OPEN ACCESS Freely available online
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 November).
DOI: https://doi.org/10.17743/jaes.2022.0028

Web MIDI API: State of the Art and Future Perspectives

ADRIANO BARATÈ, AND LUCA A. LUDOVICO*
(adriano.barate@unimi.it) (luca.ludovico@unimi.it)

Laboratory of Music Informatics Department of Computer Science University of Milan Via G. Celoria, 18 - 20133 Milan, Italy

The *Web MIDI API* is intended to connect a browser app with Musical Instrument Digital Interface (MIDI) devices and make them interact. Such an interface deals with exchanging MIDI messages between a browser app and an external MIDI system, either physical or virtual. The standardization by the World Wide Web (W3C) Consortium started about 10 years ago, with a first public draft published on October 2012, and the process is not over yet. Because this technology can pave the way for innovative applications in musical and extra-musical fields, the present paper aims to unveil the main features of the API, remarking its advantages and drawbacks and discussing several applications that could take benefit from its adoption.

0 INTRODUCTION

The *Web MIDI API* is an application programming interface (API) whose goal is to allow the connection and interaction between a browser app and MIDI devices, thus bringing MIDI into the Web. MIDI, an acronym for Musical Instrument Digital Interface, is an industry music-technology standard. This protocol aims to connect digital music equipment and make a wide range of compatible devices interact by exchanging standardized messages. Being the result of a widely accepted standardization effort by the industry, MIDI-compatible equipment embraces products from many different brands and includes not only musical instruments but also computers, tablets, smartphones, and other computing devices. MIDI is commonly used by practitioners (e.g., musicians, DJs, producers, multimedia artists), music educators, and hobbyists.

The first release of the protocol, called MIDI 1.0, dates back to 1983 [1], and it has been partially revised and integrated in 1996 [2]. More recently, a new major version, called MIDI 2.0, has been released [3]. Despite the fact that the protocol has been around for decades, old MIDI devices and tools are still perfectly compatible and can be profitably integrated into a modern MIDI system.

MIDI made its first appearance in Web languages when Internet Explorer started supporting the `<bgsound>` element, whose goal was to embed a background audio track in a Web page. Standard MIDI File (SMF) was one of the supported formats, and it enjoyed moderate success due to the lightweight dimensions of MIDI songs. Unfortunately, such an element was properly implemented in Internet Explorer only, it was never included in any HTML specification and currently is no longer supported. Moreover, the goal was to add sound to a Web page by extracting music information from a file, with no possibility for the page to interact with external MIDI devices.

To some users, MIDI has become synonymous with Standard MIDI Files and General MIDI [4], but that is not the intent of the *Web MIDI API*. Conversely, such an API is intended to support the MIDI protocol in a Web framework, enabling browser applications to

1. manage (i.e., enumerate and select) the MIDI input and output devices available on the client system. The API can be used with any MIDI-compliant physical device connected to the user's computer and with any MIDI-compliant virtual device running on it; and
2. exchange (i.e., send and receive) MIDI messages between the browser app and the rest of the MIDI system.

As claimed in a document published by the MIDI Association, "the *Web MIDI API* connects your MIDI gear directly to your browser. Your browser connects you to the rest of the world" [5].

This work aims to highlight the progress reached by the *Web MIDI API* initiative, assess its success in terms of sci-

* Tel: +39-02-503-16382; e-mail: luca.ludovico@unimi.it

entific and industry results, discuss its main advantages and drawbacks, and propose a roadmap for future perspectives. The rest of the paper is structured as follows: SEC. 1 will present the state of the art, concerning the design and development milestones of the project, the presence in the scientific literature, and already-available libraries and applications based on it; SEC. 2 will discuss the key advantages and drawbacks of such technology; SEC. 3 will bring some examples to consolidate the previous discussion; SEC. 4 will introduce the future perspectives of the API; and finally, SE. 5 will draw the conclusions.

1 STATE OF THE ART

The *Web MIDI API* is being developed under the umbrella of the W3C Audio Working Group.[1] The idea is to provide support for MIDI devices as a standard feature in Web browsers and operating systems across multiple hardware platforms. About 10 years from the release of the first public document, it is still a working draft, and as such, it is not fully supported by the totality of HTML5-compliant Web browsers.[2] More detailed information is presented next.

