A Web-Based MIDI 2.0 Monitor

Federico Avanzini, Vanessa Faschi, Luca Andrea Ludovico
Laboratory of Music Informatics (LIM), Department of Computer Science, University of Milan
{name.surname}@unimi.it

ABSTRACT

This paper presents a publicly available MIDI monitor. The application provides a web interface to list both MIDI 1.0 and MIDI 2.0 messages and aims to offer an easy-to-read interpretation of standardized message parts. In this sense, particular attention is paid to the *SysEx* messages that implement MIDI-CI communication in the context of MIDI 2.0. The tool was developed through the Web MIDI API, a standard proposal by the W3C Audio Group currently supported by most web browsers, in particular the Chromium-based ones and Mozilla Firefox. Possible uses range from diagnosing issues with MIDI devices and connections to investigating MIDI 2.0 concepts in educational scenarios.

1. INTRODUCTION

MIDI, standing for Musical Instrument Digital Interface, is a well-known technical standard that, despite its decades-long history, is still widely adopted in the field of electronic music generation and musical information description. The MIDI specification describes the communications protocol, the related digital interface, and the electrical connectors designed to support a wide range of electronic music-oriented devices (e.g., musical controllers, synthesizers, sequencers, etc.) [1]. About 12 years after the release of MIDI 1.0 specification, a file format called SMF (Standard MIDI File) was also standardized with the aim of saving and reproducing sequences of MIDI messages and related metadata [2].

A recent milestone in the evolution of this technology is MIDI 2.0 [3], first released in early 2020. MIDI 2.0 represents an extension of MIDI 1.0 rather than a stand-alone specification. In other terms, MIDI 2.0 does not replace MIDI 1.0 but builds on its core principles, architecture, and semantics. The foundational architecture for MIDI 2.0 expansion is described in the *MIDI Capability Inquiry* (MIDI-CI) specification. MIDI 2.0 compatible devices support bidirectional communication and the mutual exchange of information and profiles. MIDI-CI lets such devices agree to use extended MIDI capabilities beyond those already defined in MIDI 1.0 while preserving backward compatibility. In this way, MIDI systems can embed both MIDI 1.0 and MIDI 2.0 devices so as to support the wide range of musical instruments and SW/HW tools released in the last decades.

Backward compatibility is implemented through the use of *System Exclusive*, or simply *SysEx*, messages. This family of MIDI messages is designed to transmit information about specific functions often depending on the product's manufacturer and model. The internal freedom offered by *SysEx* messages made it possible in the past to compensate for some shortcomings of the MIDI 1.0 protocol by standardizing so-called universal *SysEx* messages, accepted by all manufacturers and models interested in adhering to certain standards (e.g., General MIDI). Please note that *SysEx* messages can be made up of any number of bytes, potentially very large. *SysEx* messages were already available in MIDI 1.0, thus using them to carry MIDI 2.0 communication is a very simple solution to guarantee backward compatibility. One of the critical issues with *SysEx* messages is that their low level of standardization combined with their typically high number of bytes make them difficult for the user to understand.

In this sense, a category of software tools known as MIDI monitors can come in handy. A MIDI monitor aims to let users view, analyze, and test MIDI data. Cook [4] describes it as a simple terminal program that displays MIDI messages coming into your system in words rather than numbers. When inserted into a MIDI layout, the monitor displays incoming and outgoing MIDI messages in real time.

The goals and applicability fields of this category of software are multiple. First, a MIDI monitor can help diagnose issues with MIDI devices or connections, which is particularly important for musicians who are not experts with technologies. Software and hardware designers can benefit from MIDI monitoring, too, in testing MIDI drivers and applications. Finally, MIDI monitors can play a fundamental role in educational activities dealing with MIDI since they show the byte values exchanged in MIDI communication at a relatively low level of abstraction. [1] Available MIDI monitors can be stand-alone products (e.g., the tools by Morningstar, Morson, and Snoize) or modules integrated into more general software (e.g., Logic Pro, MIDI-OX, Steinberg Nuendo).

This paper aims to describe a publicly available MIDI monitor designed and implemented through the Web MIDI API. Although the tool can be profitably used to analyze MIDI 1.0 communication, the major novelty compared to already existing alternatives is the attention paid to inter-

[1] The MIDI protocol is agnostic with respect to the transport layer, thus a MIDI monitor shows the messages ignoring low-level aspects such as the start and stop bit required by the electrical specification.

preting the various parts that constitute MIDI 2.0 messages. The remainder of the paper is organized as follows: Section 2 will illustrate the technologies employed to develop the MIDI monitor, Section 3 will describe such a web tool, Section 4 will underline pros and cons, and, finally, Section 5 will draw the conclusions.

