

Controller-Side Reliability of Edge-Deployed Thai Speech and Gesture Smart-Load Control on Raspberry Pi 5

Neres Supalugsanasuksakorn¹, Sakron Taeanonjiew², Apinun Khonmanee³, Saweth Hongprasit⁴, Wachirapunya Punyawong^{5*}

¹Computer Technology Department, Khon Kaen Technical College, Khon Kaen, Thailand

^{2,3,5}Electronic Technology Program, Khon Kaen Technical College, Northeastern Vocational Education Institute 3, Khon Kaen, Thailand

⁴Department of Computer Engineering, Faculty of Engineering, Rajamangala University of Technology Isan, Khon Kaen Campus, Khon Kaen, Thailand

Article Info

Article history:

Received May 7, 2026

Revised Sep 22, 2026

Accepted Oct 6, 2026

Keywords:

Embedded systems

Edge AI

Gesture recognition

Reliability evaluation

Dual-modality interaction

ABSTRACT

This paper presented an edge-deployed smart-load control system on Raspberry Pi 5 using independent Thai speech and vision-based finger-gesture channels mapped to a common ten-command GPIO actuation layer. Rather than proposing a new recognition algorithm, the study evaluated controller-side execution reliability using GPIO-confirmed binary outcomes and supplemented the human-trial evaluation with system-level engineering profiling. Structured trials examined interaction distance and background speech, yielding 90.0% execution success for both voice (54/60) and gesture (63/70), with different condition-specific performance patterns. A separate 10.5-min integrated profiling run measured computational behavior on the Raspberry Pi 5. Median processing latency was 46.90 ms for Vosk audio blocks and 72.15 ms for MediaPipe hand processing, while median GPIO software-update latency was 0.0688 ms. Median system CPU utilization was 32.30%, process memory was 427.55 MB, and CPU temperature was 83.70 °C; non-zero throttling-status values were observed during sustained workload. These results characterized both execution reliability and resource constraints for non-critical smart-load control on Raspberry Pi-class edge hardware.

Copyright © 2026 Institute of Advanced Engineering and Science.
All rights reserved.

Corresponding Author:

Wachirapunya Punyawong,
Electronic Technology Program, Khon Kaen Technical College,
Northeastern Vocational Education Institute 3, Khon Kaen, Thailand,
67 Srijun Road, Mueang District, Khon Kaen, Thailand 40000.
Email: Wachirapunyapu@kktech.ac.th

1. INTRODUCTION

Recent advances in edge computing and on-device artificial intelligence have enabled embedded systems to perform increasingly complex perception and decision tasks locally, thereby reducing cloud dependency, improving privacy, and supporting low-latency interaction in real-world environments [1]–[3]. These properties are especially important in practical smart-control applications, where timely response, availability, and deployment simplicity are often as important as raw model accuracy [2]–[4]. In intelligent home and assistive environments, Raspberry Pi-class platforms have emerged as attractive edge controllers because they offer sufficient computational capability for real-time AI pipelines while remaining affordable, compact, and easy to integrate with sensing and actuation hardware [4]–[7].

Within this context, speech and hand-gesture interaction have become two prominent touchless modalities for embedded human-machine interfaces. Speech provides a natural and accessible means of command input, while gesture offers a silent and intuitive alternative in situations where voice interaction is inconvenient or unreliable [8]–[12]. Prior studies have demonstrated the feasibility of voice-enabled smart control, embedded gesture recognition, and multimodal interfaces in smart-home, assistive, and safety-oriented systems [5]–[7], [13], [14]. More broadly, prior work on touchless and multimodal interaction suggests that

combining multiple interaction channels can improve usability, accessibility, and operational flexibility in embedded applications [9], [10], [14]–[16].

Recent edge-AI studies also emphasize that practical on-device deployment should be evaluated beyond recognition accuracy alone. Resource-constrained inference is affected by computation, memory, latency, and power limitations [17], while lightweight vision research has increasingly focused on compact architectures and efficient inference for edge devices [18]. Surveys of TinyML and on-device inference similarly identify latency and resource efficiency as important deployment considerations [19]. For speech interaction, local speech-to-text systems using offline engines such as Vosk demonstrate the feasibility of low-latency processing without continuous cloud connectivity [20]. At the system level, recent TinyML literature further argues for evaluating the interaction between models, runtimes, hardware resources, and deployment constraints rather than considering model accuracy in isolation [21]. These observations motivate the supplementary computational and thermal profiling reported in this study alongside GPIO-confirmed execution reliability.

Previous studies on voice or gesture control have usually reported recognition results, such as transcript accuracy, gesture classification accuracy, or response time [7], [12], [22], [23]. These results are useful, but they do not fully show whether the command finally changes the intended control state. In a smart-load system, the important point is not only that the system recognizes the input, but also that the controller sends the correct switching command. This issue becomes clearer when the user stands farther from the device or when background speech is present [15], [16], [24]. Therefore, this study focuses on execution at the controller side instead of reporting only recognition performance.

This study developed a smart-load control system on Raspberry Pi 5 using two independent input modalities: Thai speech commands and finger gestures. Both channels are mapped to the same ten command tokens before GPIO actuation; the architecture does not perform feature-level or score-level multimodal fusion. The contribution is therefore an execution-grounded reliability and engineering characterization of an integrated edge-control implementation rather than a new speech- or vision-recognition algorithm. The main evaluation criterion was whether the logged GPIO relay-control state changed according to the intended command. The human-trial evaluation considered interaction distance and background speech, and a separate system-level profiling run characterized processing latency, CPU utilization, memory footprint, frame rate, temperature, throttling status, and GPIO software-update latency.