1.1 Project Milestones

The goal of this section is to list the main milestones in the development, release and documentation of the *Web MIDI API*. At the moment of writing (March 2022), based on the official page of the project [4], the history of technical specifications includes the following:

- October 25, 2012 – first public draft;[3]
- December 13, 2012 – first working draft;[4]
- November 26, 2013 – second working draft;[5]
- March 17, 2015 – third working draft;[6]
- October 26, 2021 – latest editor's draft.[7]

The API specification's draft document is available in a GitHub repository that allows reconstructing the complete commit history.[8] Moreover, it is worth mentioning some articles published on the MIDI Association website which do not belong to the official documentation but represent a sort of endorsement by the MIDI industry and user community towards the initiative [5, 6].

Among the most popular desktop browsers, the *Web MIDI API* is currently supported by *Microsoft Edge*, *Google Chrome*, and *Opera*. As regards *Safari*, Apple has released a document called "Tracking Prevention in WebKit," stating that such a browser will not implement a host of APIs, including the *Web MIDI API*, because of fingerprinting concerns. Finally, the Mozilla team has recently introduced experimental support for the *Web MIDI API* that requires enabling a flag in *Firefox Nightly*. In conclusion, the current scenario is reported in Table 1, based on the data reported by "Can I use. . .".[9] Browser usage statistics show that about 75% of users are able to enjoy the *Web MIDI API* on their device.

Concerning the recent introduction of MIDI 2.0, it is worth underlining that the *Web MIDI API*, in its current form, does not explicitly address the new version of the protocol. Only its part focusing on device pairing and negotiation, being based on traditional System Exclusive communication for backward compatibility, is fully supported.

1.2 Scientific Literature

The *Web MIDI API* is surprisingly underrepresented in scientific literature. Excluding the technical documentation already cited in SEC. 1.1, a search carried out with *Google Scholar*[10] at the moment of writing (March 2022) revealed the existence of only two papers whose title contains the exact string "Web MIDI API".

In the first paper, Ludovico focuses on the potential of music-oriented educational applications based on the API. Such a contribution was presented in 2017 at the *9th International Conference on Mobile, Hybrid, and On-line Learning* (*eLmL 2017*) [7]. In the second paper, whose original Polish title can be translated into "Generating, Editing and Multi-Source Transmission of Audio Streams Using Web Audio API, WEBRTC and Web MIDI API," Walczak and Łukasik propose the latter technology to ensure communication with MIDI devices and create advanced Web audio applications in the context of the generation and transmission of audio streams from multiple sources [8]. This paper was published in 2020 in the proceedings of the *XVIII Miedzynarodowe Sympozjum Nowości w Technice Audio i Wideo* (*NTAV2020*) supported by the Polish section of the Audio Engineering Society.

Other paper titles mention the *Web MIDI API* in a more blended way, often in association with Web technologies for audio. For instance, Gurtner describes the applications of both *Web Audio* and *Web MIDI API* within a music notation editor, focusing on the way both APIs can improve the collaborative score editing experience [9]. Font and Bandiera present a visual interface taking advantage of both APIs for exploring Freesound content in a two-dimensional space and creating music by linking content in that space [10]. Finally, Stickland *et al.* propose a browser-based application that enables real-time collaboration between multiple remote instantiations of an established, mainstream, and fully featured digital audio workstation (DAW) platform over the Internet [11].

The aforementioned research in *Google Scholar*, extended to abstracts and paper bodies, returns about 100

[1] https://www.w3.org/groups/wg/audio

[2] The slowness in the development and release of Web standards is not surprising. For example, it took a decade for the *Web Audio API* to go from First Public Draft to Recommendation.

[3] https://www.w3.org/TR/2012/WD-webmidi-20121025/

[4] https://www.w3.org/TR/2012/WD-webmidi-20121213/

[5] https://www.w3.org/TR/2013/WD-webmidi-20131126/

[6] https://www.w3.org/TR/2015/WD-webmidi-20150317/

[7] https://webaudio.github.io/web-midi-api/

[8] https://github.com/WebAudio/web-midi-api/commits/

[9] https://caniuse.com/midi

[10] *Google Scholar* is a freely accessible search engine for scientific works. Such a Web tool indexes the metadata and full text of scholarly literature. It is available at https://scholar.google.com/.

Table 1. *Web MIDI API* browser compatibility. A dash in the last column represents current incompatibility, a question mark the absence of information.

