International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 10 Issue: 11 | Nov - 2026 SJIF Rating: 8.659 ISSN: 2582-3930

TinyML and Edge AI for On-Device Intelligence: A Review of Keyword Spotting and Voice-Controlled Assistive Robots

Aishwarya Yuvraj Dhane[1]* and Prof. Dr. S. M. Kulkarni [2]

[1] Dept. of Electronics & Telecommunication (VLSI & EMBEDDED SYSTEM), TSSM's Padmbhushan Vasantdada Patil Institute of Technology, Pune, India.
[2] Dept. of Electronics & Telecommunication (VLSI & EMBEDDED SYSTEM), TSSM's Padmbhushan Vasantdada Patil Institute of Technology, Pune, India.
* Correspondence: aishwarya1994dhane@gmail.com

Abstract - Voice is a natural control channel for assistive mobile robots. Yet most currently deployed voice interfaces stream audio to cloud-based recognizers, which adds latency, reliance on connectivity, and exposure of privacy. Tiny machine learning (TinyML) offers an alternative in which a quantized neural network performs keyword spotting (KWS) directly on a low-cost microcontroller. This review examines how far recent work from 2024 to 2026 supports the development of a voice-controlled assistive robot based on on-device KWS. We synthesize 40 recent studies covering compact KWS architectures, neural architecture search, quantization and pruning, noise and front-end variability, on-device personalization, speech-driven robotic systems, and safety-aware human-robot interaction. The literature indicates that efficient KWS models now fit within the memory and energy budgets of low-cost microcontrollers, and that personalization and open-set recognition are maturing. Evaluation practice, however, remains uneven: false activations per hour, end-to-end command-to-action latency, and energy per decision are seldom foregrounded, and few systems couple recognition confidence with motion safety logic. Building on these observations, we present a reference architecture for an ESP32-based assistive robot with six functional blocks, a confidence-aware command arbitration layer in which an emergency stop and obstacle checks take precedence over recognized speech, and a six-experiment evaluation protocol. Five distinct research gaps are identified, including speaker-independent validation, the effect of quantization on rare-command recall, and fail-safe behavior under sensor or model faults. The review rests on reported objectives and findings rather than reproduced experiments. It offers a structured basis for designing, benchmarking, and safely deploying offline voice-controlled assistive robots on resource-constrained hardware.

Key Words: TinyML, edge AI, keyword spotting, voice-controlled robot, assistive robotics, model quantization, ESP32, on-device inference

1. INTRODUCTION

Assistive robots are increasingly expected to operate in homes, care facilities, and public buildings where users may have limited mobility, reduced vision, or difficulty with conventional controllers. Surveys of socially assistive robots in smart environments document how this technology is moving toward everyday settings in which it attends older adults [1]. Mobile assisting robots designed specifically for blind and elderly users illustrate the same trend toward everyday assistance [2]. For people who cannot reliably use a joystick, touchscreen, or gesture interface, speech remains one of the most direct ways to issue commands, and recent systems apply spoken input to vehicle control, physically assistive manipulation, and surgical assistance [3]–[5].

Most speech interfaces nonetheless depend on remote recognition. Audio is captured locally, transmitted to a server, transcribed by a large model, and returned as text or an action. Work on speech-to-text at the edge aims to achieve ultra-low latency by moving recognition closer to the user, but such pipelines still presuppose a capable processor [6]. Hybrid edge-cloud schemes that combine automatic speech recognition with large language models add flexibility to speech-to-action systems but make responsiveness depend on how work is split between the device and the server [7]. For a robot that must stop when told to stop, variable latency and a dependency on connectivity are difficult to justify. Continuous audio streaming also raises privacy concerns in homes and care settings. Offline designs for voice-controlled smart homes have been proposed with these concerns in view, emphasizing energy efficiency and low latency [8].

Tiny machine learning (TinyML) addresses this problem by running compact neural networks on microcontrollers that draw milliwatts and cost a few dollars. The field has been surveyed from several directions over the last two years, including on-device AI models [9], the progression from TinyML toward tiny deep learning [10], embedded artificial intelligence in general [11], industrial deployments [12], and smart-city applications [13]. Practical deployment guidance and optimization strategies under extreme resource limits have also been consolidated [14], [15]. A shared conclusion is that model compression, hardware-aware design, and careful

measurement decide whether a model that works on a workstation can also work on the target device.

Keyword spotting (KWS) is the canonical TinyML audio task. A KWS model listens continuously for a small vocabulary and ignores everything else, which matches the needs of a robot that must recognize a handful of motion and alert commands. A small, closed vocabulary keeps the model compact and allows exhaustive testing, which suits safety-relevant applications such as robotics. Recent work includes compact convolutional designs for resource-constrained systems [16], multi-objective architecture search with hardware-in-the-loop evaluation [17], state-space sequence models [18], and a system-level view that treats the entire embedded pipeline, rather than the network alone, as the object of optimization [19]. Reviews of small-footprint KWS now give a structured account of these efficient models and algorithms [20].

Recognition accuracy on a benchmark, however, is only one ingredient of a dependable voice-controlled robot. Real rooms add reverberation and noise that alter the acoustic features on which classifiers depend [21]. Users differ in accent, loudness, and distance from the microphone, which has motivated on-device learning of user-specific speech characteristics [22], domain learning on extreme-edge hardware [23], and self-learning personalization on ultra-low-power audio sensors [24]. Open-set methods address a related issue, namely, rejecting words outside the command vocabulary [25]. For a robot, these issues translate directly into safety, because a false activation is not a wrong transcript but an unintended motion.

