A Simulated Multi-Connectivity Internet Radio: Digital-Twin Evaluation of Validation-First Playback and NFC-Assisted Onboarding

Alvin V. Destajo
College of Engineering
University of Perpetual Help System
Dalta - Molino
Bacoor, Cavite
alvin.destajo@perpetualdalta.edu.ph

Brian R. Dondriano
College of Engineering
University of Perpetual Help System
Dalta - Molino
Bacoor, Cavite
brian.dondriano@perpetual.edu.ph

Cyd Laurence B. Santos
College of Engineering
University of Perpetual Help System
Dalta - Molino
Bacoor, Cavite
qclbsantos@tip.edu.ph

***Abstract*—Internet radio receivers depend on community station directories whose health flags are treated as reliable, and their onboarding flows inherit evidence collected with Bluetooth Classic more than a decade ago. This paper evaluates both assumptions with a software digital twin rather than a physical build: the directory client, stream validation, NDEF tag format, control protocol and playback policy run as real code, w radio links are explicitly modelled. The directory listed 52,822 stations in 224 countries and 585 languages. Across 240 streams, 95.8% were playable, and 4.2% of entries flagged healthy delivered no audio. In a 60-station paired comparison on live streams, flag-trusting playback produced 10 false play events (16.7%) while validation-first playback produced none, and error reporting was bounded at 2.0 to 2.6 s instead of ranging from 0.9 to 13.5 s. Modelled experiments show underrun rate falling from 55 to 10 per minute as the buffer grows from 8 to 64 KB on a weak link, and a 70% reduction in modelled setup time for a tap-based workflow. The paper states which claims are measured, which are modelled, and which require hardware.**

Keywords—Internet radio, digital twin, simulation-based evaluation, station directory, validation-first playback, Bluetooth Low Energy, near field communication, ESP32-S3

I. Introduction

Radio listening has moved online, adding to terrestrial listeni r than replacing it [1]. The directory used here listed 52,822 stations in 224 countries on the day of measu 2], and a microcontroller with Wi-Fi and an I2S amplifier now suffices to build a receiver [3], [4]. Two assumptions behind that work are untested. The first is that a directory knows which of its entries work: health flags come from periodic checks [2], so a receiver that trusts them commits its buffer to a stream that may never deliver audio. The second is onboarding. Screenless devices need a phone for setup, and the evidence that near-field interaction is fast dates from 2006 to 2014 and Bluetooth Classic [5], [6]; current BLE and Android stacks are uncovered, and measured BLE timing on Android suggests the old percentages do not transfer [7].

Testing those assumptions on a physical build would cost parts, weeks and a soldering iron, and still tie the result to one board. This paper keeps every part that does not need a radio as real software and models only the radio links, parameters included, so the twin answers application-layer questions on live networks and is explicit about the ones it cannot.

The contributions are a client-side measurement of directory health against actual playability, a validation-first playback policy tested against flag-trusting playback on real streams, a modelled buffer and onboarding study, and a reproducible hardware-free implementation with an interactive demonstration, an explicit real-versus-modelled classification and a stated path to a physical build.

II. Related Work

A. Embedded Internet Radio Receivers

Stand-alone internet radio has been built on embedded hardware since the mid-2000s. Park combined a TCP/IP stack with an MP3 decoder and flash storage [3], Hwang et al. brought a non-DSP ARM9 board below 40% CPU load with a software packet tracer [8], and Cameron documents the architecture most hobby receivers still follow, an ESP32 feeding a VS1053 decoder over SPI [4]. Kodera built a complete ESP32-PICO-D4 web radio in a 50 by 50 mm enclosure [9], and Zahrah et al. found signal strength dominating stability, with buffering below about −40 dBm [10]. Fischer places these devices in the broadcast-to-OTT transition [11].

None of these designs measures the directory it depends on or provides a mobile control channel. This work keeps the low-cost architecture, swaps the decoder module for a software decoder, and adds control and measurement layers.

B. Streaming Quality, Startup, and Playout

The quality-of-experience literature establishes what listeners notice: startup delay and rebuffering dominate perceived quality [12], abandonment tracks stalls [13], quality trades against traffic volume [14], and parametric quality models exist [15].

For audio, the playout literature defines the central trade-off. Moon et al. bound achievable playout delay for a given late-loss rate [16] and Liang et al. adapt scheduling within talkspurts [17]; packet-loss concealment decides how graceful degradation looks [18], and listeners stop distinguishing codecs on consumer headphones above roughly 48 kb/s, while AAC-LC at 128 kb/s and above separates on better reproduction [19]. That literature asks how long to buffer, not whether to start, which is the question here.

C. Wi-Fi and 2.4 GHz Coexistence

Wi-Fi and Bluetooth share the 2.4 GHz band, where interference is measured and documented: dynamic fragmentation mitigates it [20], mutual interference has been quantified [21], and frequency hopping improves coexistence

[22]. The role separation here keeps the control channel off the audio path's airtime.

D. Bluetooth Low Energy as a Control Channel

BLE parameters trade energy against latency and throughput [23], and effective delay depends on connection configuration and device processing speed [24]. Measured connection-oriented BLE operations on Android handsets show that latency guarantees are difficult when a smartphone is part of the path [7], which is why the study below uses two handset profiles and reports steps separately from timing.

E. NFC Pairing, Handover, and Tag Interaction

Near-field interaction as a pairing accelerator is established prior work: touching an RFID tag made Bluetooth establishment faster and more pleasant than manual discovery in 2006 [5], NFC-mediated pairing measured 31 to 43 percent faster [6], and concurrent activation cut handover latency by about 30 percent [25]. The NFC Forum defines the NDEF handover model used here [26], and the link's threat surface is documented: eavesdropping and data corruption [27] and unsolved privacy threats [28]. The gap is contextual, since those studies used Bluetooth Classic, file transfer or headset scenarios and none instrumented a headless appliance per stage.