Browser	Support	Most recent version	First compatible version
Internet Explorer	no	11 (October 17, 2013)	···
Edge	yes	99 (March 3, 2022)	79 (January 15, 2020)
Firefox	no	98 (March 8, 2022)	···
Chrome	yes	99 (March 1, 2022)	43 (May 20, 2015)
Safari	no	15.4 (March 14, 2022)	···
Opera	yes	83 (January 19, 2022)	30 (June 9, 2015)
Safari on iOS	no	15.4 (March 14, 2022)	···
Opera Mini	no	all (March 16, 2015)	···
Android Browser	yes	99 (March 1, 2022)	?
Opera Mobile	yes	64 (February 16, 2021)	46 (September 23, 2016)
Chrome for Android	yes	99 (March 1, 2022)	?
Firefox for Android	no	96 (January 11, 2022)	···
UC Browser for Android	yes	12.12 (August 17, 2016)	?
Samsung Internet	yes	16 (December 29, 2021)	4 (April 19, 2016)
QQ Browser	no	10.4 (May 19, 2020)	···
Baidu Browser	yes	43.23 (July 12, 2019)	7.12 (April 1, 2017)
KaiOS Browser	no	2.5 (June 1, 2018)	···

results, including technical specs and duplicated works. Some of these contributions do not focus on the *Web MIDI API*; rather, they present and discuss a given Web application, whereas the API is cited as the underlying technology to implement it [12, 13]. Another relevant category of works, including, e.g., [14, 15], proposes advancements and higher-level approaches to ease Web MIDI programming or unify different technologies, typically available in the form of JavaScript libraries. This subject will be deepened in SEC. 1.3. Finally, in some papers, the *Web MIDI API* is merely mentioned as one of the technologies under development in the field of Web audio and, as such, to be monitored and possibly integrated in the future [16, 17].

For the sake of completeness, it is worth underlining the availability of many online resources dealing with the *Web MIDI API* that, because of their purely informative nature, have never been indexed by *Google Scholar*. In addition to the already cited articles from the MIDI Association website, other noticeable examples can be provided [18–21].

In conclusion, the scientific literature about the *Web MIDI API* is quite poor. One of the possible reasons is the very technical content of the subject. In this field, published works mainly consist of technical specifications or informative communications, with little interest in scientific discussion or in-depth analysis.

We compared the *Google Scholar* coverage of the *Web MIDI API* with the one of the *Web Audio API*, which is another technology developed by the W3C Audio Working Group. The difference is really noticeable: more than 110,000 results against about 100. In the authors' opinion, the most plausible reason lies in the different levels of interest between pure audio and MIDI over the Web. Even if MIDI is considered as a standard by musicians, the adoption of MIDI technologies involves a niche community. Moreover, some network applications in which the lightweight format of MIDI messages could be a point of strength (e.g., interactive sonification of Web pages) nowadays can be easily implemented through pure audio technologies, thanks to the huge availability of fiber and WiFi connections and the upcoming spread of 5G networks [22].

Nevertheless, the lack of publications about the potential of the API in application contexts is surprising, e.g., in the fields of music production, gaming, and education. In this case, a possible reason is that the standardization process has not been completed so far, and consequently, designers and developers are cautious about adopting this technology, even if there are notable exceptions (see SEC. 1.3). As a side effect, also scientific research is currently limited and mainly intended to forecast future perspectives. For example, the lack of educational applications based on the *Web MIDI API*, leaving the advantages of MIDI-based approaches unclear, is hampering related pedagogical research.

1.3 Libraries and Applications

The state of the art about Web and MIDI must take into account also already available software tools. In this field, we can recognize two categories: (1) JavaScript libraries, mostly built on top of the *Web MIDI API*, and (2) software applications that are already adopting it, even if not officially released.

Addressing the former category, the *Web MIDI API* manages communication with MIDI devices at a very low level, which is sometimes overkill compared with the developers' typical needs. This approach requires the comprehension of the rationale behind MIDI communication and knowledge about MIDI commands. Efficiency in the design and development of browser applications can be improved by adopting third-party JavaScript libraries.

An example in this sense is provided by *WEBMIDI.js*, a project developed by Jean-Philippe Côté and licensed under the Apache License, Version 2.0. Its declared goals include sending and receiving MIDI messages with ease, controlling instruments with user-friendly functions (e.g.,

playNote, *sendPitchBend*, etc.), and reacting to MIDI input with simple event listeners (e.g., *noteon*, *pitchbend*, *controlchange*, etc.). The source code is available as a GitHub repository,[11] and additional documentation can be found in the official website.[12]

A more general approach involving multiple Web APIs for sound and music is offered by the *Web Audio API eXtension* (*WAAX*). This JavaScript framework for music applications is built on top of *Web Audio API* and other Web components. The claim is "Create, Connect and Tweak." Among the most relevant features, it is worth mentioning modularity, extensibility, and robustness. This result is achieved thanks to a four-part structure: Core, Plug-ins, Musical UI, and Workflow. Once again, the interested reader can access the dedicated website on GitHub.[13]

