ESP-NOW MIDI Protocol and Enomik (ESP-NOW MIDI Kit): Wired and Wireless MIDI Devices With an Optional Code-Free Workflow

Thomas Geissl (thomas.geissl@gmail.com) [1]

[1] *Tangible Music Lab*, **Kunstuniversität Linz**, Linz, Austria

ABSTRACT

ESP-NOW MIDI is a lightweight wireless layer for low-latency links between distributed devices in musical instruments and interactive systems. It uses Espressif's ESP-NOW to carry MIDI directly between devices, without relying on external network infrastructure. Wireless hops are converted to standard MIDI streams at the edges, preserving compatibility with existing software and hardware. The API follows established MIDI libraries closely and can often be used as a near drop-in replacement. Section 4 reports round-trip latency measurements.

Enomik (ESP-NOW MIDI Kit) adds utilities for multi-device setups: a USB dongle bridging wireless nodes to a host, a WebMIDI-based configuration interface, Pure Data patches, and modular PCBs with standardized connectors and common sensors and actuators. Together, these aim to reduce infrastructural overhead, support interplay among networked devices, and keep physical controllers and digital sound-making environments tightly coupled in musical exploration.

1. INTRODUCTION

The development of new musical instruments and interfaces hinges on the interplay between creative exploration and technical implementation. In distributed setups, there is a further interplay among heterogeneous networked devices (each with its own role, timing, and control flow) and between the physical world of gesture, sensors, materials, and stage space and the digital world of MIDI, software, and synthesis. Rapid prototyping is central to practice-based research in digital musical instrument design, yet a substantial amount of work still goes into reimplementing similar low-level functionality across projects. Familiar code patterns are rewritten, communication layers are configured, and infrastructure is debugged again and again. Over time, this repetition can erode motivation, especially when building a simple prototype already demands a significant engineering effort.

Recent advances in microcontroller platforms have eased some of these constraints. Affordable Arduino-compatible hardware and more mature development workflows have lowered the entry barrier to embedded music systems. At the same time, wireless communication remains a persistent bottleneck. Implementing robust, low-latency links between distributed devices is time-consuming and often tightly coupled to venue-specific network conditions. For musical instruments and interactive systems, where timing precision is perceptually critical, wireless communication easily turns into an optimization problem instead of a creative one.

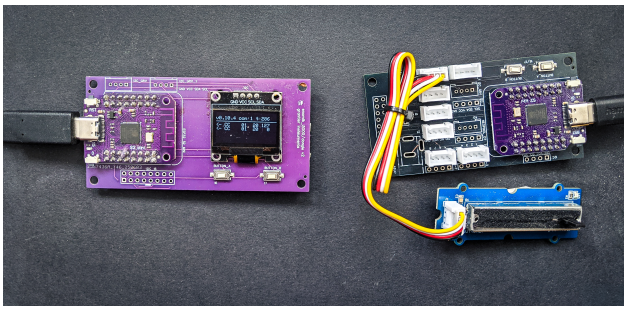


Figure 1. Enomik (ESP-NOW MIDI Kit) setup: wireless MIDI dongle (left) and client device (right).

As a result, the creative process is frequently pushed toward engineering concerns. Decisions about network topology, protocol overhead, and latency compensation can dominate early-stage experimentation. This is particularly problematic for practitioners who want to avoid reliance on venue Wi-Fi or other existing infrastructure, or who work with mobile and ad hoc instrument ensembles.

This paper introduces ESP-NOW MIDI, a lightweight wireless communication approach that addresses these issues by using Espressif's ESP-NOW protocol to transmit MIDI messages directly between devices. The system enables bidirectional, low-latency communication without external networks. MIDI data is sent wirelessly and then converted back into standard MIDI streams, maintaining compatibility with existing software and hardware ecosystems. From the programmers' perspective, the API follows established MIDI libraries closely, so ESP-NOW MIDI can often be adopted with only minor changes to existing code.

On top of this protocol, Enomik (ESP-NOW MIDI Kit) collects utilities that remove much of the friction from multi-device wireless setups. It provides a web-based configuration interface using WebMIDI, along with a set of Pure Data patches. The hardware family consists of a USB dongle board with a 0.96" OLED for MIDI/MAC monitor-

Copyright: © 2026. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

ing, client boards exposing Grove I2C plus digital/analog I/O, and an extender board connected via IDC16 (a 16-pin ribbon-cable connector); together they support repeatable hardware prototyping (Fig. 1). Taken together, these tools are intended to re-center the creative process by reducing infrastructural overhead and making it faster to move from idea to working prototype.