The robotic side of the problem is less mature than the recognition side. Speech-driven control of mobile robots has been demonstrated with wireless links and conventional recognizers [26], and KWS has been paired with person detection in a contactless elevator prototype [27]. Safety, however, is typically treated as a separate discipline. Systematic reviews of human-robot collaboration describe safety-driven approaches in detail [28], and navigation research shows how multi-sensor fusion supports reliable motion [29]. Yet, these bodies of work seldom meet in the design of a microcontroller-class voice robot, where recognition confidence, obstacle sensing, and an emergency stop must be reconciled within milliseconds.

This paper reviews literature published between 2024 and 2026 to examine that gap. The scope is deliberately specific: a mobile assistive robot that recognizes a closed set of spoken commands, such as forward, backward, left, right, stop, and help, entirely on an ESP32-class microcontroller, with obstacle sensing and an independent emergency stop. The intended user benefits from hands-free control but may speak softly, with a regional accent, or from across the room. Hence, tolerance to speaker and distance variation is a design requirement rather than an optional feature. Four research questions guide the review.

RQ1: How far can compact and quantized KWS models reduce memory and energy cost while preserving useful accuracy?

RQ2: How do background noise, speaker variation, and microphone distance effect on-device recognition?

RQ3: Can confidence-based command validation reduce false activations without causing unacceptable delay?

RQ4: How should recognition, obstacle sensing, and emergency stopping be combined so that safety decisions always take priority over speech?

Studies were identified through targeted searches of publisher and preprint repositories. They were retained when they appeared from 2024 onward and addressed KWS, TinyML deployment, speech-driven robotics, or assistive and safety-related robotics. Forty studies were examined, and their contributions were assessed from their stated objectives, methods, and reported findings. The contributions of this review are fourfold. First, it organizes recent TinyML KWS research into a structure relevant to robotic control. Second, it compares voice-driven and assistive robotic systems in terms of autonomy, safety handling, and evaluation depth. Third, it proposes a reference architecture with a confidence-aware command arbitration layer and an experimental protocol that reports false activations, latency, memory, and energy alongside accuracy. Fourth, it identifies five research gaps that can guide future work.

The remainder of the paper is organized as follows. Section 2 presents the technical foundations of on-device KWS. Section 3 reviews the literature. Section 4 describes the reference architecture. Section 5 sets out the evaluation protocol, open challenges, and research directions, and Section 6 concludes.

2. FOUNDATIONS OF ON-DEVICE KEYWORD SPOTTING

2.1 Audio Front End and Features

On-device KWS begins with a digital MEMS microphone, typically attached through the I2S bus, which delivers fixed-point samples without an analog front end. A one-second window at 16 kHz contains L = 16,000 samples. The window is divided into overlapping frames of length W with a hop size H, yielding the frame count in (1). Each frame is windowed, transformed by a fast Fourier transform, passed through a triangular Mel filter bank, and compressed logarithmically, with an optional discrete cosine transform producing Mel-frequency cepstral coefficients (MFCCs), as sketched in Figure 1. The Mel scale maps frequency f (in hertz) to m, as in (2).

$$N = \left\lfloor \frac{(L - W)}{H} \right\rfloor + 1 \quad (1)$$

$$m = 2595 \cdot \log_{10}(1 + f / 700) \quad (2)$$

The feature tensor dimensions depend on W, H, and the number of filters; these settings determine both the buffer memory and the computation required before the network

runs. Work on cross-frontend KWS shows that classifier behavior depends on the chosen front end and proposes dynamic convolution to address this variation [30]. Reverberation and additive noise degrade features in ways that clean-speech training does not anticipate [21], so noise augmentation is normally part of the training recipe.

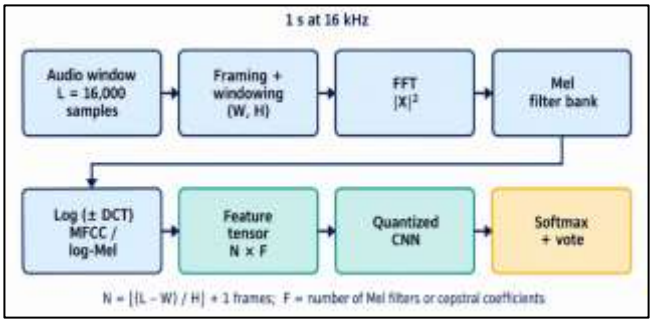


Fig -1: Audio-to-decision pipeline of an on-device KWS system for a 1s window sampled at 16 kHz (illustrative; frame length, hop, and filter count are design parameters).

2.2 Model Families

Compact CNNs and depthwise-separable CNNs remain practical baselines because their embedded runtimes well support their operators. A VGG-inspired compact network has recently been evaluated specifically for TinyML KWS [16]. Sequence models are also entering the field, as in Keyword Mamba, which applies state space models to spoken keyword spotting [18]. Neural architecture search automates the trade-off between accuracy, parameter count, and latency, and multi-objective search with hardware-in-the-loop evaluation has been compared across metaheuristic and Bayesian strategies [17]. Open-set recognition adds rejection capability by augmenting keyword features [25], which is important when most of what a robot hears is not a command.

