

padL1: A Low-Latency Drum-Learning Game With Force-Aware Piezo Pads

Authors: Caleb Song, Rishabh Cowlagi, Abishek Anand

Affiliation: Electrical and Computer Engineering, Carnegie Mellon University

Abstract— padL1 is a portable, low-cost, multi-pad drum-learning system that combines piezoelectric strike sensing with a Guitar Hero style game interface. Using piezoelectric sensing, it detects which pad is hit and estimates strike intensity (weak/normal/hard), then scores the player on timing, pad correctness, and force. Compared to existing drum sets, padL1 is much more portable and it is also the first interactive drum game that doubles as a learning platform. Our target is sub-80 ms strike-to-sound latency and 90% note accuracy to advance difficulty levels, enabling faster skill acquisition through structured, game-based practice.

Index Terms— drum pad, piezoelectric sensor, rhythm game, timing accuracy, force classification, low latency, USB, audio synthesis

1 INTRODUCTION

Learning drums typically requires a physical kit, practice space, and real-time feedback from a human. Digital rhythm games provide some engagement but lack authentic drum strike dynamics, while electronic drum kits provide realism at higher cost and complexity and a lack of learning feedback (unless you additionally pay for an instructor). PadL1 targets an accessible middle ground: a compact set of three physical pads that provide immediate audio feedback and a structured, game-like progression conducive to learning.

The core user experience is a *multi-pad* interface representing various drum sounds such as kick, snare, or cymbal. In a game learning interface inspired by Guitar Hero, notes from a selected song scroll down the screen, indicating *when* to hit, *which* pad to hit, and *how hard* to hit (weak/normal/hard). The user is then scored on (i) timing, (ii) correct pad selection, and (iii) force category. This software system runs on a user-provided computer that both renders the game UI and plays drum audio and connects with the hardware components via USB. This decision prioritizes low latency and integration speed, while preserving a clear path to future embedded audio and wireless expansion (Appendix ??).

PadL1 supports two modes: (a) a default game with three difficulty levels and curated song excerpts, and (b) user-uploaded songs. For uploaded songs, padL1 uses Demucs ([3]) to isolate the drum sounds from the rest of the song. These drum sounds are then analyzed with both frequency (for determining the type of drum sound made) and amplitude (for determining the loudness of the sound)

passes. This report describes requirements, architecture, design tradeoffs, implementation, and a validation plan for the resulting product.

2 USE-CASE REQUIREMENTS

We derive use-case requirements from the end-to-end experience of a beginner-to-intermediate player using padL1 for practice sessions.

Table 1: Use-case requirements for padL1.

Requirement	Target / Rationale
Low perceived latency	End-to-end audio feedback should feel instantaneous (target < 80 ms perceptual threshold for noticing delays).
Accurate scoring	Score reflects pad correctness, timing, and force to encourage technique (pad/force accuracy).
Clear, learnable difficulty ladder	Users progress from easy→medium→hard only if overall accuracy > 90% on an excerpt (defined in Section 4.1).
Playable in typical dorm/apartment	Compact footprint; safe force range; moderate acoustic volume via computer speakers and/or headphones.
Robust to false triggers	Avoid phantom hits due to pad ringing or vibration; no noticeable “double hits” after single strikes.
Supports user song upload	Accept WAV/MP3 upload and generate a playable note chart with demucs based music analysis.

In addition to these user-facing goals, our product’s design process prioritizes rapid iteration. Therefore, several requirements are specified as measurable engineering specifications in Section 4.

3 ARCHITECTURE AND/OR PRINCIPLE OF OPERATION

PadL1 is composed of (i) a multiple-pad piezo sensing front-end, (ii) wireless RF based communication between distributed pads and central hub, (iii) central hub for pad data aggregation and software communication, (iv) USB event transport to a user-provided computer, (v) software

learning platform that includes a game engine that renders notes and evaluates hits, and (vi) audio synthesis/playback.

3.1 System overview

Figure 1 shows our product’s planned architecture. Each pad contains multiple piezo sensors to determine where each pad was hit based on triangulation. Additionally, each pad contains an acrylic plate above the sensors to accurately transmit the hit vibrations to the sensor and foam pads underneath to provide dampening and isolation from environmental factors. When struck, the piezo produces a voltage peak which the MCU reads while sampling the pads and adds the corresponding pad id before forwarding it to the central hub over RF. The central hub is able to process the tuple and map it to its corresponding sound profile using onboard memory lookup table. Then it forwards the input in MIDI compliant format to the software system via USB. The software system then compares events to a time-aligned note chart and scores the user based on the accuracy.