In line with SMC 2026's theme, *Interplay*, we frame the contribution as a unified prototyping framework: ESP-NOW MIDI provides the wireless transport layer, while Enomik provides the surrounding toolkit (configuration workflow, reusable software components, and modular hardware) that makes distributed instrument building practical in day-to-day use. The aim is to keep both kinds of interplay in focus: coordination among networked devices, and continuous coupling between physical input and digital musical response. Within this framework, gesture and sensor input are translated into MIDI and exchanged peer-to-peer between wireless nodes, which can both send control data and react to incoming messages. Different topologies of networked interaction (point-to-point, broadcast, star-like dongle-and-clients, and reactive chains of devices) are supported without reconfiguring infrastructure, so that distributed and ad hoc ensembles can be explored within the same workflow. By lowering the technical overhead of networking and configuration, the framework lets such interplay emerge from iterative musical experimentation rather than from infrastructure constraints, and helps keep the link between embodied action and machine response close to the surface.

2. RELATED WORK

Distributed and networked musical instruments are well established in computer music and musical HCI. Prior work includes synchronized Internet-based performance systems such as *Mallo* [1], and wireless collaborative ecosystems such as ubiquitous smart instruments built around networked devices [2]. Recent Internet of Sounds literature discusses broader convergent trends in connected sonic systems [3]. These projects show the musical potential of distributed interaction, but they often depend on external infrastructure (configured routers, venue Wi-Fi, or dedicated forwarding software). In rapid prototyping, such dependencies can shift attention from interaction design to network setup and troubleshooting.

At the interoperability layer, MIDI remains a dominant protocol for musical control data across hardware instruments, controllers, and software environments. Its message vocabulary (e.g. note on/off, control change, pitch bend, and system messages) and device-agnostic semantics have enabled long-term compatibility and reuse across heterogeneous setups [4]. In practice, retaining standard MIDI I/O is valuable because it allows novel controllers and distributed devices to integrate with existing DAWs, patching environments, and hardware synths without requiring bespoke host software.

Beyond MIDI, Open Sound Control (OSC) is a network-oriented, transport-independent protocol for real-time control and is widely used in computer music practice [5, 6]. OSC commonly runs over UDP in local and wide-area setups. Wireless router-based deployments can suffer from packet loss and latency variability, as addressed by utilities such as LANdini [7]; timing precision may also require explicit synchronization [8]. For MIDI over IP, RTP-MIDI supports networked performance and streaming with packet-loss recovery mechanisms [9], but usually assumes an existing network and session setup. At the application-synchronization layer, Ableton Link provides decentralized tempo and phase synchronization over local networks without fixed master/client roles [10]. Ableton also announced Link Audio in the Live 12.4 public beta, extending local-network collaboration to real-time audio streaming between Link Audio-enabled devices [11]. In practice, wireless MIDI is commonly implemented with Bluetooth or Wi-Fi-based approaches, trading ease of adoption against pairing procedures, access-point dependence, and environment-specific variability. Section 4 reports complementary end-to-end latency measurements for RTP-MIDI and Bluetooth MIDI alongside ESP-NOW MIDI. ESP-NOW offers an alternative model for short-range communication on ESP microcontrollers. Pasic et al. describe it as a lightweight protocol for ESP32 systems supporting direct device-to-device exchange without requiring a local Internet network [12]. Espressif documentation notes that ESP-NOW encapsulates application data in vendor-specific IEEE 802.11 action frames and transmits them connectionlessly between Wi-Fi devices, i.e. without conventional association to an access point [13]. This link-layer approach motivates ESP-NOW as a transport for infrastructure-independent, topology-flexible musical links.

The present work introduces ESP-NOW MIDI, a MIDI-oriented communication layer that encapsulates standard MIDI messages over ESP-NOW. Devices can exchange musical events without exposing users directly to the underlying wireless protocol. Rather than proposing a new wireless standard, the contribution is a protocol and toolkit that adapt ESP-NOW to established MIDI workflows and rapid prototyping of distributed musical interfaces.