2.3 Compression, Deployment, and Benchmarking

Quantization maps a real-valued tensor r to an integer q using a scale s and zero-point z, as in (3). Eight-bit integer storage is four times smaller than 32-bit floating point and often runs faster on integer-capable cores. However, its effect on accuracy and latency must be measured on the target board. A recent survey classifies quantization methods, platforms, and applications for microcontrollers [31]. Pruning is another approach, and DTMM demonstrates how to adapt models for extremely resource-constrained IoT devices [32]. When a single device cannot hold the entire model, splitting learning across devices offers a partial remedy, with latency as a central concern [33]. Benchmarking frameworks that record inference time, CPU load, and operation counts make such comparisons reproducible [34], and system-level studies caution that end-to-end efficiency also depends on acquisition and post-processing stages [19].

$$q = clip\left(round\left(\frac{r}{s}\right) + z, q_{\min}, q_{\max}\right) \quad (3)$$

2.4 Hardware Platforms

ESP32-class microcontrollers combine dual-core processors, a few hundred kilobytes of SRAM, external flash, and built-in wireless interfaces at very low cost, which makes them attractive for educational and prototype robots. Their memory budget must accommodate the audio buffer, feature tensor, interpreter arena, and robot control code simultaneously. Purpose-built silicon pushes efficiency further, as in a heterogeneous TinyML system-on-chip with systematic minimum-energy search reported for KWS [35]. Energy awareness extends beyond recognition to communication, as shown by work on energy-efficient TinyML object detection and transmission [36]. These platforms and techniques define the design space in which the architecture of Section 4 is positioned.

2.5 Command Vocabulary and Always-Listening Operation

The command set is itself a design variable. Short, acoustically distinct words reduce confusion between classes, and explicit unknown and silence classes let the network decline to act. Commands with safety consequences, namely stop and help, deserve extra representation in the training set and a dedicated check of recall, because a missed stop is costlier than a missed turn. Always-listening operation also fixes the duty cycle of the front end and the network, so energy spent in idle listening can weigh as heavily on a battery budget as the occasional action, which motivates the energy-oriented designs in [35], [36]. Finally, a confirmation rule across consecutive windows can trade a small increase in delay for fewer false activations, a trade-off examined in Section 4.

3. LITERATURE REVIEW

3.1 Compact KWS and On-Device Adaptation

Table 1 summarizes representative studies. Garai and Samui organize small-footprint KWS into efficient models and algorithms [20] and, in later work, apply multi-objective search with hardware-in-the-loop evaluation, which grounds the search in measured rather than estimated device behavior [17]. Chango et al. design and evaluate a compact VGG-inspired CNN for TinyML [16], while Ding et al. apply state-space models to KWS [18]. Dong et al. approach efficiency from the silicon side [35]. Liu et al. and Ko et al. address front-end variability and out-of-vocabulary rejection, respectively [25], [30]. Three studies address adaptation to the user: learnable user speech characteristics [22], on-device domain learning [23], and self-learning personalization [24]. A recurring limitation is that evaluation is usually framed around public corpora with closed vocabularies. In contrast, a robot needs a purpose-chosen command set tested with unseen speakers in the intended acoustic setting.

Table -1: Representative KWS and On-Device Adaptation Studies (2024–2026).

Ref.	Focus	Reported contribution and scope note
[20]	Review	Organizes efficient small-footprint models and algorithms; actuated platforms are outside the scope
[17]	Architecture search	Multi-objective search with hardware-in-the-loop evaluation; metaheuristic versus Bayesian
[16]	Compact CNN	VGG-inspired network designed and evaluated for resource-constrained TinyML
[18]	State space models	Keyword Mamba for spoken KWS; MCU footprint needs separate confirmation
[35]	Dedicated silicon	Heterogeneous SoC with minimum-energy search; not an off-the-shelf board
[25]	Open-set KWS	Keyword feature augmentation is relevant to rejecting non-command speech
[30]	Front-end variability	Dynamic convolution across front ends; relevant to microphone changes
[19]	System-level view	End-to-end efficiency of the embedded KWS pipeline on microcontrollers
[22]	Personalization	On-device learning of user speech characteristics
[23]	Domain learning	On-device domain learning on low-power extreme-edge systems
[24]	Self-learning	Personalized KWS on ultra-low-power audio sensors

3.2 Surveys, Deployment, and Benchmarking

Broader surveys supply the context in which KWS deployment decisions are made. Wang et al. review on-device AI models [9], and Somvanshi et al. trace the shift from TinyML to tiny deep learning [10]. Huang et al. cover embedded AI broadly [11], Alharthi et al. focus on industrial IoT [12], and Trigkas et al. on smart-city applications [13]. Cordova-Cardenas et al. pair a survey with a deployment framework for embedded neural networks [14], and Echchidmi and Bouayad discuss optimization under extreme resource constraints [15]. Taken together, these works agree that quantization, pruning, and hardware-aware design are standard practice, while reproducible measurement of latency, memory, and energy on the target device is what separates credible results from estimates [31]–[34]. Object detection on low-power hardware shows the same pattern, with energy-efficient inference and communication evaluated jointly [36].

3.3 Speech-Driven and Assistive Robotic Systems

Table 2 compares speech-driven systems at the application level. Gupta et al. analyze a speech-recognition-based wireless control system for mobile robots [26], and Hussein et al. examine deep learning parameters that influence speech-based driving control [3]. Padmanabha et al. use large language models as speech interfaces for physically assistive robots [4], and Davila et al. describe a voice interface for surgical robot assistants [5]. The edge-cloud scheme of Torkamani and Zarin chooses between local and remote processing for speech-to-action [7]. Offline recognition for smart homes [8], edge speech-to-text [6], and deep learning speech command recognition [37] complete the recognition-centered end of the range. The elevator prototype of Pimpalkar and Niture is the closest pairing of on-device KWS with a physical actuator, although it involves no locomotion [27]. Across these systems, language flexibility increases with model size, whereas determinism and independence of connectivity favor small closed-vocabulary models.