F. Onboarding and Provisioning Security

Provisioning is the weakest link in consumer IoT: six of eight SmartCfg-style schemes leak Wi-Fi credentials [29] and onboarding flows have systematized failure modes [30]. Bluetooth pairing is no safer, with reflection attacks across all pairing modes [31], initiator-side downgrades breaking Secure Connections Only on 18 commercial products and Android BLE applications [32], man-in-the-middle attacks succeeding against current wearables [33], and a demonstrated logic flaw in BLE Secure Connections [34]. Vendor channels such as BluFi belong to the class the WiSec study criticizes [35]. This design therefore keeps credentials inside an encrypted BLE GATT session and places no secret on the tag.

G. Simulation and Emulation Practice

Simulation-based evaluation is established, but its credibility rests on stated assumptions and, where possible, validation. Digital twins reflect selected properties of a physical object, with fidelity defined by application context rather than physical identity [36], [37], and network simulators earn trust only after validation, as when ns-3's 802.11 models were checked against testbeds and analytical models [38], [39]. Embedded verification treats model-, software- and hardware-in-the-loop as successive stages with hardware last [40], and firmware re-hosting shows the fragility: FirmAE raised emulation success from about 16% to 79% of images [41].

This work applies the same discipline at the application layer: what can run for real does, modelled parts carry explicit parameters, and no modelled number is presented as a measurement.

H. Usability Measurement and Reliability

Because the twin cannot run a user study, the paper fixes the usability measures it would use instead of inventing results. The System Usability Scale remains the standard low-cost instrument [42], though its fit for voice-first devices is contested [43], and smart-speaker studies show task-completion time, error counts and feedback quality discriminating between designs [44]; the demo exposes the same measures. Reconnection policy and payload sizing govern recovery after a drop [45], and protocol choice changes reception energy [46]. These sources set the protocol for the hardware phase and support no claims here.

No study measures how often a directory-declared healthy stream fails for an embedded client, and none evaluates a validation-first policy against a flag-trusting baseline on live streams. Those two absences bound the contribution. The radio combination is not claimed as novel, and the onboarding comparison is a modelled workflow plus a protocol, not a hardware result (Table I).

Work	Radios	Control	Directory	Eval.
Park [3], Hwang [4]	Wi-Fi	none	no	HW
Cameron [5], Kodera [6]	Wi-Fi	web UI	no	HW
Zahrah [7]	Wi-Fi	none	no	HW
Salminen [8], Seewoonauth [9]	RFID/NFC + BT	none	no	HW
Sim [10]	NFC + BT	none	no	HW
This work	Wi-Fi, BLE, NFC	BLE app	yes	twin

III. Simulation Architecture

A. Role Separation in a Hardware-Free Setting

Each radio keeps exactly one job, because overlapping duties make multi-radio designs fragile: Wi-Fi is the data plane, BLE the control plane, and NFC a one-shot bootstrap handing the phone the device name and address. Removing the board does not change the roles; the data plane becomes the host's network stack and the control plane a protocol object (Fig. 1).

The same three transports are also the three ways a listener starts playback: a tap resolves the device and opens the session, a BLE write carries the command, and an HTTP request reaches the same command surface from the local network. All three terminate in one JSON command set, so the choice between them is ergonomic rather than functional, and a listener can switch station, adjust volume or recall a favourite over whichever is at hand. What the control plane does not do is carry listener audio: it selects and starts a station stream, and pushing arbitrary audio into a station for broadcast would need an ingest endpoint operated by that station plus the licence the Discussion returns to.

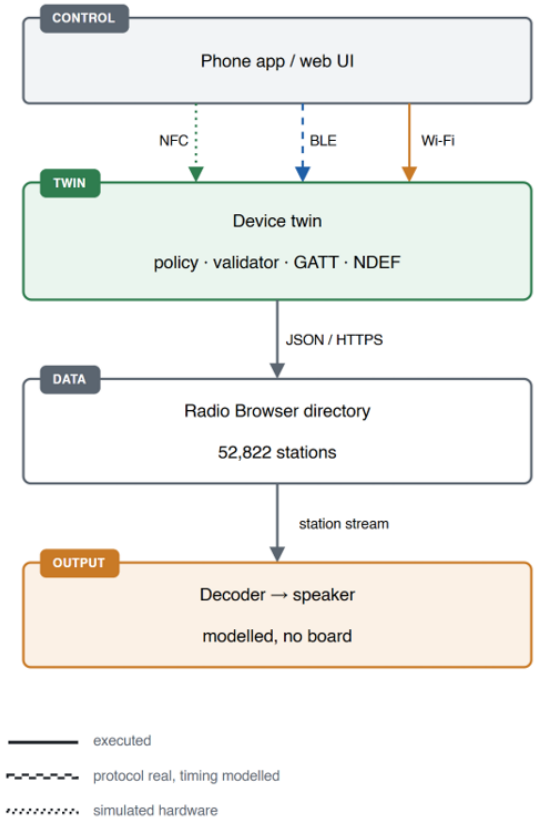


Fig. 1. Simulation architecture. NFC, BLE and Wi-Fi reach one command surface; solid edges run as code, dashed and dotted edges are modelled.

The ESP32-S3 stays the reference target because it carries both radios natively and documents coexistence [47]. It also fixes a limitation the twin inherits honestly: no Bluetooth Classic, so A2DP audio is impossible and BLE carries only control messages.

B. Component Mapping and Tool Selection

Wokwi offers ESP32 Wi-Fi through a gateway but no Bluetooth and no I2S on the S3, so it cannot exercise the control plane. Renode supports the ESP32 at core translation only, not the SoC and peripherals. QEMU has an Espressif fork but no BLE radio model, and the Android Emulator exposes a virtual Bluetooth controller from API 31 yet no NFC and would need a full app build. ns-3 has validated 802.11 models but no BLE module and no application layer, so it cannot run the directory client. A Python twin runs the actual client code, needs no licence and is auditable line by line (Table II).