3. SYSTEM OVERVIEW

The system consists of four components: a lightweight wireless MIDI protocol, an accompanying software library, a browser-based configuration environment, and dedicated hardware modules. These can be used together or independently, depending on project needs.

3.1 ESP-NOW MIDI Protocol

ESP-NOW MIDI transmits standard MIDI byte sequences between ESP32-based devices over Espressif's ESP-NOW communication layer. The protocol does not redefine MIDI; it encapsulates existing MIDI messages in ESP-NOW payloads. The software stack follows established MIDI programming patterns, so existing code can usually be adapted with minor changes.

For fixed-length messages (e.g. common channel voice messages), the transmitted payload corresponds directly

to the MIDI bytes. For variable-length SysEx, the byte stream is segmented into fixed-size chunks (128 bytes in our implementation) to keep overhead predictable and to fit comfortably within the ESP-NOW payload budget. As a result, SysEx communication is supported but not intended for time-critical musical interaction; it is primarily aimed at configuration, parameter exchange, and other non-performance-critical use cases.

3.2 ESP-NOW MIDI Library

The ESP-NOW MIDI protocol is accompanied by an Arduino library available through the Arduino Library Manager. Its API mirrors common MIDI programming patterns, so standard messages such as note on/off, control change, and pitch bend can be sent and received with minimal code changes. The library is released under the GNU Lesser General Public License (LGPL) and is publicly available at `http://github.com/grantler-instruments/ESP-NOW-MIDI/`. Its API follows the standard Arduino MIDI library closely.

The library also includes a configurable SysEx client, designed for device configuration, parameter exchange, and non-performance-critical communication; peer MAC addresses for ESP-NOW can be set in code or supplied through this SysEx path. In addition, it supports USB MIDI alongside wireless transmission, enabling devices to function in hybrid setups or as conventional USB controllers without modifying application logic.

A collection of example sketches demonstrates typical use cases, including point-to-point messaging, multi-device broadcasting, and interaction with external MIDI software.

3.3 Network topologies and distributed interaction

ESP-NOW MIDI supports several interaction topologies for distributed musical setups. *Point-to-point* links allow two devices to exchange MIDI bidirectionally with minimal latency, suitable for a single controller and a reactive sound module. *Broadcast* (one-to-many) lets one sender address multiple receivers, e.g. one controller driving several sound sources or actuators. *Star-like* configurations are common with Enomik: a USB-connected dongle bridges the wireless ESP-NOW network to a host (DAW, Pd, etc.), while multiple wireless clients send or receive MIDI through the same dongle. This host bridge is optional; devices can also communicate directly peer-to-peer without a USB MIDI host in the signal path. Because devices can both transmit and react to MIDI, *reactive chains* are possible (e.g. sensor A triggers device B, which in turn drives device C) without central coordination. All patterns use the same peer-to-peer radio layer. No router or access point is required, and switching between topologies is largely an addressing and application-logic change rather than an infrastructure reconfiguration.

3.4 Enomik

The kit provides utilities for multi-device wireless setups. A no-code workflow is supported through a WebMIDI-based configuration interface, complemented by Pure Data patches. Its hardware side currently uses three board types: a USB dongle board (with a 0.96" OLED as MIDI monitor and MAC-address display), client boards (with Grove connectors for I2C sensors and digital/analog I/O), and an extender board that connects via IDC16.

In a typical workflow, a user connects the USB dongle PCB to a host computer running a DAW, Pure Data (Pd), or another MIDI-capable environment. In the provided dongle example, the dongle bridges the wireless ESP-NOW MIDI network to USB MIDI and can optionally show a simple on-device MIDI monitor on an attached OLED screen. Each ESP-NOW peer is identified by its Wi-Fi MAC address. Addresses can be registered in code through the Arduino library or, in the no-code path, by sending MIDI SysEx configuration over a temporary USB connection (e.g. from the Enomik WebMIDI interface, where users also assign pin-to-MIDI and MIDI-to-pin mappings for the intended topology; Fig. 2). After configuration, the client can be unplugged and run standalone (for example from a power bank) while continuing to exchange MIDI messages wirelessly with the dongle and other devices.