Table -2: Speech-Driven and Assistive Systems Reviewed

Ref.	Application	Processing and relevance to this review
[26]	Mobile robot control	Speech recognition with wireless link; control chain analyzed
[3]	Driving control	Deep learning recognition shows speech as a control channel
[4]	Assistive manipulation	LLM-based speech interface; relies on large models
[5]	Surgical assistance	Voice control interface in a safety-critical setting
[7]	Speech-to-action	Edge-cloud ASR and LLM; connectivity-dependent path
[8]	Smart homes	Offline recognition proposal: energy and latency motivation
[6]	Edge speech-to-text	Ultra-low-latency prototype; transcription, not command spotting
[27]	Contactless elevator	TinyML person detection and KWS; no mobility
[37]	Command recognition	Deep learning command classifier; recognition-centered

3.4 Assistive Context, Perception, and Safety

Socially assistive robots in smart environments are surveyed by Cruces et al. [1], and Garrosa et al. design a mobile assisting robot for blind and older adults [2]. Alternative input channels are developing in parallel: Nguyen et al. steer a smart wheelchair with hand gestures recognized by a YOLOv8n model [38], and Fertl et al. review gesture recognition on edge devices [39]. Such channels can complement speech when noise makes the voice unreliable. On the motion side, Ušinskis et al. show how sensor fusion supports navigation for autonomous mobile robots [29], and Salih et al. implement perception-driven navigation for quadruped robots on microcontroller hardware [40]. Khan et al. review safety-driven approaches in human-robot collaboration [28]. These contributions provide components, namely perception, navigation, and safety principles. Still, none of them define how a spoken command should be accepted, delayed, or refused by a microcontroller's safety layer.

3.5 Processing Paradigms for Voice-Controlled Robots

Table 3 contrasts the processing paradigms that appear in the reviewed work. Cloud and hybrid recognition [4], [6], [7] offer open vocabularies and natural language, whereas compact on-device KWS [16], [19] offers a closed vocabulary with predictable behavior. The comparison is qualitative and reflects architectural properties rather than measured figures. It explains why spoken motion commands suit on-device KWS, while open-ended requests may justify remote processing.

Table -3: Processing Paradigms for Voice-Controlled Robots (Qualitative)

Property	Cloud or hybrid ASR	On-device KWS
Vocabulary	Open, language-model driven	Closed, command-specific
Connectivity	Needed for the remote path	Not needed for recognition
Response time	Varies with the link and server	Set by the local pipeline
Audio privacy	Audio leaves the device	Audio stays on the device
Compute and memory	Large models off-device	Limited by MCU RAM and flash
Behavior under fault	Degrades with the network	Deterministic, testable offline

3.6 Synthesis

Figure 2 shows the distribution of the 40 reviewed studies by theme and year. Three observations follow. Studies on efficient KWS and TinyML deployment make up more than half of the corpus, and most of them appeared in 2025 or 2026. Work on speech-driven and assistive systems is smaller and mostly recognition-centered. Studies that join recognition with actuation under explicit safety logic are scarce within the reviewed set, which motivates the architecture proposed next.

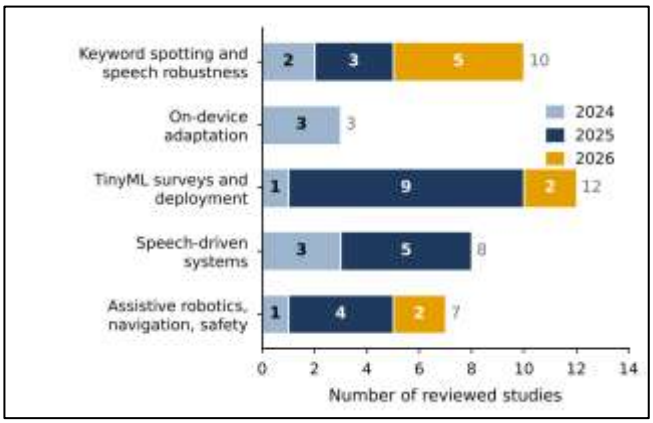


Fig -2: Distribution of the 40 reviewed studies by theme and publication year (compiled by the authors from the reference list).

4. REFERENCE ARCHITECTURE

4.1 Functional Blocks

The proposed system is a two-wheel mobile robot that listens for predefined commands, recognizes them locally, and acts only after a safety layer approves. Figure 3 shows six functional blocks. A digital microphone captures speech (block 1), the ESP32 segments the audio and computes log-Mel or MFCC features (block 2), and a quantized network predicts the command class (block 3). A command arbitration layer (block 4) decides whether the prediction may be executed, a motor driver moves the robot (block 5), and ranging, an audible buzzer, and a display provide safety and feedback (block 6). The hardware choices are listed in Table 4. An I2S microphone is preferred over an analog sound sensor because it provides digital audio and avoids the limits of a simple analog front end. The motor driver is adequate only when the stall current remains within its rating, and motor noise and voltage drops must be controlled through regulation and decoupling to ensure the microphone and processor see a stable supply.