3.2 Timing model and latency budget

The user perceives correctness primarily through *audio timing*. Humans can detect audiovisual delays on the order of tens of milliseconds in controlled settings [4]. Rhythm games commonly use tight judgment windows (tens of milliseconds) for high-accuracy scoring [6]. We predict our product’s latency budget to be dominated by (a) sensor-to-event detection time, (b) USB transmission and scheduling, and (c) audio buffer + OS scheduling. We hope to largely mitigate (c) by using low-latency audio libraries and utilizing OS buffers.

3.3 Default game content (3 levels)

The default game includes three short drum-driven excerpts. We select songs based on widely accepted drum difficulty categorizations and instructional materials:

- **Easy:** “Seven Nation Army” (The White Stripes) — commonly recommended as a beginner drum song [1].
- **Medium:** “Smells Like Teen Spirit” (Nirvana) — often taught as an intermediate progression with multiple groove levels [5].
- **Hard:** “YYZ” (Rush) — available as “Advanced” drum sheet music, reflecting complex patterns and coordination [7].

In our final product, we will enable a checkpointing mechanism which allows a user to dynamically adjust the difficulty of their input by specifically selecting either easy or hard parts of the song.

4 DESIGN REQUIREMENTS

We translate the use-case requirements (Section 2) into verifiable design specifications. Each specification includes a verification method that is implemented in the test plan (Section 7).

Table 2: Design requirements mapped from use-case requirements.

DR ID	Engineering specification	Verification
DR-1	End-to-end audio latency from pad strike to audible output < 25 ms.	Oscilloscope and/or audio loopback
DR-2	Hit detection false double-trigger rate < 5% of strikes during normal play.	Repeated strike trials
DR-3	Pad classification accuracy $\geq 95\%$ (kick/snare/cymbal) under isolated strikes.	Confusion matrix
DR-4	Force bin classification accuracy $\geq 90\%$ over 3 bins in controlled trials.	Confusion matrix
DR-5	Scoring correctness uses timing+pad+force (Section 4.1) and reports per-song accuracy.	Software unit tests
DR-6	User-upload pipeline produces a playable chart for > 90% of tested songs.	Dataset evaluation

4.1 Scoring definition

Each note target is represented as (t_i, p_i, f_i) where t_i is the desired strike time, $p_i \in \{\text{kick, snare, cymbal}\}$, and $f_i \in \{\text{weak, normal, hard}\}$. A player event is $(\hat{t}_j, \hat{p}_j, \hat{f}_j)$.

Matching rule: An event matches a target if (i) $\hat{p}_j = p_i$, (ii) $\hat{f}_j = f_i$, and (iii) $|\hat{t}_j - t_i| \leq W$ where W is the timing window. We set W based on perceptual considerations and rhythm-game practice [6]:

- **Perfect:** $|\Delta t| \leq 25$ ms
- **Good:** $25 < |\Delta t| \leq 50$ ms
- **Ok:** $50 < |\Delta t| \leq 90$ ms
- **Miss:** $|\Delta t| > 90$ ms, or wrong pad, or wrong force

Accuracy: We define per-song accuracy as

$$\text{Accuracy} = \frac{\#(\text{matched targets})}{\#(\text{targets})}$$

where “matched” requires correct pad+force and timing within the *Ok* window. Users advance to the next difficulty if accuracy > 90% on the excerpt.