Tool	What it covers for this system	Why it was not selected
Wokwi	ESP32/S3 core, GPIO, Wi-Fi with an internet gateway	no Bluetooth and no I2S on the S3, so the control plane cannot run
Renode	many SoCs, Cortex-M and Cortex-A, RISC-V	ESP32 support is core translation only, not the SoC and peripherals
QEMU	full-system emulation	no BLE or Wi-Fi radio model for this target
Android Emulator	Android app with a virtual Bluetooth controller	no NFC support and it requires a full app build
ns-3	validated 802.11 PHY and MAC models	no BLE module and no application layer to run the client
Python twin	policy, protocol, API, buffer and link models	no radio hardware, which is accepted and declared

C. Implementation Classification

Every feature sits in exactly one class (Table III): Class A is executed by the code in this repository, Class B is simulated hardware, Class C is modelled behaviour with stated parameters, and Class D is not implemented and not claimed.

Class	Feature	Status
A real	directory discovery, search, by-UUID resolution, click	executed on live API
A real	stream validation, TTFB, six-way failure classification	executed on live streams
A real	NDEF encoder/parser, AAR + MIME record	unit tested
A real	GATT JSON command handler, 12 operations	unit tested
A real	playback policy, backoff ladder, false-play accounting	executed
A real	web demo, audio playback in browser	executed
B simulated	NFC tap, virtual tag read	virtual tag
B simulated	BLE link and onboarding workflow	protocol object
B simulated	device buffer occupancy and underruns	byte-accurate model
C modelled	Wi-Fi delay, jitter, loss profiles	parameters stated
C modelled	BLE connection, discovery, notification timing	parameters stated
D not implemented	physical board, DAC, speaker, power, thermal, EMI	not claimed
D not implemented	user study with participants	not claimed

D. Validation-First Playback Policy

The policy is the core contribution and runs for real. Instead of starting playback on a 200 response, the receiver sends a HEAD request and a ranged GET of about 4 KB with a short timeout, and a station counts as playable only when audio bytes arrive. Failures are classified (HTTP error, timeout, DNS, TLS, connection) so the interface can say what went wrong, and a bounded ladder of five backoff steps ends in an error state (Fig. 2). The same machine runs in Python here and in the firmware sketch.

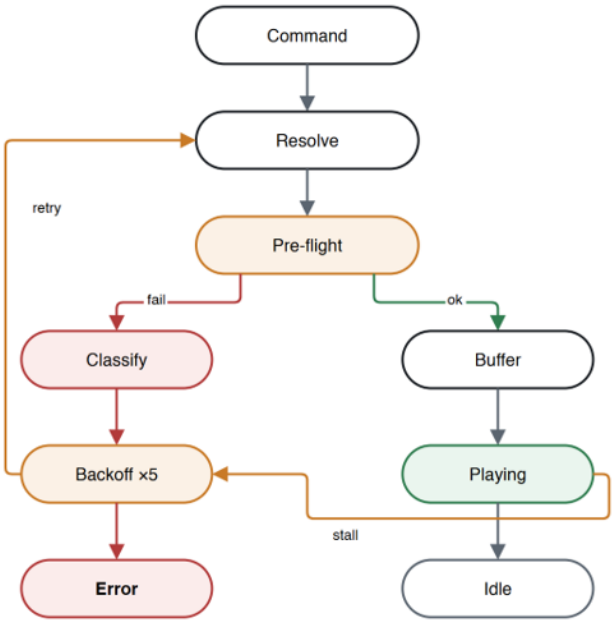


Fig. 2. Validation-first policy: playback is declared only after audio bytes arrive, and failures climb a bounded backoff ladder.

IV. Implementation

A. Real Components

The directory client follows the documented flow: resolve the discovery host, reverse-look-up the answer to learn the live node, then use JSON endpoints with a descriptive user agent, addressing stations by UUID and registering a click on play. The validator opens the stream, reads the first chunk and reports elapsed time and failure class. The NDEF module implements the binary record layout directly, with an Android Application Record and a MIME record carrying device name, address and service UUID; no credential goes on the tag. The GATT handler answers eleven JSON operations with volume range checks and favorites.

B. Modelled Components

The Wi-Fi model is a delay distribution with jitter, spikes and loss, anchored on the measured time-to-first-byte distribution of Section VI. The BLE model derives stage costs from connection interval, MTU and per-operation overhead, with a spread reflecting handset variance. The buffer model is byte-accurate: chunks carry 50 ms of audio and arrive with a log-normal inter-arrival spread whose mean equals the audio rate, and a starved buffer pauses and rebuffers. Every parameter appears in the scenario files and each CSV row.

C. Interactive Demonstration

The prototype includes a browser interface (Fig. 3) that exposes the twin through a radio client rather than a diagnostic panel. Discovery, browsing, search and the nearest list query the live directory, and every stream is validated before its URL reaches the audio element, so the playback in the capture is real audio. The nearest list is ordered by great-circle distance from directory coordinates, and the interface states how many entries carry them, keeping that metadata's sparsity visible instead of hiding it behind a fabricated local-transmitter claim. The spectrum display runs on the actual signal through an analyser node when the stream sends Access-Control-Allow-Origin; otherwise the interface says the signal cannot be tapped and disables the enhancement controls rather than drawing a synthetic trace. The same tap feeds a short in-memory recording, so capture-permitting streams expose a rewind window, where dragging the elapsed-time marker left replays audio that was broadcast and the right edge reconnects the live stream. NFC and BLE panels sit in a settings view whose text states that the tap and radio timings are modelled.