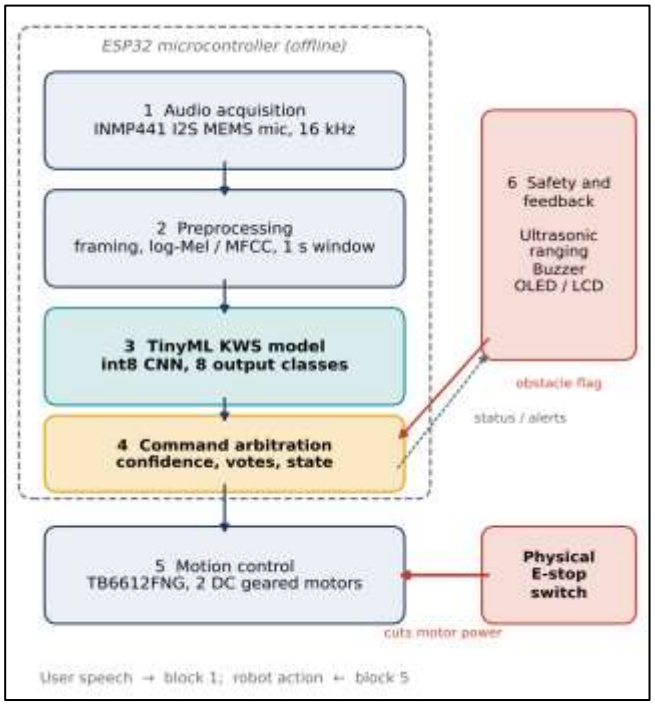


Fig -3: Proposed six-block architecture

Blocks 1 to 4 run on the ESP32 without network access; the obstacle flag feeds the arbitration layer, and the physical E-stop cuts motor power independently of software.

Table -4: Hardware of the First Prototype and Design Notes

Component	Role	Design note
ESP32 board	Inference, control	RAM and flash must hold buffers, the tensor arena, and firmware
INMP441 (I2S)	Digital audio	Avoids analog front-end limits
TB6612FNG, 2 DC motors	Locomotion	Check stall current against driver rating
Ultrasonic sensor (1–2)	Obstacle sensing	Gates forward motion
Buzzer, OLED/LCD	Feedback	Command and fault messages
Battery, regulator	Power	Decoupling against motor noise
Physical E-stop	Independent safety	Disconnects motor power in hardware

4.2 Model and Dataset Design

The classifier has eight output classes: the six commands (forward, backward, left, right, stop, help), an unknown or background class, and silence. Treating unknown speech and silence as explicit classes prevents the system from assuming that every sound is a valid command. A starting point of 100 to 300 recordings per command from several speakers is reasonable for a prototype, and a stronger study should vary speakers, accents, loudness, distance, room acoustics, and noise. The speaker should split the training, validation, and test sets, since a random split lets the model memorize familiar voices. Two configurations, a compact CNN and a depthwise-separable CNN, should be compared, followed by 8-bit quantization evaluated on the same test set. The review

above suggests drawing on noise augmentation [21], open-set rejection [25], and, where user variation is large, lightweight personalization [22], [24].

4.3 Confidence-Aware Command Arbitration

Mapping each prediction directly to a motor action is unsafe. Figure 4 shows the proposed arbitration sequence. The emergency stop is checked first; then the prediction confidence is compared against a threshold τ selected on validation data rather than chosen arbitrarily; then the agreement of k out of n consecutive predictions is assessed; then the validity of the command in the current robot state is assessed; and finally, the obstacle distance is assessed. A command is executed only when all conditions in (4) hold, where p is the confidence, V the vote result, S the state-validity check, E the emergency flag, and d the measured distance.

$$execute(c) \Leftrightarrow p \geq \tau \wedge V(c) \wedge S(c,s) \wedge \neg E \wedge d > d_{\min} \quad (4)$$

A command timeout limits each action to a short burst, so a single utterance cannot cause unbounded motion. The command "help" stops the robot, sounds the buzzer, and displays an assistance message. An external alert can be sent over a separate wireless link, but it cannot be guaranteed when the network is unavailable, whereas recognition itself continues to work offline. Gating motion based on recognition confidence and independent sensors aligns with the emphasis on safety-driven design in human-robot collaboration [28].

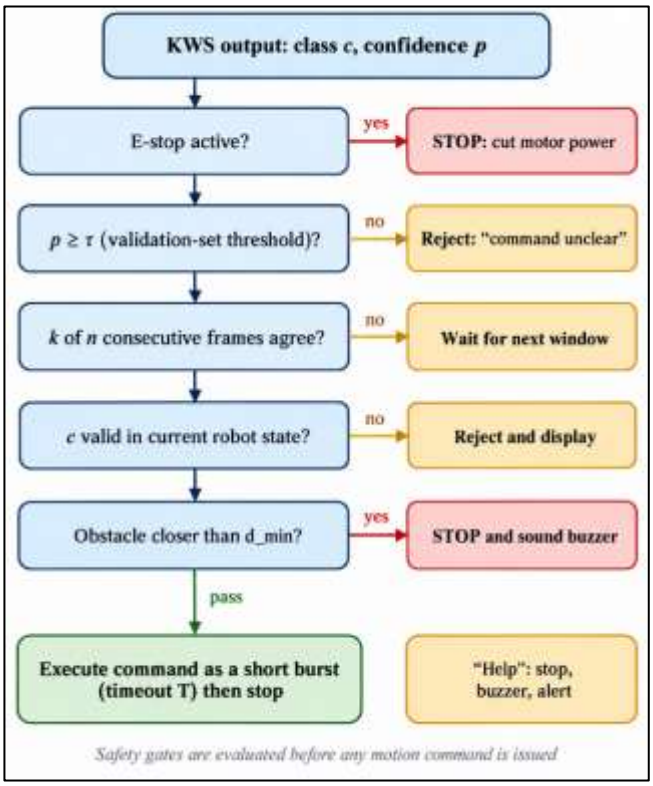


Fig -4: Command arbitration sequence. Safety gates (red) override recognized speech; amber outcomes reject the command without motion.

