A Prototype of a MIDI 2.0 Controller for Inclusive Music Interaction

Vanessa Faschi[1][0000−0002−9815−1127], Guido Orsomaria Basi[1], Federico Avanzini[1][0000−0002−1257−5878], Luca Andrea Ludovico[1][0000−0002−8251−2231], and Emanuele Parravicini[2]

[1] Laboratory of Music Informatics (LIM), Dept. of Computer Science, University of Milan, via G. Celoria 18, Milan, Italy
[2] Audio Modeling, Via Bernardino Luini 65, Meda, Italy

Abstract. This paper presents a software/hardware solution designed to facilitate music performance that is responsive to the various abilities and contexts of users, thus narrowing the gap between disability and technological access to sonic expression. We examine how the MIDI 2.0 protocol, particularly its profile configuration and property exchange capabilities, can be leveraged to create adaptive musical interfaces that dynamically respond to the spatial and situated conditions of performance. In this approach, the configuration of digital instruments becomes sensitive not only to individual users' motor or cognitive abilities but also to the specific environments and cultural practices in which music-making occurs. We describe the development of the Inclusive MIDI Controller, a digital musical instrument designed to foster inclusive sonic practices and to expand the spatial and experiential possibilities of music performance for a wider range of practitioners. Finally, we detail the design and implementation of a MIDI 2.0 Device Manager developed using the Core MIDI workbench and the JUCE framework. This system was evaluated in conjunction with the ProtoZOA board, an open-source, flexible prototyping platform for MIDI 2.0, demonstrating its capacity to automatically configure instruments and interfaces.

Keywords: MIDI 2.0 · Controller · Inclusivity · Accessibility · Accessible Devices · Inclusion.

1 Introduction

Music is a universal form of expression that fosters communication, creativity, and emotional connection across varied cultural spaces and social environments. However, conventional modes of musical interaction typically presume a standard

Proc. of the 17th Int. Symposium on Computer Music Multidisciplinary Research, London, United Kingdom, 2025

level of physical, cognitive, or sensory capability, which can marginalize individuals with disabilities or those without access to traditional instruments. These limitations underscore the importance of developing innovative approaches that support inclusive and context-aware musical practices, enabling participation in diverse sonic settings and situated performance spaces.

The field of inclusive design in music technology has undergone significant advancements in recent years. Scholars and practitioners have explored the development of systems that democratize music making, focusing on accessibility and usability [23, 9]. Such efforts often involve the integration of assistive technologies, multimodal interfaces, and adaptive software to lower barriers and ensure the participation of individuals with diverse abilities.

A promising approach is the creation of hardware/software ecosystems that simplify musical control through intuitive interaction paradigms. Bridging the gap between human intention and machine execution, empower musicians to push the boundaries of sonic exploration and unlock new avenues of creative expression, also for impaired users. By abstracting technical complexity and focusing on user-centered design, these systems open new possibilities for music education, performance, and rehabilitation [13].

The MIDI protocol has been extensively used in inclusive music applications, but primarily as a communication standard that enables interoperability between hardware and software devices. Indeed, MIDI is only used as a transport layer: it facilitates message exchange between standard devices capable of MIDI communication, but it does not directly contribute to the inclusivity of the system itself. The recent release of MIDI 2.0 specifications introduced numerous advancements, especially in the context of inclusive music applications. MIDI 2.0 builds on the foundation of its predecessor by introducing a more sophisticated protocol that enhances interoperability, precision, and ease of use. While the full range of benefits will be clarified in subsequent sections, two key features stand out as particularly advantageous for inclusive applications: profiles and auto-configuration.

The design and implementation of the *Inclusive MIDI Controller* [8] fits within this framework. This software application addresses young people and adults with physical and cognitive disabilities. The proposed system leverages simple controls to facilitate rich musical interaction, drawing inspiration from human-computer interaction (HCI) principles and inclusive design methodologies. A detailed discussion of the prototype is presented in [8].

The paper presents a specific component of the overall architecture, namely the *MIDI 2.0 Device Manager* (*M2DM*). The *M2DM* is programmed in C++ and its implementation leverages the Core MIDI workbench and the JUCE framework to interface with the *ProtoZOA* by AmeNote, a prototyping board for the developer to experiment with MIDI 2.0. Our analysis will be conducted from both a theoretical and a practical perspective. The new MIDI 2.0 protocol is examined to highlight its innovations and strengths, followed by a description of the project and its implementation. The code of the device manager developed in this context is publicly available on GitHub.