2. RESEARCH METHOD

This section presents the objective and methodological framework used to design, implement, and evaluate an edge-deployed dual-modality smart-load control system on Raspberry Pi 5. Thai speech recognition and vision-based finger-gesture interaction operate as independent perception channels and are mapped to a shared command and GPIO actuation layer. No feature-level or score-level fusion is performed. In addition to controller-side execution reliability, a separate engineering profiling experiment was conducted to characterize computational latency and resource behavior during integrated on-device operation.

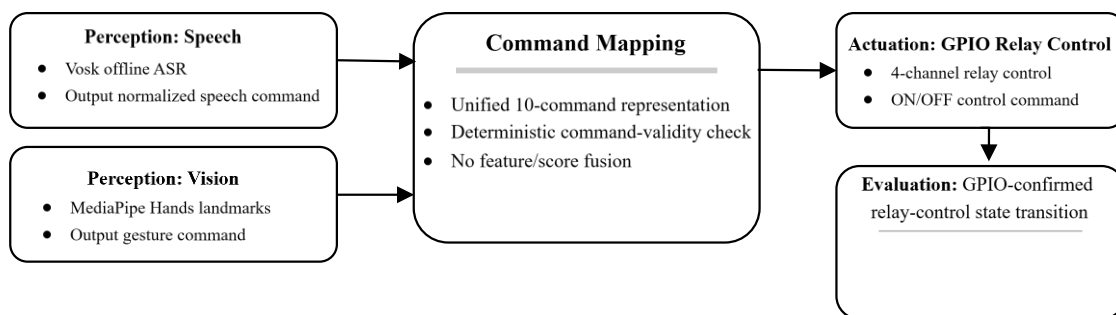


Figure 1. Overall architecture of the proposed edge-deployed dual-modality smart-load control system

2.1. System Overview

This section describes how the system was built and tested. The prototype was implemented on a Raspberry Pi 5 and used two input channels: Thai speech input and vision-based finger gesture input. Both channels were processed on the device and converted into the same command set before relay control. The experiment measured whether each command produced the intended GPIO relay-control state transition. The main parts of the method include the hardware setup, command processing, test conditions, outcome definition, latency measurement, and statistical analysis. Figure 1 shows the overall system flow.

2.2. Hardware Platform and Experimental Setup

The prototype was built on a Raspberry Pi 5 running Raspberry Pi OS. Hardware interfaces included a USB microphone for Thai speech recognition and a USB webcam for hand-gesture detection. In the implemented configuration used for the profiling code, speech input was acquired at 48 kHz, mono, 16-bit PCM with an audio block size of 16,000 samples. Camera frames were configured at 640×480 pixels with a target capture rate of 30 frames/s. All speech processing, gesture processing, command checking, and GPIO control were performed locally on the Raspberry Pi 5 without cloud inference.

The relay-control part used two 2-channel electromechanical relay modules, giving four controllable output channels. The Raspberry Pi GPIO pins sent ON/OFF control signals to the relay modules. For each trial, the program logged the GPIO relay-control states before and after the command. These logs were used to check whether the intended controller-side state transition occurred.

Figure 2 shows the laboratory setup and the relay-control prototype. Figure 2(a) shows the sensing devices and the control platform used during the tests. Figure 2(b) shows the inside of the relay-control enclosure, including the Raspberry Pi 5, relay modules, and wiring connections. In this study, the result was confirmed from GPIO state logs at the controller side. Direct load-side electrical measurement was not included.

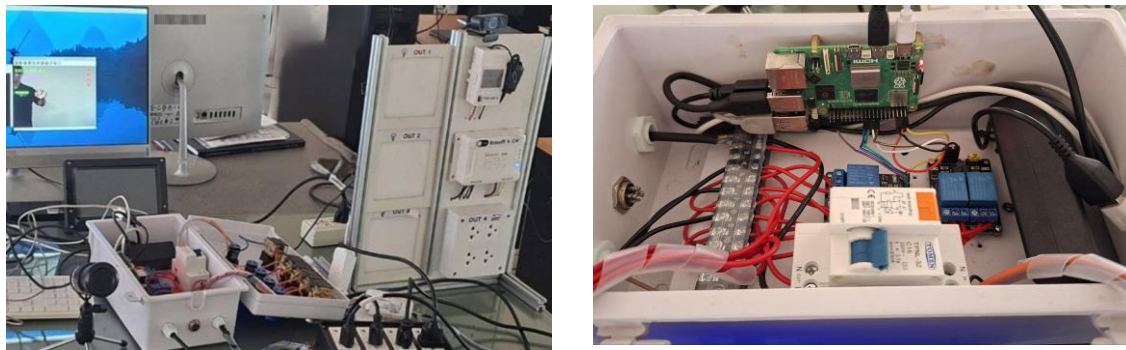


Figure 2. Experimental setup and relay-control prototype: (a) laboratory test environment; (b) internal hardware implementation of the Raspberry Pi 5 and relay-control enclosure

2.3. Dual-Modality Command-Processing Method

The system employs two independent perception pipelines for speech and gesture. The term dual-modality is used here to describe the availability of two interaction channels, not multimodal feature or score fusion. Although the branches differ in sensing and preprocessing, both are mapped deterministically into a common ten-command representation before GPIO actuation. The processing flows are summarized in Figure 3.