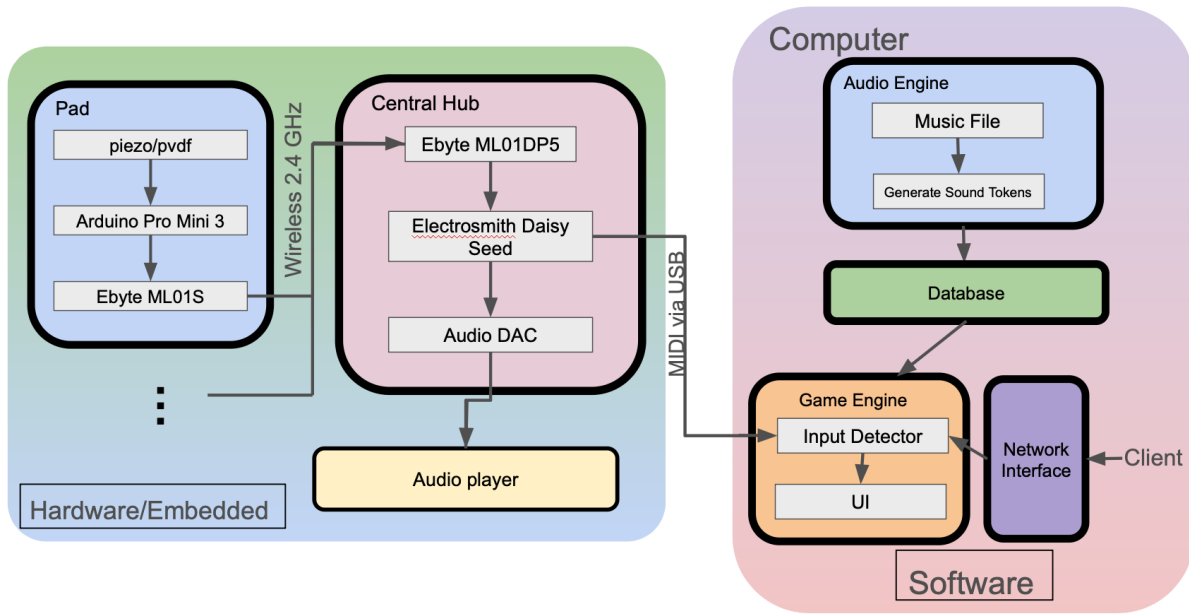


Figure 1: padL1 MVP architecture: distributed piezo pads feed an MCU for hit detection and force classification; events are sent from pads via wireless communication to central hub for aggregation; events are then forwarded via USB to a computer for game engine and scoring and audio player for audio output.

5 DESIGN TRADE STUDIES

This section motivates key engineering decisions for the MVP and documents alternatives.

5.1 Trade Study 1: 35mm Piezo Sensors vs PVDF and FSR

Options:

1. **35mm Piezo Sensors (chosen for MVP):** Small, cheap disc sensor capable of generating $10V_{pp}$ through vibration of piezo material.
2. **PVDF and FSR:** Lightweight resistor and sensors able to pick up higher range of frequencies and offers high sensitivity for strike sensing.

Decision factors: Sensitivity and characterizing strikes. 35mm Piezo disc sensors are cheap, \$0.86 per disc, and take up $3.85mm^2$ space compared to the FSR and PVDF which combined take up $3.5mm^2$ but cost \$15.83 for one of each. Thus in terms of prototyping the product, the cheaper option was more viable for quick iterative testing on different types of hit classifications and configurations for location sensing.

Furthermore, For the PVDF and FSR combination setup the drum head can no longer be a rubber drum head with an acrylic impact plate as that dampens the impact of the user. The head needs to be replaced with a mesh head with plastic rims that provide the tension on the mesh head necessary for the PVDF and FSR to pick up rim shots and regular hits. The FSR and PVDF do not provide as much peak to peak voltage as the piezo discs requiring an amplifier circuitry to reliably distinguish the measurements.

Although the latter option provides greater range of specific hits, the risks of hardware integration, size of the entire pad, additional circuitry, and cost are not viable for rapid prototyping of the MVP.

Outcome: We choose the 35mm Piezo Sensors for the MVP to ensure a working end-to-end system that provides enough measurements for impact classification while maintaining a reasonable price.

5.2 Trade Study 2: nRF24L01+ and Arduino Pro Mini 3.3V vs ESP32

Options:

1. **nRF24L01+ and Arduino Pro Mini 3.3V (chosen for MVP):** Low power, low network jitter, and up to $2Mbps$ transmission rate allowing for average latency of $2ms$.
2. **ESP32:** Module comes with ESPNOW communication protocol with large amounts of documentation. Also contains onboard lightweight MCU that can packetize and communicate with RF module.

Decision factors: Pad-to-hub latency and performance. The ESP32 does have ample documentation and does not require extra circuitry to interface the communication between an MCU and the RF module onboard. However, the ESPNOW protocol does have some overhead and is susceptible to network jitters which can be prevalent during demonstrations with each unit costing around \$8.00.

Comparatively, the nRF24L01+ and Arduino Pro Mini configuration ensures low latency and low network jitters while providing a strong transmission rate. Furthermore,

the cost of the combined configuration is \$7.67 but does take slightly more space on the pad.

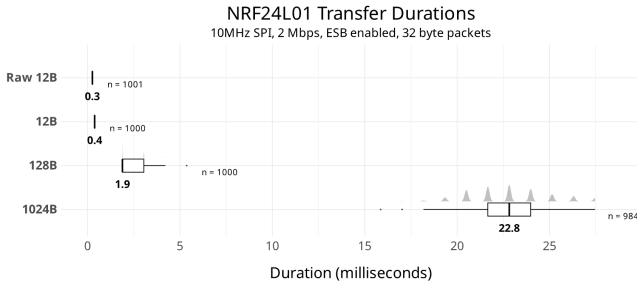


Figure 2: Overall on the 2Mbps settings, the nRF24L01+ module is able to transfer 12B packets with a latency of 0.9ms which will provide a larger latency budget for the software processing and latency of the end-to-end system.[2]

