

Toward a Distributed Cardiac Auscultation Simulator: Feasibility of MQTT-Synchronized Multi-Node Audio Playback

I Kadek Agus Wahyu Raharja^{1*}, Adi Pratama Putra P.²

¹ Computer Engineering, Universitas Warmadewa, Denpasar, Indonesia

² Medical Education, Universitas Warmadewa, Denpasar, Indonesia

ARTICLE INFO

Article history:

Initial submission

12-08-2026

Received in revised form

02-09-2026

Accepted 18-09-2026

Available online 30-09-2026

Keywords:

Distributed Audio

Synchronization, MQTT,

Cardiac Auscultation

Training, ESP32, Multi-

node Embedded Systems

DOI:

<https://doi.org/10.59356/smart-techno.v8i2.241>

ABSTRACT

Cardiac auscultation training traditionally relies on passive manikins or direct patient contact, both of which limit exposure to varied heart sound patterns and remain costly to acquire and maintain. This study investigated the feasibility of MQTT combined with NTP-based timestamp synchronization for coordinating audio playback across distributed low-cost microcontroller nodes, as a first step toward a distributed cardiac auscultation simulator. A prototype of two ESP32 microcontrollers, each driving two independent I2S audio channels through MAX98357A amplifiers, was connected to a Mosquitto MQTT broker on a laptop acting as a temporary central controller. Three scenarios were tested: synchronized playback on a single channel per node, on a second channel per node, and simultaneous two-channel playback within a single node. Across 25-30 valid trials per scenario, median command latency ranged from 44.5 to 124 milliseconds, and median inter-node synchronization error ranged from 243.5 to 369 milliseconds.

Simultaneous two-channel playback within one node revealed a median 9.7-second inter-channel delay, attributed to blocking audio playback in the current firmware. These results demonstrate that MQTT based synchronization is technically feasible on low-cost hardware, though accuracy falls short of a ± 50 millisecond target that informed by empirical audio-latency perception thresholds. A secondary finding was a substantial mismatch between the firmware's configured 16 kHz I2S sample rate and the audio files' likely true rate near 22.05 kHz, contributing to compressed multi-channel playback timing.

1. INTRODUCTION

Cardiac auscultation remains a fundamental diagnostic skill in medical education, yet conventional training methods present persistent limitations. Passive manikins reproduce a narrow range of heart sounds and cannot represent pathological variation across multiple auscultation points simultaneously, while direct patient examination is constrained by case availability and ethical considerations for novice learners. Commercial auscultation simulators exist but are typically priced beyond the reach of many medical education institutions, particularly in resource-limited settings.

Several efforts have addressed this affordability gap through open-source and low-cost hardware. A low-cost AI-empowered stethoscope has been developed, pairing a lightweight diagnostic model with inexpensive single-board computers (Zhang et al., 2023). An open-access 3D-printed digital stethoscope designed for socially distanced tele-auscultation during the COVID-19 pandemic has also been reported (Nuttall & Teng, 2021). Remote auscultation has been evaluated as a viable alternative to in-person examination during patient isolation (Haskel et al., 2022), and a review of stethoscope technology has noted a clear trend toward digital and networked designs (Seah et al., 2023). These works collectively demonstrate growing acceptance of low-cost, technology-enhanced auscultation tools, but none address

the specific challenge of synchronized playback across multiple physical points on a single training platform.

The need for affordable simulation extends beyond auscultation specifically. A systematic review covering healthcare training tools in low- and middle-income countries found consistent evidence that technology-enhanced, low-cost simulation improves learning outcomes even when compared against traditional methods, though long-term clinical impact remains understudied (Ferreira et al., 2025). Parallel developments in Internet of Things (IoT) based healthcare systems, such as an industrial smart healthcare architecture (Zang & You, 2023), a fall-risk monitoring system for Parkinson's patients (Bhattacharjee et al., 2023), and a cardiovascular risk prediction platform (Islam et al., 2023), illustrate a broader pattern: low-cost microcontroller-based sensing and communication infrastructure is increasingly viable for healthcare-adjacent applications previously requiring specialized, expensive equipment.

A central technical requirement for a multi-point auscultation simulator is reliable synchronization of audio playback across physically separated nodes, since heart and lung sounds recorded at different anatomical locations must play in coordinated fashion to represent a coherent physiological scenario. MQTT, a lightweight publish-subscribe messaging protocol, has been widely adopted for this class of problem in IoT contexts, with comparative evaluations showing its suitability for latency-sensitive, resource-constrained communication relative to alternative protocols such as CoAP and WebSocket (Silva et al., 2021; Wytrębowicz et al., 2021). In healthcare-specific contexts, MQTT has been used to implement low-latency, reliable patient monitoring (Alshammari, 2023), and extended with lightweight encryption for secure remote diagnosis using ESP32 hardware (Kumar et al., 2025). These studies establish MQTT as a technically sound candidate for the present application, though none evaluate it specifically for audio playback synchronization.

This study evaluates the core synchronization mechanism of such a distributed audio architecture using a two-node prototype, isolating the communication and timing layer from the broader system. The objective was to determine whether MQTT-based communication, paired with NTP timestamp synchronization, can achieve playback coordination accurate and consistent enough for multi-point auscultation training, and to identify specific firmware limitations relevant to further development of this approach.

2. LITERATURE REVIEW

Prior work relevant to this study spans two domains: performance evaluation of MQTT in latency-sensitive, resource-constrained systems, and the deployment of low-cost microcontroller networks for coordinated or synchronized data collection.