The remainder of the paper is structured as follows: Sec. 2 presents an analysis of the pertaining scientific literature; Sec. 3 proposes the use of MIDI 2.0 in inclusive scenarios; Sec. 4 introduces the *Inclusive MIDI Controller*; Sec. 5 provides details on the MIDI 2.0 Device Manager and describes the tests performed to verify the correct functioning of the devices and applications; finally, Sec. 6 presents the conclusions and outlines potential future developments.

2 State of the Art

Inclusive music technologies aim to enable individuals of all abilities to participate in music-making by overcoming physical, cognitive, or sensory barriers. Traditional instruments often assume normative motor and sensory functions, making them inaccessible for many. This challenge has led to the emergence of Accessible Digital Musical Instruments (ADMIs), custom-designed systems that leverage adaptive hardware and multimodal interfaces to broaden participation in musical expression [9, 16].

Research in ADMIs includes a diverse range of interaction paradigms: gaze-based instruments such as EyeHarp [24], gesture-based systems using Leap Motion or inertial sensors [2, 5, 11], and touchless controllers that utilize proximity sensing or computer vision [10]. Gaze-controlled systems have shown particular promise for users with limited motor ability, enabling control over pitch, timing, and dynamics through eye movements [7, 6, 20]. Gesture-based systems similarly offer expressive input while removing the need for precise contact or fine motor control.

Beyond input modalities, the field has embraced user-centered and participatory design approaches, recognizing that accessibility must be tailored and co-developed with end users [3, 12]. These approaches ensure that musical tools are not only technically accessible but also creatively empowering and socially meaningful.

MIDI (Musical Instrument Digital Interface) has historically played a critical role in inclusive music-making. Its original specification, MIDI 1.0, established in 1983, provided a standardized protocol for musical communication across devices [14, 18]. This allowed alternative controllers, such as sip-and-puff devices, custom button grids, or eye trackers, to interface with commercial DAWs and sound modules [15, 21, 22]. However, MIDI 1.0 exhibited well-documented limitations, including low resolution (7-bit CC messages), unidirectional communication, and the need for manual mapping, all of which pose significant barriers to users with impairments [19, 25].

As already mentioned in Sect. 1, the release of MIDI 2.0 in 2020 marked a transformative shift. The new protocol retains backward compatibility but introduces critical enhancements, including 32-bit parameter resolution, bidirectional communication, profile configuration, and property exchange via the MIDI-CI (Capability Inquiry) mechanism [17]. These innovations support auto-configuration, allowing devices to announce their capabilities and dynamically

adapt to host software, a crucial feature for assistive setups, where manual configuration may be impractical.

Recent literature has begun to explore MIDI 2.0's implications for inclusivity. For instance, MASSIG (Music Accessibility Standard Special Interest Group), coordinated by the MIDI Association, brings together stakeholders from industry and disability advocacy to ensure that MIDI 2.0 remains accessibility-conscious [4]. Projects such as Sound Without Sight[3] have further emphasized the need for adaptive interfaces that support blind and partially sighted musicians. Despite its promise, real-world implementations of MIDI 2.0 in accessible instruments remain limited. Some commercial DAWs have begun to experiment with auto-configuration features, but very few systems actively integrate MIDI 2.0 property exchange for the purpose of inclusive interaction [15]. Some recent research efforts have begun to bridge this gap by demonstrating real-time adaptation to individual user needs, but they often remain experimental or hardware-specific [5, 4].

Our work addresses this gap by presenting a MIDI 2.0-enabled ecosystem comprising a flexible hardware controller and a device manager that applies auto-configuration and dynamic property exchange to inclusive music performance. By supporting real-time adaptation, automatic mapping, and intuitive control for users with diverse abilities, this system contributes a concrete and scalable example to the state of the art in accessible digital music systems.

3 MIDI 2.0 for Inclusive Music Interaction

The most straightforward use of property exchange and profiles for accessibility is the automatic adaptation of user interfaces, a new feature of MIDI 2.0. The official specifications describe a minimal implementation with a computer running a DAW and a MIDI-compatible pedal [17] that, after being connected to the system, is automatically detected and impacts the graphical user interface of the DAW. This example can be adapted to an inclusive scenario, where MIDI applications and devices query each other to find out what inclusive parameters are supported and configurable [4].