Because of its historical importance, it is also worth mentioning the *Jazz-Plugin API*,[14] now deprecated in favor of *JZZ.js*. The former API is a free software tool to play individual notes and control external MIDI In/Out devices via JavaScript. It worked in the browsers supporting NPAPI[15] and included binaries required to enable MIDI in browsers that did not support NPAPI. *JSS.js*[16] is a JavaScript MIDI library that works with *Node.js* in all major browsers under Linux, macOS and Windows and, with some limitations, on iOS and Android devices, too. *JZZ.js* enables the *Web MIDI API* in *Node.js*, and those browsers that do not support it natively.

Concerning the latter category, namely, Web applications based on the *Web MIDI API* or integrating it, many examples are currently available, and the following list does not claim to be complete. Trying to identify software families based on their features and goals, we can find online DAWs such as *Bandlab* [17] and *Soundation*.[18] Moreover, there are Web-based professional control interfaces, such as those implemented by *Traxus Interactive*,[19] the synths collected in the *WebSynths* portal,[20] *Heisenberg*,[21] *Yamaha Soundmondo*,[22] and *Viktor*.[23] Another use case is online scoring and collaborative music notation; among the most relevant products, it is worth mentioning *Flat* [24] and *Noteflight*.[25] A gamification approach for the development of musical skills is followed in *PlayDrumsOnline*.[26] Finally, an honorary mention goes to *RGBmidi*,[27] an online drawing tool that lets the user change colors via MIDI controller, for its original approach.

2 DISCUSSION

In this section, we will review the most relevant advantages that should encourage the standardization and adoption of the *Web MIDI API*, but we will also critically analyze the current drawbacks to be solved in order to obtain a successful technology.

2.1 Main Advantages

It's the Internet!—A browser application works on all platforms and devices, is accessible anywhere, often for free, provided that an Internet connection is available. There is no need to install software components, updates are automatic and the latest version is immediately available for users. Moreover, browsers make sharing and interaction easy also via social media and online communities. Finally, the *Web MIDI API* can be a bridge to connect music devices over a network.

A full orchestra at your fingertips—Before the advent of the *Web MIDI API*, the ways to have timbral richness in a web application were (1) the availability of sampled sounds for each note (when needed, even with different dynamics, articulation effects, etc.) and (ii) the explicit implementation of sound synthesis, e.g., via the *Web Audio API*. Now a key advantage is the possibility to delegate sound synthesis to an external MIDI synth, responsible for the audio generation and timbre management. In this way, a Web application can easily support at least the 128 programs granted by General MIDI. These programs embrace not only "traditional" musical instruments but also synthesizers and audio effects. The instrumental set can be further extended according to the MIDI specs, e.g., by using additional sound banks. In the case of a virtual synth, the quality of samples can be improved by using sound fonts, if supported by the software.

Another brick in the wall—By using the *Web MIDI API*, the resulting browser applications can take part in a MIDI system, potentially complex and made up of heterogeneous devices (controllers, synthesizers, sequencers, etc.), virtual and/or physical. Thus, the Web developer can add functions to the system, implement missing elements, design new ways to input, manipulate, and show MIDI information, and so on. For hardware device makers, instrument control panels that previously needed to be produced in multiple versions can now be implemented once in HTML5, and consumers can run them on any Web device.

Easy-peasy—A Web programmer with a basic knowledge of MIDI's rationale and main messages can implement even complex browser applications with very little effort. In fact, the functions provided by the *Web MIDI API* are mainly limited to establishing a connection toward in/out ports, sending MIDI messages (also in their standard format made of status and data bytes), and supporting callback functions to manage incoming messages. Moreover, the li-

[11] https://github.com/djipco/webmidi
[12] https://webmidijs.org/
[13] https://hoch.github.io/WAAX/
[14] https://jazz-soft.net/doc/Jazz-Plugin/
[15] NPAPI stands for Netscape Plugin Application Programming Interface. It is an application programming interface to support plugin integration in compatible web browsers.
[16] https://jazz-soft.net/doc/JZZ/
[17] https://www.bandlab.com/
[18] https://chrome.soundation.com/
[19] https://traxusinteractive.com/category/interfaces/webmidi/
[20] https://www.websynths.com/
[21] https://www.audiotool.com/product/device/heisenberg/
[22] https://soundmondo.yamahasynth.com/voices
[23] http://nicroto.github.io/viktor/
[24] https://flat.io/
[25] https://www.noteflight.com/
[26] https://www.playdrumsonline.com/
[27] http://www.h3nk.com/midi.htm

braries listed in SEC. 1.3 aim to further reduce the burden by providing simplified approaches to Web programming.