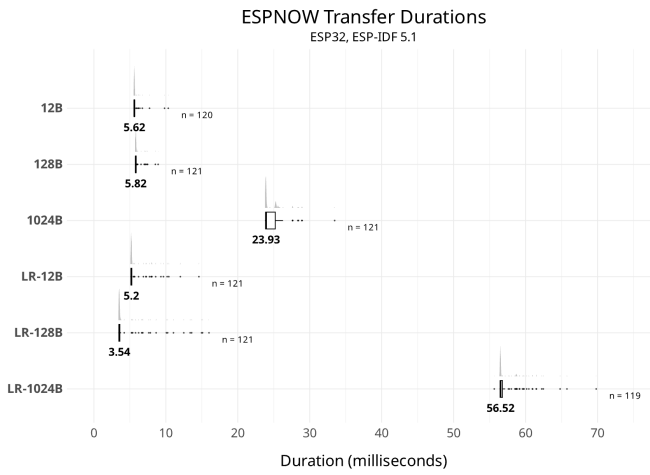


Figure 3: On average, the ESPNOW module is able to transfer 12B, 128B packets with a latency of 5.6ms which is acceptable for the system depending on any packet size. [2]

Outcome: We choose the nRF24L01+ and Arduino Pro Mini 3.3V configuration because of its reasonable price and fast transmission rates which gives us a larger latency budget for the rest of our system. Although, it takes up more space compared to the ESP32 and extra wiring overhead the benefits of the latency improvement are better for our system which requires quick impact-to-audio feedback to make it an authentic instrument. If we have to send bigger packets, then ESP32 would be a more viable option due to its consistent latency measurements.

5.3 Trade Study 3: USB-to-PC audio (MVP) vs embedded audio (future)

Options:

1. **USB-to-PC (chosen for MVP):** MCU sends events to a PC, and the PC plays audio.
2. **Embedded audio (future):** MCU triggers local audio playback and optionally sends events to PC for UI and scoring.

Decision factors: latency, implementation complexity, portability, and integration risk. USB-to-PC reduces hardware risk and leverages existing audio libraries, but may introduce OS scheduling jitter. Embedded audio reduces dependence on OS buffering, but requires more complex real-time audio firmware and DAC/codec integration.

Outcome: We choose USB-to-PC for MVP to ensure a working end-to-end system while hardware parts ship. We document embedded audio as an extension in Appendix ??.

5.4 Trade Study 4: Force metric representation (3 bins vs continuous velocity)

Options:

1. **3 bins (chosen):** weak / normal / hard guidance; robust and easy to communicate.
2. **Continuous velocity:** a 0–127 scale (MIDI-style) allowing finer feedback.

Tradeoff: Continuous velocity improves nuance but requires a stable calibration curve across users and pad mounting variability. Three bins can be tuned per-pad with pre-defined thresholds and yields clear instruction for beginners.

Chosen approach: Bin by peak amplitude using per-pad thresholds ($\theta_1, \theta_2, \theta_3$):

$$f = \begin{cases} \text{weak} & \text{if } \theta_1 < A < \theta_2 \\ \text{normal} & \text{if } \theta_2 \leq A < \theta_3 \\ \text{hard} & \text{if } A \geq \theta_3 \end{cases}$$

where A is the measured peak amplitude within a strike window. Based on oscilloscope testing, we found $\theta_1 \approx 280$ mV for noise; $\theta_2 \approx 1.28$ V; and $\theta_3 \approx 2.0$ V Threshold selection is validated in Section 7.

5.5 Trade Study 5: User-upload chart generation (Demucs + transients)

For user-uploaded songs, our goal is to produce (t_i, p_i, f_i) targets with minimal manual authoring.

Chosen pipeline: (1) separate drum stem using Demucs [3]; (2) detect onsets/transients; (3) classify each onset to one of {kick, snare, cymbal} using spectral features; (4) estimate force from transient energy.

Alternatives: Manual chart authoring (high quality but labor-intensive), or end-to-end drum transcription ML models. We select Demucs because it is widely used, open source, and reduces non-drum interference before onset detection, and is the simplest option.

6 SYSTEM IMPLEMENTATION

6.1 Subsystem A: Piezo pad hardware

Each pad is a 20 cm × 20 cm striking surface with an overall thickness of 3.5 cm. At the top, the neoprene rubber acts as a drum head, absorbing all impacts to the

drum and dampening it to a degree that does not destroy the pad. The acrylic plate is the impact plate and essentially is rigid enough that it preserves the vibrations from the drum head which can get picked up by the piezoelectric disks mounted under the surface. The piezo output is conditioned for ADC safety using voltage regulator circuitry that maps the impact value from $[-380 \text{ mV}, 3.3 \text{ V}]$. Lastly, the EVA foam and the plywood backing improve the durability of the system and dampen environmental noise from activating the piezo sensors. The mechanical stack aims to (i) preserve sensitivity, (ii) minimize cross-talk between pads through mounting isolation, and (iii) provide the system with physical protection.