2.3.1. Thai Speech-Command Processing

The Thai speech-command pipeline is shown in Figure 3(a). The USB microphone captured the speech input, and the Raspberry Pi 5 decoded the command locally using the Vosk offline speech-recognition engine [25], [26]. After decoding, the Thai transcript was normalized before command matching. This step was used to reduce small differences in spacing, wording, and recognition output.

For each decoded utterance, the normalized transcript was compared with a predefined Thai command dictionary. The implementation used deterministic transcript matching and normalization rules; the experimental code used for the reported trials did not apply a separate word-confidence threshold. Unmatched transcripts therefore produced no command output and did not trigger relay-control actuation.

Accepted speech commands were converted into ten internal control tokens. O1–O4 represented opening relay channels 1–4. C1–C4 represented closing relay channels 1–4. OA represented opening all channels, and CA represented closing all channels. These tokens were then passed to the shared command and actuation layer for deterministic command validation and GPIO relay-control execution. Only mapped command tokens were forwarded for relay-control actuation.

2.3.2. Vision-Based Gesture-Command Processing

The vision-based gesture-command pipeline is presented in Figure 3(b). RGB frames from the USB webcam are processed using MediaPipe hand-landmark extraction to obtain hand keypoints [27], [28]. These landmarks are interpreted using rule-based finger-state and hand-orientation logic to determine the corresponding command. To reduce transient triggering during hand motion, the implementation requires the same candidate command to remain stable for 0.8 s before execution. This temporal hold-validation interval is intentionally longer than a video-frame period and is used to suppress transient triggering during hand motion. A visual progress bar indicates the hold interval, followed by an EXEC confirmation and channel-state indicators on the connected display.

2.3.3. Unified Command Representation and GPIO Actuation

Speech and gesture outputs are mapped independently into a unified ten-command label space consisting of Open 1–4, Close 1–4, Open All, and Close All. This common representation provides a shared actuation interface but does not constitute feature-level or score-level multimodal fusion. Before GPIO actuation, deterministic command-validity checks are applied to the mapped command token.

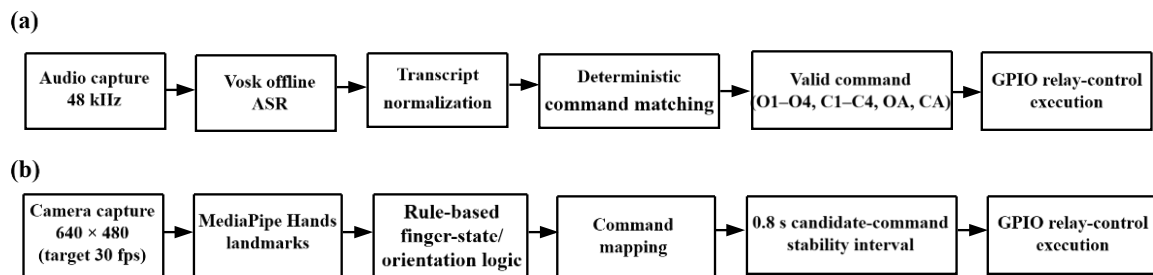


Figure 3. Modality-specific command-processing pipelines: (a) Thai speech-command processing; (b) vision-based gesture-command processing

2.4. Experimental Conditions and Trial Procedure

The experimental evaluation was conducted in an indoor laboratory environment using a structured trial protocol. Twelve adult participants, aged 18-60 years, were involved: six in the voice trials and six in the gesture trials. No children or minors participated in the study. The voice evaluation comprised 60 trials and the gesture evaluation comprised 70 trials. Because the participant groups were small and non-overlapping, the human-trial component is treated as a pilot engineering reliability evaluation rather than a population-level HCI or usability study. Participants were informed of the purpose of the study and were introduced to the command vocabulary, operating procedure, and required interaction distance before testing. Verbal informed consent was obtained from each participant before the experimental trials began; no written consent form was used.

For the voice branch, the four condition codes were D1 (0.5 m, quiet), D2 (0.5 m, background speech), D3 (1.0 m, quiet), and D4 (1.0 m, background speech). Gesture trials were conducted using the same distance labels and condition codes to support cross-modality comparison. However, background speech was retained only as a contextual environmental condition for gesture interaction and was not treated as a primary gesture-side factor. The gesture evaluation consisted of 70 trials distributed across the same condition-coded structure. Each trial was performed as a discrete smart-load switching task through the four-channel relay-control interface. Each trial followed the same operational sequence: input acquisition, modality-specific processing, command mapping, command validation, GPIO relay-control command issuance, and outcome logging. The logged GPIO states before and after command issuance were used to determine whether the intended relay-control state transition was achieved at the controller side.

2.5. Outcome Measures and Statistical Analysis

The primary outcome measure of this study was binary execution success based on GPIO-confirmed controller-side relay-control state-transition labeling. For each trial, the system logged the GPIO relay-control states before and after command issuance. A trial was labeled as successful only when the logged post-command GPIO relay-control state matched the intended state transition defined for the issued command. If the logged state did not match the intended command outcome, the trial was labeled as a failure. Failure cases included non-actuation, incorrect channel actuation, or a mismatch between the intended relay-control state and the logged controller-side GPIO state. The execution success rate for each condition was calculated as

$$\hat{p} = \frac{x}{n} \quad (1)$$

where x is the number of successful executions and n is the total number of trials under the corresponding condition. Success rates and Wilson 95% confidence intervals were reported to summarize controller-side execution reliability across experimental conditions [29].