5. EVALUATION PROTOCOL, RESEARCH GAPS, AND DIRECTIONS

5.1 Evaluation Protocol

Accuracy alone is insufficient, as a model can appear accurate while activating the robot during background speech. Table 5 lists six experiments, and Figure 5 shows how they fit into the development workflow. Besides accuracy, precision, recall, F1-score, and the confusion matrix, the protocol reports the false rejection rate and the false activation rate defined in (5) and (6), where M is the number of missed command utterances among N_k command utterances. A is the number of unintended activations in T hours of non-command audio.

$$FRR = \frac{M}{N_k} \quad (5)$$

$$\frac{FA}{h} = \frac{A}{T} \quad (6)$$

Table -5: Proposed Experimental Protocol

Exp.	Measurements	Answers
E1	Accuracy, precision, recall, F1, confusion matrix, FAR, FRR on unseen speakers	RQ1
E2	FP32 versus int8: accuracy, model size, peak RAM, latency, energy per inference	RQ1
E3	Accuracy and false activations at several measured noise levels and distances	RQ2
E4	Command-to-action delay is split into acquisition, features, inference, validation, and motor response.	RQ3, RQ4
E5	Current draw in idle listening, preprocessing, inference, motion, alert	RQ1
E6	E-stop success, obstacle blocking, false motion, behavior on the microphone fault, or invalid output	RQ4

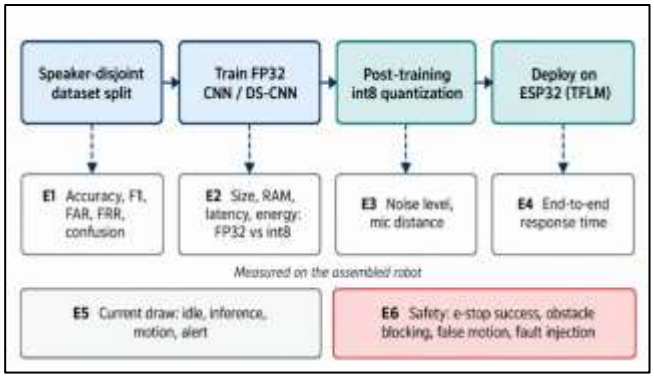


Fig -5: Evaluation workflow. E1 to E4 are tied to model development and deployment; E5 and E6 are measured on the assembled robot.

Hardware-in-the-loop measurement is consistent with the evaluation used in architecture search [17] and with the benchmarking practice of reporting inference time, CPU load, and operations [34]. Energy per inference and idle-listening current deserve equal weight with accuracy because an assistive robot must operate on a battery [35], [36].

5.2 Research Gaps

Five gaps follow from the review. G1, speaker-independent and noise-aware validation: in the reviewed abstracts and stated findings, unseen speakers and measured noise levels are seldom emphasized, although both shape real-

room performance [21], [30]. G2, quantization effects on rare commands: aggregate accuracy can hide a drop in recall for "stop" or "help," the two commands where a miss matters most [31]. G3, coupling of confidence and safety: recognition studies and safety studies develop separately [28], [29]. G4, end-to-end reporting: within the reviewed set, few studies split command-to-action latency and energy per decision across the pipeline [19], [34]. G5, fail-safe behavior: responses to microphone loss, invalid model output, or low supply voltage are seldom described for MCU-class robots in the reviewed set. Table 6 summarizes the five gaps and maps each to a measurable response in Table 5.

Table -6: Research Gaps and Proposed Responses

Gap	Observation in the reviewed set	Proposed response
G1	Unseen-speaker and measured-noise validation are seldom emphasized	Speaker-disjoint splits; E3 at measured levels
G2	Aggregate accuracy can hide the loss of stop and help recall after quantization.	Per-class recall, int8 versus FP32 (E2)
G3	Recognition and safety studies develop separately	Arbitration layer of Figure 4, tested in E6
G4	Latency and energy seldom split across pipeline stages	Stage-wise timing and current (E4, E5)
G5	Fault behavior is rarely described for MCU-class robots	Fault injection in E6

5.3 Future Directions

Personalization from a few user recordings [22]–[24] may reduce errors in atypical speech, which is especially relevant for assistive users, at the cost of additional memory and the need to guard against drift. Open-set rejection [25] and multimodal confirmation, for example, pairing a spoken command with a gesture [38], [39], could lower false activations without raising the threshold τ to the point of missed commands. Edge-cloud fallback [7] and language-model interfaces [4] may serve non-critical requests such as questions, provided that motion commands remain on the deterministic offline path. Finally, the use of compact neural architecture search [17] and silicon-level energy management [35] deserves study for battery-limited assistive platforms.

Shared benchmark protocols would also help. A community test set of assistive-robot commands, recorded with unseen speakers at several distances and noise levels, together with a standard report sheet listing model size, peak RAM, latency, energy per inference, FRR, and false activations per hour, would enable future studies to be comparable. Such reporting would extend existing TinyML benchmarking practice [34] and the system-level view of keyword spotting [19] toward the actuation-aware setting emphasized here. User studies with the intended population, including older adults and people with speech differences, remain necessary before any claim of assistive benefit can be made [1], [2].

5.4 Limitations of the Review

Two limitations apply. The corpus was assembled by targeted search rather than a registered protocol, so that some relevant studies may be missing, and contributions were assessed from reported objectives and findings, which means that implementation details, such as exact memory footprints, were not independently verified. The reference architecture is a design proposal. Its parameters, including the threshold τ, the vote length k of n, the distance d_{min}, and the burst timeout T, must be determined experimentally and are not claimed here.