2. EMPLOYED TECHNOLOGIES

In order to make our MIDI monitor easily available and updatable, we chose to implement it in the form of a web application. This category of software tools refers to applications accessed via a web browser over a network and developed using browser-supported languages. For their execution, web applications depend on web browsers [5].

The *Web MIDI Monitor* was developed using only client-side formats and languages for web programming. In particular, the platform combines HTML and CSS for static content and its presentation, respectively. In addition, its active behavior is governed by JavaScript, a well-known client-side programming language. Since no server-side language is implied, the *Web MIDI Monitor* can operate also in local mode on the user's device.

Concerning the JavaScript code, one project is worth specific attention. The management of the connections and the communication with the external MIDI system was realized through the Web MIDI API, a W3C [2] Audio group initiative that is currently striving to become an officially recognized standard. At the moment of writing, the specification is at the stage of Editor's Draft, i.e. a work-in-progress document allowing the group to iterate internally on its content for consideration [6]. Drafts have not received a formal review and are not endorsed by W3C. Even if not recognized as a standard so far, nowadays the Web MIDI API is well supported by most web browsers. [3] In this sense, a relevant exception is still represented by Apple Safari.

As reported in [6], this specification describes an API for MIDI-protocol support which enables web applications to enumerate and select MIDI I/O devices on the client system and send and receive MIDI messages. By providing access to a wide range of musical instruments, MIDI devices, and compatible software tools, the Web MIDI API enables both music and non-music MIDI applications. Concerning the former aspect, the scientific literature reports research works dealing with its direct use [7–9], higher-level libraries based on it [10, 11], and experiments mixing it with the Web Audio API, another JavaScript interface developed by the W3C Audio Group [12]. On the other side, the scientific literature has also explored applications based on music concepts but addressing extra-musical aspects, e.g. the development of soft skills, problem-solving attitude, and computational thinking abilities [13, 14].

3. THE WEB MIDI MONITOR

In this section, we will describe the software tool developed to perform MIDI monitoring via the web. Such a platform, called *Web MIDI Monitor*, is publicly available at `https://midimonitor.lim.di.unimi.it/`.

Figure 1. Example of a MIDI 1.0 message list.

3.1 Basic Functions

The *Web MIDI Monitor* aims to display in real-time the MIDI messages arriving on all input interfaces recognized in the client system. Since multiple input sources may be available, the message's input port is also displayed.

As an example, a monitoring session is shown in Figure 1. The list includes an *All Notes Off* (corresponding to a specific *Control Change*), a *Program Change*, a *Note-On*, and a *Note-Off* message. In accordance with the expected functions of a MIDI monitor, the tool presents not only the sequence of status and data bytes (in binary, decimal, and hexadecimal format) but also a human-readable interpretation of data. Flags and meaningful aggregations of bits within a byte are highlighted (e.g., the most significant bit to distinguish between a status and a data byte, or the low nibble in the status byte of channel messages to identify the current channel). When possible, numeric information is matched against MIDI tables to offer a straightforward interpretation to the user; in Figure 1, this is the case of the CC number that identifies the *All Notes Off* message within the *Control Change* family, the name of the instrument for *Program Change*, and the note pitch translated into Scientific pitch notation for *Note-On* and *Note-Off*.

Another goal when parsing MIDI information is to reconstruct values that span across multiple bytes. It is the case of the *Pitch Bend Change* message, where 2^{14} detuning levels are expressed on 2 data bytes and remapped onto the range $[-8192, +8191]$, or the scenario of MIDI Universal IDs introduced in MIDI 2.0, that are pseudo-random 28-bit values represented on 4 data bytes. This approach can be further extended to the visualization of values that span across multiple MIDI messages, as in the case of *MIDI time code* that splits an SMPTE timecode into a series of 8 messages.

3.2 Advanced Features

The features listed above are present in most MIDI monitors. A novelty aspect of our proposal is the use of web technologies, but, apart from this, there are other characteristics that are worth underlining.

[2] World Wide Web Consortium, `https://www.w3.org/`
[3] Source: `https://caniuse.com/midi`

The main motivation of the project is to provide the user with better support to understand MIDI 2.0 messages. As briefly mentioned in Section 1, MIDI 2.0 introduces the so-called MIDI-CI communication that relies on suitably-formatted universal *SysEx* messages. MIDI 2.0 specification defines how to interpret a part of the data-byte sequence that, in a manufacturer's *SysEx* message, would be completely free and depending on implementation choices. A notable example is the unique addresses of the sender and recipient device, which must be reconstructed by combining 4 bytes of data each; another case is the value of MIDI profiles expressed in JSON format, which therefore spans across numerous bytes to be interpreted as text characters. In the mentioned scenarios, the mere list of bytes that constitute the *SysEx* message, as shown by generic MIDI 1.0-compatible monitors, would be hardly readable by the user.