As a concrete illustration, consider a flex sensor wired to an analog input on a wireless client: bending the sensor is read as a continuous value and mapped to a MIDI Control Change message, which is sent over ESP-NOW MIDI to the USB dongle and appears on the host as ordinary MIDI. With the dongle connected to a laptop running a DAW or soft synth, the user maps that CC to a synthesis parameter in software—for example oscillator pitch, filter cutoff, or another musically meaningful control. The performer can then move freely while the sensor drives sound on the host without a cable run to the instrument body, within the latency characteristics reported in Section 4.

4. EVALUATION

4.1 Benchmarks

Figure 3 shows four topologies in section order: ESP-NOW MIDI (main results), direct USB MIDI (host/USB baseline), RTP-MIDI over Wi-Fi, and Bluetooth MIDI (BLE).

We measured ESP-NOW MIDI round-trip latency on macOS (MacBook Air M4 plus i9 MacBook for baselines) using two Lolin ESP32-S2 Mini boards and the setup in Fig. 3(a). Traffic followed a request-response loop (each new MIDI message after the previous echo). Latency is round-trip time (RTT) from host send to echoed return at the host. All Pd runs used `pd-patches/benchmark_client-echo_rtt.pd`[1] in the ESP-NOW-MIDI repository (v0.10.8); tables aggregate the analysis outputs under `benchmarks/espnowMidi`, `benchmarks/rtpMidi`, `benchmarks/bleMidi`, and `benchmarks/usbMidi` [14]. We report mean, sample standard deviation (σ), median, p95/p99, IQR and MAD, and min–max. σ and p95/p99 capture overall and

[1] `https://github.com/grantler-instruments/ESP-NOW-MIDI/blob/main/pd-patches/benchmark_client-echo_rtt.pd`

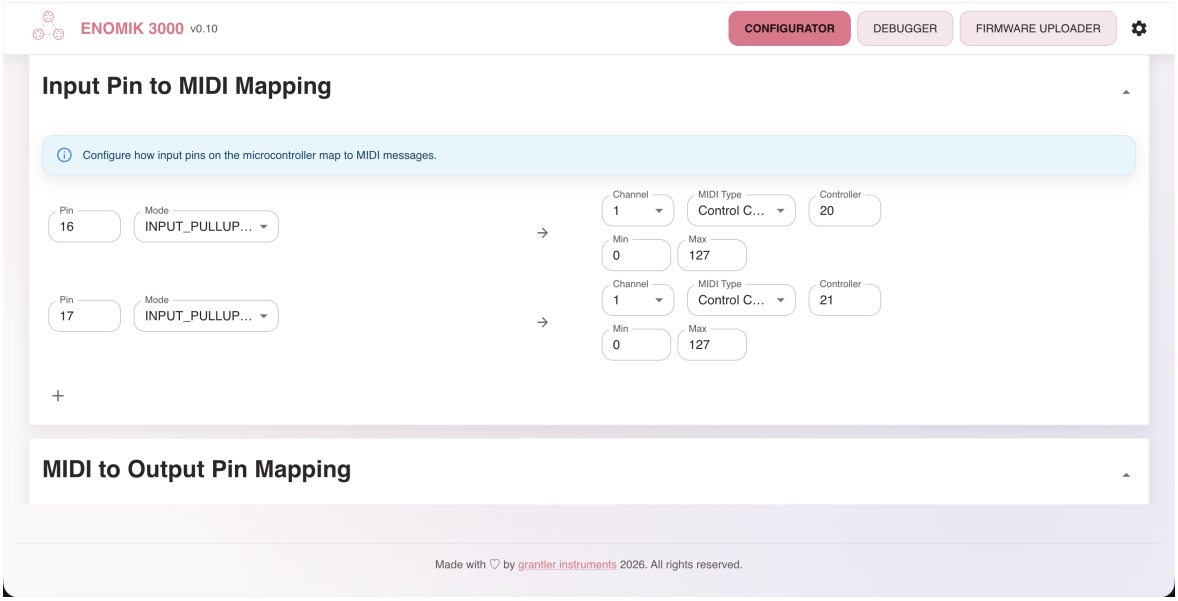


Figure 2. Enomik browser-based configuration interface: pin-to-MIDI mapping, MIDI-to-pin mapping, and wireless MIDI configuration.

tail spread; IQR and MAD are robust to outliers. For a single wireless hop, half the RTT is a rough one-way estimate (paths need not be symmetric). Obstacles (*Obs.*) are interior walls between the wireless devices (0 = line-of-sight, 1 = one wall, 2 = two walls).