In addition to binary execution success, response latency was measured from input detection to GPIO relay-control state confirmation. Median latency and 95th-percentile latency were reported to describe typical response time and worst-case response behavior. These latency measures were used to characterize end-to-end response timing for the non-critical smart-load switching prototype. The structured condition-wise comparisons were organized according to standard experimental-design principles [30].

A pooled logistic regression was re-examined as an exploratory sensitivity analysis of the binary GPIO-confirmed execution outcome. However, some condition cells contained no observed failures, producing quasi-complete separation and unstable maximum-likelihood estimates for the pooled model. Because the pilot sample was small, the modality groups were non-overlapping, and failure events were sparse, odds ratios and regression-based confidence intervals were not retained for inferential interpretation. The statistical summary therefore emphasizes condition-wise execution success rates, Wilson 95% confidence intervals, and latency distributions. A larger within-subject dataset with more balanced outcome events would be required for stable multivariable effect estimation.

3. RESULTS AND DISCUSSION

This section presents the controller-side execution performance of the proposed edge-deployed dual-modality smart-load control system and the supplementary engineering profiling results. The human-trial results report GPIO-confirmed execution outcomes for the independent voice and gesture channels, whereas the profiling experiment characterizes computational and thermal behavior during integrated on-device operation.

3.1. Overall Execution Performance

System performance was evaluated using GPIO-confirmed relay-control state-transition outcomes at the controller interface. As summarized in Table 1, the voice modality achieved 54 successful executions out of 60 trials, corresponding to a success rate of 90.0% with a Wilson 95% confidence interval of 79.9–95.3%. The gesture modality achieved 63 successful executions out of 70 trials, also corresponding to 90.0%, with a Wilson 95% confidence interval of 80.8–95.1%. The two modalities therefore showed the same observed aggregate success rate in this pilot dataset. However, the overlapping confidence intervals should not be interpreted as evidence of statistical equivalence between the modalities, particularly given the small, non-overlapping participant groups and the limited number of execution failures. Accordingly, no claim of equivalent reliability or superiority of either modality is made from these aggregate results. The following subsections instead describe the condition-specific performance patterns observed for each modality under the tested conditions.

Table 1. Overall GPIO-confirmed relay-control state-transition performance

Modality	Trials	Success	Failure	Success Rate	Wilson 95% CI
Voice	60	54	6	90.0%	79.9–95.3%
Gesture	70	63	7	90.0%	80.8–95.1%

3.1.1. Voice modality under factorial conditions

The condition-level execution results for the voice modality are summarized in Table 2. In quiet conditions, the speech interface achieved 15/15 successful executions in D1 (0.5 m, quiet) and 15/15 in D3 (1.0 m, quiet). With background speech, the success rate was 13/15 in D2 (86.7%) and 11/15 in D4 (73.3%). Thus, within this pilot dataset, the lowest voice success rate occurred when background speech and the longer interaction distance were combined. Median latency increased from 1.0 s in D1 to 2.0 s in D4, while p95 increased from 1.5 s to 3.0 s. These condition-wise results are descriptive and are not used as population-level causal evidence; the observed direction is consistent with prior work on speech-recognition sensitivity to noise [24].

Table 2. Voice modality performance under factorial conditions based on GPIO-confirmed relay-control state transitions

Condition	Trials	Success (n/N)	Success (%)	Median Latency (s)	p95 (s)
D1 (0.5 m, Quiet)	15	15/15	100.0%	1.0	1.5
D2 (0.5 m, Background speech)	15	13/15	86.7%	1.5	2.5
D3 (1.0 m, Quiet)	15	15/15	100.0%	1.1	2.0

Condition	Trials	Success (n/N)	Success (%)	Median Latency (s)	p95 (s)
D4 (1.0 m, Background speech)	15	11/15	73.3%	2.0	3.0

3.1.2. Gesture modality under condition-coded comparison

The gesture results are summarized in Table 3 using the same D1–D4 condition codes for cross-modality comparison. Gesture control maintained high execution reliability in three conditions, with 17/18 successful trials in D1 (94.4%), 16/17 in D2 (94.1%), and 16/17 in D4 (94.1%). The largest performance drop occurred in D3, where the success rate decreased to 14/18 (77.8%). In contrast to the voice results, this pattern suggests that the observed gesture performance may be more sensitive to distance-related visual constraints. Background speech in the gesture trials was retained only as a contextual condition for cross-modality coding and is not interpreted as a direct perturbation of camera-based landmark detection.

Gesture latency medians ranged from 0.9 s to 1.3 s, and p95 values ranged from 1.6 s to 2.2 s across the four condition codes. These values are reported descriptively. Because background speech was only a contextual condition for gesture trials and the voice and gesture participant groups were non-overlapping, the latency results are not used to claim comparative robustness of gesture over voice.

Table 3. Gesture modality performance under condition codes based on GPIO-confirmed relay-control state transitions

Condition	Trials	Success (n/N)	Success (%)	Median Latency (s)	p95 (s)
D1	18	17/18	94.4%	0.9	1.6
D2	17	16/17	94.1%	1.0	1.8
D3	18	14/18	77.8%	1.2	2.0
D4	17	16/17	94.1%	1.3	2.2

3.1.3. Cross-modality descriptive comparison

Although both modalities achieved the same overall execution success rate of 90.0%, their lowest condition-wise success rates occurred under different condition codes. Voice execution success was lowest in D4 (11/15, 73.3%), whereas gesture execution success was lowest in D3 (14/18, 77.8%). Because background speech was only a contextual condition for gesture trials and the modality groups were non-overlapping, these descriptive differences are not interpreted as comparative causal evidence.