MQTT Protocol Performance

MQTT, CoAP, and OPC UA have been compared across representative IoT scenarios, concluding that MQTT combined lightweight overhead with acceptable latency for near-real-time applications, though its performance degraded as network conditions worsened (Silva et al., 2021). A similar conclusion has been reached for MQTT, WebSocket, and additional messaging protocols, characterizing MQTT as a pragmatic default choice for constrained IoT devices exchanging small, frequent payloads (Wytrębowicz et al., 2021). MQTT over TCP has been directly measured against MQTT over WebSocket using ESP32 hardware for a digital-twin application, reporting mean latencies of 290.5 and 342.0 milliseconds respectively, with standard deviations exceeding 190 milliseconds in both cases (Amirkhanov et al., 2025). This finding is directly comparable to the present study's testbed and suggests that latency variability of several hundred milliseconds should be expected as a baseline characteristic of consumer WiFi combined with MQTT, rather than as evidence of implementation error.

ESP32-Based Multi-Node Systems

Several studies specifically characterize ESP32 performance in synchronized or multi-node configurations. A fully functional ESP32-based sensor network has been evaluated,

reporting that the platform maintained stable high-rate sampling but noted variability originating from the built-in analog-to-digital converter rather than from network transmission itself (Espinosa-Gavira et al., 2024). General design and implementation considerations for ESP32-based IoT devices have been documented (Hercog et al., 2023), and the energy consumption implications of running MQTT-based real-time monitoring on ESP32 hardware have been specifically measured, a consideration relevant to any eventual battery-powered or field-deployed version of the present system (Syahmi Md Dzahir & Seng Chia, 2023). A five-node ESP32 architecture coordinated through a lightweight publish-subscribe protocol for home automation has also been described, demonstrating that multi-node ESP32 deployments using MQTT are practically achievable with off-the-shelf components, consistent with the architecture adopted here (Pérez-Ayala et al., 2026). A comparable multi-node wireless approach has been applied to wearable motion sensing, underscoring the broader applicability of low-cost ESP32 networks beyond the present healthcare-training context (Dobra et al., 2025).

MQTT in Healthcare Applications

In healthcare-specific applications, an MQTT-based patient vital-sign monitoring system prioritizing low latency has been implemented (Alshammari, 2023), and this direction has been extended by adding lightweight one-time-pad encryption to MQTT-based remote diagnosis without introducing meaningful additional overhead on ESP32 hardware (Kumar et al., 2025). Controlled comparisons of MQTT power and timing characteristics against alternative transport mechanisms have also been conducted, concluding that MQTT remained favorable for intermittent, small-payload transmission typical of embedded sensing and control tasks (Derlatka & Kopniak, 2023; Jara Ochoa et al., 2023). Broader IoT-healthcare deployments illustrate that similar low-cost, networked microcontroller architectures are already being applied to disease monitoring and risk-prediction tasks, reinforcing the practical viability of the approach for medical-adjacent applications generally (Bhattacharjee et al., 2023; Islam et al., 2023; Zang & You, 2023).

MQTT Latency Components

Round-trip latency in MQTT-based IoT communication has been formally decomposed into propagation, transmission, processing, and queueing components at each network hop (Abid et al., 2025). This decomposition indicates that even a single WiFi hop introduces multiple simultaneous delay sources, providing a theoretical basis for the latency magnitude and run-to-run variability observed in the present study (Table 2), despite the short physical distances between nodes and controller.

Research Gap

Despite this substantial body of work characterizing MQTT performance and ESP32-based multi-node systems individually, none of the reviewed studies specifically evaluate MQTT-coordinated audio playback synchronization, nor do they address the particular requirement of a cardiac auscultation training platform: coordinated, low-latency triggering of pre-recorded physiological sounds across multiple physically distributed speakers. This represents the specific gap addressed by the present study.

3. METHOD

System Architecture

The system architecture follows a hybrid centralized-distributed design, in which a central controller coordinates synchronized playback across independent ESP32 audio nodes over MQTT.

Figure 1 shows the overall system architecture. A central controller, running a local Mosquitto MQTT broker, publishes synchronized playback commands to two or more ESP32 audio nodes. Each node independently synchronizes its clock against a public NTP server,

receives commands specifying which channel and audio file to play along with a target synchronization timestamp, and reports timing status back to the controller after playback.