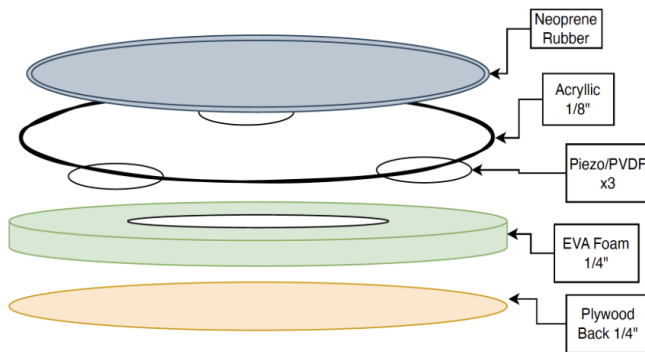


Figure 4: Piezo pad mechanical stack-up (illustrative).

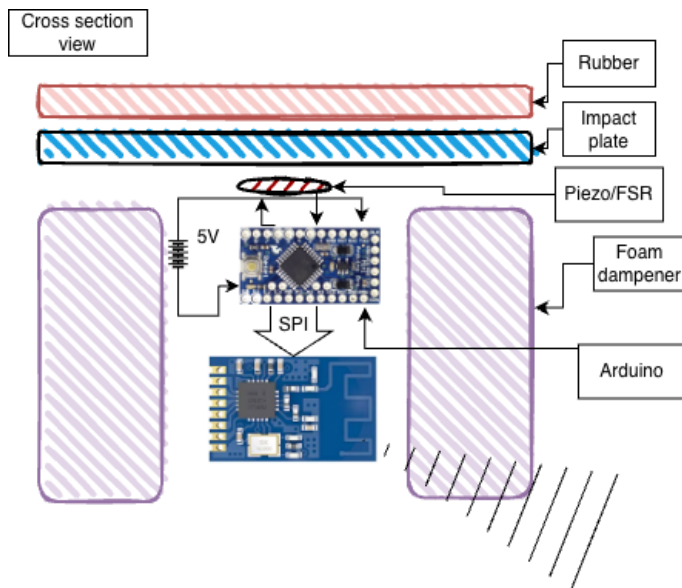


Figure 5: Piezo pad mechanical cross-section (illustrative).

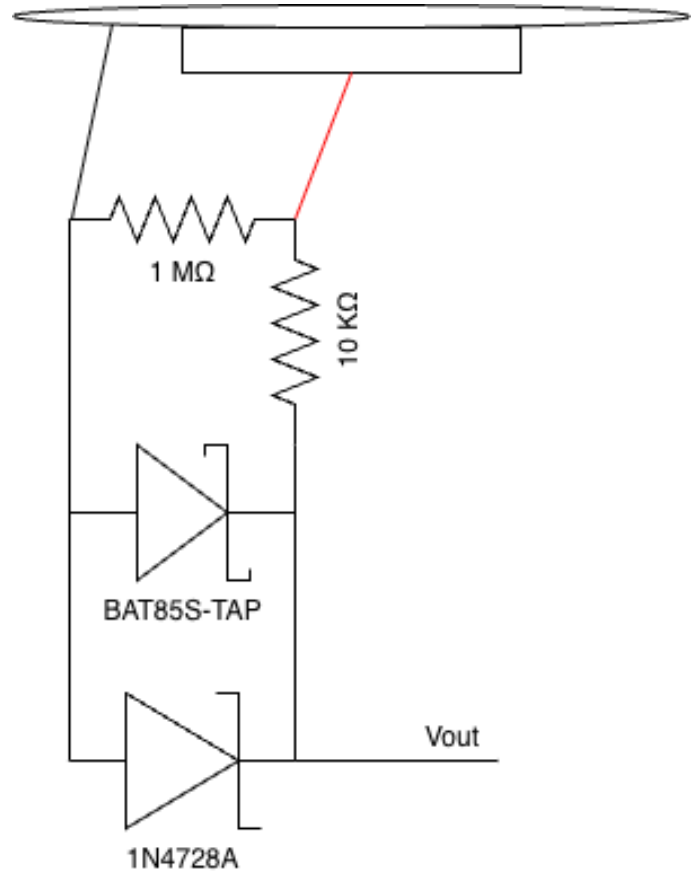


Figure 6: Voltage regulator circuit which clamps the voltage levels to $[-380 \text{ mV}, 3.3 \text{ V}]$, the BAT85S-TAP clamps the negative voltage to -380 mV and the 1N4728A clamps the voltage to be 3.3 V which protects the Arduino from harmful voltage levels.

6.2 Subsystem B: Embedded sensing firmware (hit detect + force)

The MCU samples three analog channels (one per pad) at 115200 baud and runs a lightweight state machine:

- **Idle:** monitor for threshold crossing indicating a strike onset.
- **Peak capture:** within a fixed window (e.g., 5–10 ms), track peak amplitude A and timestamp $\hat{t}$.
- **Debounce/ring suppression:** ignore subsequent peaks for a refractory interval (e.g., 20–40 ms) to prevent double triggers.
- **Force binning:** map A to weak/normal/hard using thresholds θ_1, θ_2 .