When a musician connects an accessible MIDI controller (e.g., a single-button device, a specific set of buttons, a breath controller, or an eye tracker), the software automatically recognizes its features and adjusts accordingly. This function removes the barrier of having to set up MIDI mappings manually, which can be challenging for users with motor or cognitive impairments. If a musician uses a simplified input device, the DAW can provide an alternative interface, e.g., with larger buttons, visual cues, or spoken feedback. For blind musicians, the DAW might activate screen reader support or haptic feedback based on the detected controller. The cognitive load can be further reduced by suggesting mappings based on common accessibility presets.

The following case illustrates the exchange of MIDI-CI messages between a computer running a DAW, playing the role of transaction initiator, and an

[3] https://soundwithoutsight.org/

accessible controller with a 2-axis joystick and a push button, with the role of transaction responder. Provided that both devices are capable of MIDI 2.0 communication, the following exchange of messages occurs:

1. A complete MIDI-CI *Discovery Transaction*, in order to exchange key information between the DAW and the controller (e.g., their *MIDI Universal ID* (MUIDs));
2. A *Property Exchange Capabilities* inquiry sent by the software to request details of *Property Exchange Support* from the controller;
3. A *Reply to Property Exchange Capabilities* message returned by the controller;
4. A *Get Property Data* inquiry sent by the software to request the resource list from the controller;
5. A *Reply to Get Property Data* return message by the controller containing the available resources, i.e., information on the device and the list of all its components, namely two axes and a button;
6. A new *Get Property Data* inquiry sent by the software to obtain some general information about the controller, such as the manufacturer id and name, the family id and name, the model id and name, and the version id and name;
7. A *Reply to Get Property Data* return message carrying the required device information
8. Another *Get Property Data* inquiry sent by the software focusing on the list of controls in the controller;
9. A *Reply to Get Property Data* return message describing single controls (i.e., *X axis*, *Y axis*, and *Button*) and their current values.

MIDI-CI implements the subscribe function, a mechanism of communication between devices that enables them to dynamically discover and negotiate their capabilities. In the scenario described above, the DAW can subscribe to the specific parameters of the controller and receive updates when they change. In this way, the DAW can read these parameters and their ranges, either representing them on the screen or by speech synthesis. The latter behavior, for instance, makes parameter values accessible to people with visual impairments.

Profile Exchange can also facilitate customized interaction models, such as a single button triggering chords or arpeggios instead of single notes, a joystick translating movement into pitch bend or volume control, or a pressure-sensitive switch enabling expressive dynamics for musicians with limited dexterity.

Considering a system with multiple controllers, musicians who use many adaptive devices simultaneously (e.g., foot pedals and sip-and-puff tools) can benefit from Profile Exchange, ensuring that all devices work together seamlessly. The software can even suggest layered control strategies, such as using one device for note input and another for modulation.

4 Inclusive MIDI Controller

The already mentioned *Inclusive MIDI Controller* [8] is a versatile software application developed to provide accessible MIDI control across multiple plat-

Fig. 1. The main interface of the *Inclusive MIDI Controller*.

forms. The concept originated from Manuele Maestri, founder of the musical project Musica Senza Confini,[4] during his work on inclusive ensemble music performance. In seeking solutions for a musician with highly limited mobility—restricted to finger, toe, and eye movements, Maestri devised a tailored setup using the Logic Pro DAW, the Max environment, and manual configuration of each track to support non-conventional interaction.

This configuration enabled the user to perform live within ensembles alongside musicians and singers. The system leveraged an eye tracker for selecting colored buttons displayed on screen and micromuscular sensors to trigger MIDI events. These events included notes, chords, short sequences, or other sounds.

The next phase aimed to evolve this bespoke solution into a customizable, user-adaptive software offering a range of playable pieces. This marked the beginning of the *Inclusive MIDI Controller* project. The system's architecture includes the main software that runs on a central computer, receives input from various control devices, and sends MIDI messages to a separate system running a DAW. A multitrack project loaded into the DAW allows different tracks to be controlled by different input devices, enabling collaborative musical performance.

The software is developed in C++ using the JUCE framework and is compatible with both macOS and Windows. It features a standalone user interface that provides two configurations, one that can be used with the eye tracker and also includes a rest zone to allow the gaze to be able to fixate on an area that cannot be activated, and another one that involves the use via a touch device. Both these interfaces provide configurable colored pads, ranging from 2 to 12 per screen, that can be activated through standard or assistive input devices, enhancing accessibility for diverse user needs.