Table 1 lists all ESP-NOW conditions (runs up to 15 m labeled indoor, above 15 m outdoor). Mean RTT is about 13.4–14.9 ms up to 15 m and about 16.3–19.0 ms from 20 m to 75 m.

Baselines use the same Pd echo loop: *RTP-MIDI* (Fig. 3(c)) between the two MacBooks via a mobile router (∼3 m, only those clients); *BLE MIDI* (Fig. 3(d)) with built-in radios at 1–10 m (Table 2). They are laptop-to-laptop paths, whereas ESP-NOW includes the dongle and ESP32, so Table 2 is a practical reference, not a fair transport shootout.

Distributions are right-skewed (host/USB scheduling), so medians and tails matter for musical timing. Indoors, ESP-NOW σ is about 9.3–10.3 ms and rises at longer outdoor ranges (about 11.4–14.2 ms). At 3 m indoor (0 obstacles), ESP-NOW mean RTT is 14.49 ms versus BLE central latency near 47–50 ms and RTP-MIDI p99 118 ms in our runs. USB MIDI alone (Fig. 3(b), Table 2) isolates the M4 host path (mean 1.25 ms); the ∼6 ms minima in Table 1 reflect the full dongle-plus-radio bridge. Results are consistent with real-time use in these conditions; further work should dissect jitter sources (host scheduling, USB polling, radio contention). Dataset: `https://github.com/grantler-instruments/ESP-NOW-MIDI/tree/main/benchmarks`.

ESP-NOW MIDI shows lower central and tail latency than BLE and RTP-MIDI *here*, but we do not claim universal superiority—paths differ. Deployment still benefits from MAC-based peers (firmware API or SysEx via the Enomik web app) instead of repeated BLE pairing or one cable per link; Espressif allows up to twenty peers per device [13].

4.2 Informal performance use

Beyond these benchmarks, ESP-NOW MIDI has been used informally in the Klang-/Licht *STROM* project [2] and with Turntangilism at Ars Electronica Festival 2025, without systematic measurement but without noticeable dropouts or unusable latency. Controlled studies remain needed for usability and robustness.

5. CONCLUSION

5.1 Summary

ESP-NOW MIDI and the Enomik toolkit provide a framework for prototyping wired and wireless musical instruments and interfaces, combining a wireless MIDI protocol, browser-based configuration, reusable software components, and modular hardware.

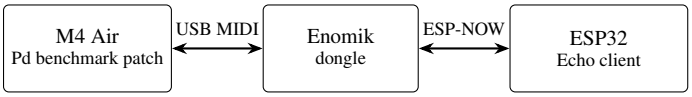
The evaluation suggests that ESP-NOW MIDI is suitable for real-time musical control under the tested conditions. Across line-of-sight and obstructed runs up to 75 m, round-trip latency remained in a usable range, with increasing central/tail latency and jitter at longer distances.

5.2 Future Work

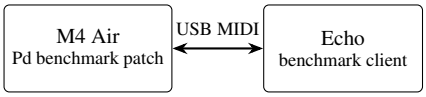
5.2.1 Performance and Stability

Future work will focus on technical refinement and user-centered evaluation. A priority is deeper jitter analysis, including source characterization (e.g. host scheduling, radio collisions, USB polling) and mitigation strategies for latency-sensitive performance, with repeated runs across additional BLE peripherals and non-macOS hosts.

[2] `https://klanglichtstrom.de`



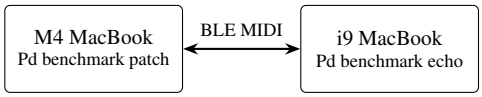
(a) ESP-NOW MIDI: MacBook Air M4 with Pd benchmark patch, USB MIDI dongle, wireless client (bidirectional).



(b) USB MIDI only: direct M4-to-echo USB link (no dongle, no ESP-NOW), used as the host/USB latency baseline.



(c) RTP-MIDI: M4 Pd benchmark patch to i9 Pd benchmark echo patch via a mobile-router Wi-Fi network (only these two MacBooks connected as clients).



(d) Bluetooth MIDI (BLE): M4 runs the Pd benchmark patch, i9 the Pd benchmark echo patch.