The firmware emits a compact event packet:

$$(\hat{t}, \hat{p}, \hat{f})$$

where $\hat{t}$ is a local timestamp (ms), $\hat{p}$ is pad ID, and $\hat{f}$ is force bin. We will be using the daisy seed microcontroller provided by Electrosmith because of its 64 MB of SRAM for audio file storage, SPI pins for communication with the central hub, and USB interface with support for MIDI communication.

6.3 Subsystem C: USB event transport

The MVP uses a wired USB connection to the PC. Events may be encoded as:

- **USB MIDI:** standardized event framing

We will implement MIDI and quantify event-to-receive latency in Section 7.

6.4 Subsystem D: PC game engine + audio

The PC software maintains a chart of targets (t_i, p_i, f_i) and performs online matching as the song plays. The UI renders falling notes with pad icons and force markers. Audio playback uses a per-pad sample set (kick/snare/cymbal) with amplitude scaling for force bins.

6.4.1 Default mode

Default mode contains three curated excerpts (Section 3). The player begins at easy and advances if accuracy > 90%. To encourage learning, the game reports per-dimension statistics:

- timing histogram of Δt ,
- pad confusion counts,
- force confusion counts.

6.4.2 User-upload mode

User-upload mode performs offline preprocessing:

1. User uploads an audio song file (such as a .WAV, .MP4, etc).
2. Using demucs, isolate drum sounds [3].
3. Using FFT, determine frequency of each sound and based on the frequency assign to one of kick, snare, and cymbals.
4. Using amplitude analysis, determine the loudness of each sound.
5. Combine resulting sound and loudness with timestamp to construct (timestamp, sound, loudness) tuples.

The resulting chart is stored in a file as a list of (timestamp, sound, loudness) tuples.

7 TEST & VALIDATION

We validate each design specification in Section 4.

7.1 Tests for DR-1: End-to-end latency

Goal: verify strike-to-audio latency < 80 ms.

Method:

1. Attach an oscilloscope probe to the piezo output (or MCU digital “strike detected” GPIO) to mark strike time.
2. Record the audio output using a loopback (audio interface) or microphone near the speaker.
3. Measure the delay between strike marker and the first significant audio waveform onset.

Acceptance: mean latency < 80 ms and 95th percentile < 100 ms.

7.2 Tests for DR-2: Double-trigger and false positive rate

Goal: ensure strike detection does not create spurious hits.

Method:

- Perform 200 controlled strikes per pad using a method that produces a forces.
- Count extra events within the refractory interval after each strike.
- Measure false positives during 2 minutes of idle vibration (table taps nearby).

7.3 Tests for DR-3: Pad classification accuracy

Method: strike each pad 200 times (isolated), log predicted pad ID, and compute confusion matrix. Expect near-perfect accuracy because each pad is physically distinct (one sensor per pad).

7.4 Tests for DR-4: Force bin classification accuracy

Method: perform labeled trials for weak/normal/hard by using a simple rig which can drop a stick consistently from a configurable height (where a higher height would lead to a higher force bin). Compute confusion matrix and tune $\theta_1, \theta_2, \theta_3$ per pad.

7.5 Tests for DR-5: Scoring correctness

Method: create synthetic event logs that hit targets at known offsets (e.g., $\Delta t = -80, -40, -10, 0, +10, +40, +80$ ms) and verify the assigned judgments and accuracy computation. This is a software unit test for Section 4.1.

7.6 Tests for DR-6: User-upload pipeline success rate

Method: evaluate on a set of 10 songs across genres. For each, verify that:

- Demucs produces a drum stem [3],
- onset detector produces non-empty targets,
- chart plays end-to-end without crashes,

This should have a 100% success rate.

8 PROJECT MANAGEMENT

8.1 Schedule

Figure 7 summarizes milestones: software MVP (in progress), hardware integration (in progress), calibration/testing, and final demo.

8.2 Team Member Responsibilities

- **Abishek Anand:** game engine, scoring logic, user-upload preprocessing pipeline, integration testing.
- **Rishabh Cowlagi:** embedded firmware, hit detection and force classification, USB transport.
- **Caleb Song:** pad mechanical design, piezo mounting, analog conditioning, hardware integration and calibration.

8.3 Bill of Materials and Budget

Table 3 lists the BOM.

8.4 TechSpark Usage Plans

We plan to use TechSpark for mechanical prototyping (pad mounting plates) and basic electronics assembly/testing (soldering, wire management).

8.5 Risk Mitigation Plans

Table 4: Key risks and mitigations for padL1 MVP.