[4] https://www.musicasenzaconfini.com/

Through the Settings menu, each colored pad can be configured with the following parameters:

- The associated event: MIDI Note, MIDI Control Change, Chord, MIDI Sequence, or page navigation;
- Trigger behavior: Trigger (sound activated on mouse hover), Hold (sound activated on hover, deactivated when the cursor leaves), Latch (sound toggled on hover, deactivated on the next hover);
- the MIDI channel;
- the MIDI note;
- the velocity and release-velocity values;
- the delay in milliseconds for Note-On and Note-Off messages;
- the pad color;
- Two primary keyboard shortcuts;
- An additional shortcut for repeating the last triggered note—especially useful for eye-tracker users.

The Aux Pads menu offers additional configurable pads, visually identical to the main ones but accessible only via keyboard shortcuts and not displayed on the main interface. Users can save, load, reset, and create new configurations. Once activated, these pads generate MIDI messages sent to connected MIDI-compatible devices (e.g., DAWs or synthesizers). The activation method depends on the user's hardware or assistive tools (Human Interface Devices, HIDs). A core strength of *Inclusive MIDI Controller* is its deep MIDI integration. As MIDI is a universal communication protocol, messages from the software can be routed to DAWs, plugin hosts, music software, or external MIDI hardware–allowing for flexible and compatible workflows.

MIDI connectivity is handled differently across platforms:

- On macOS, the software utilizes CoreMIDI's virtual ports for direct communication with MIDI-enabled software and hardware;
- On Windows, third-party virtual MIDI drivers such as LoopBe or LoopMIDI may be required to establish connectivity.

Thanks to its cross-platform support, accessible interface, and robust MIDI capabilities, *Inclusive MIDI Controller* empowers users to creatively trigger samples, play virtual instruments, or manipulate sound effects with enhanced flexibility and control.

5 MIDI 2.0 Device Manager (M2DM)

In this section, we focus on the implementation of *MIDI 2.0 Device Manager* (*M2DM*), capable of connecting different MIDI devices and elaborating incoming and outgoing data. The code is made publicly available on GitHub.[5]

[5] https://github.com/LIMUNIMI/MIDI2DevManager

In order to develop the *M2DM*, we need a set of functions to manage the MIDI ecosystem. In a typical scenario, the applications themselves take care of managing the configuration of MIDI devices autonomously, as it happens in JUCE [6] and AudioKitEX2.[7] At present, JUCE, the framework used to develop the *M2DM*, does not offer classes for cross-platform support of MIDI 2.0 nor does it support communication through Universal MIDI Packet (UMP). Official Windows MIDI 2.0 drivers had not yet been released at the time of development; "beta" versions were first released in June 2023. Consequently, it was necessary to develop an ad hoc library for the Apple ecosystem through the Core MIDI framework.[8]

The basic principles behind a device manager are:

- Creation of a virtual MIDI client;
- Creation and arrangement of MIDI IN/OUT ports associated with a virtual client;
- Identification of connected compatible MIDI devices and their ports;
- Connection and disconnection between the device's port and the corresponding virtual port.

The device used for the *M2DM* configuration is *ProtoZOA* (see Fig. 2), which consists of two *Raspberry Pi Pico* microprocessors, 6+1 touch buttons, two potentiometers, a rotary encoder, two MIDI 1.0 IN/OUT ports, one display expansion port, an expansion port to connect the board via expansion cards such as Bluetooth or Ethernet, several configurable jumpers, and a port for external power supply.

The *Raspberry* units play two different roles: the unit called "UUT" processes MIDI 1.0 and 2.0 messages in both input and output, converts MIDI 1.0 messages into MIDI 2.0 and vice versa, and is responsible for the conversion between UMP messages and Bluetooth and IP protocols. Conversely, the unit called "MAIN" performs managing and debugging functions.

To implement the *M2DM*, we choose to mimic the one provided by JUCE, consisting of a list containing active MIDI devices, with the ability to select each of them. The JUCE framework library used to recreate the device list provides several methods to create and modify a table. A callback function has to be implemented to put the Device Manager (GUI) in communication with the MIDI2 library methods in the AudioProcessor.