The general form of MIDI-CI messages, reported in hexadecimal values, is:

`F0 7E cc 0D xx 01 ss ss ss ss dd dd dd dd [...] F7`

Analyzing the message structure byte by byte:

- `F0` represents the *Start of Exclusive* message, as defined by MIDI 1.0 specification;
- `7E` identifies any non-real time universal *SysEx*, once again in accordance with MIDI 1.0;
- `cc` is the channel identifier. Value `7F` is reserved to identify the whole MIDI port, namely all channels;
- `0D` is the value of the *Sub-ID #1* field for a generic MIDI-CI message. *Sub-ID #1* and the following *Sub-ID #2* work together to specify the function and sub-function of a universal *SysEx*.[4] This byte is what turns a generic non-real time universal *SysEx*, as already defined in MIDI 1.0, into a MIDI-CI message for MIDI 2.0;
- `xx` is a placeholder for the *Sub-ID #2* part that determines a more specific function with respect to *Sub-ID #1*;
- the following byte provides the MIDI-CI version, currently set to 1;
- the subsequent 4 + 4 bytes contain the MIDI universal IDs (MUIDs) of the source and destination device respectively;
- the variable-length part represents the message's payload;
- finally, `F7` stands for *End of Exclusive*.

Please note that the described structure is far more articulated than in the case of typical MIDI 1.0 messages, whose length ranges from 1 to 3 bytes (with the obvious exception of *SysEx* messages). Moreover, such a structure ignores the content of the payload, which could be further decomposed into smaller information units. Taking into account complexity and making MIDI 2.0 messages readable are the main aspects that motivated the release of a new monitoring tool aware of the MIDI 2.0 extension. We presented the generic MIDI-CI message structure in detail because this internal division guided us in designing MIDI 2.0 visualization.

Minor features include the possibility to filter incoming messages by input interface, include/exclude single messages or message families from the monitoring, export the received data in Comma-Separated Values (CSV) format, and set the color palette to improve visualization.

3.3 Using the *Web MIDI Monitor*

In order to see the *Web MIDI Monitor* in action, it is worth clarifying some operating aspects.

First, the web browser has to be compatible with the Web MIDI API. In this sense, the *Web MIDI Monitor* is supported also by mobile devices, provided that they have a suitable browser installed.

Secondly, the web tool must be enabled to receive MIDI messages, which means that: i) an upstream part of the MIDI system is required to generate messages, and ii) the client where the *Web MIDI Monitor* is running must have at least one MIDI input port. The MIDI system to monitor can be a full physical, virtual, or mixed one. The MIDI layout can include both MIDI 1.0 and MIDI 2.0 devices.

Finally, in order to allow the management of *SysEx* messages by the Web MIDI API, the user is required to explicitly grant access to local MIDI devices. Usually, the browser asks for such permission via a dialog window.

To see the *Web MIDI Monitor* in action, it is possible to introduce it into a physical MIDI setup by making the host PC talk with the other MIDI devices in the setup. The simplest way to achieve such a mixed physical/virtual MIDI system is to attach a MIDI controller to the computer via a USB cable, provided that such a connection is supported by the MIDI device, and employ the USB-MIDI transport protocol. A more articulated scenario relies on the use of an external MIDI computer interface that, in addition to IN/OUT ports for traditional 5-pin DIN cables, can deliver MIDI messages via USB. For example, the *Web MIDI Monitor* was tested by attaching a *Roland PC200* master keyboard to an *M-AUDIO Fast Track Pro* interface, in turn, connected to the computer via USB (see Figure 2). The result of the test session shown in the figure includes a *Note-On* and a *Note-Off* 3-byte message surrounded and interleaved by a number of 1-byte *Active Sensing* messages automatically produced by the *Fast Track Pro* to keep the communication channel alive.

When no dedicated hardware is available, the simplest way to enjoy the potential of the *Web MIDI Monitor* is to launch a software MIDI controller and attach it to a virtual MIDI port, which, in turn, is seen as a virtual input by the browser app. Unfortunately, this approach is unlikely effective to send (and, consequently, track) *SysEx* messages. Rather, this scenario can be emulated using a MIDI sequencer integrating a message editor, so as to enter and send any kind of MIDI commands, including MIDI

[4] The list of MIDI 1.0 supported values is available at `https://www.midi.org/specifications-old/item/table-4-universal-system-exclusive-messages`.