6. CONCLUSION

This review examined 40 studies from 2024 to 2026 to assess how well current TinyML research supports a voice-controlled assistive robot that recognizes commands entirely on a microcontroller. Compact and quantized KWS models, architecture search, open-set rejection, and on-device personalization are advancing quickly, and benchmarking practice is improving. The integration of recognition with obstacle sensing, an emergency stop, and confidence-aware arbitration remains thinly documented. We therefore proposed a six-block reference architecture, an arbitration sequence in which safety conditions precede speech, and a six-experiment protocol that reports false activations, latency, memory, and energy next to accuracy. Five research gaps were identified. Because the reviewed studies were assessed from their reported objectives and findings rather than reproduced, the conclusions should be read as a map of the field. Experimental validation of the proposed architecture on a physical prototype is the natural next step.

Conflicts of Interest: The authors declare no conflict of interest

REFERENCES

1. Cruces, A.; Jerez, A.; Bandera, J.P.; Bandera, A. Socially Assistive Robots in Smart Environments to Attend Elderly People—A Survey. Applied Sciences 2024, 14, 5287. https://doi.org/10.3390/app14125287.
2. Garrosa, M.; Ceccarelli, M.; Russo, M.; Yang, B. Design of a Mobile Assisting Robot for Blind and Elderly People. Applied Sciences 2025, 15, 10474. https://doi.org/10.3390/app151910474.
3. Hussein, H.H.; Karan, O.; Kurnaz, S. Enhancing Driving Control via Speech Recognition Utilizing Influential Parameters in Deep Learning Techniques. Electronics 2025, 14, 496. https://doi.org/10.3390/electronics14030496.
4. Padmanabha, A.; et al. VoicePilot: Harnessing LLMs as Speech Interfaces for Physically Assistive Robots. arXiv 2024, arXiv:2404.04066. https://doi.org/10.1145/3654777.3676401.
5. Davila, A.; Colan, J.; Hasegawa, Y. Voice Control Interface for Surgical Robot Assistants. arXiv 2024, arXiv:2409.10225. https://doi.org/10.1109/MHS63891.2024.10856297.
6. Di Leo, S.; De Cicco, L.; Mascolo, S. Real-Time Speech-to-Text on Edge: A Prototype System for Ultra-Low Latency Communication with AI-Powered NLP. Information 2025, 16, 685. https://doi.org/10.3390/info16080685.