MIDI beyond MIDI—The MIDI protocol was originally conceived to make music-related machinery from different producers cooperate by exchanging standardized messages in a shared format. Nevertheless, such a protocol can be seen as a way to exchange general-purpose binary information as well. On one side, you can adapt Channel Voice, Channel Mode, System Common, and System Real-Time messages to purposes different from the original ones; on the other side, if you want to preserve the original intent and function of MIDI messages, you can deliver custom information through System Exclusive messages, whose payload is not standardized by MIDI.

2.2 Main Drawbacks

A wannabe standard—Despite its 10-year-long lifetime and the endorsement by W3C, the *Web MIDI API* is still at the development stage, and the official documentation is a working draft. Even if several prototypes are being developed, this aspect can discourage the adoption of such technology in publicly available applications in which the expectation is transparency with respect to the browser type and version in use.

The Internet is for everyone, but MIDI is not!—Browser applications adopting the *Web MIDI API* are supposed to interact with a physical, virtual, or mixed external MIDI system. If MIDI equipment is not available, such tools are totally useless or, in the best case, lose much of their efficacy. Moreover, as a direct consequence of the previous item, users equipped with an incompatible browser (e.g., *Firefox* or *Safari*) cannot enjoy the MIDI experience.

When the going gets tough—Even if an expert would easily complete the configuration and setup of the system, for a common user, making the browser application work can be a tricky task. The input and output ports must be correctly set, above all, when multiple concurrent devices are available; in the other case, apparently, the app would not receive or produce any message, and the perceived result would be a malfunction of the system. In addition, some browsers present disclaimers that sound obscure and alarming to "uninitiated" users. For example, *Google Chrome* opens a dialog window reporting: "[Deprecation] Web MIDI will ask a permission to use even if the sysex is not specified in the MIDI Options since around M82, around May 2020."

(Un)chained melody—Most users erroneously overlap the concepts of MIDI protocol and Standard MIDI Files, namely, digital documents that provide a way for MIDI music sequences to be saved, transported, and opened. A MIDI file stores MIDI messages, which are commands that tell a musical device what to do in order to make music. As explicitly stated in [4], the use case of simply playing back a file in SMF format is not within the purview of the *Web MIDI API*. Conversely, a Web application could receive MIDI messages coming from a MIDI file opened in a sequencer, parse, manipulate, and consume them or send them in output to an external synthesizer.

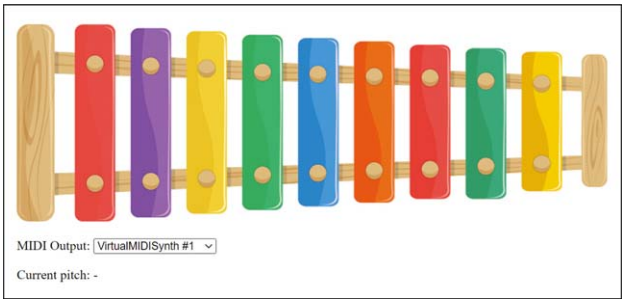

Fig. 1. Interface of the *Xylophone* example.

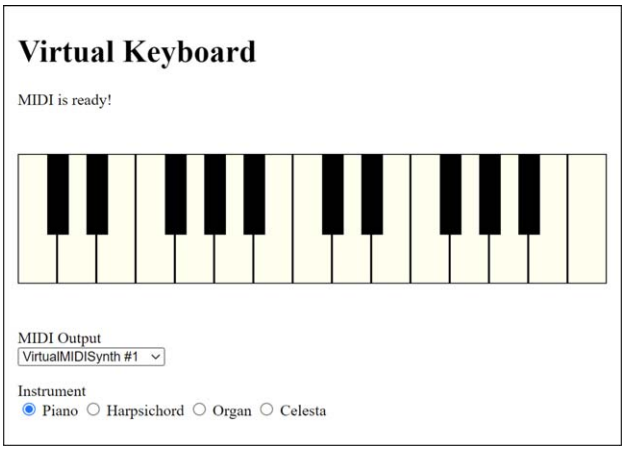

Fig. 2. Interface of the *Keyboard* example.

3 EXAMPLES

For the sake of clarity, in this section, we will present some basic examples aiming to show the potential of the *Web MIDI API*. The HTML and CSS code has been kept intentionally simple, and the implementation of the MIDI part requires the use of the API only, with no additional library. Needless to say, the examples can be enjoyed only if the following conditions are met: (1) a network connection is available, (2) the Web browser in use is compatible with the *Web MIDI API*, and (3) either virtual or physical input/output MIDI devices are available. The source code of each example can be inspected using the suitable developer tools offered by the browser.