In order to integrate the *M2DM*, it is necessary to set up the *Inclusive MIDI Controller* [8].[9] The first step is to update the GUI, introducing a page dedicated exclusively to the *M2DM* (see Fig. 3), visible through a button located in the header. The second step is to add functionalities to allow the user to map or remove the MIDI 2.0 parameters associated with the pad (these are represented by the two buttons, "Learn MIDI2" and "Unlearn MIDI2") After updating the

[6] https://juce.com
[7] https://www.audiokit.io
[8] https://developer.apple.com/documentation/coreMIDI
[9] The code we shared in this paper, does not include this part.

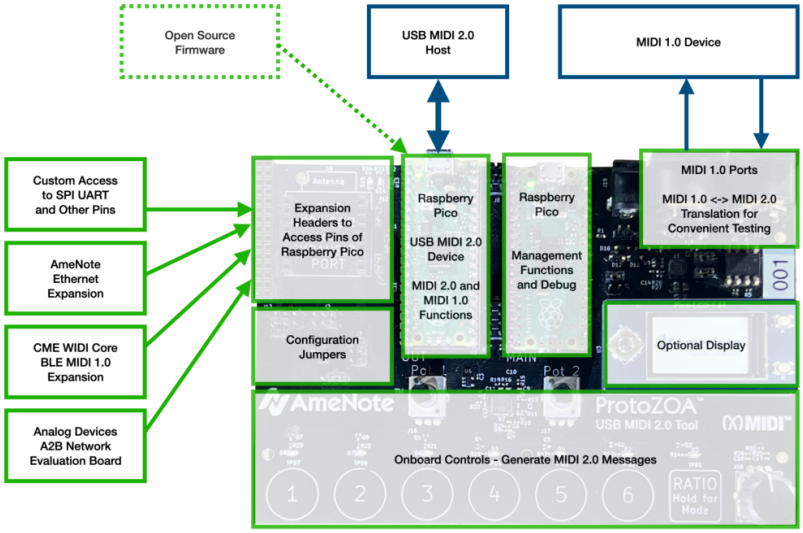

Fig. 2. *ProtoZOA* board component specifications.

GUI, the pre-existing data structures were updated to allow parameters related to the MIDI data to be saved.

When a MIDI 2.0 message is received, if the "Learn MIDI2" button is active, the parameters of the message are parsed and are saved in the data structure related to the selected pad; otherwise, the parameters of the received message are compared with those saved in the memory of the current pads: if they match, the event associated with the parsed pad is invoked; if they do not match, no event is triggered. A schematic example of the proposed system is shown in Figure 4.

Various functional tests were conducted on the implemented device manager. By keeping the "UMP" board connected and opening the *M2DM*, the list of active MIDI devices is displayed. Any active device in the list can be connected and disconnected by selecting/deselecting it through the GUI (see Fig. 3). By running these tests, it was noted that Core MIDI automatically encapsulates all MIDI messages of type 1.0 into type 2.0 (UMP Message Type 2, line 1 in the code example above). Consequently, at the present time, it cannot be ensured that the connected device is MIDI 2.0 only.

One further test was performed is to verify the proper operation of the *M2DM* within the *Inclusive MIDI Controller*. This test consists of connecting the *ProtoZOA* board to the application via the *M2DM*, mapping MIDI messages with their events to be triggered (e.g., Note on, Note off, Chord, CC, ...) and verifying their correct reception/transmission via a MIDI monitoring application such as the *Web MIDI Monitor* [1].[10] Next, a pad of the *Inclusive MIDI Con-*

[10] https://midimonitor.lim.di.unimi.it/

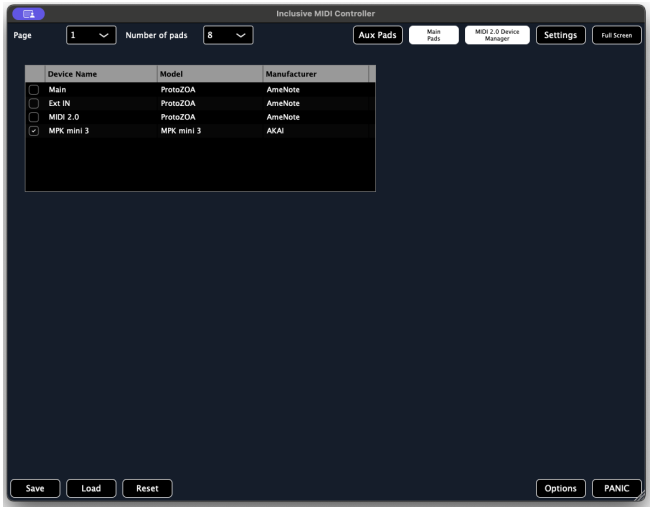

Fig. 3. *M2DM* integrated into the *Inclusive MIDI Controller*.