Risk	Impact	Mitigation
Piezo ringing causes false hits	Inflated score errors, user frustration	Debounce + refractory interval; damping layer; per-pad threshold tuning
Computer audio buffering adds latency/jitter	“Laggy” feedback; timing feels off	Low-latency audio API; small OS buffers; extensive latency benchmarking and optimization
User-upload charts misaligned	Poor playability	Test driven synchronization or move audio output to computer thus eliminating the issue
Parts shipping delays	Delays hardware integration	Parallel software development; simulate pad events; use temporary sensors if needed

9 RELATED WORK

Commercial electronic drum kits provide multi-pad sensing and low-latency audio but are significantly more expensive and less portable than the padL1 MVP. Rhythm games (e.g., Guitar Hero/Rock Band style) provide engaging visuals and structured difficulty ladders, motivating our UI design and progression rule. For audio preprocessing of user-uploaded tracks, modern source separation tools such as Demucs [3] enable more reliable onset extraction by isolating drums from harmonic content.

10 SUMMARY

padL1 is a three-pad, force-aware rhythm game designed to make drum practice more accessible. The MVP architecture prioritizes rapid integration and low latency by using USB-to-PC transport and PC audio synthesis. Key design choices include three discrete force bins, a timing+pad+force scoring rule, and a default curriculum based on curated excerpts with increasing drum complexity [1, 5, 7]. Our validation plan measures end-to-end latency, hit robustness, and classification accuracy, and we outline clear extensions for embedded audio and wireless connectivity.

Glossary of Acronyms

- ADC – Analog-to-Digital Converter
- API – Application Programming Interface
- BPM – Beats Per Minute

Table 3: Bill of Materials and Budget (padL1 MVP)

Description	Model #	Manufacturer	Qty	Cost (\$)
Piezo film sensor	DT1-052K	TE Connectivity	2	29.63
Force sensing resistor (FSR 406)	30-73258	Interlink Electronics	2	21.64
Piezo disc 27mm	CEB-2704-L100	Same Sky (CUI Devices)	10	6.80
Piezo disc 35mm	CEB-35035-L100	Same Sky (CUI Devices)	18	26.45
Neoprene rubber pad sheet	72"x17"x1/4"	XCEL	2	31.98
Acrylic sheet 12"x12"	2-Pack	CALPALMY	2	19.98
Double-sided foam mounting tape	4026-style	3M	1	5.99
EVA foam tiles (isolation base)	24"x24"x1/2"	We Sell Mats	1	36.99
Embedded audio MCU (future extension)	Daisy Seed	Electrosmith	1	30.07
2.4GHz transceiver module (2-pack)	E01-ML01DP5	Ebyte	1	14.99
2.4GHz transceiver module	E01-ML01S	Ebyte	1.77	10.61
Capacitor assortment kit	13698	SparkFun Electronics	1	18.49
Schottky diode	BAT85S-TAP	Vishay	20	8.91
Zener diode	1N4728A	onsemi	20	9.07
Resistors 10k Ω	CF14JT10K0	Stackpole Electronics	25	7.36
Resistors 1M Ω	CF14JT1M00	Stackpole Electronics	25	7.44
Dual-gang potentiometer (10k log)	PDB182-K415K-103A1	Bourns	2	11.03
ATmega328P Pro Mini (3.3V)	Pro Mini Pack	AITRIP	1	17.99
Wired earbuds (3.5mm)	–	AmazonBasics	1	9.49
Spray adhesive	Super 77	3M	1	13.48
Pine plywood base	23/32" x24" x48"	Unbranded	1	26.24
Solderless breadboard (400 tie)	64	Adafruit	6	35.77
Panel-mount headphone jack	3.5mm stereo	BLLNDX	2	11.98
Total Budget				408.40

- DAC – Digital to Audio Convertor
- FSR – Force Sensitive Resistor
- MCU – Microcontroller Unit
- MIDI – Musical Instrument Digital Interface
- PVDF – Polyvinylidene Fluoride
- SPI – Serial Peripheral Interface
- SRAM – Static Random Access Memory
- USB – Universal Serial Bus

References

- [1] *15 Easy Drum Songs for Beginners*. Gear4music Blog. Accessed 2026-02-23. 2023. URL: <https://www.gear4music.com/blog/15-easy-drum-songs-for-beginners/>.
- [2] *Benchmarking latency across common wireless links for microcontrollers*. Electric UI Blog Post. Accessed 2026-02-23. n.d. URL: <https://electricui.com/blog/latency-comparison#microbenchmarking-embedded-hardware>.
- [3] *Demucs: Music Source Separation*. GitHub repository. Accessed 2026-02-23. n.d. URL: <https://github.com/facebookresearch/demucs>.
- [4] R. Eg et al. “Perceived synchrony for realistic and dynamic audiovisual content”. In: *PLoS ONE / PubMed Central* (2015). Accessed 2026-02-23. URL: <https://pmc.ncbi.nlm.nih.gov/articles/PMC4451240/>.
- [5] *How To Play "Smells Like Teen Spirit" (Beginner To Pro)*. Drumeo. Accessed 2026-02-23. 2022. URL: <https://www.drumeo.com/beat/how-to-play-smells-like-teen-spirit-beginners/>.
- [6] *Rhythm Game Crash Course – Native Audio*. Exceed7. Accessed 2026-02-23. n.d. URL: <https://exceed7.com/native-audio/rhythm-game-crash-course/index.html>.
- [7] *YYZ (Advanced Level) (Rush) – Drums Sheet Music*. Tomplay. Accessed 2026-02-23. 2024. URL: <https://tomplay.com/drums-sheet-music/rush/yyz-advanced-level-drums-score>.