The first scenario concerns the generation of MIDI messages from a Web user interface in order to send them to a synthesizer. In particular, the *Xylophone* example[28] employs a mapped image to trigger NoteOn and NoteOff messages via mouse clicks (see Fig. 1). An evolution of this case study is the *Keyboard* example,[29] which emulates a keyboard controller and supports also Program Change messages (see Fig. 2). These examples also present basic management of MIDI out connections.

As another scenario, we focus on the retrieval of incoming MIDI messages and the triggering of an event handler. A

[28]https://www.lim.di.unimi.it/download/midi/web/xylophone.html

[29]https://www.lim.di.unimi.it/download/midi/web/keyboard.html

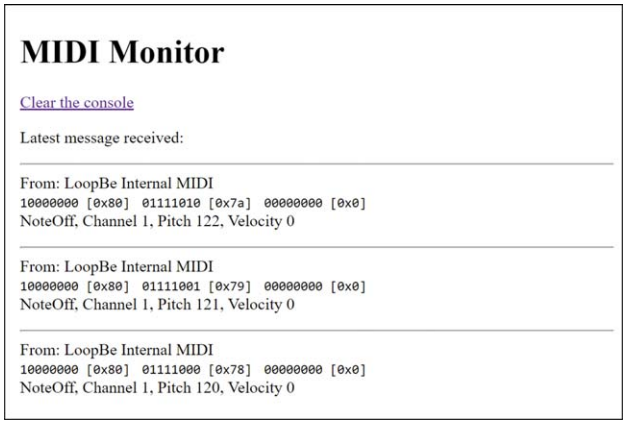

Fig. 3. Interface of the *MIDI monitor* example.

simple example is provided by a *MIDI monitor*,[30] namely, an app to display MIDI messages received from the MIDI system. In our implementation, whose interface is shown in Fig. 3, there is no selection of the MIDI input port because we want to display all incoming messages.

These two approaches can be combined in order to implement browser applications designed to stay in the middle of a MIDI chain. This kind of software tool, for instance, can read Channel Voice messages in input, filter them based on a customizable condition, add new messages through its on-screen interface, and finally, route MIDI commands to an output port.

4 FUTURE PERSPECTIVES

After presenting the state of the art, mentioning the main advantages and drawbacks, and offering some clarifying examples, the research question becomes: How can the *Web MIDI API* expand the frontiers of Web audio?

The major premise to answer such a question is that the API has not yet been standardized by W3C, so its support by Web browsers is not mandatory. Whatever tool currently integrates the *Web MIDI API*, it does so experimentally, and to maintain full compatibility with browsers, it should handle the case in which the API is not supported. In some cases, the problem can be solved by adopting alternative strategies. For example, Web DAWs that integrate MIDI sequencing can provide a backup synthesizer based on the already-supported *Web Audio API* instead of General MIDI programs. Unfortunately, when user interaction is strongly linked to the *Web MIDI API*, the only possibility is to show a disclaimer message. Clearly, this is a strong limitation, which, hopefully, in the not-too-distant future, will be overcome by the conclusion of the Web Audio Group works. It is worth underlining that the authors of this paper do not belong to this working group and have no additional information about the standardization process but the official documentation.

Going back to the research question, the scientific literature and some already-available Web tools are suggesting a better integration among Web technologies for sound and music computing. The *Web Audio API* involves handling audio operations, whereas the *Web MIDI API* supports real-time music information exchange with musical instruments and other equipment. Their functions do not overlap; rather, they can be combined to cover a wide range of operations, for instance, everything a user expects to find inside a modern software system to manage audio. It is no coincidence that these technologies are often mentioned together in works that discuss online DAWs.

Another future perspective concerns the further development of higher-level libraries based on the *Web MIDI API*. Historically, a similar process involved the *Web Audio API*, which soon originated *Howler.js*, *SoundJS*, *Tone.js*, and many other spin-off libraries currently in use by Web developers. Likewise, we can expect the birth and evolution of something similar for MIDI. The initiatives mentioned in SEC. 1.3, e.g., *WEBMIDI.js*, are proof of that.