The condition-wise results show different descriptive patterns across the two input channels: voice performance was lowest when longer distance and background speech were combined, whereas gesture performance was lowest in the D3 distance-coded condition. A detailed failure taxonomy separating non-actuation, incorrect-channel actuation, and other controller-side mismatches was not established and should be included in future work. The worst-condition p95 latency was 3.0 s for voice and 2.2 s for gesture. These values are reported as engineering timing observations for this pilot prototype and are not interpreted as general HCI responsiveness targets.

3.2. Exploratory Model Check and Statistical Limitation

Re-examination of the pooled binary execution outcomes showed that a conventional maximum-likelihood logistic regression was not stable for this pilot dataset. In particular, some voice-condition cells contained no observed failures, which produced quasi-complete separation and prevented reliable finite estimation of several pooled model coefficients. The previously tabulated odds ratios and forest plot were therefore not retained in the final analysis.

This model limitation does not change the observed trial counts, execution-success rates, Wilson 95% confidence intervals, or latency summaries reported in Tables 1-3. Accordingly, the study interprets the human-trial results descriptively and makes no inferential claim of modality superiority, statistical equivalence, or causal environmental effects. Larger within-subject trials with more balanced failure observations are needed before multivariable odds-ratio estimates can be interpreted reliably.

3.3. Discussion from an Embedded Intelligent-Control Perspective

The results show why controller-side checking is important in a smart-load control system. Recognition accuracy alone does not show whether the relay-control command was executed as intended. In this study, each trial was checked from the logged GPIO state before and after command issuance. This made it possible to evaluate the system at the execution stage, not only at the perception stage.

The results also show that voice and gesture exhibited different failure patterns. Voice control was more affected when background speech and longer distance were combined. For gesture, background speech was only a contextual session label and should not be interpreted as a causal visual factor. The observed gesture variation is therefore discussed primarily in relation to interaction distance and trial-to-trial variability.

At the same time, the results should be interpreted with appropriate practical caution. In the present implementation, execution success was confirmed at the GPIO controller interface rather than through direct load-side electrical verification. In addition, the gesture conditions were reported using matched condition codes for parity with the speech experiments, but background speech served only as contextual environment on the gesture side. These aspects do not invalidate the observed trends, but they do define the scope of the present findings as controller-side reliability evidence rather than full end-to-end load verification.

3.4. Computational and System-Level Profiling

To address deployment-level behavior beyond recognition and execution outcomes, a separate integrated profiling run was conducted for approximately 10.5 min using the same Raspberry Pi 5 implementation. The profiler recorded Vosk audio-block processing time, MediaPipe hand-processing time, effective frame rate, GPIO software-update latency, system CPU utilization, Python-process CPU utilization, process resident memory, CPU temperature, and Raspberry Pi throttling status. This profiling experiment was supplementary to the 60 voice and 70 gesture human trials and was not used to recompute the execution-success rates.

Table 4. Engineering performance during the 10.5-min integrated profiling run

Metric	n	Mean	Median	p95
Vosk block processing (ms)	1,890	53.74	46.90	89.44
MediaPipe hand processing (ms)	8,551	66.18	72.15	99.69
Effective frame rate (frames/s)	8,550	14.57	11.51	—
GPIO software-update latency (ms)	8,551	0.0750	0.0688	0.1065
System CPU utilization (%)	631	32.57	32.30	35.75
Python process CPU utilization (%)	631	120.51	120.50	126.40
Python process RAM (MB)	631	425.16	427.55	444.63
CPU temperature (°C)	631	82.54	83.70	85.90

Table 4 shows that the median Vosk audio-block processing time was 46.90 ms (p95 89.44 ms), while median MediaPipe hand-processing time was 72.15 ms (p95 99.69 ms). Median GPIO software-update latency was only 0.0688 ms (p95 0.1065 ms), indicating that software GPIO state updates contributed negligibly compared with perception processing. The median effective frame rate was 11.51 frames/s during the sustained integrated run. System CPU utilization had a median of 32.30%, while the Python process used a median resident memory of 427.55 MB. Process CPU utilization exceeded 100% because the measurement aggregates CPU time across multiple cores.

During the sustained profiling run, median CPU temperature was 83.70 °C, p95 was 85.90 °C, and the maximum observed temperature was 87.55 °C. Non-zero throttling-status values were observed. Because the profiling experiment did not isolate the causes represented by the status values or independently manipulate cooling and power-supply conditions, these observations are reported descriptively and are not attributed to a specific thermal or power mechanism. GPU-core utilization was not instrumented in this experiment and is therefore not reported.

3.5. Summary of Findings

Across the pilot human trials, both voice and gesture achieved 90.0% GPIO-confirmed execution success. The supplementary 10.5-min profiling run further showed median Vosk and MediaPipe processing times of 46.90 ms and 72.15 ms, respectively, with a median GPIO software-update latency of 0.0688 ms. During sustained integrated operation, the median CPU temperature was 83.70 °C and non-zero throttling-status values were observed. These measurements are reported descriptively because the profiling run did not isolate thermal and power effects. The human-trial reliability results remain exploratory because of the small, non-overlapping participant groups.

