A Multi-Protocol Adaptive Microtuning Architecture: Bridging MTS, MPE, and MIDI 2.0 for Struck-String Just Intonation

Rui Su[1], **Joseph Timoney**[1], and **Damián Keller**[2,3]

[1]**Maynooth University**, Maynooth, Ireland
[2]**Federal University of Acre**, Brazil
[3]**Ubiquitous Music Group**

ABSTRACT

Current implementations of Real-Time Just Intonation (JI) are faced with a conundrum of precision versus compatibility. Tuning within MIDI is supported by the MIDI Tuning Standard (MTS) and pitch bend-based methods. MTS features good resolution but suffers from limited hardware support. Pitch bend-based methods provide ample compatibility but not enough granular resolution for microtonal tuning. This paper outlines an adaptive, multi-stage tuning architecture that extends the capabilities of the Web MIDI API and solves both of these problems. Within the ubiquitous music (Ubimus) research agenda for struck-string interactions, this proposed system utilises a hierarchical fallback strategy: (1) MTS for scale-based tuning, (2) an MPE mode optimised for locking the sensitivity of pitch bend to ± 2 semitones for high resolution tuning, and (3) a forward-compatible prototype engine that complies with the emerging MIDI 2.0 standard. Both MTS and MPE techniques provide resolution at least one order of magnitude finer than the deviation intervals present in equal temperament, enabling perceptually transparent adaptive tuning via native deployment in the browser.

1. INTRODUCTION

Ubiquitous Music (Ubimus) is a research field that encompasses the design of music-creation, artistic and educational tools, supported by diverse social and community-based methodological approaches [1]. Current Ubimus frameworks have opened new avenues for developing conceptual approaches and for improving existing legacy frameworks [2]. An important aim of Ubimus is "aesthetic pliability," or the ability of sociotechnical systems to accommodate culturally varied perspectives and preferences [1, 2]. Therefore, support for tuning scales beyond 12-TET is a requirement to achieve aesthetic pliability in Ubimus pitch-oriented creative endeavours.

Recent Ubimus research proposes struck-string interaction as a framework that fosters the use of generative techniques to explore the relationship between tuning and sonic qualities [4]. This approach focuses on piano-like timbre without incorporating a keyboard-centric perspective. Su *et al.* [5] implemented an initial toolkit to support Just Intonation (JI) in struck-string interaction, and demonstrated a prototype for converting MIDI data to JI. Furthermore, Su *et al.* [6] reviewed existing systems, identifying the need for a comprehensive browser-based architecture to support both real-time alternative tunings and access to external hardware support. This work suggests diverse areas of application, highlighting caveats in browser-based support and pointing to techniques with potential to bypass keyboard-centric interaction.

One area where struck-string interaction may foster aesthetic diversification is adaptive tuning. The assignment of specific frequencies to discrete pitches has implications beyond the equal temperament standard (12-TET) adopted by most keyboard-based musical devices. Historically, alternative tunings have driven the development of specialised microtonal keyboards [21]. Keyboard customisation demands considerable investment in hardware, rendering most solutions inaccessible to practitioners in peripheral or low-income countries. Hence, struck-string proposals have skewed the reconfiguration of hardware while fostering streamlined software-based alternatives. Our approach to adaptive tuning is consistent with this sustainability-oriented view. Rather than altering the material structure of the keyboard, we deal with the tuning information before it reaches the audio-synthesis stage.

The 12-TET model divides the octave into twelve equal parts; consequently, its intervals are approximate representations of pure integer frequency ratios. JI is based on exact simple-integer frequency ratios – which are usually perceived as being more consonant than equal-tempered intervals; however, there is no single fixed JI scale that can accommodate all key signatures [3]. Therefore, to deploy JI within tonal systems it is necessary to retune the scales when the harmonic context changes. We treat JI as one example of a technically demanding tuning system—since it needs the fine, context-dependent retuning that the default 12-TET MIDI pipeline cannot provide—and as a representative case of a large family of alternative tuning systems. In fact, JI has been at the basis of a significant body of 20th-century music—exemplified by James Tenney's composition "Arbor Vitae", which unfolds within a larger-than-usual (11-limit) just-intonation pitch

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.

space [22]. Equal temperaments with many divisions, notably 53-TET, place comparable demands on the transport layer and have recently been the focus of dedicated computational efforts [23,24]. An architecture that can carry JI to unmodified hardware from the browser may, in principle, support these alternative tunings as well.