Figure 2. Testing the *Web MIDI Monitor* with a physical MIDI setup.

2.0 messages. Another option is to produce ad-hoc MIDI messages via a dedicated application, such as the *MIDI Sysex Editor* available at `https://midi.toys/`.

Since, at the moment of writing, few hardware devices support MIDI 2.0 and the *Web MIDI Monitor* was specifically designed for such an extension, the generation and monitoring of MIDI 2.0 messages deserve particular attention. To this end, we have installed and configured a couple of MIDI 2.0 software products designed to work together, namely the *Bome MIDI-CI Initiator* and the *Bome MIDI-CI Responder* (see Figure 3). In this way, we have simulated a typical MIDI 2.0 information exchange, made of:

- a *Discovery* inquiry by the initiator;
- a *Reply to Discovery* inquiry by the responder;
- an inquiry of *Property Data Exchange Capabilities* by the initiator;
- the *Reply to Property Data Exchange Capabilities* by the responder;
- a number of *Get Property Data* inquiries by the initiator and the corresponding *Reply to Get Property Data* messages by the responder.

The interface of the *Web MIDI Monitor* (see Figure 4) shows the alternating messages sent by the two devices on their output ports. Each port is characterized by a different color, shown on the left side of the message. The structure of each message is interpreted and clearly presented to the user. An example is provided by the 4+4 bytes dedicated to the sender's and receiver's MUIDs.

4. DISCUSSION

In this section, we will critically analyze the strengths and weaknesses of the *Web MIDI Monitor*.

The main advantage of implementing a MIDI monitor as a web tool is the ease of embedding it into a (potentially complex) MIDI system. MIDI communication does not strictly require the 5-pin DIN connectors standardized by the MIDI 1.0 specification, which are still present in most MIDI musical instruments but are available on standard PCs only through ad hoc interfaces (e.g., joystick ports on old audio cards and external audio peripherals). Luckily, current devices can receive and send MIDI data using recent standard ports and formats through MIDI over USB, Bluetooth, and Ethernet.

Another benefit of a web-based solution is the intrinsic extensibility, with updates made available to users on a rolling base. In the case of MIDI 2.0, which is a quite recent standard extension, the possibility to refine message visualization and management is fundamental, also depending on the needs of developers and the release of new products. For example, the only keyboard controller on the market that claims to be "MIDI 2.0 ready", namely the Roland A-88MKII MIDI Keyboard Controller, never mentions MIDI 2.0 in its operating manual. In the climate of uncertainty that characterizes a moment in which the standard is not yet considered mature by the market, the possibility of quickly updating the MIDI monitor is a potential success factor.

The Web MIDI API is a commonly accepted JavaScript interface under development in the W3C context. On one side, this constitutes a clear advantage, since the management of MIDI messages and the connection with external devices can occur at a higher level of abstraction. On the other side, there are some potential drawbacks that are worth remarking on. First, the client computer has to be equipped with one of the compatible browsers, even if the support of this API is getting more and more extensive. Secondly, the management of some features typical of MIDI communication is delegated to the design choices of the Web MIDI API. Examples include how the running status and the occurrence of real-time messages in the middle of other messages are handled. Finally, the communication delay introduced by putting a MIDI monitor in the middle of a MIDI chain is a minor problem in educational and offline debugging activities, but it could impact, e.g., a live-performance scenario. A deeper insight into the Web MIDI API would be beyond the scope of this paper. For a critical discussion about its pros and cons, please refer to [15].

5. CONCLUSIONS

In this paper, we presented a browser app based on the Web MIDI API to monitor MIDI messages. It is worth noting that MIDI-CI messages are fully backward compatible thanks to the adoption of MIDI 1.0 *SysEx* messages; consequently, any MIDI monitor, even if developed before the release of the MIDI 2.0 extension, would be able to provide monitoring for MIDI-CI messages. Nevertheless, in our opinion, it is important to have a free support tool capable to guide the user in understanding the standard parts of these messages. To this end, the proposed *Web MIDI Monitor* is made publicly available to users via the web.

Possible use modes include technical applications, such as the debugging of MIDI communication in a scenario that embeds both MIDI 1.0 and MIDI 2.0 devices, and educational activities, allowing a better comprehension of

MIDI-CI message exchange.

6. REFERENCES

[1] IMA, *MIDI 1.0 Specification*. Sun Valley, CA: International MIDI Association (IMA), August 1983.

[2] MMA, *The Complete MIDI 1.0 Detailed Specification*. Los Angeles, CA: MIDI Manufacturers Association (MMA), 1996.