4. CONCLUSION

This study developed and tested a Raspberry Pi 5-based smart-load control system using Thai speech commands and finger gestures. The system processed both input methods on the device and mapped them into

the same ten-command space. Instead of reporting only recognition performance, this study evaluated whether each command produced the intended GPIO relay-control state transition at the controller side.

Experimental results showed that both voice and gesture modalities achieved the same observed overall execution success rate of 90.0%, while exhibiting different condition-specific performance patterns. In the voice branch, the lowest observed success rate occurred in D4 (1.0 m with background speech), whereas in the gesture branch the lowest observed success rate occurred in D3 (1.0 m, quiet). Because background speech was retained only as a contextual condition in the gesture trials, these patterns are not interpreted as evidence of a common environmental effect across modalities. Because sparse failure events produced quasi-complete separation in a conventional pooled logistic model, inferential odds-ratio estimates were not retained; conclusions are therefore based on the descriptive success rates, Wilson confidence intervals, and latency summaries. In addition, the worst-condition p95 latency was 3.0 s for voice and 2.2 s for gesture, and is reported as an engineering timing observation for this non-critical prototype rather than as a general HCI responsiveness target.

This study contributes an execution-grounded reliability and engineering characterization of dual-modality edge control rather than a new recognition model or multimodal-fusion algorithm. The independent speech and gesture channels provide alternative input paths mapped to a common actuation layer, while the profiling results document computational load and observed temperature/throttling-status behavior during sustained operation on Raspberry Pi 5.

This study also has limitations. Execution was verified from GPIO state logs at the controller side rather than from direct electrical measurement at the load side. The human-trial sample was small (12 participants divided between non-overlapping modality groups), so the reported reliability patterns should be interpreted as exploratory engineering evidence rather than population-level HCI findings. The pooled logistic model was not retained because sparse failure events produced quasi-complete separation and unstable maximum-likelihood estimates. The system used independent speech and gesture channels without feature- or score-level fusion. GPU utilization was not instrumented in the profiling run. CPU temperature was recorded, and non-zero throttling-status values were observed; these measurements are reported descriptively because thermal and power effects were not isolated. The gesture interface provides on-screen recognition state, a stability/progress bar, an EXEC confirmation, and channel-state indicators, but equivalent auditory acknowledgment was not implemented for speech commands. No automated retry or recovery policy was evaluated. Moreover, the binary GPIO-confirmed endpoint is an engineering reliability measure and does not substitute for standardized HCI measures of perceived responsiveness, workload, learnability, or usability. Future work should include a larger within-subject cohort using both modalities, modality-specific disturbance factors, load-side electrical confirmation, explicit command acknowledgment and controlled retry/recovery mechanisms, standardized usability measures, controlled power/thermal benchmarking, and comparison with optimized lightweight inference pipelines.

ACKNOWLEDGMENTS

The authors would like to express their sincere gratitude to the Northeastern Vocational Education Institute 3 for providing support for this research and publication. The authors also would like to thank the Computer Technology Department and the Electronics Technology Program, Khon Kaen Technical College, for their valuable support and contribution to the development of this research.

CONFLICT OF INTEREST STATEMENT

The authors declare no conflict of interest

ETHICS AND PARTICIPANT CONSENT STATEMENT

Prior to participation, all participants were informed of the purpose of the study and the experimental procedures. All 12 participants were adults aged 18-60 years; no children or minors were involved in the study. Verbal informed consent was obtained from all individual participants before the experimental trials began; no written consent form was used. Participants did not disclose their identities as part of the study record. At the time of the study, the authors' unit did not have an institutional research ethics committee, and the pilot engineering evaluation was not submitted to an external ethics committee. The absence of formal ethics review is reported transparently and is not presented as an ethics approval, exemption, or waiver.

DECLARATION OF GENERATIVE AI USE

During the preparation and revision of this manuscript, the authors used ChatGPT (OpenAI) to assist with language refinement, academic phrasing, structural organization, consistency checking, and the programmatic preparation or checking of schematic and plotting layouts based exclusively on author-provided implementation details and numerical results. The tool was not used to create, alter, or fabricate raw research data or experimental outcomes. All AI-assisted text, figure layouts, and reference-format checks were independently reviewed and verified by the authors, who take full responsibility for the accuracy, integrity, and final content of the manuscript.