Problems arise when implementing microtonal pitch variation in an ecosystem that utilises standard data-exchange protocols. Historically, overcoming the limitations of equally tempered tuning in MIDI has presented challenges. The MIDI protocol provides no straightforward method to alter its pitch-to-frequency mapping. Universal System Exclusive messages or Registered Parameter Numbers are usually employed in addition to the data sent through the note-on/note-off message stream. Therefore, developers tend to implement one of two alternatives: (1) use the officially supported MIDI Tuning Standard (MTS) while adopting System Exclusive (SysEx) messages for the precise control of pitch; as a downside, there are extremely few commercial grade products that support MTS natively [7]; or (2) use Polyphonic Pitch Bend Methods, which work on most hardware devices; these solutions limit the available resolution head room and require significant channel management [8]. When considering browser-based implementations, the reception of SysEx messages is blocked by default, thus specific authorization mechanisms need to be deployed.

Our adaptive tuning system *Instant Harmonies* features a unified architecture that addresses precision, compatibility, and security by means of a hierarchical fallback strategy. The system implements MTS for scale-based tuning when a SysEx permission is available, a precision-optimised MPE layer is proposed as a second-level fallback, and a proof-of-concept MIDI 2.0 engine provides support for forward compatibility. Through a browser-native approach utilising the Web MIDI API, it supports adaptive JI on any MIDI-compatible controller without requiring the installation of third-party software. Web infrastructure as a basis for accessible, cross-platform music applications tends to remove the barriers of installing, licensing and selecting an appropriate OS. This browser-based design was deliberate and motivated by the limitations of previously developed adaptive-tuning systems [1,2,25].

2. BACKGROUND

2.1 Related Work

In this section we review examples of software algorithms designed to operate as real-time adaptive tuning systems. Sethares [11] developed a system in Max that employs gradient descent to minimise sensory dissonance. The implementation was based upon his earlier formulation of adaptive tuning as the dynamic maximisation of consonance, computed directly from the spectrum of the sound [24]. In contrast to our tonality-based strategy, this method does not require explicit knowledge of key or tonal centre; because it works from timbre rather than notation, it can also be applied to inharmonic spectra. Code created Groven.Max [12], a Max-based adaptation of Eivind Groven's tuning system that maps keyboard input to a 36-tone just-intonation scale distributed over three separate MIDI channels. Hermode Tuning [3], featured both in Logic Pro and Cubase, adjusts the intervals toward just ratios while constraining the shift from the equal-tempered reference line to $\pm$20 cents ($\pm$30 cents for each pitch). Stange *et al.* [13] implemented adaptive tuning as a spring network of linear equations, generating MIDI output through fifteen separate channels for pitch bend. In a related vein, Trueman *et al.* [25] modelled tuning as a system of virtual springs in bitKlavier, where every sounding note is "tethered" to a reference scale so that its intonation can be pulled between fixed and microtuned behaviours; tellingly, the first implementation was a browser-based application in p5.js before the engine was implemented in C++. Volkov [14] developed Pivotuner, a VST3/AU plug-in with a configurable key-determiner algorithm and built-in MPE support.

Moussa [7] identified three common methods of MIDI microtuning protocols—MTS, manufacturer-specific SysEx, and polyphonic pitch bend—highlighting that although MTS provides a high resolution (0.0061 cents), its adoption rate has remained relatively low. Celik [8] created a real-time dynamic microtonal channel manager in Max/MSP. OddSound MTS-ESP [15] was proposed as a software-only solution featuring inter-process communication to share tuning information among multiple processes within a DAW session. As with the majority of DAW-based systems, it remained limited to desktop computers.