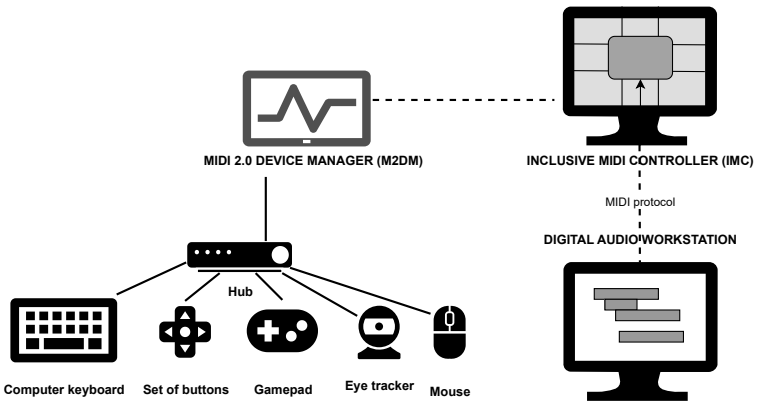

Fig. 4. Scheme of the described system.

troller is selected and the MIDI 2.0 message is mapped by activating a specific "Learn MIDI2" functionality available in the application, and sending the MIDI 2.0 message from the physical controller. After this mapping phase, the MIDI 2.0 controller can be manipulated through the mapped buttons, potentiometers, and rotary encoders. The triggered events are correctly visualized within the *Web MIDI Monitor*.

6 Conclusion

This paper has presented a prototype MIDI 2.0 controller designed to facilitate inclusive music interaction. By leveraging key features of MIDI 2.0, such as profile configuration and property exchange, the proposed system enables automatic adaptation to individual user needs, making music technology more accessible to individuals with disabilities.

As discussed in Section 3, the potential of MIDI 2.0 for inclusive music focuses mainly on three aspects. Auto-configuration simplifies the creation of a MIDI setup from both a physical and cognitive perspective. Software interfaces adapt to the devices in the system, introducing custom visualizations that depend on both the type of device and the form of impairment. The values generated by controls and their ranges can be represented in an accessible manner.

The implementation of the *MIDI 2.0 Device Manager* (*M2DM*) demonstrates how contemporary MIDI ecosystems can enhance usability and interoperability. Through the integration of the *ProtoZOA* board and the JUCE framework, the proposed solution successfully allows for automatic device discovery, parameter mapping, and adaptive user interfaces. The testing phase confirmed the effectiveness of the system, showing that MIDI 2.0 messages were correctly interpreted and managed within an accessible performance environment.

Future work will focus on expanding the range of supported input devices, refining real-time adaptation mechanisms, and conducting user studies to evaluate the practical impact of the system on diverse user groups. Additionally, further integration with assistive technologies such as speech synthesis, haptic feedback, and eye-tracking will be explored to enhance accessibility and inclusivity.

By bridging the gap between technological advancements and inclusive music-making, this work contributes to the growing field of accessible digital musical instruments, opening new possibilities for musicians of all abilities.

References

1. Avanzini, F., Faschi, V., Ludovico, L.A.: A web-based midi 2.0 monitor. In: Bresin, R., Falkenberg, K. (eds.) Proceedings of the Sound and Music Computing Conference 2023, Stockholm, Sweden. pp. 148–153. SMC (2023). https://doi.org/10.5281/zenodo.8341243
2. Baratè, A., Ludovico, L.A., Oriolo, E.: Investigating embodied music expression through the leap motion: Experimentations in educational and clinical contexts. In: McLaren, B.M., Reilly, R., Uhomoibhi, J., Zvacek, S. (eds.) Computer Supported Education - 10th International Conference, CSEDU 2018, Funchal, Madeira, Portugal, March 15–17, 2018, Revised Selected Papers. Communications in Computer and Information Science, vol. 1022, pp. 532–548. Springer International Publishing (2019). https://doi.org/10.1007/978-3-030-21151-6_25
3. Bowers, J., Archer, P.: Designing dmi with disabled musicians: An interdisciplinary approach to accessible music technology. In: Proceedings of the International Conference on New Interfaces for Musical Expression (NIME) (2022)