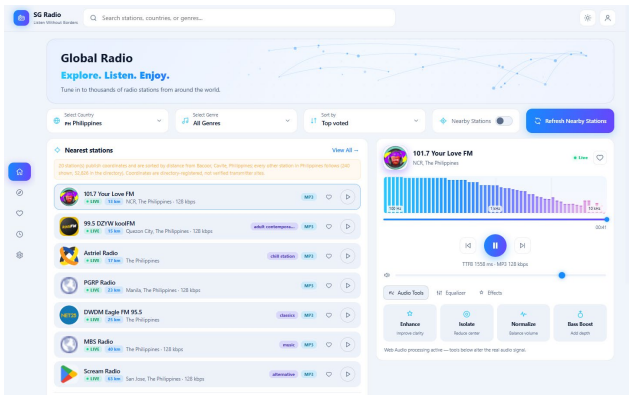


Fig. 3. Interactive demonstration. Search, validation and playback are real; the NFC tap and BLE timings are modelled.

D. Reproducibility

The package contains the library, three experiment scripts, JSON scenario files, a fifteen-case unit test suite, the figure generator and the Wokwi check with its firmware sketch, wiring and serial log. Experiments write one CSV row per observation plus a summary naming measured and modelled fields, and fixed seeds with a frozen station UUID list keep the method reproducible while the directory changes.

V. Methodology

A. Scenarios and Metrics

Three scenarios are defined. The policy comparison runs both policies over the same frozen station list on live streams, randomizing order per station to avoid warming bias, and the list is stratified: every station that failed in the earlier 240-stream probe plus a random sample of healthy ones, with per-stratum rates reported so the stratification stays visible. The buffer sweep runs four impairment profiles against four buffer sizes with five seeds and two minutes of audio each, and the onboarding study runs thirty trials per path on two handset profiles; each metric's operational definition is given in Table IV.

Two rates are used throughout: availability, the share of attempts that delivered audio, and the false-play rate, the share of attempts where the interface claimed playback without audio.

$$A = n_{ok}/N \tag{1}$$

$$F = n_{fp}/N \tag{2}$$

where A is availability, n_ok playable attempts, F the false-play rate, n_fp false-play events and N attempts. Both rates are reported per stratum in Section VI so the deliberate oversampling of dead entries stays visible.

Metric	Definition	Source
False play	interface claims playback while no audio bytes arrived	measured
Time to audio	command to first audio byte	measured
Time to terminal	command to playing or to an error being shown	measured
Availability	playable attempts over attempts	measured
Underrun rate	buffer starvation events per minute of audio	modelled
Setup time	tap or scan start to GATT ready	modelled
Step count	discrete user actions in the workflow	counted

B. Statistics and Validity

The policy comparison is paired: each station is fetched once per policy, so differences reflect the policy rather than the network. Distributions are reported as median and interquartile range alongside mean and standard deviation, because time-to-first-byte is skewed. The modelled experiments report means over seeds and state the number of runs. Three threats are acknowledged. The probe runs from one vantage point, so latency may not generalize. The directory changes continuously, so the numbers describe a snapshot. And the buffer and BLE models encode assumptions that a hardware build would replace with measurements.

VI. Results

A. Directory Snapshot

The directory advertised 52,822 stations in 224 countries and 585 languages, includi stations tagged to the Philippines and 89 in Filipino languages. Coverage is skewed: the United States accounts for 6,097 listed stations, Germany 5,663, the Russian Federation 2,795 and China 1,973. The long tail still leaves hundreds of options for a listener outside the dominant markets, which is the practical argument for a directory-based receiver over a fixed station list.

B. Endpoint Latency and Stream Availability

Light metadata endpoints answer in about one second, while 100-station search lists of roughly 150 KB take 1.5 to 1.7 s from this vantage point (Fig. 7). Of 240 probed streams, 230 delivered audio within the window, an availability of 95.8%. The ten failures comprised three connect timeouts, two HTTP 404 responses, two connection errors, and one each of read timeout, DNS failure and TLS failure (Fig. 8). The directory had flagged all 240 as healthy, so 4.2% of flagged-healthy entries were stale. A further practical observation from the demo: HLS manifest URLs are correctly rejected by the validator, because a manifest is not an audio stream, which means some large broadcasters cannot be played by a simple client at all.

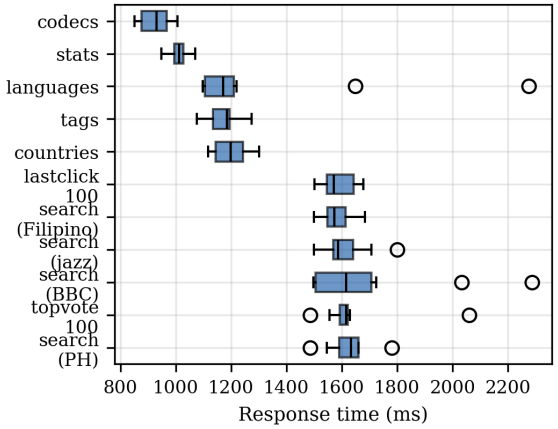


Fig. 4. Directory endpoint response times measured from the Philippines, ten requests each.

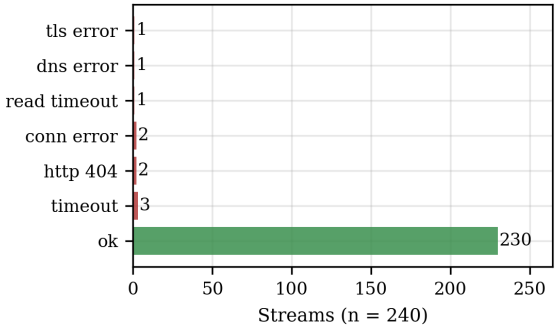


Fig. 5. Outcome classes of the 240-station probe: 230 playable, 10 failed.