Recently, the development of the Web MIDI API [16] has been proposed as a general framework for browser-based MIDI interactions but, as Baratè and Ludovico [17] point out, it currently remains a W3C working draft supporting primarily Chromium-based browsers. Our architecture addresses these limitations by providing a hierarchical fallback mechanism between MTS and precision-optimised MPE, along with browser-native deployment, support for external hardware, and future-oriented MIDI 2.0 compatibility.

2.2 Just Intonation and Resolution Requirements

Just Intonation is a method of tuning musical intervals as a small number of frequency ratios to create a unique tonal quality [10, 18]. When the interval between two fundamentals gets closer to a simple ratio, the harmonic partials associated with each fundamental begin to align, and eventually the "beating" effect is eliminated or reduced. If the ratios used become more complex, the fusion of the two sounds creates a characteristic harshness, typical of tempered intervals. We have adopted a conservative design threshold of about 3 cents based upon the smallest musically meaningful interval deviation (approximately 2 cents for the ET fifth) and the smallest audibly distinguishable tuning increment [3]. A system's quantisation error should therefore be at least an order of magnitude smaller than the above value. All operating layers of our system provide much greater resolution than the minimum required.

The MTS Scale/Octave 2-Byte format provides a resolution of about 0.012 cents, based upon the 14-bit word

length and the total number of discrete steps available:

$$R_{\text{mts}} = \frac{200 \text{ cents}}{2^{14}} = \frac{200}{16384} \approx 0.0122 \text{ cents} \quad (1)$$

The Registered Parameter Number (RPN)-negotiated MPE mechanism we have developed restricts pitch-bend sensitivity to ± 2 semitones and provides a resolution of about 0.024 cents:

$$R_{\text{mpe}} = \frac{400 \text{ cents}}{2^{14}} = \frac{400}{16384} \approx 0.0244 \text{ cents} \quad (2)$$

In both cases, these values are considerably smaller than our established design threshold of 3 cents. The standard MPE specification allows for a pitch-bend range of ± 48 semitones on member channels. If we use the full 9600 cent range provided by the specification (which would yield a 14-bit resolution of $9600/16384 \approx 0.59$ cents per step), there would be very little margin left over before exceeding our established design threshold. On the other hand, our MPE layer uses only one quarter of the total pitch-bend range provided by the standard MPE specification; nonetheless, it offers a 24-fold improvement in resolution while remaining MPE-compliant through RPN negotiation on the member channels. The RPN negotiation mechanism, as defined within the MIDI 1.0 standard, allows a controller to specify the configuration of an instrument parameter (in this case, the pitch-bend sensitivity) prior to sending any performance data.

3. SYSTEM ARCHITECTURE

Instant Harmonies utilises the Web MIDI API [16] to allow browser access to external MIDI devices. Upon activation of the proposed system, the first step is to verify if the user has granted SysEx permissions; this is a required safety measure for the transmission of MTS messages. Once SysEx permissions have been granted, the system will assume that the connected external device supports MTS and attempt to transmit tuning messages. Since a single MIDI 1.0 connection carries no return channel and the standard defines no capability-inquiry handshake, there is no established method to determine whether an external device supports MTS [19]; therefore, SysEx messages are classified as "fire and forget". If a user finds that the MTS tuning mode does not yield the expected output, she may opt to manually select the MPE mode via the user interface. This option will then be saved as default for future usage.

The proposed permission-based architecture addresses both the trade-offs of compatibility vs. precision and provides a layer of security for the browser environment. Given that a browser-based connection could potentially be used to transmit any type of data to the local device, including but not limited to firmware updates or factory resets, browsers prohibit by default the transmission of SysEx messages. The MPE mode serves as both a compatible and secure bridge by utilising only Standard Channel Voice Messages that do not require special permissions.

3.1 Primary Method: MIDI Tuning Standard (MTS)

In tonality-based adaptive JI, the system determines the currently occurring tonal centre and retunes the 12 pitch classes of the chromatic scale to the pure JI ratios relative to this tonal centre [13]. Since JI is defined as being relative to a particular tonal centre, the tuning must be recalculated whenever the tonal centre changes. A tonality-based approach therefore differs from the methods that address the perceived dissonance of a set of simultaneous pitches by changing the pitch of each sounding note independently [11, 26].