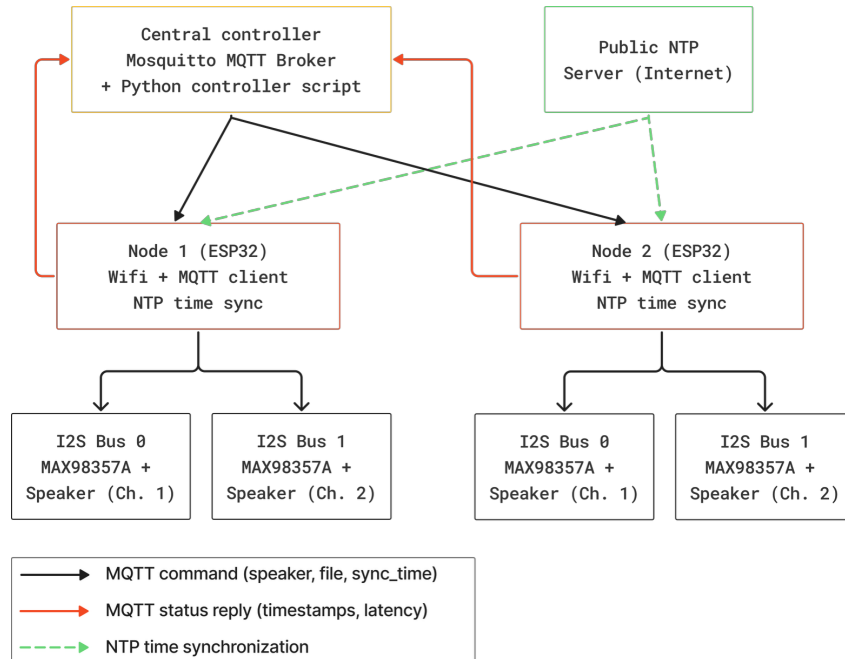


Figure 1. System architecture: MQTT command/status flow between the central controller and audio nodes, NTP time synchronization, and the node-to-speaker signal path.

Figure 2 details the firmware logic executed on each node. After connecting to WiFi and synchronizing its clock via NTP, a node connects to the MQTT broker and waits for an incoming command. Upon receiving one, it parses the target speaker channel, audio file name, and synchronization timestamp, then busy-waits until its local clock reaches that timestamp before triggering playback and reporting timing status. This loop repeats for the duration of the experiment.

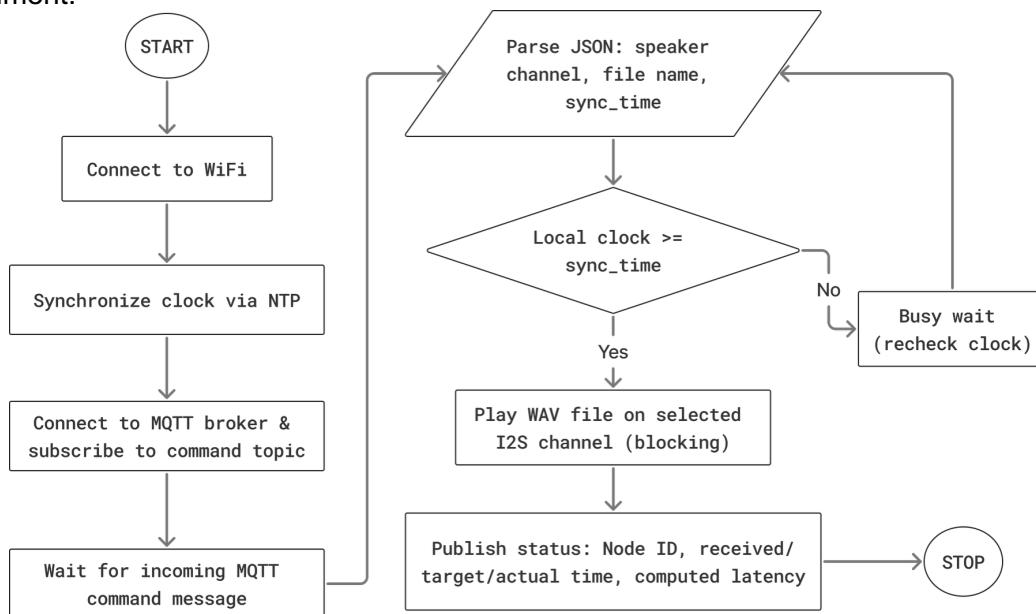


Figure 2. Firmware operation flowchart executed on each ESP32 audio node.

Figure 3 illustrates the resulting message sequence between the controller and two nodes for a single synchronized playback trial, showing how the shared synchronization timestamp T is distributed via MQTT, used locally by each node to time its own playback, and

then used by the controller to compute the latency and synchronization-error metrics defined in Equations 1 and 2 below.

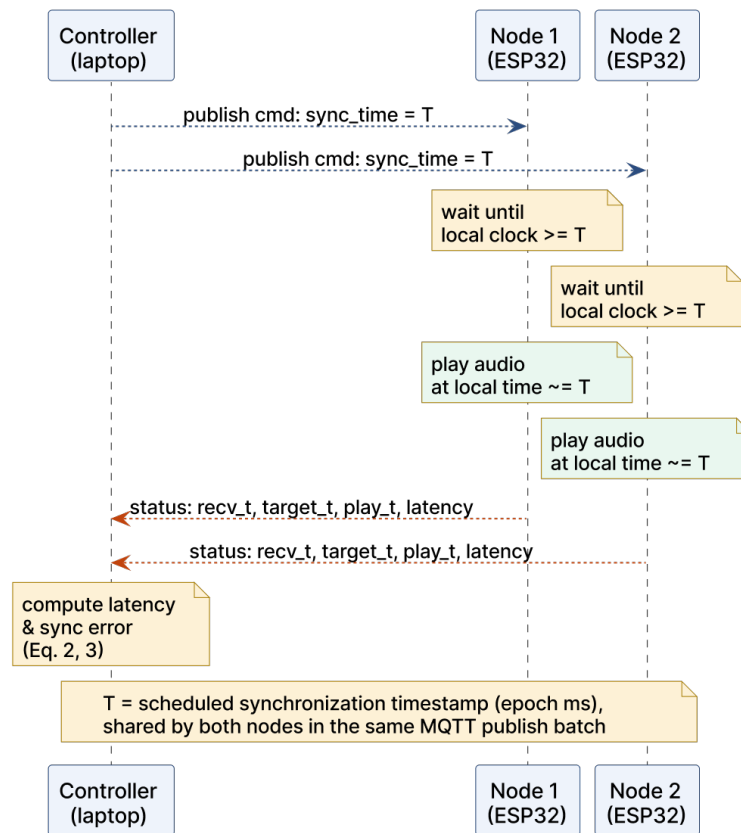


Figure 3. Sequence diagram of one synchronized playback trial, from command publication to status collection.