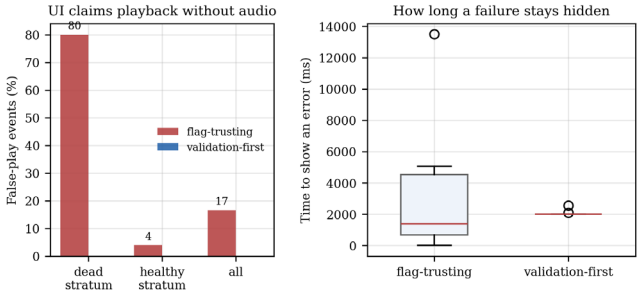


Fig. 6. Policy comparison on live streams. Flag-trusting playback declares playback before audio exists; validation-first never does, and bounds error reporting.

C. Policy Comparison on Live Streams

Fifty of the 60 stations played under both policies, so availability is identical by construction; what differs is what the user sees while a failure is discovered (Table V, Fig. 6). Flag-trusting playback produced 10 false play events, 16.7% of attempts, declaring playback and then going silent, while validation-first produced none because it never declares before bytes arrive. The effect concentrates where it matters: 80% of dead-stratum stations produced a false play under flag-trusting against 4% of the healthy stratum. Error reporting ranged from 0.9 ms to 13.5 s with a median of 1.38 s under flag-trusting, and stayed between 2.0 and 2.6 s under validation-first, at the cost of a later message because the policy retries once after a two-second backoff. Time to audio was indistinguishable, with medians of 1.22 s and 1.30 s for the same 50 stations.

Measure	Flag-trusting	Validation-first
Playable	50 / 60	50 / 60
False play events	10 (16.7%)	0 (0.0%)
False plays, dead stratum	80%	0%
False plays, healthy stratum	4%	0%
Time to audio, median	1224 ms	1305 ms
Time to audio, IQR	1001-1994 ms	943-1932 ms
Time to error, median	1380 ms	2000 ms
Time to error, range	0.9-13495 ms	2000-2570 ms

D. Buffer Behaviour Under Modelled Impairment

A tap-based workflow cuts the step count from five to two by removing scan, wait and select, and modelled setup time falls from 7.3 s to 2.1 s on the faster handset and from 8.9 s to 2.5 s on the slower one, about 71% (Table VII, Fig. 8). The absolute numbers rest on stage-cost assumptions and are not hardware performance, but the step count follows from the protocol definition and the direction of the effect matches the historical studies [5], [6]. Success rate is deliberately unmodelled, since scan and connection failures depend on real radios.

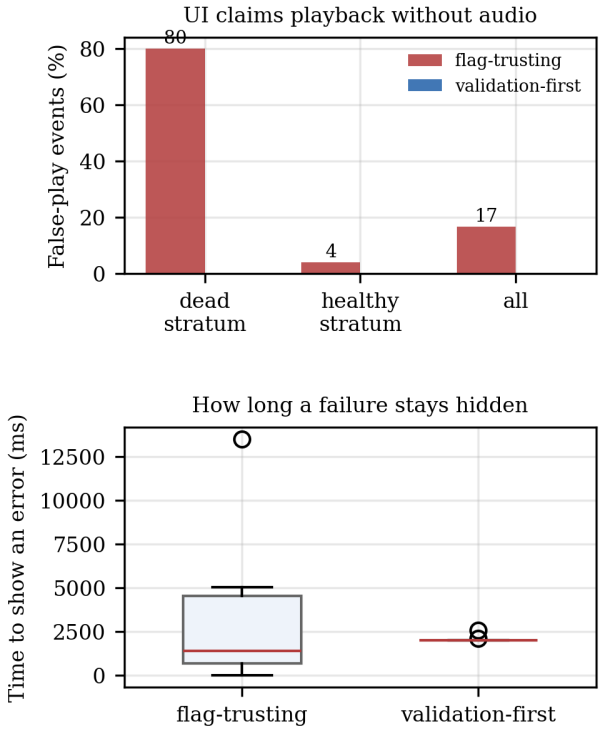


Fig. 7. False plays and error-reporting delay under both policies on live streams.

E. Buffer Behaviour Under Modelled Impairment

Under the weakest profile the underrun rate falls from 55.2 to 9.8 per minute as the buffer grows from 8 to 64 KB, and under a clean link from 2.2 to zero. Each doubling roughly halves the rate, the shape the playout literature predicts [16], and the trade-off is startup, since a larger buffer takes longer to reach its start threshold (Table VI, Fig. 7). These are model predictions, not device measurements.

Profile	8 KB	16 KB	32 KB	64 KB
lan (0% loss)	2.2	0.9	0.2	0.0
broadband (0.5%)	9.6	4.6	2.5	1.2
congested (3%)	30.1	17.3	9.5	5.0
weak (8%)	55.2	32.0	18.4	9.8

$$U = u / (T / 60) \qquad (3)$$

where U is the modelled underrun rate per minute, u the starvation events and T the audio duration in seconds; the rate is a property of the model, not a device measurement.

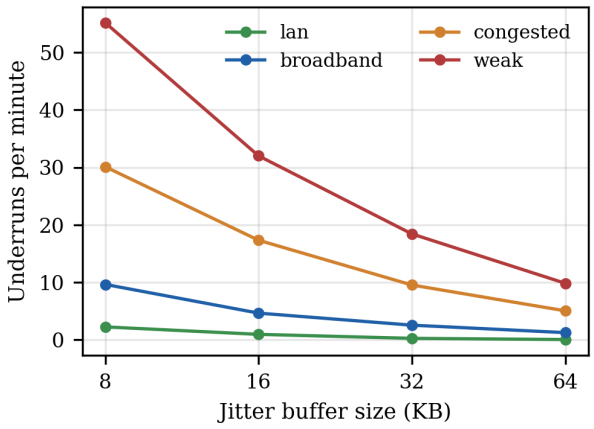


Fig. 8. Modelled underruns against buffer size; each doubling roughly halves the rate.