Tonality-based techniques will usually tune the 12 pitches simultaneously, as this is a computationally efficient way to implement a scale. An efficient way to support this type of adaptive tuning is the MTS Scale/Octave Tuning 2-Byte message—a Universal Real-Time System Exclusive message (Sub-ID#1 = 0x08, 'MIDI Tuning Standard'; Sub-ID#2 = 0x09). [1] The Scale/Octave Tuning 2-Byte message retunes all 12 pitch classes (i.e., C through B) for all the selected MIDI channel(s), making this method suitable for adaptive tuning applications where the fundamental frequencies are aligned with the key changes. For each pitch class, the system computes the cent offset between the predefined 5-limit JI ratio and its equal-tempered counterpart, then encodes this offset as a 14-bit value in the MTS Scale/Octave 2-Byte message. A 5-limit ratio is one whose numerator and denominator factorise into primes no greater than 5 (i.e. 2, 3 and 5)—a basis of common-practice consonances such as the just major third, 5/4.

3.2 Fallback Method: Precision-Optimised MPE

When MTS is unavailable, MPE is used. We changed the pitch-bend sensitivity from the standard range (± 12 semitones) to ± 2 semitones with registered parameter number (RPN) 0, thus limiting the range of expression but improving tuning accuracy to 0.0244 cents per step. As per the MPE specification, the Master Channel for the zone-wide messages is Channel 1, while the Member Channels (2 through 16) carry the individual notes. (The MPE specification also allows for dual zones, with an upper zone supported by Channel 16; however, for simplicity we use a single lower zone.) RPN 0 is sent to all the Member Channels with the data-entry MSB at 2 semitones to fix the pitch-bend sensitivity.

As MPE assigns a specific member channel to every sounding MIDI note, there is a polyphony limit imposed by the 15 channels available, which demands an allocation strategy. The method implemented keeps a pool of free member channels, alongside a queue ordered by member-channel usage that tracks which note currently occupies each member channel. If a new note-message arrives and a member channel is available in the free pool, the next available member channel is used; if all member channels are occupied, the member channel associated with the least recently activated note is evicted. This eviction policy is sim-

[1] MIDI Manufacturers Association & Association of Musical Electronics Industry (2020), *MIDI Tuning Updated Specification*, documents CA-020 and CA-021, available from `https://midi.org/midi-tuning-updated-specification`.

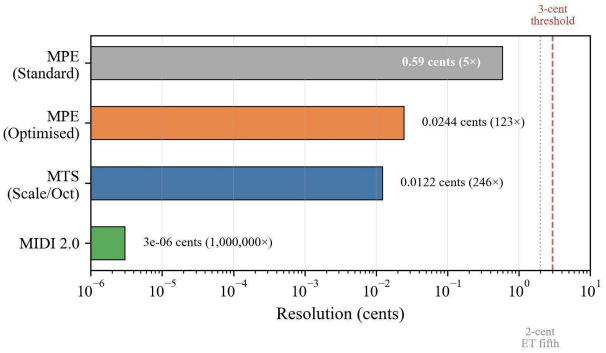


Figure 1. Logarithmic comparison of tuning resolution across MIDI protocols. The 3-cent perceptual threshold (dashed red line) and the 2-cent ET fifth deviation (dotted grey line) are shown for reference. All proposed modes substantially exceed the threshold.

ilar to cache-replacement techniques, and has been applied to the multi-channel pitch-bend allocation problem, as first described by Moussa [7]. The Least Recently Used (LRU) strategy ensures that the evicted note is the one whose pitch bend is least likely to be perceptually relevant.

3.3 Proof-of-Concept: MIDI 2.0 Integration

We have developed a MIDI 2.0 Tuning Engine which generates Universal MIDI Packet (UMP) Structures. Since the Web MIDI API has no support for sending UMP messages [17], this module is currently running in File Export Mode. The Tuning Engine utilises the MIDI 2.0 per-note pitch control carried by Registered Per-Note Controller index 3, whose 32-bit value uses a '7.25' fixed-point format: the top 7 bits give the note in semitones (0–127) and the remaining 25 bits furnish the fractional semitone (a resolution of 1/33,554,432 of a semitone). This yields a theoretical resolution of 0.000003 cents: roughly one million times finer than the perceptual threshold.