A Gantt Chart

Gantt chart summarizing work schedule for product design split between three team members.

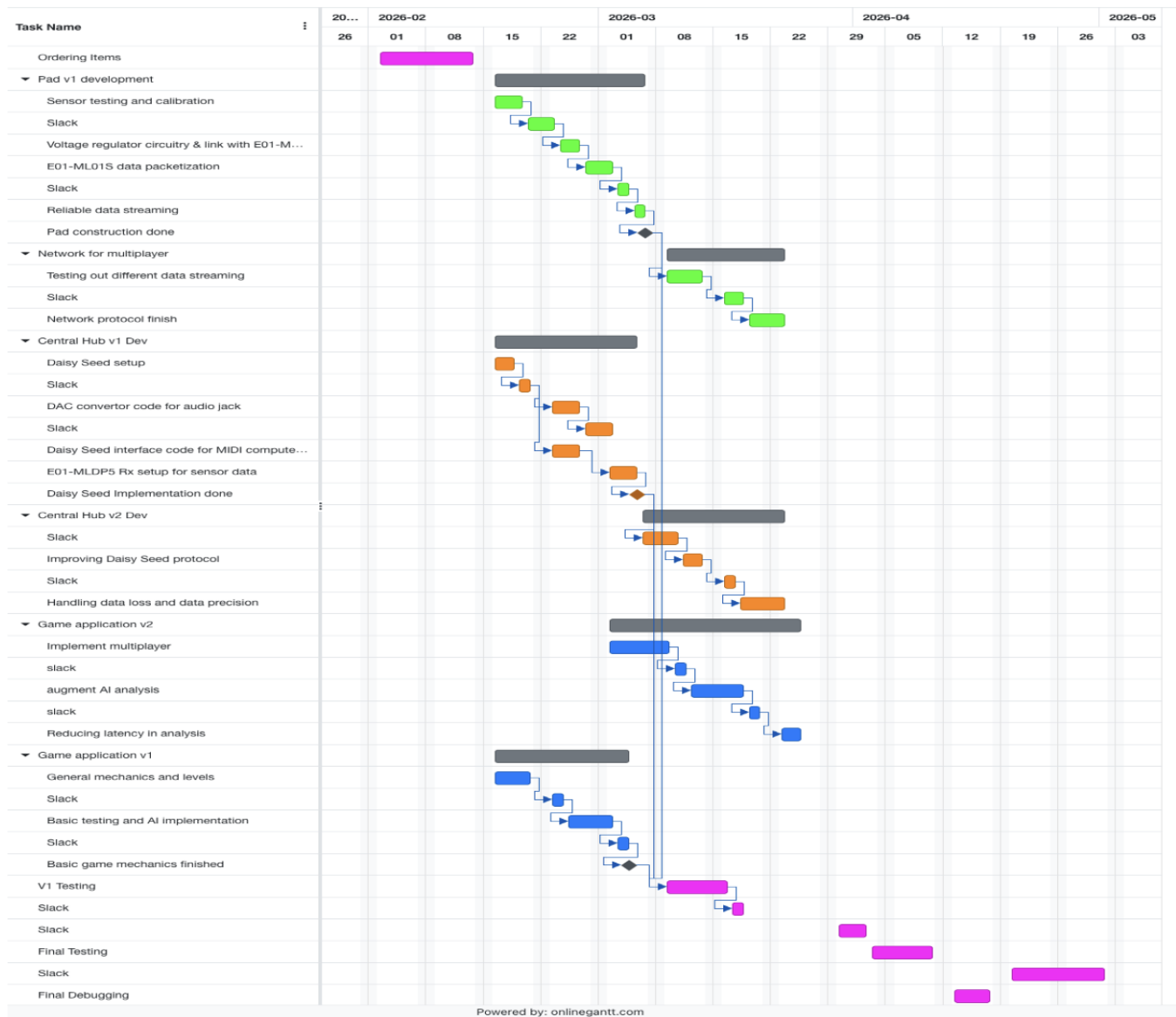


Figure 7: Gantt Chart