Hardware Implementation

Each audio node is built around a single ESP32 development board driving two independent audio channels, each channel routed through a separate I2S bus to its own MAX98357A Class-D amplifier module (Figure 4). This dual-bus configuration was adopted specifically to avoid the channel volume imbalance commonly reported when multiple I2S outputs share bus resources. A custom printed circuit board design consolidates three such ESP32 modules and six MAX98357A amplifiers per node group for denser multi-point deployments (Figure 4b). The experiments reported here were conducted on an equivalent breadboard implementation of the same circuit, containing two independently powered ESP32 units wired identically to the schematic in Figure 4 (Figure 5).

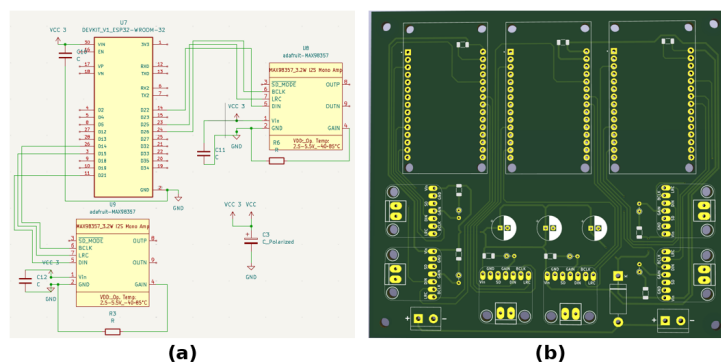


Figure 4. Circuit schematic and layout of the custom printed circuit board, showing dual I2S buses driving two independent MAX98357A amplifier modules.

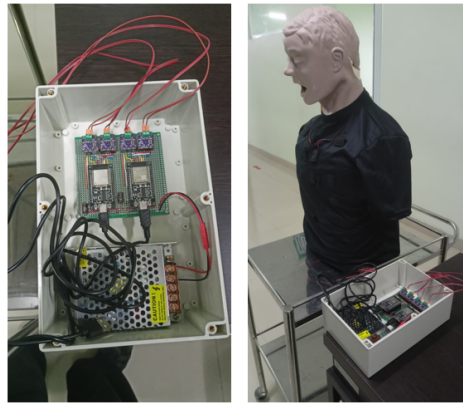


Figure 5. Enclosed prototype housing containing two ESP32 nodes, positioned alongside a training manikin.

Firmware and Controller Implementation

Firmware on each node performs four functions: connecting to a WiFi network and an MQTT broker using the PubSubClient library; synchronizing an internal epoch-millisecond clock against a public NTP server; hosting a lightweight web interface for uploading WAV audio files to onboard flash storage via SPIFFS; and subscribing to a per-node MQTT command topic carrying a JSON payload specifying which speaker channel to activate, which uploaded file to play, and a target synchronization timestamp. Upon receiving a command, the node busy-waits until its local clock reaches the specified timestamp before beginning playback, then publishes a status message reporting the command-received time, the target time, and the actual playback-start time, all expressed as 64-bit epoch-millisecond values to avoid the 32-bit integer overflow that produces spurious multi-day timing errors when computed at millisecond resolution over realistic epoch values.

The central controller ran a local Mosquitto MQTT broker and a Python script that published synchronized play commands to one or more node topics with a fixed lead time before the intended play instant, then subscribed to and logged each node's status replies for offline analysis.

Experimental Scenarios

Three experimental scenarios were conducted to evaluate synchronization performance under increasing timing demand. In the baseline scenario, both nodes played the same audio file on their first channel simultaneously, repeated for 20 trials. In the second scenario, both nodes played the same file on their second channel, repeated for 30 trials. In the third scenario, a single node was commanded to play different audio files on both of its channels simultaneously, repeated for 15 trials, isolating the effect of the firmware's blocking playback implementation on channels sharing a single microcontroller. Across all scenarios, WAV files of matched nominal duration (14 seconds) were used, and the interval between repeated commands was set to exceed the expected playback duration to prevent command backlog from accumulating between trials.

This third scenario was deliberately conducted using the firmware's existing, unmodified playback implementation. Its purpose was diagnostic rather than evaluative: to isolate and characterize the specific limitation affecting same-node multi-channel playback, rather than to represent the expected performance of a deployment-ready simulator.

4. RESULT AND DISCUSSION

Functional Testing

Before analyzing timing accuracy, the prototype's core functions were verified using black-box functional testing, confirming that each stage of the communication and playback pipeline operated as designed prior to quantitative analysis (Table 1).

Table 1. Functional (Black-box) Testing Results for The Prototype's Core Operations.