F. Onboarding Workflow

A tap-based workflow cuts the step count from five to two by removing scan, wait and select, and modelled setup time falls from 7.3 s to 2.1 s on the faster handset and from 8.9 s to 2.5 s on the slower one, about 71% (Table VII, Fig. 8). The absolute numbers rest on stage-cost assumptions and are not hardware performance, but the step count follows from the protocol definition and the direction of the effect matches the historical studies [5], [6]. Success rate is deliberately unmodelled, since scan and connection failures depend on real radios.

Handset profile	Manual	NFC tap	Gain	Steps
15 ms interval, MTU 247	7287 ms	2121 ms	70.9%	5 to 2
30 ms interval, MTU 185	8863 ms	2486 ms	72.0%	5 to 2

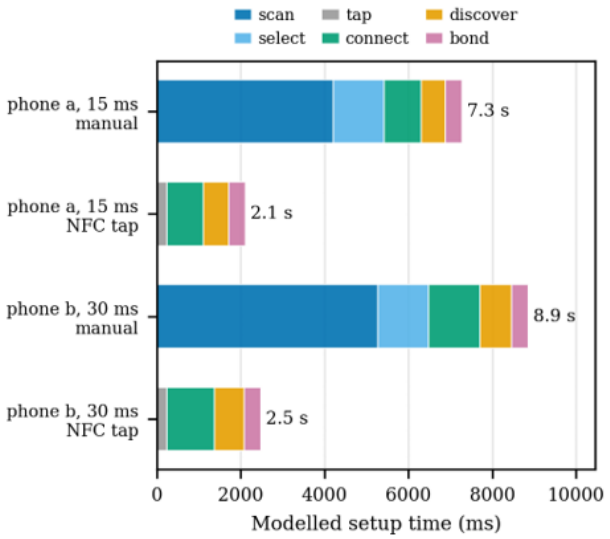


Fig. 9. Modelled setup time by stage; step counts are counted, timings are modelled.

VII. Discussion and Limitations

The results support a specific claim: for an embedded client, directory-declared health does not reliably predict playability, and verifying before announcing playback removes false play states at no measurable startup cost while making error reporting predictable. They say nothing about radio performance, audio fidelity, power or usability, and no such claims are made.

Four limitations bound the work. The measurement is one snapshot from one country, so latency and failure rates may not generalize. The paired comparison uses 60 stations with deliberate oversampling of dead entries, making the 16.7% false-play rate a property of that sample rather than an estimate for the whole directory, which is why per-stratum figures accompany it. The buffer and onboarding numbers come from models with stated but unvalidated parameters. And the twin cannot reproduce radio behaviour, so the coexistence and security arguments remain design reasoning backed by literature.

One constraint is independent of the simulation: the design avoids over-the-air broadcast, since a broadcast station in the Philippines needs a congressional franchise and regulator

permits [48], so the device streams over IP only and the twin inherits that boundary rather than testing it.

The path to hardware is additive. The NDEF bytes can be written to an NTAG213 unchanged, the JSON command set maps onto a NimBLE GATT server, the policy state machine already exists in the firmware sketch, and the buffer parameters become ring-buffer settings. The next step is to re-run the buffer sweep on a board, replace modelled underruns with measured ones, then repeat the onboarding study with two phones and the physical tag. Reviewers should read the onboarding comparison as a protocol plus a model and the policy comparison as a real result.

VIII. Conclusion

A multi-connectivity internet radio was redesigned as a software digital twin so the research could proceed without hardware. The directory client, stream validation, NDEF format, control protocol and playback policy run as real code; only the radio links and buffer behaviour are modelled, and every experiment states which is which. On live streams flag-trusting playback produced false play events in 16.7% of attempts against none for validation-first, and the modelled studies reproduce the buffering trade-off and a 71% setup-time reduction for tap-based onboarding. The application-layer contribution is supported by measurement; the radio-layer contribution remains a documented design awaiting hardware.

Acknowledgment

The authors thank the Radio Browser community for maintaining the open directory used in this work, the ns-3 and firmware-emulation communities whose validation studies informed the evaluation design, and the maintainers of Python, requests, pandas, numpy and matplotlib. The ESP32-S3 reference design and provisioning documentation are Espressif's. An AI assistant was used to draft code and manuscript text under author direction; it is not an author, and all claims remain the responsibility of the human authors.

References

[1] C. A. Lin, "Exploring the Online Radio Adoption Decision-Making Process: Cognition, Attitude, and Technology Fluidity," *Journalism & Mass Communication Quarterly*, vol. 86, no. 4, pp. 884–899, 2009, doi: 10.1177/107769900908600410.

[2] Radio Browser, "API documentation," 2026. [Online]. Available: https://docs.radio-browser.info/. [Accessed: Nov. 16, 2026].

[3] I.-G. Park, "The study on development of a digital internet radio receiver," *J. KIISE: Computing Practices and Letters*, vol. 12, no. 2, 2006. [Online]. Available: http://koreascience.or.kr/article/ArticleFullRecord.jsp?cn=JBGHIF_2006_v12n2_102

[4] N. Cameron, "Internet radio," in *Electronics Projects with the ESP8266 and ESP32*, Apress, pp. 1–25, 2021, doi: 10.1007/978-1-4842-6336-5_1.

[5] T. Salminen, S. Hosio, and J. Riekki, "Enhancing Bluetooth Connectivity with RFID," in *Fourth Annual IEEE International Conference on Pervasive Computing and Communications (PERCOM'06)*, None, pp. 36–41, doi: 10.1109/percom.2006.23.

[6] K. Seewoonauth, E. Rukzio, R. Hardy, and P. Holleis, "Touch & connect and touch & select," in *Proceedings of the 11th International Conference on Human-Computer Interaction with Mobile Devices and Services*, 2009, pp. 1–9, doi: 10.1145/1613858.1613905.