7. Torkamani, M.J.; Zarin, I. Adaptive Edge-Cloud Inference for Speech-to-Action Systems Using ASR and Large Language Models. arXiv 2025, arXiv:2512.12769.
8. Huang, P.; Ullah, I.; Wei, X.; Ahanger, T.A.; Hassan, N.; Shah, Z.H. Towards Energy-Efficient and Low-Latency Voice-Controlled Smart Homes: A Proposal for Offline Speech Recognition and IoT Integration. arXiv 2025, arXiv:2506.07494.
9. Wang, X.; et al. Empowering Edge Intelligence: A Comprehensive Survey on On-Device AI Models. arXiv 2025, arXiv:2503.06027. https://doi.org/10.1145/3724420.
10. Somvanshi, S.; et al. From Tiny Machine Learning to Tiny Deep Learning: A Survey. arXiv 2025, arXiv:2506.18927. https://doi.org/10.1145/3776588.
11. Huang, X.; Wang, H.; Qin, S.; Tang, S.-K. Embedded Artificial Intelligence: A Comprehensive Literature Review. Electronics 2025, 14, 3468. https://doi.org/10.3390/electronics14173468.
12. Alharthi, S.; Rashid, M.; Aljabri, M. TinyML in Industrial IoT: A Systematic Review of Applications, System Components, and Methodologies. Sensors 2026, 26, 2550. https://doi.org/10.3390/s26082550.
13. Trigkas, A.; Piromalis, D.; Papageorgas, P. Edge Intelligence in Urban Landscapes: Reviewing TinyML Applications for Connected and Sustainable Smart Cities. Electronics 2025, 14, 2890. https://doi.org/10.3390/electronics14142890.
14. Cordova-Cardenas, R.; Amor, D.; Gutiérrez, Á. Edge AI in Practice: A Survey and Deployment Framework for Neural Networks on Embedded Systems. Electronics 2025, 14, 4877. https://doi.org/10.3390/electronics14244877.
15. Echchidmi, M.; Bouayad, A. TinyML for Sustainable Edge Intelligence: Practical Optimization under Extreme Resource Constraints. Technologies 2026, 14, 215. https://doi.org/10.3390/technologies14040215.
16. Chango, W.G.; et al. Design and Evaluation of a Compact VGG-Inspired CNN for Keyword Spotting in Resource-Constrained TinyML Systems. Computation 2026, 14, 112. https://doi.org/10.3390/computation14050112.
17. Garai, S.; Samui, S. Multi-Objective Neural Architecture Search for TinyML Keyword Spotting: A Comparative Study of Metaheuristic and Bayesian Optimization with Hardware-in-the-Loop Evaluation. Neural Computing and Applications 2026, 38, 784. https://doi.org/10.1007/s00521-026-12484-3.
18. Ding, H.; et al. Keyword Mamba: Spoken Keyword Spotting with State Space Models. Computer Speech & Language 2026, 99, 101909. https://doi.org/10.1016/j.csl.2025.101909.
19. Bartoli, P.; Bondini, T.; Veronesi, C.; Giudici, A.; Antonello, N.; Zappa, F. End-to-End Efficiency in Keyword Spotting: A System-Level Approach for Embedded Microcontrollers. arXiv 2025, arXiv:2509.07051.
20. Garai, S.; Samui, S. Advances in Small-Footprint Keyword Spotting for TinyML: A Comprehensive Review of Efficient Models and Algorithms. Neurocomputing 2026, 695, 134028. https://doi.org/10.1016/j.neucom.2026.134028.
21. Cesarini, V.; Costantini, G. Reverb and Noise as Real-World Effects in Speech Recognition Models: A Study and a Proposal of a Feature Set. Applied Sciences 2024, 14, 11446. https://doi.org/10.3390/app142311446.
22. Cioflan, C.; Cavigelli, L.; Benini, L. Boosting Keyword Spotting through On-Device Learnable User Speech Characteristics. arXiv 2024, arXiv:2403.07802.
23. Cioflan, C.; Cavigelli, L.; Rusci, M.; de Prado, M.; Benini, L. On-Device Domain Learning for Keyword Spotting on Low-Power Extreme Edge Embedded Systems. arXiv 2024, arXiv:2403.10549.
24. Rusci, M.; Paci, F.; Fariselli, M.; Flamand, E.; Tuytelaars, T. Self-Learning for Personalized Keyword Spotting on Ultra-Low-Power Audio Sensors. IEEE Internet of Things Journal 2024. https://doi.org/10.1109/JIOT.2024.3515143.
25. Ko, K.; et al. KFA: Keyword Feature Augmentation for Open Set Keyword Spotting. IEEE Signal Processing Letters 2024, 31, 2985–2989. https://doi.org/10.1109/LSP.2024.3484932.
26. Gupta, S.; Mamodiya, U.; Al-Gburi, A.J.A. Speech Recognition-Based Wireless Control System for Mobile Robotics: Design, Implementation, and Analysis. Automation 2025, 6, 25. https://doi.org/10.3390/automation6030025.
27. Pimpalkar, A.S.; Niture, D.V. Towards Contactless Elevators with TinyML Using CNN-Based Person Detection and Keyword Spotting. arXiv 2024, arXiv:2405.13051.
28. Khan, A.; Akhtar, M.; Qureshi, S.M.; Mustafa, M.; Alsaleh, N.A.; Ahmad, I. A Systematic Review of Safety-Driven Approaches in Human–Robot Collaborative Systems. Sensors 2026, 26, 2079. https://doi.org/10.3390/s26072079.
29. Ušinskis, V.; Nowicki, M.; Dzedzickis, A.; Bučinskas, V. Sensor-Fusion-Based Navigation for Autonomous Mobile Robot. Sensors 2025, 25, 1248. https://doi.org/10.3390/s25041248.
30. Liu, R.; et al. Dynamic Convolution Models for Cross-Frontend Keyword Spotting. Scientific Reports 2025, 15, 16745. https://doi.org/10.1038/s41598-025-00304-y.
31. Abushahla, H.A.; Varam, D.; Panopio, A.J.N.; AlHajri, M.I. Neural Network Quantization for Microcontrollers: A Comprehensive Survey of Methods, Platforms, and Applications. arXiv 2025, arXiv:2508.15008.
32. Han, L.; Xiao, Z.; Li, Z. DTMM: Deploying TinyML Models on Extremely Weak IoT Devices with Pruning. arXiv 2024, arXiv:2401.09068.
33. Jenhani, Z.; Bensalem, M.; Dizdarević, J.; Jukan, A. Optimizing Split Learning Latency in TinyML-Based IoT Systems. arXiv 2025, arXiv:2507.16594.
34. Dey, A.; Srivastava, S.; Singh, G.; Pettit, R.G. Real-Time Performance Benchmarking of TinyML Models in Embedded Systems (PICO: Performance of Inference, CPU, and Operations). arXiv 2025, arXiv:2509.04721.
35. Dong, Y.; et al. A Heterogeneous TinyML SoC with Systematic Minimum Energy Searching and Management for Keyword Spotting. IEEE Journal of Solid-State Circuits 2026, 61, 3098–3112. https://doi.org/10.1109/JSSC.2025.3619427.
36. Bhushan, C.M.; et al. Deploying TinyML for Energy-Efficient Object Detection and Communication in Low-Power Edge AI Systems. Scientific Reports 2025, 15, 44299. https://doi.org/10.1038/s41598-025-27818-9.
37. Vujičić, D.; Damnjanović, Đ.; Marković, D.; Stamenković, Z. Deep Learning System for Speech Command Recognition. Electronics 2025, 14, 3793. https://doi.org/10.3390/electronics14193793.
38. Nguyen, T.-H.; Ngo, B.-V.; Nguyen, T.-N. Vision-Based Hand Gesture Recognition Using a YOLOv8n Model for the Navigation of a Smart Wheelchair. Electronics 2025, 14, 734. https://doi.org/10.3390/electronics14040734.
39. Fertl, E.; Castillo, E.; Stettinger, G.; Cuéllar, M.P.; Morales, D.P. Hand Gesture Recognition on Edge Devices: Sensor

Technologies, Algorithms, and Processing Hardware. Sensors 2025, 25, 1687. https://doi.org/10.3390/s25061687.

40. Salih, F.H.; Mazinan, A.H.; Modaresi, S.M. An Integrated Tiny-YOLO v3 and Q-Iteration Framework for Stable, Energy-Efficient Autonomous Navigation of Quadruped Robots on AMB82-Mini Microcontrollers. Scientific Reports 2026, 16, 1021. https://doi.org/10.1038/s41598-025-30610-4.