Finally, much remains to be done in the field of innovative applications based on user interaction. First, the *Web MIDI API* can be profitably applied to formal and nonformal music education, in which the exchange of MIDI messages either is automatically evaluated by a remote software system or provides the educator with a support tool. This may require the adoption by musicians of MIDI-compatible musical instruments whose interfaces recall those of traditional instruments (e.g., MIDI keyboard controllers, wind instruments, drums, etc.). In this sense, an ad hoc tool could automatically assess, e.g., the perfect timing of a drum loop or the correct execution of a musical scale on a keyboard. This kind of approach is becoming more and more relevant in the context of remote education, a subject particularly felt in the COVID-19 pandemic era.

Nevertheless, even more interesting educational applications could emerge when adopting nontraditional interfaces for music expression (e.g., wearable devices, motion-based controllers, building blocks, etc.). In these scenarios, a more natural and/or amusing user interaction can foster not only the acquisition of basic musical concepts but also the development of soft skills, even in very young or impaired learners.

The *Web MIDI API* has been described so far as a consumer of messages coming from an upstream MIDI chain. Nevertheless, as shown in SEC. 3, a Web tool can also operate as a generator of MIDI messages to be sent in output to a MIDI system. This option allows the design and implementation of original interfaces for musical creativity and expression in the form of Web pages. Once again, the acquisition of musical skills can benefit from a gamification process. Moreover, ad hoc user interfaces can foster accessibility for impaired users, thus paving the way toward inclusive music expressiveness.

Finally, the *Web MIDI API* can be employed in nonmusical applications in which MIDI is merely used as a numeric protocol to exchange control values between compatible devices. This means that popular MIDI hardware can be used to control any kind of software in the browser and provide

[30] https://www.lim.di.unimi.it/download/midi/web/midi_monitor.html

a low-cost and/or highly specialized alternative solution to standard hardware/software interfaces. Some applications falling under this definition (e.g., *RGBmidi*) have been presented in SEC. 1.3. Thanks to this approach, to cite but a few examples, physical buttons and knobs can substitute on-screen sliders, a multipad device can be operated by a user equipped with protection gloves in an industrial setting, or a lightpad can provide an easy-to-use alternative to send color information to a remote system. In this context, possible applications are countless.

5 CONCLUSION

MIDI is one of the oldest protocols still in use in computing in its original form. Even if the recent version 2.0 has introduced some relevant novelties to expand its potential (e.g., a higher number of available MIDI channels and bidirectional communication), the key concept of maintaining full compatibility with MIDI 1.0 poses some technical limitations.

Problems to be solved with MIDI in general, and MIDI over the Web in particular, include the low transmission speed between devices (limited to 31.25 Kbit/s for MIDI 1.0), the consequent issue of MIDI choke (i.e., the condition that occurs when a MIDI device tries to send data at a rate exceeding the cable transmitting capability), and the management of latency. Some of these issues have been practically solved by adopting other communication protocols, e.g., MIDI over USB or over Bluetooth for local devices and MIDI over Ethernet for distributed MIDI systems. Moreover, a MIDI-based approach obviously requires the availability of MIDI-compatible devices, but virtual tools can provide a valid alternative to the lack of physical equipment.

The *Web MIDI API* and all the related efforts aiming to integrate the MIDI protocol into browser applications can be considered promising technologies that will contribute to expanding the frontiers of Web audio. Thanks to the widespread use of mobile devices and the ubiquitous presence of network connections, when the W3C standardization initiative finally comes to an end, Web MIDI will have the potential to be one of the most disruptive music technologies.

6 REFERENCES

[1] International MIDI Association, *MIDI 1.0 Specification* (1983 Aug.).

[2] MIDI Manufacturers Association, *The Complete MIDI 1.0 Detailed Specification* (1996 Feb.).

[3] Association of Musical Electronics Industry (AMEI) and MIDI Manufacturers Association (MMA), *MIDI 2.0 Specification* (2020 Jan.).

[4] C. Wilson and J. Kalliokoski, "Web MIDI API," https://webaudio.github.io/web-midi-api/ (accessed Mar. 17, 2022).

[5] The MIDI Association, "About Web MIDI," https://www.midi.org/midi-articles/about-web-midi (accessed Mar. 17, 2022).

[6] The MIDI Association, "Web MIDI (MIDI Support in Web Browsers)," https://www.midi.org/developer-web-midi-info (accessed Mar. 17, 2022).

[7] L. A. Ludovico, "The Web MIDI API in On-Line Applications for Music Education," in *Proceedings of the Ninth International Conference on Mobile, Hybrid, and On-line Learning (eLmL)*, pp. 72–77 (Nice, France) (2017 Mar.).