[7] J. Siva and C. Poellabauer, "Connection-Oriented BLE Traffic Servicing Characteristics on Android Devices," in *2020 IEEE International Workshop Technical Committee on Communications Quality and Reliability (CQR)*, 2020, pp. 1–6, doi: 10.1109/cqr47547.2020.9101377.

[8] L. J. Hwang, C. C. Shih, and I. T. Kuo, "A Novel Technique for Real-Time Internet Radio Recorder on Non-DSP Embedded System," in *2008 International Conference on Multimedia and Ubiquitous Engineering (mue 2008)*, 2008, pp. 96–101, doi: 10.1109/mue.2008.36.

[9] T. Kodera, "Adaptive antenna system by ESP32-PICO-D4 and its application to web radio system," *HardwareX*, vol. 3, pp. 91–99, 2018, doi: 10.1016/j.ohx.2018.03.001.

[10] N. N. Zahrah, D. Darlis, and S. Hadiyoso, "Desain dan implementasi penerima radio internet berbasis ESP32 untuk radio komunitas D3 teknologi telekomunikasi," *eProceedings of Applied Science*, Telkom University, 2025. [Online]. Available: https://openlibrarypublications.telkomuniversity.ac.id/index.php/appliedscience/en/article/view/28467

[11] W. Fischer, "Broadcast over Internet, HbbTV, OTT, Streaming," in *Signals and Communication Technology*, Springer International Publishing, pp. 903–913, 2020, doi: 10.1007/978-3-030-32185-7_44.

[12] M. Seufert, S. Egger, M. Slanina, T. Zinner, T. Hoßfeld, and P. Tran-Gia, "A Survey on Quality of Experience of HTTP Adaptive Streaming," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 1, pp. 469–492, 2015, doi: 10.1109/comst.2014.2360940.

[13] S. Takahashi, K. Yamagishi, and J. Okamoto, "Classification of Viewing Abandonment Reasons for Adaptive Bitrate Streaming," in *2020 Twelfth International Conference on Quality of Multimedia Experience (QoMEX)*, 2020, pp. 1–6, doi: 10.1109/qomex48832.2020.9123146.

[14] T. Kimura, T. Kimura, A. Matsumoto, and K. Yamagishi, "Balancing Quality of Experience and Traffic Volume in Adaptive Bitrate Streaming," *IEEE Access*, vol. 9, pp. 15530–15547, 2021, doi: 10.1109/access.2021.3052552.

[15] K. Yamagishi and T. Hayashi, "Parametric Quality-Estimation Model for Adaptive-Bitrate-Streaming Services," *IEEE Transactions on Multimedia*, vol. 19, no. 7, pp. 1545–1557, 2017, doi: 10.1109/tmm.2017.2669859.

[16] S. B. Moon, J. Kurose, and D. Towsley, "Packet audio playout delay adjustment: performance bounds and algorithms," *Multimedia Systems*, vol. 6, no. 1, pp. 17–28, 1998, doi: 10.1007/s005300050073.

[17] Y. Liang, N. Farber, and B. Girod, "Adaptive playout scheduling and loss concealment for voice communication over ip networks," *IEEE Transactions on Multimedia*, vol. 5, no. 4, pp. 482–493, 2003, doi: 10.1109/tmm.2003.819095.

[18] M. Y. Zhu, M. Zhang, X. Q. Yu, and W. G. Wan, "Streaming audio packet loss concealment based on sinusoidal frequency estimation in MDCT domain," *IEEE Transactions on Consumer Electronics*, vol. 56, no. 2, pp. 811–819, 2010, doi: 10.1109/tce.2010.5506006.

[19] A. Hines, E. Gillen, D. Kelly, J. Skoglund, A. Kokaram, and N. Harte, "Perceived Audio Quality for Streaming Stereo Music," in *Proceedings of the 22nd ACM international conference on Multimedia*, 2014, pp. 1173–1176, doi: 10.1145/2647868.2655025.

[20] A. Hsu, D. Wei, and C. c. Kuo, "Coexistence Mechanism Using Dynamic Fragmentation for Interference Mitigation between Wi-Fi Bluetooth," in *MILCOM 2006*, 2006, pp. 1–7, doi: 10.1109/milcom.2006.302042.

[21] S. He, B. Zhang, J. Luo, and M. Qu, "Research on coexistence and mutual interference of Wi-Fi and Bluetooth," in *2021 International Conference on Microwave and Millimeter Wave Technology (ICMMT)*, 2021, pp. 1–3, doi: 10.1109/icmmt52847.2021.9617988.

[22] A. A. Eltholth, "Improved Spectrum Coexistence in 2.4 GHz ISM Band Using Optimized Chaotic Frequency Hopping for Wi-Fi and Bluetooth Signals," *Sensors*, vol. 23, no. 11, pp. 5183, 2023, doi: 10.3390/s23115183.

[23] C. Gomez, J. Oller, and J. Paradells, "Overview and Evaluation of Bluetooth Low Energy: An Emerging Low-Power Wireless Technology," *Sensors*, vol. 12, no. 9, pp. 11734–11753, 2012, doi: 10.3390/s120911734.

[24] R. Rondon, K. Landernas, and M. Gidlund, "An analytical model of the effective delay performance for Bluetooth low energy," in *2016 IEEE 27th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, 2016, pp. 1–6, doi: 10.1109/pimrc.2016.7794553.

[25] J. Sim, Y. Lee, and J. S. Kim, "Concurrent Bluetooth module activation for reducing the latency of NFC connection handover," in *The 18th IEEE International Symposium on Consumer Electronics (ISCE 2014)*, 2014, pp. 1–2, doi: 10.1109/isce.2014.6884323.

[26] NFC Forum, “Connection handover technical specification,” NFC Forum. [Online]. Available: https://nfc-forum.org/learn/technical-specifications/. [Accessed: Nov. 16, 2026].

[27] E. Haselsteiner and K. Breitfuß, “Security in near field communication (NFC): Strengths and weaknesses,” in *Proc. Workshop on RFID Security (RFIDSec)*, Graz, Austria, 2006.