4. COMPARATIVE EVALUATION

The basis of adaptive JI is psychoacoustic research. Our sensitivity to beats, or periodic amplitude changes resulting from two close partials. This is what generally determines how we perceive mistuned intervals [9]. The fifth of equal temperament deviates from the pure interval by roughly 2 cents, while the major third deviates by about 14 cents [9, 10]. According to Mohrlok [3], the audible limit for retuning steps is about 3–4 cents. Therefore, we set our design threshold at approximately 3 cents.

To validate our design, we compared the effective resolution of each tuning protocol to the human beat-detection threshold. We applied a conservative 3-cent margin of error, as outlined by Mohrlok [3] and Vos [9]. The Perceptual Margin column in Table 1 provides the resolution as a multiple of the 3-cent margin, i.e., a multiple of 123× indicates that the resolution is 123 times finer than the margin of error. Figure 1 displays the resolution of the two tuning methods on a logarithmic scale.

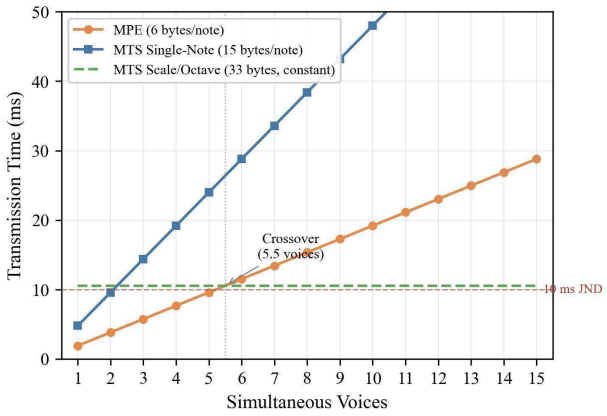


Figure 2. Transmission time as a function of simultaneous voices. MPE (6 bytes/note) has lower overhead for sparse textures, while MTS Scale/Octave (33 bytes, constant) becomes more efficient beyond 5.5 voices. MTS Single-Note (15 bytes/note) scales linearly. Per-note figures include the 3-byte Note-On, whereas the MTS Scale/Octave curve counts only its one-off 33-byte tuning message. The 10 ms. percussive JND is shown for reference [20].

The transmission structure and timing for each tuning protocol is provided in Table 2. Both MTS and MPE have resolutions exceeding the human beat-detection threshold. However, there is a significant cost associated with bandwidth and latency. Classic serial MIDI 1.0[2] transmits at 31,250 baud, and each byte is framed by one start bit and one stop bit around its eight data bits (10 bits per byte on the wire). The effective throughput is therefore $31{,}250 \div 10 = 3{,}125$ bytes per second, i.e. 0.32 ms. per byte, which is the basis for the transmission times in Table 2.[3]

There are clear architectural trade-offs between the two tuning protocols. First, MTS Scale/Octave has constant time-scale retuning (a single 33-byte message retunes all 12 pitch-classes regardless of how many events are being played, making it ideal for use in JI applications that rely on tonality-based adaptive methods). Second, MPE's per-note granularity features lower latency in sparse textures and the ability to tune a single note without affecting other pitches. However, as illustrated in Figure 2, the transmission time increases linearly with polyphony.

5. DISCUSSION

Both tuning layers achieve a resolution finer than perceptual thresholds. In this section we examine two architectural aspects of *Instant Harmonies*' design and point to three limitations to be addressed in future implementations.

[2] MIDI Manufacturers Association & Association of Musical Electronics Industry (1996), *The Complete MIDI 1.0 Detailed Specification*, Document Version 96.1, available from `https://midi.org`.

[3] These times use the 31.25 kbaud serial rate as a transport-independent worst case; USB-MIDI and Web MIDI transports are faster, so the figures are conservative upper bounds.