REFERENCES

- [1] C. Kim, D. Gowda, D. Lee, J. Kim, A. Kumar, S. Kim, A. Garg, and C. Han, "A review of on-device fully neural end-to-end automatic speech recognition algorithms," arXiv preprint arXiv:2012.07974, 2021, doi: <https://doi.org/10.48550/arXiv.2012.07974>.
- [2] X. Wang, Z. Tang, J. Guo, T. Meng, C. Wang, T. Wang, and W. Jia, "Empowering Edge Intelligence: A Comprehensive Survey on On-Device AI Models," ACM Computing Surveys, vol. 57, no. 9, Art. no. 228, 2025, doi: <https://doi.org/10.1145/3724420>.
- [3] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," IEEE Internet of Things Journal, vol. 3, pp. 637–646, 2016, doi: <https://doi.org/10.1109/JIOT.2016.2579198>.
- [4] C. Toma, M. Popa, B. Iancu, M. Doinea, and A. Pascu, "Edge Machine Learning for the Automated Decision and Visual Computing of the Robots, IoT Embedded Devices or UAV-Drones," Electronics, vol. 11, no. 21, Art. no. 3507, 2022, doi: <https://doi.org/10.3390/electronics11213507>.
- [5] C. Irugalbandara et al., "A secure and smart home automation system with speech recognition and power measurement capabilities," Sensors, vol. 23, no. 13, Art. no. 5784, 2023, doi: <https://doi.org/10.3390/s23135784>.
- [6] L. Lazzaroni, F. Bellotti, and R. Berta, "An embedded end-to-end voice assistant," Engineering Applications of Artificial Intelligence, vol. 136, Art. no. 108998, 2024, doi: <https://doi.org/10.1016/j.engappai.2024.108998>.
- [7] T. Hobbs and A. Ali, "Smart Home Control Using Real-Time Hand Gesture Recognition and Artificial Intelligence on Raspberry Pi 5," Electronics, vol. 14, no. 20, Art. no. 3976, 2025, doi: <https://doi.org/10.3390/electronics14203976>.
- [8] S. Kaewwichai, K. Thangthai, P. Tipakorn, and W. Limprasert, "Development and Evaluation of a Thai Automatic Speech Recognition Model Using the Conformer Model," Science & Technology Asia, vol. 30, no. 3, pp. 60–69, 2025, doi: <https://doi.org/10.14456/scitechasia.2025.47>.
- [9] M. Z. Iqbal and A. G. Campbell, "From luxury to necessity: Progress of touchless interaction technology," Technology in Society, vol. 67, Art. no. 101796, 2021, doi: <https://doi.org/10.1016/j.techsoc.2021.101796>.
- [10] E. Dritsas, M. Trigka, C. Troussas, and P. Mylonas, "Multimodal interaction, interfaces, and communication: A survey," Multimodal Technologies and Interaction, vol. 9, no. 1, Art. no. 6, 2025, doi: <https://doi.org/10.3390/mti9010006>.
- [11] Y. D. Du, F. Li, and X. Huang, "Research on multimodal human-robot interaction based on speech and gesture," Computers & Electrical Engineering, vol. 72, pp. 443–454, 2018, doi: <https://doi.org/10.1016/j.compeleceng.2018.09.014>.
- [12] W. Kaczmarek, J. Panasiuk, S. Borys, and P. Banach, "Industrial robot control by means of gestures and voice commands in off-line and on-line mode," Sensors, vol. 20, no. 21, Art. no. 6358, 2020, doi: <https://doi.org/10.3390/s20216358>.
- [13] P. Huang, I. Ullah, X. Wei, T. A. Ahanger, N. Hassan, and Z. H. Shah, "Towards Energy-Efficient and Low-Latency Voice-Controlled Smart Homes: A Proposal for Offline Speech Recognition and IoT Integration," arXiv preprint arXiv:2506.07494, 2025, doi: <https://doi.org/10.48550/arXiv.2506.07494>.
- [14] A. P. Rocha, M. Ketsmur, N. Almeida, and A. Teixeira, "An Accessible Smart Home Based on Integrated Multimodal Interaction," Sensors, vol. 21, no. 16, Art. no. 5464, 2021, doi: <https://doi.org/10.3390/s21165464>.
- [15] N. Malamas, K. Panayiotou, A. Karabatea, E. Tsardoulas, et al., "A multilayer architecture towards the development and distribution of multimodal interface applications on the edge," Sensors, vol. 24, no. 16, Art. no. 5199, 2024, doi: <https://doi.org/10.3390/s24165199>.
- [16] G. Diraco, G. Rescio, P. Siciliano, and A. Leone, "Review on human action recognition in smart living: Sensing technology, multimodality, real-time processing, interoperability, and resource-constrained deployment," Sensors, vol. 23, no. 11, Art. no. 5281, 2023, doi: <https://doi.org/10.3390/s23115281>.
- [17] M. M. H. Shuvo, S. K. Islam, J. Cheng, and B. I. Morshed, "Efficient acceleration of deep learning inference on resource-constrained edge devices: A review," Proceedings of the IEEE, vol. 111, no. 1, pp. 42–91, 2023, doi: <https://doi.org/10.1109/JPROC.2022.3226481>.
- [18] P. Mittal, "A comprehensive survey of deep learning-based lightweight object detection models for edge devices," Artificial Intelligence Review, vol. 57, Art. no. 242, 2024, doi: <https://doi.org/10.1007/s10462-024-10877-1>.
- [19] S. Heydari and Q. H. Mahmoud, "Tiny Machine Learning and On-Device Inference: A Survey of Applications, Challenges, and Future Directions," Sensors, vol. 25, no. 10, Art. no. 3191, 2025, doi: <https://doi.org/10.3390/s25103191>.