4. Burgess, T.: Tim burgess: improving hardware accessibility using midi. https://soundwithoutsight.org/tim-burgess-improving-hardware-accessibility-using-midi/ (2024), accessed: 2025-01-24
5. Davanzo, N., Avanzini, F.: Hands-free accessible digital musical instruments: conceptual framework, challenges, and perspectives. IEEE Access **8**, 163975–163995 (2020)
6. Davanzo, N., Avanzini, F., et al.: Design concepts for gaze-based digital musical instruments. In: Proceedings of the 19th Sound and Music Computing Conference. pp. 472–478. SMC network (2022)
7. Davanzo, N., Dondi, P., Mosconi, M., Porta, M.: Playing music with the eyes through an isomorphic interface. In: Proceedings of the Workshop on Communication by Gaze Interaction. pp. 1–5 (2018)
8. Faschi, V., Ludovico, L.A., Avanzini, F.: An accessible software interface for collaborative music performance. In: Proceedings of the 21st Sound and Music Computing Conference. pp. 150–157. SMC (2024). https://doi.org/10.5281/zenodo.14337032
9. Frid, E.: Accessible digital musical instruments—a review of musical interfaces in inclusive music practice. Multimodal Technologies and Interaction **3**(3) (2019). https://doi.org/10.3390/mti3030057, https://www.mdpi.com/2414-4088/3/3/57
10. Gillian, N.E., Paradiso, J.A.: The gesture recognition toolkit. Journal of Machine Learning Research **15**, 3483–3487 (2014)
11. Godøy, R.I., Haga, E., Jensenius, A.R.: Playing "air instruments": mimicry of sound-producing gestures by novices and experts. In: International Gesture Workshop. pp. 256–267. Springer (2005)
12. Gurevich, M., Stapleton, P.: Composing for hyperinstruments: Bridging the real and virtual. In: Proceedings of the International Conference on New Interfaces for Musical Expression (NIME) (2008)
13. Hunt, A., Kirk, R.: Mapping strategies for musical performance. Trends in gestural control of music **21**(2000), 231–258 (2000)
14. Jungleib, S.: The Complete SCI MIDI. Sequential Circuits, Inc. (1983)
15. Krout, R., Burnham, A., Moorman, S.: Computer and electronic music applications with students in special education: From program proposal to progress evaluation. Music Therapy Perspectives **11**(1), 28–31 (1993)
16. McPherson, A.P., Kim, Y.M.S.: The problem of the second performer: Interaction models for musical human-computer interaction. Computer Music Journal **44**(4), 17–34 (2020). https://doi.org/10.1162/comj_a_00548
17. MIDI Association: Common Rules for MIDI-CI Property Exchange. MIDI Association Document M2-103-UM. The MIDI Association and Association of Musical Electronics Industry (AMEI) (11 2024)
18. MMA: Complete MIDI 1.0 Detailed Specification. The MIDI Manufacturer Association (1996)
19. Moore, F.R.: The dysfunction of MIDI. Computer Music Journal **12**(1), 19–28 (1988). https://doi.org/10.2307/3680054
20. Riegel, C., Robinson, K.M., Larsen, T., Larsen, P.: Eye-tracking digital music creation and performance: disability and ableism. International Journal of Performance Arts and Digital Media pp. 1–17 (2024)
21. Samuels, K.: The meanings in making: Openness, technology and inclusive music practices for people with disabilities. Leonardo Music Journal **25**, 25–29 (12 2015). https://doi.org/10.1162/LMJ_a_00929, https://doi.org/10.1162/LMJ_a_00929
22. Savage, J.: An inclusive priority. Music Teacher **103**(9), 54–55 (2024)

23. Tanaka, A.: Sensor-based musical instruments and interactive music. In: The Oxford Handbook of Computer Music, chap. 12. Oxford University Press (04 2011). https://doi.org/10.1093/oxfordhb/9780199792030.013.0012
24. Vamvakousis, Z., Ramirez, R.: The EyeHarp: A Gaze-Controlled Digital Musical Instrument. Frontiers in Psychology **7**, article 906 (2016). https://doi.org/10.3389/fpsyg.2016.00906
25. Young, D., Murphy, J.: Midi 1.0 limitations and the future of musical interaction. In: AES Conference on Audio for Games (2017)