Method	Bit Depth	Range	Resolution (cents)	Perceptual Margin
MPE (Standard)	14 bit	$\pm$48 semitones	$\sim$0.59	$\sim$5$\times$
MPE (Optimised)	14 bit	$\pm$2 semitones	0.0244	$\sim$123$\times$
MTS (Scale/Oct)	14 bit	$\pm$1 semitone	0.0122	$\sim$246$\times$
MIDI 2.0	32 bit	Per-note absolute	0.000003	$\sim$1,000,000$\times$

Table 1. Resolution analysis of the proposed tuning methods.

Message Type	Structure	Bytes	Time (ms)	Source
MTS Scale/Octave 2-Byte	`F0 7F id 08 09 ff gg hh [ss tt]×12 F7`	33	10.56	CA-021[1]
MTS Single-Note	`F0 7F id 08 02 tt ll [kk xx yy zz] F7`	12	3.84	CA-020[1]
MPE Pitch Bend	`En ll mm`	3	0.96	MIDI 1.0[2]
Note On/Off	`9n/8n kk vv`	3	0.96	MIDI 1.0[2]

Table 2. MIDI message byte counts and transmission times. Superscripts refer to the specification footnotes: [1] MIDI Tuning Updated Specification (CA-020, CA-021); [2] The Complete MIDI 1.0 Detailed Specification.

Hierarchical fallback

Moussa [7] established the limits of single-port channel assignment and introduced an overall routing method that makes use of three different physical MIDI output channels: one for instruments that can generate polyphonic aftertouch, another for MTS capable instruments and a third one for instruments that do not have either of these features. Similar to Moussa's system, *Instant Harmonies* allows for graceful degradation through software-level protocol-switching in a single browser session using SysEx permissions as the fallback criteria. This mechanism adds security to desktop-based systems, given that browsers block SysEx by default. Therefore, the IH's precision-optimised MPE layer functions simultaneously as a compatibility bridge and a security-compliance mechanism. As illustrated previously, both techniques surpass the conservative 3-cent threshold by over two orders of magnitude. Therefore, this software-fallback strategy may supplement FPGA hardware-based flexibility and reconfigurability, as described by Keller *et al.* in their Ubimus plugging framework [4].

Browser-Native Deployment

While several systems support sending data to external MIDI hardware [7, 8, 12, 13], they do so through commercial applications or custom-made hardware. The spring-tuning engine in bitKlavier was initially deployed as a web application [25], though this implementation was replaced by a desktop-based tool. *Instant Harmonies* routes protocol-level tuning messages—MTS, precision-optimised MPE, and MIDI 2.0—to external MIDI hardware from a zero-installation environment, using the Web MIDI API. This allows a browser-based application to interface with complex MIDI configurations, potentially comprising several devices [17]. As a result, *Instant Harmonies* zero-installation model aligns with the Ubimus goal of expanding opportunities for creative expression and engagement by casual stakeholders [1].

Limitations

Three technical limitations need to be acknowledged. Firstly, MIDI 1.0 defines no capability-inquiry handshake, so MTS support cannot be identified automatically [19]. Secondly, our MPE layer relies on synthesisers that correctly identify and respond to RPN 0 messages; if the synthesis engine ignores non-default sensitivity values, the resolution decreases from 0.024 to about 0.59 cents. Thirdly, there is a potential for timing drifts when sending data over a USB-MIDI interface. The Web MIDI API is currently at Working Draft status, with support limited to Chromium-based browsers. Since version 108, the Firefox version on a first run prompts the user to install an add-on. Neither implementation supports real-time MIDI 2.0 transport. Furthermore, Apple Safari/WebKit does not implement Web MIDI at all, a decision Apple attributes to fingerprinting concerns [26]. For non-expert users, this dependence on MIDI hardware implies some familiarity with MIDI configuration that can present a barrier [17]. The extension of support to a second, independent engine suggests a trajectory of broadening adoption that may, in time, encompass most real-time MIDI 2.0-compatible platforms.