[8] M. Walczak and E. Łukasik, "Generowanie, edycja i transmisja wielośródłowych strumieni audio z wykorzystaniem WEB AUDIO API, WEBRTC i WEB MIDI API," *XVIII Miedzynarodowe Sympozjum Nowości w Technice Audio i Wideo NTAV2020*, pp. 9–13 (Wrocław, Poland) (2020 Oct.).

[9] C. Gurtner, "Applications of Audio and MIDI API Within a Music Notation Editor," in *Proceedings of the 2nd Web Audio Conference (WAC)* (Atlanta, Georgia) (2016 Apr.).

[10] F. Font and G. Bandiera, "Freesound Explorer: Make Music While Discovering Freesound!" in *Proceedings of the 3rd Web Audio Conference (WAC)* (London, UK) (2017 Aug.).

[11] S. Stickland, R. Athauda, and N. Scott, "Design of a Real-Time Multiparty DAW Collaboration Application Using Web MIDI and WebRTC APIs," in *Proceedings of the 5th Web Audio Conference (WAC)*, pp. 59–64 (Trondheim, Norway) (2019 Dec.).

[12] A. Baratè, L. A. Ludovico, and D. A. Mauro, "A Web Prototype to Teach Music and Computational Thinking Through Building Blocks," in *Proceedings of the 14th International Audio Mostly Conference: A Journey in Sound*, pp. 227–230 (Nottingham, UK) (2019 Sep.). https://doi.org/10.1145/3356590.3356625.

[13] T. Bazin and G. Hadjeres, "Nonoto: A Model-Agnostic Web Interface for Interactive Music Composition by Inpainting," *arXivpreprint arXiv:1907.10380* (2019). https://doi.org/10.48550/ARXIV.1907.10380.

[14] J. Kleimola, "DAW Plugins for Web Browsers," in *Proceedings of the 1st Web Audio Conference (WAC)* (Paris, France) (2015 Jan.).

[15] S. Kachalo, "JZZ.js – A Unified API for MIDI Applications," in *Proceedings of the 2nd Web Audio Conference (WAC)* (Atlanta, Georgia) (2016 Apr.).

[16] H. Choi and J. Berger, "WAAX: Web Audio API eXtension," in *Proceedings of International Conference on New Interfaces for Musical Expression (NIME)*, pp. 499–502 (Daejeon, Korea) (2013 May).

[17] F. L. Schiavoni, L. L. Gonçalves, and J. M. da Silva Sandy, "Mosaicode and the Visual Programming of Web Application for Music and Multimedia," *Revista Música Hodie*, vol. 18, no. 1, pp. 132–146 (2018 Jun.). https://doi.org/10.5216/mh.v18i1.53577.

[18] J.-P. Côté, "Web MIDI: Music and Show Control in the Browser," https://tangiblejs.com/posts/web-midi-music-and-show-control-in-the-browser (accessed Mar. 17, 2022).

[19] A. Hronco, "Making Music in the Browser – Web MIDI API," https://www.keithmcmillen.com/blog/

making-music-in-the-browser-web-midi-api/ (accessed Mar. 17, 2022).

[20] J.-P. Côté, "MIDI is Staging a Comeback... in Your Browser!" https://fitc.ca/presentation/midi-staging-comeback-browser-2/ (accessed Mar. 17, 2022).

[21] P. Anglea, "Getting Started With The Web MIDI API," https://www.smashingmagazine.com/2018/03/web-midi-api/ (accessed Mar. 17, 2022).

[22] A. Baratè, G. Haus, and L. A. Ludovico, "Advanced Experience of Music through 5G Technologies," *IOP Conf Mater Sci Eng*, vol. **365**, no. 1, paper 012021 (2018 Jun.). https://doi.org/10.1088/1757-899X/364/1/012021.

THE AUTHORS

Adriano Baratè

Luca A. Ludovico

Adriano Baratè is currently a researcher with the Computer Science Department, the University of Milan. He is a member of the Laboratory of Music Informatics (LIM) and secretary of the IEEE Working Group for XML Musical Application.

•

Luca A. Ludovico is a professor of Music Informatics, Sound Synthesis, and MIDI Programming at the University of Milan, Italy. He received his Master's Degree in Computer Engineering from the Polytechnic of Milan in 2003 and his Ph.D. in Computer Science from the University of Milan in 2006. Since 2003, he has been a member of the Laboratory of Music Informatics (LIM) of the University of Milan. His research interests include the formalization and encoding of symbolic music, multimedia, and cultural heritage. He is a member of the W3C Community on Music Notation, a member of the MIDI Association, an IEEE member and the vice-chair of the IEEE Working Group for XML Musical Application.