No	Tested Function	Test Method	Expected Result	Result	Status
1	WiFi connection	Power on node, observe Serial Monitor log	Node connects to the configured WiFi network	Connected	Successful
2	NTP time synchronization	Observe node epoch-ms clock after boot	Clock synchronizes to current UTC time	Synchronized (one transient anomaly observed and auto-corrected; see Discussion)	Successful
3	MQTT broker connection & subscription	Observe node and broker logs	Node connects to broker and subscribes to its command topic	Connected and subscribed	Successful
4	Command reception & parsing	Publish a JSON command from the controller	Node parses speaker, file, and sync_time fields correctly	Parsed correctly	Successful
5	Synchronized audio playback trigger	Send command with a future sync_time	Node begins WAV playback at the scheduled time	Playback triggered (timing accuracy analyzed in Table 2)	Successful
6	Status reporting	Observe controller-side MQTT log after playback	Node publishes timestamps and computed latency	Status received and logged	Successful
7	Web-based WAV upload	Open node IP address in a browser, upload a file	File is stored to SPIFFS and listed on the page	File uploaded and listed	Successful

Latency and Synchronization Error Analysis

Table 2. Command Latency and Inter-Node Synchronization Error by Scenario.

Scenario	n	Median Latency	Mean Latency	SD Latency	Median Sync Error	Mean Sync Error	SD Sync Error
Baseline (Channel 1)*	25	124	351.1	637.6	369	571.8	744.9
Second channel (Channel 2)	30	44.5	182.6	259.6	243.5	285.8	260.1
Simultaneous two-channel (same node)	15	—	—	—	9704	9716.1	655.7

*Four trials in the baseline scenario were excluded due to an NTP synchronization anomaly (see discussion below); n reflects the remaining valid trials.

Command latency for a given trial is defined as the difference between a node's actual playback-start time and the scheduled synchronization target:

$$Li = T_{play,i} - T_{target,i} \quad (1)$$

where $T_{play,i}$ is the epoch-millisecond time node i began playback and $T_{target,i}$ is the scheduled synchronization timestamp for that trial. Inter-node (or inter-channel) synchronization error for a trial involving N simultaneous playback events is defined as the spread between the earliest and latest actual playback start:

$$E_{run} = \max(T_{play}) - \min(T_{play}) \quad (2)$$

As an illustrative example drawn from the baseline dataset, one trial scheduled for a synchronization target of 1,786,354,730,316 milliseconds resulted in Node 1 beginning playback at 1,786,354,730,411 milliseconds and Node 2 at 1,786,354,730,780 milliseconds, yielding individual latencies of 95 and 464 milliseconds respectively (Equation 1) and an inter-node synchronization error of 369 milliseconds (Equation 2), matching the scenario median reported in Table 2.

Table 2 summarizes command latency and synchronization error across the three test scenarios. In the baseline scenario, four of 29 collected trials exhibited implausible timing values on the order of hundreds of billions of milliseconds, traced to an intermittent malformed response from the public NTP server that temporarily corrupted one node's internal clock for approximately one resynchronization interval (up to 60 seconds) before self-correcting. These trials were excluded from analysis and are noted here as a documented limitation of relying on public NTP infrastructure for synchronization accuracy at the millisecond scale; a production deployment would likely require either a dedicated local NTP source or a validated clock-sanity check before accepting a synchronization timestamp.

Public NTP was adopted in this study primarily for its zero-infrastructure simplicity, requiring no additional hardware or network services beyond what each node already needed for internet connectivity; this convenience is traded against the reliability risk demonstrated by the observed clock anomaly. Alternatives such as a dedicated local NTP server, IEEE 1588 Precision Time Protocol (PTP), or a GPS-disciplined clock which typically achieve microsecond level accuracy compared to NTP's millisecond-level accuracy over the public internet (Raman et al., 2020), would likely reduce or eliminate this class of anomaly at the cost of additional infrastructure, and represent natural directions for a production deployment. Notably, the four anomalous trials in the baseline scenario, and a substantially reduced command delivery success rate observed for the same physical node in supplementary dual-channel trials, were both associated with Node 1, suggesting a node-specific factor (such as WiFi signal quality or power stability) that warrants physical inspection independent of the synchronization-method question.

Figure 6 presents the baseline-scenario data with these anomalous trials already excluded. Median command latency ranged from 44.5 milliseconds in the second-channel scenario to 124 milliseconds in the baseline scenario, with corresponding median inter-node synchronization error of 243.5 and 369 milliseconds respectively (Figures 6 and 7). This magnitude of error is broadly consistent with the 290 to 342 millisecond mean latencies reported for MQTT over consumer WiFi using comparable ESP32 hardware (Syahmi Md Dzahir & Seng Chia, 2023), and with the multi-component latency decomposition described in (Abid et al., 2025), in which propagation, transmission, processing, and queueing delay each contribute independently to total round-trip time; the observed variability therefore reflects a general characteristic of best-effort WiFi and MQTT transport rather than a defect specific to the present implementation.

Baseline Scenario (Channel 1, Both Nodes) — Command Latency and Inter-Node Synchronization Error

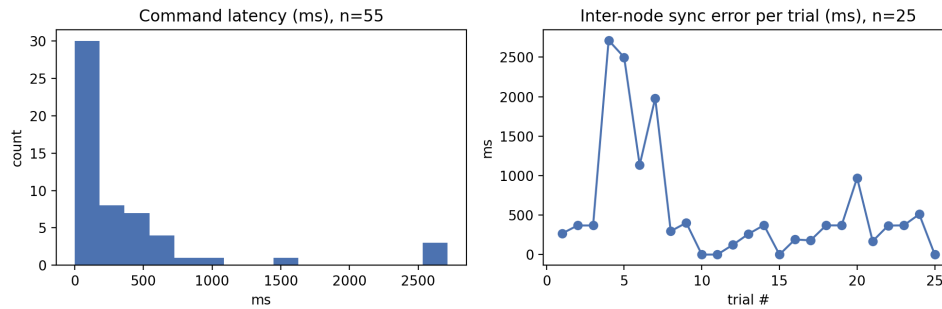


Figure 6. Command latency distribution and per-trial inter-node synchronization error, baseline scenario (Channel 1), NTP-anomaly trials excluded.