Future work on *Instant Harmonies* will focus on addressing the following areas: (1) provide support for real-time MIDI 2.0 UMP transmissions for an enhanced Web MIDI API, thereby eliminating the 15 voice polyphony limit and the overhead associated with allocating MPE channels; (2) incorporate symbolic representations (such as MusicXML) to avoid the latency associated with key-identification algorithms, which demand a temporal window on the order of one beat or more—before a stable estimate is available and a scale retuning message can be issued; and (3) employ MIDI 2.0 Capability Inquiry (MIDI-CI) to automatically negotiate the setup, yielding an entirely automated device-discovery process.

6. CONCLUSION

In this paper, we have documented a multi-protocol MIDI-based architecture that addresses the longstanding trade-off between tuning accuracy and hardware compatibility in adaptive JI: *Instant Harmonies*. IH employs a hierarchical fallback strategy. MTS Scale/Octave Tuning is utilised when SysEx permissions are available, and a precision-

optimised MPE mode—which locks pitch-bend sensitivity to ± 2 semitones by RPN 0 negotiations—is used as a fallback if MTS is either unavailable or ineffective. A proof-of-concept MIDI 2.0 engine ensures forward compatibility with future browser-based standards.

Both operational mechanisms surpass perceptual requirements: MTS provides an approximate 0.012 cent resolution (ca. 246 times a conservative 3-cent threshold); the precision-optimised MPE layer furnishes a 0.024-cent resolution (approximately 123 times the threshold). The proposed contribution is architectural rather than algorithmic: to the best of our knowledge, this is the first adaptive tuning system to furnish a hierarchical fallback mechanism between standard MIDI-tuning protocols—as distinct from other browser-based systems. Furthermore, IH addresses direct external hardware output from a zero-installation web-browser environment. As part of the Ubimus research agenda on struck-string interaction [4–6], our system enables adaptive JI without specialised hardware or software requirements. Audio examples (JI vs ET, adaptive retuning, and songs) and a glossary of MIDI tuning terms are available at `https://doi.org/10.5281/zenodo.21251590`.

Acknowledgements

We wish to extend our gratitude to the anonymous reviewers for allowing us to improve this paper with their comments, specifically the observation that MIDI uses RPN and SysEx universal methods to replace its default equal-temperament pitch mapping. Partial funding to DK was provided by the CNPq research grant [402250/2024-9]. This work is part of the "Instant Harmonies" project, a doctoral research project at Maynooth University that aims to make Just Intonation tools easily accessible and adaptable. The goal of the project is to make microtonal music-making more accessible by creating a free, browser-based platform that lets both professional musicians and amateurs experience Just Intonation on their existing MIDI devices without specialised hardware or software.

7. REFERENCES

[1] D. Keller, V. Lazzarini, and M. S. Pimenta (eds.), "Ubiquitous Music." Berlin and Heidelberg: Springer International Publishing, 2014, doi: 10.1007/978-3-319-11152-0.

[2] V. Lazzarini, D. Keller, N. Otero, and L. Turchet (eds.), "Ubiquitous Music Ecologies." London: Taylor & Francis (Routledge), 2021.

[3] W. Mohrlok, "The Hermode Tuning system," 2003. [Online]. Available: `https://sethares.engr.wisc.edu/paperspdf/hermode.pdf`

[4] D. Keller, A. Jagwani, and V. Lazzarini, "The ubimus plugging framework: Deploying FPGA-based prototypes for ubiquitous music hardware design," *Computers*, vol. 14, no. 4, p. 155, 2025, doi: 10.3390/computers14040155.

[5] R. Su, J. Timoney, and D. Keller, "Just intonation and inharmonicity in struck-string interaction," in *Proc. Ubiquitous Music Symp. (UbiMus 2024)*, Macau, China, 2024.

[6] R. Su, J. Timoney, and D. Keller, "MIDI adaptive tuning strategies by means of AI-based struck-string interaction in ubimus," in *Proc. Ubiquitous Music Symp. (UbiMus 2025)*, 2025.

[7] A. S. Moussa, "Perception-based microtuning over MIDI networks," *IEEE Multimedia*, vol. 13, no. 1, pp. 56–64, Jan.–Mar. 2006, doi: 10.1109/MMUL.2006.18.