Figure 3. Benchmark measurement topologies (schematic), in section order. All runs used macOS on a MacBook Air (M4) and an i9 MacBook. **(a)** ESP-NOW MIDI path for the tabulated runs: M4 as USB MIDI host, Enomik dongle, and wireless ESP32 client; the peer board relays USB MIDI to the host. **(b)** USB MIDI only (no wireless hop): direct USB MIDI path from the M4 host to the echo client (no dongle); used to isolate host and USB stack latency (mean ∼1.25 ms on the M4 Air; Table 2). **(c)** RTP-MIDI endpoints over a mobile-router Wi-Fi network at ∼3 m, with only the two MacBooks connected as clients. **(d)** Wireless Bluetooth MIDI (BLE) baseline at ∼3 m with request-response traffic (next message sent after echo return).

Table 1. ESP-NOW MIDI round-trip latency (ms) across measured conditions from `benchmarks/espnowMidi/*_analysis.txt`. Host: MacBook Air M4 (macOS); board: Lolin ESP32-S2 Mini; 1000 samples per condition; request-response traffic (next message sent after echo return). Obs.: number of obstacles (interior walls) between devices. Indoor/outdoor label follows this paper's distance split (up to 15 m indoor; above 15 m outdoor).

Cond.	Dist.	Obs.	Mean	σ	Med.	p95	p99	IQR	MAD	Min	Max
Indoor	1 m	0	13.79	9.85	6.00	32.67	38.00	16.00	0.00	6.00	64.67
Indoor	3 m	0	14.49	10.01	11.33	32.67	38.00	16.00	5.33	6.00	96.67
Indoor	5 m	0	14.90	10.18	11.33	32.67	38.00	16.00	5.33	6.00	54.00
Indoor	7 m	0	13.41	9.31	6.00	32.67	38.00	16.00	0.00	6.00	38.00
Indoor	10 m	1	14.19	10.28	6.00	32.67	38.00	16.00	0.00	6.00	48.67
Indoor	15 m	2	13.91	9.57	9.67	32.67	38.00	16.00	3.67	6.00	48.67
Outdoor	20 m	0	16.30	11.38	11.33	32.67	48.67	16.00	5.33	6.00	155.33
Outdoor	40 m	0	18.88	13.42	16.67	38.00	64.67	21.33	10.67	6.00	128.67
Outdoor	50 m	0	18.98	14.21	16.67	43.33	59.33	21.33	10.67	6.00	187.33
Outdoor	75 m	0	18.29	12.46	16.67	38.00	54.00	21.33	10.67	6.00	102.00

5.2.2 Features

Both the library and the toolkit could be extended in several directions. First, the library currently does not support MIDI Polyphonic Expression (MPE); adding MPE support would make the system more suitable for highly expressive digital instruments. MIDI 2.0 and Universal MIDI Packets (UMP) could extend expressive control beyond MPE, but we still carry MIDI 1.0 byte streams for compatibility with the Arduino MIDI stack and host tools we build for. UMP is on the roadmap once those libraries support it. Second, while the library can already be used independently of the toolkit, the toolkit itself could be expanded to support more complex sensors and actuators, such as I2C-based devices. For example, accelerometer data could be exposed more directly in the web interface, allowing a mapping such as the x-axis acceleration of an accelerometer to a specific MIDI Control Change message.

Further work may investigate *Enomiga* (ESP-NOW MIDI Gateway), a host-side ESP-NOW-to-MIDI bridge, and under what conditions it would be useful despite ESP-NOW's proprietary link layer and the lack of standard host APIs. We have sketched the idea and run early packet-sniffing experiments on the host Wi-Fi interface, but have not pursued a working gateway beyond that. The USB dongle remains the intended host bridge in the current toolkit.

5.2.3 User Study

Planned work includes a controlled user study with instrument designers and performers to evaluate usability systematically—task completion, learnability, and subjective workload—and to measure how much the toolkit reduces prototyping effort versus ad hoc wiring and one-off firmware. The study will also examine whether lower infrastructural overhead translates into broader creative exploration and faster iteration in realistic instrument-building tasks.