Second-Channel Scenario (Channel 2, Both Nodes) — Command Latency and Inter-Node Synchronization Error

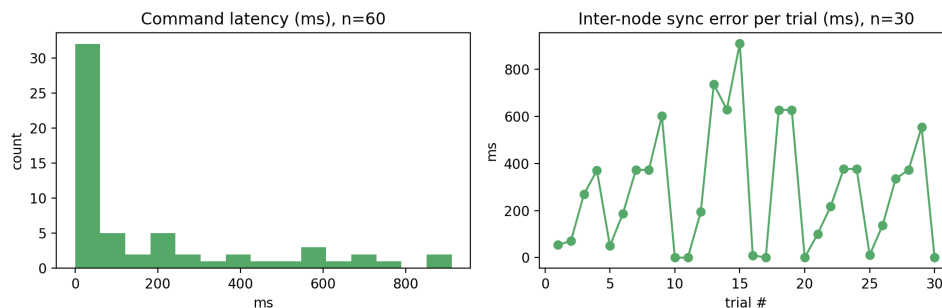


Figure 7. Command latency distribution and per-trial inter-node synchronization error, second-channel scenario (Channel 2).

Both latency and synchronization-error distributions were right-skewed, with standard deviations exceeding or approaching their respective means (Table 2), indicating that a small number of high-latency outlier trials disproportionately influenced the mean; median values are therefore reported as the primary descriptive statistic, consistent with the skewed-distribution pattern visible in Figures 6 and 7.

Because worst-case behavior is more consequential than central tendency for a training-scenario deployment, the 90th, 95th, and 99th percentiles were additionally computed for each scenario. For the baseline scenario, latency and synchronization-error 95th percentiles were 1,861 and 2,395 milliseconds respectively, with 99th percentiles of 2,714 and 2,662 milliseconds. For the second-channel scenario, the corresponding 95th percentiles were 715 and 688 milliseconds, and 99th percentiles were 907 and 861 milliseconds. In both cases the tail values substantially exceed the median, confirming that a minority of trials experienced markedly degraded synchronization and that median-only reporting understates worst-case risk.

Effect of Simultaneous Multi-Channel Playback

The third scenario, in which a single node played two different audio files simultaneously on its two channels, produced a substantially different result: a median inter-channel timing gap of 9,704 milliseconds, with the two channels never beginning playback within less than 8,702 milliseconds of one another across 15 trials (Figure 8). Applying Equation 2 to one representative trial from this scenario, a synchronization target of 1,786,356,511,251 milliseconds produced a Channel 1 playback start of 1,786,356,511,281 milliseconds and a Channel 2 playback start of 1,786,356,520,974 milliseconds, giving an inter-channel gap of 9,693 milliseconds, close to the scenario median. This gap is attributable to the current firmware's blocking implementation of the I2S playback routine, which processes the two channel commands sequentially rather than concurrently; the second channel's

playback cannot begin until the first channel's full audio buffer has finished writing to the I2S peripheral.

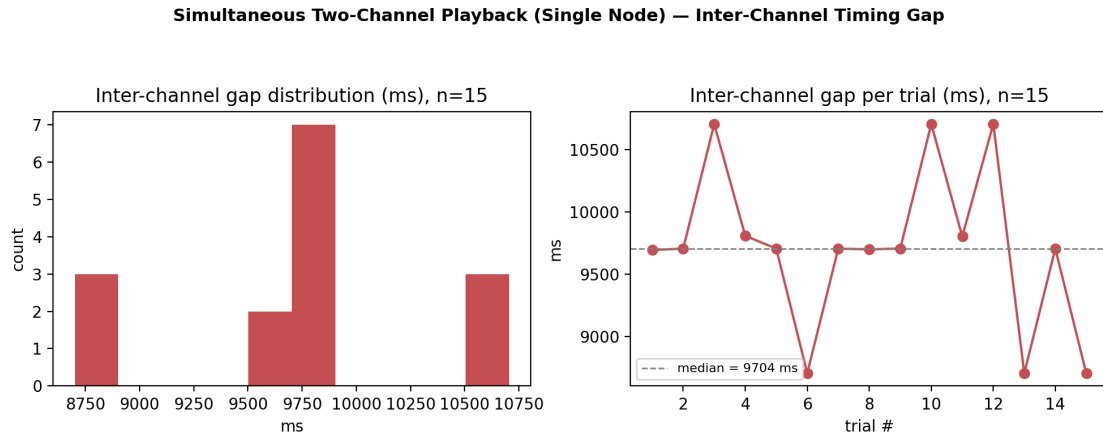


Figure 8. Inter-channel timing gap for simultaneous two-channel playback within a single node.

