para compartir —copiar y redistribuir el material en cualquier medio o formato— y adaptar el documento —remezclar, transformar y crear a partir del material— para cualquier propósito, incluso para fines comerciales, siempre que cumpla la condición de:

Atribución: Usted debe dar crédito a la obra original de manera adecuada, proporcionar un enlace a la licencia, e indicar si se han realizado cambios. Puede hacerlo en cualquier forma razonable, pero no de forma tal que sugiera que tiene el apoyo del licenciante o lo recibe por el uso que hace de la obra.

[Resumen de licencia](#) - [Texto completo de la licencia](#)