Table 2. Baseline round-trip latency (ms) from repository analysis files (USB, RTP-MIDI, and BLE). Same request-response traffic as ESP-NOW runs (next message sent after echo return). "USB" is the direct host loop baseline on MacBook Air M4. "RTP" is MacBook Air M4 to i9 MacBook over a mobile-router Wi-Fi network (1000 samples). "BLE" rows use the same two laptops at 1 m (0 obstacles), 3 m (1 obstacle), 5 m (2 obstacles), and 10 m (2 obstacles).

Transport	Mean	σ	Med.	p95	p99	IQR	MAD	Min	Max
USB (1 m)	1.25	0.76	1.33	1.33	2.67	0.00	0.00	0.67	13.33
RTP (mobile router)	30.86	19.01	22.00	59.33	118.00	10.67	0.00	16.67	139.33
BLE (1 m, 0 obs.)	49.30	12.39	44.99	75.46	91.43	11.61	5.81	36.28	107.39
BLE (3 m, 1 obs.)	50.03	13.30	44.99	79.82	97.23	11.61	5.81	36.28	110.29
BLE (5 m, 2 obs.)	49.81	13.09	44.99	81.27	92.94	10.16	4.35	36.28	105.94
BLE (10 m, 2 obs.)	47.22	7.89	43.33	64.67	80.67	5.33	0.00	32.67	107.33

6. REFERENCES

[1] Z. Jin, R. Oda, A. Finkelstein, and R. Fiebrink, "Mallo: A distributed, synchronized instrument for internet music performance," 2015.

[2] L. Turchet and M. Barthet, "An ubiquitous smart guitar system for collaborative musical practice," *Journal of New Music Research*, vol. 48, no. 4, pp. 352–365, 2019.

[3] L. Turchet, M. Lagrange, C. Rottondi, G. Fazekas, N. Peters, J. Østergaard, F. Font, T. Bäckström, and C. Fischione, "The internet of sounds: Convergent trends, insights, and future directions," *IEEE Internet of Things Journal*, vol. 10, no. 13, pp. 11 264–11 292, 2023.

[4] J. Rothstein, *MIDI: A comprehensive introduction*. AR Editions, Inc., 1995, vol. 7.

[5] M. Wright and A. Freed, "Open soundcontrol: A new protocol for communicating with sound synthesizers," in *Proceedings of the International Computer Music Conference (ICMC)*, Thessaloniki, Greece, 1997, pp. 101–104.

[6] M. Wright, A. Freed, and A. Momeni, "Opensound control: State of the art 2003," in *Proceedings of the International Conference on New Interfaces for Musical Expression (NIME)*, 2003, pp. 153–159.

[7] J. Narveson and D. Trueman, "Landini: A networking utility for wireless lan-based laptop ensembles," *eContact!*, vol. 16, no. 3, 2013. [Online]. Available: https://econtact.ca/16_3/narveson-trueman_landini.html

[8] S. Madgwick, T. Mitchell, C. Barreto, and A. Freed, "Simple synchronization for open sound control," in *Proceedings of the International Computer Music Conference (ICMC)*, 2015.

[9] J. Lazzaro and J. Wawrzynek, "An RTP payload format for MIDI," in *Proceedings of the 117th Audio Engineering Society Convention*, San Francisco, CA, Oct. 2004. [Online]. Available: http://john-lazzaro.github.io/sa/pubs/pdf/aes117.pdf

[10] F. Goltz, "Ableton link – a technology to synchronize music software," in *Proceedings of the Linux Audio Conference (LAC)*, 2018. [Online]. Available: https://lac.linuxaudio.org/2018/pdf/LAC2018_proceedings.pdf#page=51

[11] Ableton, "Live 12.4 is coming – with link audio, updated devices and more," 2026, ableton blog post (public beta announcement), accessed 2026-03-24. [Online]. Available: https://www.ableton.com/en/blog/live-12-4-is-coming/

[12] R. Pasic, I. Kuzmanov, and K. Atanasovski, "Esp-now communication protocol with esp32," *Journal of Universal Excellence*, vol. 6, no. 1, pp. 53–60, 2021.

[13] Espressif Systems, "ESP-NOW programming guide," 2026, accessed: 2026-03-17. [Online]. Available: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/network/esp_now.html

[14] Grantler Instruments, "ESP-NOW-MIDI benchmarks," 2026, accessed: 2026-03-11. [Online]. Available: https://github.com/grantler-instruments/ESP-NOW-MIDI/tree/main/benchmarks