[28] C. H. Chen, I. C. Lin, and C. C. Yang, “NFC Attacks Analysis and Survey,” in *2014 Eighth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, 2014, pp. 458–462, doi: 10.1109/imis.2014.66.

[29] C. Li et al., “Passwords in the Air,” in *Proceedings of the 11th ACM Conference on Security & Privacy in Wireless and Mobile Networks*, 2018, pp. 1–11, doi: 10.1145/3212480.3212496.

[30] R. A. K. Fezeu, T. J. Salo, A. Zhang, and Z.-L. Zhang, “Dissecting IoT device provisioning process,” arXiv:2310.14125, Oct. 2023. [Online]. Available: https://arxiv.org/abs/2310.14125

[31] T. Claverie and J. L. Esteves, “BlueMirror: Reflections on Bluetooth Pairing and Provisioning Protocols,” in *2021 IEEE Security and Privacy Workshops (SPW)*, 2021, pp. 339–351, doi: 10.1109/spw53761.2021.00054.

[32] Y. Zhang, J. Weng, R. Dey, Y. Jin, Z. Lin, and X. Fu, “Breaking secure pairing of Bluetooth Low Energy using downgrade attacks,” in *Proc. 29th USENIX Security Symp.*, 2020. [Online]. Available: https://www.usenix.org/conference/usenixsecurity20/presentation/zhang-yue

[33] H. Greß, B. Krüger, and E. Tischhauser, “The Newer, the More Secure? Standards-Compliant Bluetooth Low Energy Man-in-the-Middle Attacks on Fitness Trackers,” *Sensors*, vol. 25, no. 6, pp. 1815, 2025, doi: 10.3390/s25061815.

[34] M. Shi, J. Chen, K. He, H. Zhao, M. Jia, and R. Du, “Formal analysis and patching of BLE-SC pairing,” in *Proc. 32nd USENIX Security Symp.*, 2023. [Online]. Available: https://www.usenix.org/conference/usenixsecurity23/presentation/shi-min

[35] Espressif Systems, “BluFi: Wi-Fi network configuration via Bluetooth channel,” ESP-IDF Programming Guide. [Online]. Available: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-guides/ble/blufi.html. [Accessed: Nov. 16, 2026].

[36] B. R. Barricelli, E. Casiraghi, and D. Fogli, “A Survey on Digital Twin: Definitions, Characteristics, Applications, and Design Implications,” *IEEE Access*, vol. 7, pp. 167653–167671, 2019, doi: 10.1109/access.2019.2953499.

[37] R. Minerva, G. M. Lee, and N. Crespi, “Digital Twin in the IoT Context: A Survey on Technical Features, Scenarios, and Architectural Models,” *Proceedings of the IEEE*, vol. 108, no. 10, pp. 1785–1824, 2020, doi: 10.1109/jproc.2020.2998530.

[38] D. Magrin, S. Avallone, S. Roy, and M. Zorzi, “Validation of the ns-3 802.11ax OFDMA implementation,” in *Proceedings of the Workshop on ns-3*, 2021, pp. 1–8, doi: 10.1145/3460797.3460798.

[39] N. Baldo et al., “Validation of the IEEE 802.11 MAC model in the ns3 simulator using the EXTREME testbed,” in *Proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques*, 2010, doi: 10.4108/icst.simutools2010.8705.

[40] D. Amalfitano, N. Amatucci, V. De Simone, V. Riccio, and F. A. Rita, “Towards a Thing-In-the-Loop approach for the Verification and Validation of IoT systems,” in *Proceedings of the 1st ACM Workshop on the Internet of Safe Things*, 2017, pp. 57–63, doi: 10.1145/3137003.3137007.

[41] M. Kim, D. Kim, E. Kim, S. Kim, Y. Jang, and Y. Kim, “FirmAE: Towards Large-Scale Emulation of IoT Firmware for Dynamic Analysis,” in *Annual Computer Security Applications Conference*, 2020, pp. 733–745, doi: 10.1145/3427228.3427294.

[42] J. Brooke, “SUS: A 'Quick and Dirty' Usability Scale,” in *Usability Evaluation In Industry*, CRC Press, pp. 207–212, 1996, doi: 10.1201/9781498710411-35.

[43] D. S. Zwakman, D. Pal, T. Triyason, and V. Vanijja, “Usability of Voice-based Intelligent Personal Assistants,” in *2020 International Conference on Information and Communication Technology Convergence (ICTC)*, 2020, pp. 652–657, doi: 10.1109/ictc49870.2020.9289550.

[44] T. Doi and Y. Nishikawa, “Usability Analysis of Smart Speakers from a Learnability Perspective for Novel Users,” *Applied System Innovation*, vol. 7, no. 3, pp. 36, 2024, doi: 10.3390/asi7030036.

[45] M. Domingues, J. N. Faria, and D. Portugal, “Dimensioning payload size for fast retransmission of MQTT packets in the wake of network disconnections,” *EURASIP Journal on Wireless Communications and Networking*, vol. 2024, no. 1, 2024, doi: 10.1186/s13638-023-02327-3.

[46] Y. Liu, L. Guo, F. Li, and S. Chen, “An empirical evaluation of battery power consumption for streaming data transmission to mobile devices,” in *Proceedings of the 19th ACM international conference on Multimedia*, 2011, pp. 473–482, doi: 10.1145/2072298.2072360.

[47] Espressif Systems, “ESP32-S3 series datasheet,” 2025. [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32-s3_datasheet_en.pdf. [Accessed: Nov. 16, 2026].

[48] Republic of the Philippines, “Act No. 3846: An act providing for the regulation of public and radio communications in the Philippines and for other purposes (as amended),” 1931. [Online]. Available: https://www.chanrobles.com/acts/act3846.htm. [Accessed: Nov. 16, 2026].