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

[4] M. Cook, "Basic MIDI," *Arduino Music and Audio Projects*, pp. 31–47, 2015.

[5] S. Al-Fedaghi, "Developing web applications," *International journal of software engineering and its applications*, vol. 5, no. 2, pp. 57–68, 2011.

[6] C. Wilson and J. Kalliokoski, "Web MIDI API W3C editor's draft," https://webaudio.github.io/web-midi-api/.

[7] M. Walczak and E. Łukasik, "Generowanie, edycja i transmisja wieloźródłowych 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, 2020.

[8] C. Gurtner, "Applications of audio and MIDI API within a music notation editor," in *Web Audio Conference WAC–2016, April 4-6, 2016, Atlanta, USA*. Georgia Institute of Technology, 2016.

[9] T. Bazin and G. Hadjeres, "Nonoto: A model-agnostic web interface for interactive music composition by inpainting," in *10th International Conference on Computational Creativity (ICCC 2019), UNC Charlotte, North Carolina*, 2019.

[10] S. Kachalo, "JZZ.js – a unified API for MIDI applications," in *Web Audio Conference WAC–2016, April 4-6, 2016, Atlanta, USA*. Georgia Institute of Technology, 2016.

[11] J.-P. Côté, "User-Friendly MIDI in the Web Browser," in *NIME 2022*, jun 16 2022, https://nime.pubpub.org/pub/user-friendly-midi-in-the-web-browser.

[12] 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 International Web Audio Conference*, 2019, pp. 59–64.

[13] 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 2017)*, L. A. Ludovico and A. M. F. Yousef, Eds. IARIA XPS, 2017, pp. 72–77.

[14] 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*, 2019, pp. 227–230.

[15] A. Baratè and L. A. Ludovico, "Web MIDI API: State of the art and future perspectives," *Journal of the Audio Engineering Society*, vol. 70, no. 11, pp. 918–925, 2022.

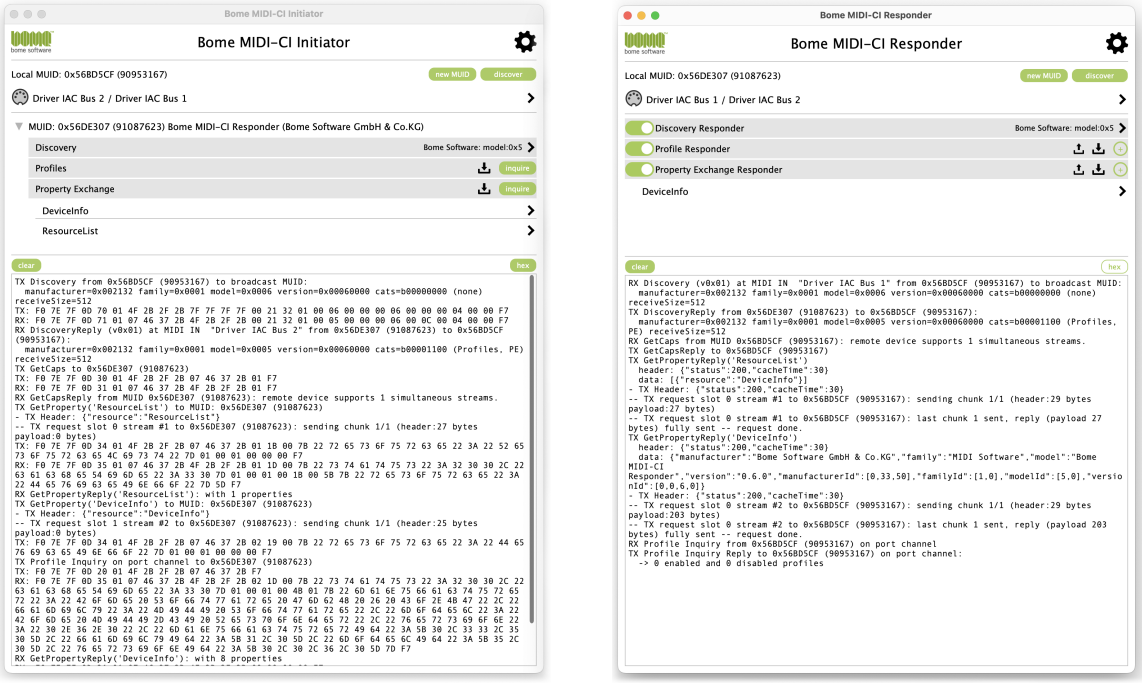

Figure 3. The *Bome MIDI-CI Initiator* and the *Bome MIDI-CI Responder*.

Figure 4. Monitoring MIDI 2.0 message exchange through the *Web MIDI Monitor*.