Because the measured gap is shorter than the nominal 14-second file duration, the firmware's fixed 16-kilohertz sample-rate configuration likely does not match the true sample rate of the source WAV files, causing faster-than-intended playback. This can be estimated directly from the observed timing. If the configured sample rate is scaled by the ratio of nominal to measured duration, the implied true sample rate is:

$$f_{\text{actual}} = f_{\text{configured}} \times \frac{D_{\text{nominal}}}{D_{\text{measured}}} \quad (3)$$

where $f_{\text{configured}}$ is the 16,000 hertz value currently set in firmware, D_{nominal} is the 14,000ms nominal file duration, and D_{measured} is the 9,704ms median inter-channel gap observed in the third scenario, taken as an approximation of true single-channel playback duration. Substituting these values gives $f_{\text{actual}} = 16,000 \text{ Hz} \times (14,000 / 9,704) = 16,000 \text{ Hz} \times 1.443 = 23,083 \text{ Hz}$, close to the common 22,050 hertz consumer audio sample rate. This suggests the source WAV files were most likely recorded or exported at approximately 22.05 kilohertz rather than 16 kilohertz, and that correcting the firmware's I2S clock configuration to match should independently improve measured playback-duration accuracy in future testing, separate from the blocking-playback correction discussed below.

Implications for Synchronization Accuracy

Relative to a ± 50 millisecond synchronization target considered desirable for perceptually coherent multi-point auscultation playback, the present results fall short by a factor of five to seven for cross-node synchronization, and represent an outright failure to meet the target for same-node multi-channel playback under the current firmware. The ± 50 millisecond target itself is grounded in empirical work on human sensitivity to audio-timing deviation, which has reported a mean just-noticeable-difference for audio latency of approximately 49 milliseconds under near-zero baseline conditions (Schmid et al., 2024). This gap should not be interpreted as evidence against the overall architectural approach; rather, it identifies three specific, addressable engineering priorities for subsequent development: first, replacing blocking I2S playback with a non-blocking, task-based implementation, for example using separate FreeRTOS tasks per channel, to eliminate the sequential-playback delay observed in the third scenario; second, either adopting a more precise time-synchronization mechanism than public-pool NTP, or introducing clock-sanity validation to guard against the anomaly observed in the baseline scenario; and third, correcting the firmware's I2S sample-rate configuration per the calculation above.

Source level inspection clarifies the mechanism underlying the first priority above: the WAV-playback routine performs a fully blocking I2S write within the main loop, during which the MQTT client's message-processing routine is never invoked. A command for the second channel, published with negligible delay after the first, therefore remains unprocessed until the first channel's playback completes, producing a gap that tracks the file's playback duration, the same mechanism likely also explains the reduced command-delivery reliability observed on one node in supplementary trials, since MQTT keep-alive handling is similarly suspended during playback. The proposed fix moves I2S playback into per-channel FreeRTOS tasks with independent command queues, allowing the main loop to service MQTT continuously regardless of playback state; this refactoring is scoped for subsequent development.

5. CONCLUSION

This study evaluated the feasibility of MQTT-based, NTP-synchronized audio playback coordination for a distributed cardiac auscultation training application. The two-node prototype confirmed that the core synchronization mechanism functions correctly, though observed accuracy does not yet meet a ± 50 millisecond target. Two specific, firmware-addressable limitations were identified: reliance on public NTP infrastructure and the current blocking audio-playback routine. Future work will implement non-blocking, task-based playback, evaluate alternative time-synchronization strategies, and extend testing to a larger multi-node configuration.

6. ACKNOWLEDGEMENT

This research was conducted as part of a Penelitian Dosen Pemula grant awarded by the DPPM, Universitas Warmadewa. The authors thank student research assistants Putu Gede Parama Wikananda and I Putu Galih Surya Saputra for their contributions to hardware assembly and testing.

7. REFERENCES

- Abid, A., Jazib, M., & Riaz, M. (2025). RCD-IoT: Enabling Industrial Monitoring and Control with Resource-Constrained Devices Under High Packet Transmission Rates. *ArXiv Preprint*. <https://arxiv.org/abs/2501.07895>
- Alshammari, H. H. (2023). The internet of things healthcare monitoring system based on MQTT protocol. *Alexandria Engineering Journal*, 69, 275–287. <https://doi.org/10.1016/j.aej.2023.01.065>
- Amirkhanov, B., Amirkhanova, G., Kunelbayev, M., Adilzhanova, S., & Tokhtassyn, M. (2025). Evaluating HTTP, MQTT over TCP and MQTT over WEBSOCKET for digital twin applications: A comparative analysis on latency, stability, and integration. *International Journal of Innovative Research and Scientific Studies*, 8(1), 679–694. <https://doi.org/10.53894/ijirss.v8i1.4414>
- Bhattacharjee, P., Biswas, S., Chattopadhyay, S., Roy, S., & Chakraborty, S. (2023). Smart Assistance to Reduce the Fear of Falling in Parkinson Patients Using IoT. *Wireless Personal Communications*, 130(1), 281–302. <https://doi.org/10.1007/s11277-023-10285-8>
- Derlatka, G., & Kopniak, P. (2023). Comparative analysis of transfer protocols asynchronous messages on systems queuing. *Journal of Computer Sciences Institute*, 26, 29–32. <https://doi.org/10.35784/jcsi.3074>
- Dobra, V.-A., Dobra, I.-M., & Folea, S. (2025). Real-Time Paddle Stroke Classification and Wireless Monitoring in Open Water Using Wearable Inertial Nodes. *Sensors*, 25(17), 5307. <https://doi.org/10.3390/s25175307>
- Espinosa-Gavira, M. J., Agüera-Pérez, A., Palomares-Salas, J. C., Sierra-Fernandez, J. M., Remigio-Carmona, P., & González de-la-Rosa, J. J. (2024). Characterization and Performance Evaluation of ESP32 for Real-time Synchronized Sensor Networks. *Procedia Computer Science*, 237, 261–268. <https://doi.org/10.1016/j.procs.2024.05.104>
- Ferreira, J. M. G., Generoso, J. R., Marra, A. R., Deliberato, R. O., Dewan, M., Loeb, D., Real, F., Collins, K., Ely, K., Zackoff, M., Lopes, G. de O. V., Pardo, I., Gutfreund, M. C., Callado, G. Y., & Mendonça, E. A. (2025). Effectiveness of Low-cost, Technology-enhanced Simulation Training for