- [20] S. Di Leo, L. De Cicco, and S. Mascolo, "Real-Time Speech-to-Text on Edge: A Prototype System for Ultra-Low Latency Communication with AI-Powered NLP," *Information*, vol. 16, no. 8, Art. no. 685, 2025, doi: <https://doi.org/10.3390/info16080685>.
- [21] L. A. Pazmiño Ortiz, I. F. Maldonado Soliz, and V. K. Guevara Balarezo, "Advancing TinyML in IoT: A Holistic System-Level Perspective for Resource-Constrained AI," *Future Internet*, vol. 17, no. 6, Art. no. 257, 2025, doi: <https://doi.org/10.3390/fi17060257>.
- [22] A. González-Docasal, J. Alonso, J. Olaizola, M. Mendicute, M. P. Franco, A. del Pozo, D. Aguinaga, A. Álvarez, and E. Lleida, "Design and Evaluation of a Voice-Controlled Elevator System to Improve the Safety and Accessibility," *IEEE Open Journal of the Industrial Electronics Society*, vol. 5, pp. 1239–1250, 2024, doi: <https://doi.org/10.1109/OJIES.2024.3483552>.
- [23] A. El mrabet, A. Tber, M. Benaly, L. Hlou, and R. El gouri, "Unlocking Doors: A TinyML-Based Approach for Real-Time Face Mask Detection in Door Lock Systems," *Indonesian Journal of Electrical Engineering and Informatics*, vol. 11, no. 2, pp. 515–527, 2023, doi: <https://doi.org/10.52549/ijeet.v11i2.4756>.
- [24] A. Pervaiz, F. Hussain, H. Israr, M. A. Tahir, F. R. Raja, N. K. Baloch, F. Ishmanov, and Y. B. Zikria, "Incorporating Noise Robustness in Speech Command Recognition by Noise Augmentation of Training Data," *Sensors*, vol. 20, no. 8, Art. no. 2326, 2020, doi: <https://doi.org/10.3390/s20082326>.
- [25] D. Povey et al., "The Kaldi speech recognition toolkit," in *Proc. IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, 2011. <https://infoscience.epfl.ch/entities/publication/4a69ed57-34b0-41e8-aac7-5b4442e9cf0f/files>
- [26] Alphacephei, "Vosk offline speech recognition API." Available: <https://alphacephei.com/vosk/> (accessed Jan. 30, 2026).
- [27] C. Lugaresi, J. Tang, H. Nash, C. McClanahan, E. Uboweja, M. Hays, F. Zhang, C.-L. Chang, M. G. Yong, J. Lee, W.-T. Chang, W. Hua, M. Georg, and M. Grundmann, "MediaPipe: A framework for building perception pipelines," *arXiv preprint arXiv:1906.08172*, 2019, doi: <https://doi.org/10.48550/arXiv.1906.08172>.
- [28] F. Zhang, V. Bazarevsky, A. Vakunov, A. Tkachenka, G. Sung, C.-L. Chang, and M. Grundmann, "MediaPipe Hands: On-device real-time hand tracking," *arXiv preprint arXiv:2006.10214*, 2020, doi: <https://doi.org/10.48550/arXiv.2006.10214>.
- [29] E. B. Wilson, "Probable inference, the law of succession, and statistical inference," *Journal of the American Statistical Association*, vol. 22, no. 158, pp. 209–212, 1927, doi: <https://doi.org/10.1080/01621459.1927.10502953>.
- [30] D. C. Montgomery, *Design and Analysis of Experiments*, 9th ed. Hoboken, NJ, USA: John Wiley & Sons, 2017.

BIOGRAPHIES OF AUTHORS



Neres Supalugsanasuksakorn received the B.S.Tech.Ed. degree in Electrical Engineering from the Institute of Technology and Vocational Education, Thewa Campus, Bangkok, Thailand, in 1997. He obtained the M.Sc. degree in Information Technology from King Mongkut's University of Technology North Bangkok in 2007. He is a senior lecturer in the Computer Technology Department at Khon Kaen Technical College, Thailand. His interests include the Internet of Things, automatic control systems, programmable logic controller systems, and embedded systems. His email is naressu@kktech.ac.th.



Sakron Taanonjiew received the B.Ind.Tech. degree in Electronic Engineering from North Eastern College, Khon Kaen, Thailand, in 1997. He also completed the M.S.Ind.Ed. degree at King Mongkut's University of Technology Thonburi, Bangkok, Thailand. He is a senior lecturer in Electronic Technology at Khon Kaen Technical College, Northeastern Vocational Education Institute 3, Thailand. His academic interests are automatic control systems, electronic circuits and signals, embedded systems, and artificial intelligence. His email is Sakornth@kktech.ac.th.



Apinun Khonmanee earned the B.S.Ind.Ed. degree in Electronics and Telecommunications Engineering from the Pathumwan Institute of Technology, Bangkok, Thailand, in 2005. He received the M.S.Ind.Ed. degree in Electrical Engineering from King Mongkut's University of Technology North Bangkok in 2007. He is a senior lecturer in the Electronic Technology Program at Khon Kaen Technical College, Northeastern Vocational Education Institute 3, Thailand. His interests include electronic circuit systems, electrical signaling, and automated control. His email is apinunko@kktech.ac.th.



Saweth Hongprasit received the B.Ind.Tech. in electrical engineering from Mahanakorn University of Technology (MUT), M.Tech.Ed. in electrical technology from King Mongkut's Institute of Technology North Bangkok (KMITNB) and Ph.D. in Electrical and Computer Engineering from Mahasarakham University (MSU), Thailand in 1995, 2007 and 2012, respectively. He is a lecturer in the Department of Computer Engineering, Faculty of Engineering, Rajamangala University of Technology Isan, Khon Kaen Campus, Thailand. His research interests include CMOS circuit design, analog signal processing, active filters and oscillators. He can be contacted at email: saweth12@gmail.com



Wachirapunya Punyawong earned his B.Ind.Tech. in electronic engineering from North Eastern College, Khon Kaen, Thailand, in 1991. He later received an M.Sc. in Information Technology from Khon Kaen University in 2005 and a Ph.D. in Electrical and Computer Engineering from Mahasarakham University in 2016. He is now an Assistant Professor in the Electronic Technology Program at Khon Kaen Technical College, Northeastern Vocational Education Institute 3, Thailand. His work focuses on automatic control systems, remote control systems, embedded systems, and artificial intelligence. His email is Wachirapunya@kktech.ac.th.