[8] S. Celik, "Micro-MIDI: A real time, dynamic microtonal MIDI application," *Int. J. Eng. Sci. Invention*, vol. 5, no. 9, pp. 1–7, 2016.

[9] J. Vos, "The perception of pure and mistuned musical fifths and major thirds: Thresholds for discrimination, beats, and identification," *Perception & Psychophysics*, vol. 32, no. 4, pp. 297–313, 1982, doi: 10.3758/BF03206236.

[10] S. Schwär, S. Rosenzweig, and M. Müller, "A differentiable cost measure for intonation processing in polyphonic music," in *Proc. 22nd ISMIR Conf.*, 2021, pp. 626–633.

[11] W. A. Sethares, "Real-time adaptive tunings using Max," *J. New Music Res.*, vol. 31, no. 4, pp. 347–355, 2002, doi: 10.1076/jnmr.31.4.347.14163.

[12] D. L. Code, "Groven.Max: An adaptive tuning system for MIDI pianos," *Comput. Music J.*, vol. 26, no. 2, pp. 50–61, 2002, doi: 10.1162/014892602760137185.

[13] K. Stange, C. Wick, and H. Hinrichsen, "Playing music in just intonation: A dynamically adaptive tuning scheme," *Comput. Music J.*, vol. 42, no. 3, pp. 47–62, 2018, doi: 10.1162/comj_a_00478.

[14] D. Volkov, "Pivotuner: Automatic real-time pure intonation and microtonal modulation," in *Proc. Int. Comput. Music Conf. (ICMC)*, Shenzhen, China, 2023.

[15] O. Cash, D. Gamble, and D. Hancock, "MTS-ESP Mini user guide," ver. 1.08, OddSound, 2021.

[16] C. Wilson and J. Kalliokoski, "Web MIDI API," W3C Working Draft, 2015. [Online]. Available: `https://www.w3.org/TR/webmidi/`

[17] A. Baratè and L. A. Ludovico, "Web MIDI API: State of the art and future perspectives," *J. Audio Eng. Soc.*, vol. 70, no. 11, pp. 918–925, 2022, doi: 10.17743/jaes.2022.0028.

[18] T. Nicholson and M. Sabat, "Fundamental principles of just intonation and microtonal composition," Universität der Künste Berlin, Studio für Intonationsforschung und mikrotonale Komposition, 2018.

[19] S. Luke and L. A. Ludovico, "MIDI 1.5, so to speak," in *Proc. Sound Music Comput. Conf. (SMC 2024)*, 2024, pp. 373–382.

[20] R. Jack, A. Mehrabi, T. Stockman, and A. McPherson, "Action-sound latency and the perceived quality of digital musical instruments: Comparing professional percussionists and amateur musicians," *Music Perception*, vol. 36, no. 1, pp. 109–128, 2018, doi: 10.1525/mp.2018.36.1.109.

[21] D. Keislar, "History and principles of microtonal keyboards," *Comput. Music J.*, vol. 11, no. 1, pp. 18–28, 1987, doi: 10.2307/3680175.

[22] M. Winter, "On James Tenney's Arbor Vitae for string quartet," *Contemp. Music Rev.*, vol. 27, no. 1, pp. 131–150, 2008, doi: 10.1080/07494460701671566.

[23] D. Dalmazzo, K. Déguernel, and B. Sturm, "A computer application to explore 53-tone equal temperament harmonies through modal interchange," in *Proc. Int. Conf. New Interfaces for Musical Expression (NIME)*, 2025, doi: 10.5281/zenodo.15698843.

[24] W. A. Sethares, "Adaptive tunings for musical scales," *J. Acoust. Soc. Am.*, vol. 96, no. 1, pp. 10–18, 1994, doi: 10.1121/1.410471.

[25] D. Trueman, A. Bhatia, M. Mulshine, and T. Trevisan, "Tuning playfully: Composed and adaptive tunings in bitKlavier," *Comput. Music J.*, vol. 43, no. 2–3, pp. 67–88, 2019, doi: 10.1162/comj_a_00519.

[26] "Web MIDI API," Can I use…, 2026. [Online]. Available: `https://caniuse.com/midi` (accessed Jun. 18, 2026).