- Healthcare Training in Low—and Middle-income Countries (LMICs): A Systematic Literature Review. *Journal of General Internal Medicine*, 40(16), 4040–4050. <https://doi.org/10.1007/s11606-025-09794-y>
- Haskel, O., Itelman, E., Zilber, E., Barkai, G., & Segal, G. (2022). Remote Auscultation of Heart and Lungs as an Acceptable Alternative to Legacy Measures in Quarantined COVID-19 Patients—Prospective Evaluation of 250 Examinations. *Sensors*, 22(9), 3165. <https://doi.org/10.3390/s22093165>
- Hercog, D., Lerher, T., Truntič, M., & Težak, O. (2023). Design and Implementation of ESP32-Based IoT Devices. *Sensors*, 23(15), 6739. <https://doi.org/10.3390/s23156739>
- Islam, M. N., Raiyan, K. R., Mitra, S., Mannan, M. M. R., Tasnim, T., Putul, A. O., & Mandol, A. B. (2023). Predictis: an IoT and machine learning-based system to predict risk level of cardio-vascular diseases. *BMC Health Services Research*, 23(1), 171. <https://doi.org/10.1186/s12913-023-09104-4>
- Jara Ochoa, H. J., Peña, R., Ledo Mezquita, Y., Gonzalez, E., & Camacho-Leon, S. (2023). Comparative Analysis of Power Consumption between MQTT and HTTP Protocols in an IoT Platform Designed and Implemented for Remote Real-Time Monitoring of Long-Term Cold Chain Transport Operations. *Sensors*, 23(10), 4896. <https://doi.org/10.3390/s23104896>
- Kumar, N. R., Krishnan, R. B., Vairavasundaram, S., Manikandan, G., V. I., & Ravi, L. (2025). An IoT based remote medical diagnosis system using one time pad cipher over MQTT protocol. *Scientific Reports*, 15(1), 42117. <https://doi.org/10.1038/s41598-025-26208-5>
- Nuttall, C., & Teng, C.-C. (2021). Design and Implementation of a Low Cost, Open Access Digital Stethoscope for Social Distanced Medical Care and Tele Auscultation. 2021 5th International Conference on Medical and Health Informatics, 129–133. <https://doi.org/10.1145/3472813.3473178>
- Pérez-Ayala, P. C., Salvador-Boza, L. D., Lopez-Lopez, M. A., Soto-Oroya, C. A., Cabana-Cáceres, M. R., & Castro-Vargas, C. (2026). A Low-Cost Multi-Node Architecture with ESP32 Microcontrollers for Assisted Home Automation of Elderly and Disabled Residents. *EAI Endorsed Transactions on Internet of Things*, 11. <https://doi.org/10.4108/eetiot.12871>
- Raman, C., Tan, S., & Hung, H. (2020). A Modular Approach for Synchronized Wireless Multimodal Multisensor Data Acquisition in Highly Dynamic Social Settings. *Proceedings of the 28th ACM International Conference on Multimedia*, 3586–3594. <https://doi.org/10.1145/3394171.3413697>
- Schmid, A., Ambros, M., Bogon, J., & Wimmer, R. (2024). Measuring the Just Noticeable Difference for Audio Latency. *Audio Mostly 2024 - Explorations in Sonic Cultures*, 325–331. <https://doi.org/10.1145/3678299.3678331>
- Seah, J. J., Zhao, J., Wang, D. Y., & Lee, H. P. (2023). Review on the Advancements of Stethoscope Types in Chest Auscultation. *Diagnostics*, 13(9), 1545. <https://doi.org/10.3390/diagnostics13091545>
- Silva, D., Carvalho, L. I., Soares, J., & Sofia, R. C. (2021). A Performance Analysis of Internet of Things Networking Protocols: Evaluating MQTT, CoAP, OPC UA. *Applied Sciences*, 11(11), 4879. <https://doi.org/10.3390/app11114879>
- Syahmi Md Dzahir, M. A., & Seng Chia, K. (2023). Evaluating the Energy Consumption of ESP32 Microcontroller for Real-Time MQTT IoT-Based Monitoring System. 2023 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT), 255–261. <https://doi.org/10.1109/3ICT60104.2023.10391358>
- Wytrębowicz, J., Cabaj, K., & Krawiec, J. (2021). Messaging Protocols for IoT Systems—A Pragmatic Comparison. *Sensors*, 21(20), 6904. <https://doi.org/10.3390/s21206904>
- Zang, J., & You, P. (2023). An industrial IoT-enabled smart healthcare system using big data mining and machine learning. *Wireless Networks*, 29(2), 909–918. <https://doi.org/10.1007/s11276-022-03129-z>
- Zhang, M., Li, M., Guo, L., & Liu, J. (2023). A Low-Cost AI-Empowered Stethoscope and a Lightweight Model for Detecting Cardiac and Respiratory Diseases from Lung and Heart Auscultation Sounds. *Sensors*, 23(5), 2591. <https://doi.org/10.3390/s23052591>