The Web MIDI API in On-Line Applications for Music Education

Luca A. Ludovico

Laboratorio di Informatica Musicale
Dipartimento di Informatica
Università degli Studi di Milano
Email: luca.ludovico@unimi.it

Abstract—**This work discusses a recent browser technology known as the Web MIDI API to design and release browser applications for music. This application programming interface (API), still under development by the Audio group of the World Wide Web Consortium (W3C), provides support to the Musical Instrument Digital Interface (MIDI) protocol, enabling web applications to interface with MIDI input and output devices on the client system and send and receive MIDI messages. The paper critically analyzes advantages and drawbacks currently presented by the Web MIDI API in the context of web-based music education. In the final part of this work, a case study will be discussed.**

Keywords–MIDI; Web applications; Music education; Web MIDI API.

I. Introduction

In computing, a *web application* – or *web app* – is a client-server software application in which the client or user interface runs in a web browser. Web applications are very popular, since they offer a number of advantages: they are cross-platform, easy to update, often free, and they should not require ad hoc installations. The distribution of web applications is immediate and their audience potentially includes all users owning a network-attached device equipped with a web browser. These characteristics are very important for educational applications too. For instance, portability and compatibility are at the base of the *Bring Your Own Device* (BYOD) paradigm, whose advantages, risks and perspectives in the educational field have been discussed in [1], [2], [3]. As stated in [4], the most important quality criteria for the success of web applications include reliability, usability, security, availability, scalability, maintainability, and time-to-market. Most of these aspects must be considered also in the design and implementation of applications for students.

The Musical Interface for Digital Instruments (MIDI) is a well-known and widely-adopted protocol to exchange messages among compatible music devices. Even if nowadays MIDI could seem out of date and naive to non-experts – suffice it to recall that the maximum bitrate of the protocol is 31.25 kbps and the structure of messages is composed by 7-bit packages – it is still largely supported by professional music and audio equipment. Some recent initiatives are bringing new life to the format. In this sense, it is worth citing: the recent reorganization of the official web site [5], with updated contents, a new graphical layout and a *News* section; the release on the marketplace of innovative music controllers, such as the *ROLI Seaboard Grand*, able to extend the potential of the original MIDI protocol still preserving full compatibility [6]; and finally, the establishment of interest groups which join academia and industry in order to apply MIDI to new contexts and meet recent technological requirements. In this light, the World Wide Web Consortium (W3C) launched an initiative aiming to design and implement the so-called Web MIDI API. This specification defines an Application Programming Interface (API) supporting the MIDI protocol, enabling web applications to enumerate and select MIDI input and output devices on the client system and send and receive MIDI messages [7]. As stated in the official W3C page, the Web MIDI API is intended to enable non-music MIDI applications as well as music ones, by providing low-level access to the MIDI devices available on the users' systems. Further details will be provided in Section IV.

The goal of this paper is to match the advantages of web applications in the music education field with the flexibility and power of the MIDI protocol for music reproduction. The point of intersection of these two worlds lies in the technology known as the Web MIDI API. The early development stage of the API, the limited support currently offered by browsers and the consequent lack of available applications make this promising field relatively novel.

The paper is structured as follows: Section II will discuss web resources for music teaching and learning; Section III will provide an overview of the MIDI protocol and its applicability to music education; Section IV will give details about the Web MIDI API initiative; Section V will present a critical analysis of this approach concerning the design, development and release of web applications; finally, Section VI will discuss the case study of an on-line application for music coding.

II. Web Resources for Music Teaching and Learning

Computer-based technologies can help music education from many points of view, ranging from teaching strategies to new approaches to composition and performance, from assistive technology and music therapy to K-12 listening skills development [8]. In recent times, the pervasiveness of network technologies and the availability of high-speed connections have resulted in a rise of meaningful Internet-based music resources. Just to mention some examples:

- The official web sites of important institutions – see the digitalization project of the *Bach Archiv Leipzig* [9], the music section of the *Beic Digital Library* [10], and the archive of the *Teatro alla Scala* [11], to name but a few – nowadays offer high-quality materials to Internet users. The mentioned examples cover heterogeneous aspects of music, including scores [12], audio recordings [13], and heterogeneous opera-related contents [14] respectively. These materials are of great interest for the community of experts, scholars and music students, but they cannot be easily reused in an educational context, due to both technical limitations and copyright issues;

- There are collections of downloadable music resources, often based on the efforts of community members. In this light, two relevant examples are the *Internet Archive*, a site that retrieves digitized content from the web including music scores and audio files [15], and the *IMSLP/Petrucci Music Library* [16], a popular platform to share public domain music [17];
- Other on-line initiatives providing non-interactive resources for music education and dissemination, including static web pages, video tutorials, etc. For example, the YouTube official channel of the Philharmonia Orchestra – London (UK) collects in the *Instrument Guides* section a number of videos aiming to present musical instruments to young people.

Many studies discussed the integration of web-based material into music teaching, documenting phases of integration that include supplemental links to resources, web-based teaching sequences, and various media to support course content [18]. All the mentioned initiatives are potentially useful for music teaching and learning, nevertheless, in order to achieve effective educational results some additional features would be desirable. The ideal web interface should not only grant high-quality and certified resources, achieve cross-platform compatibility, and be always available, but it should also support an active and customizable experience of music contents, present multi-modal interactivity, and foster peer-to-peer and student-teacher interactions.

Keeping these goals in mind, we can provide a non-exhaustive state of the art on recent on-line initiatives that embody such a vision. An example of web platform containing high-quality materials for music education is *DREAM – Digital Resource Exchange About Music* [19], a virtual space for exchanging information about digital learning tools [20]. Another relevant example is *EMIPIU – Enhanced Music Interactive Platform for Internet Users* [21], a web environment that adopts the IEEE 1599 standard to encode music in all its aspects according to a multi-layer structure and presents an advanced web player to enjoy such contents in a synchronized way [22]. As an evidence of its pedagogical valence, the same technology has been employed in the on-line version of a music textbook for children published by Pearson [23]. Finally, let us mention the *Chrome Music Lab* [24] a set of simple on-line tools that let anyone explore how music works. This initiative is a collaboration between musicians and coders, and its core technology lies in the freely available *Web Audio API* [25].

In our opinion, a more detailed survey about web applications currently available for music education would go beyond the scope of this work.

III. MIDI AND MUSIC EDUCATION

The *Musical Instrument Digital Interface* (MIDI) is an industry standard music technology protocol that connects products from many different companies including digital musical instruments, computers, tablets and smartphones. The MIDI specification describes the protocol, the digital interface and low-level hardware aspects, such as ports, cables and connectors.

The original protocol was designed and released in the 1980's. Despite significant enhancements over the years, the MIDI specification officially remains at version 1.0 [26]. Later extensions include the Standard MIDI File format, MIDI Show Control, MIDI Time Code, and MIDI Machine Control. Nowadays the activities of the MIDI Manufacturers Association (MMA) are focusing on new transfer protocols, such as MIDI over USB and over wireless.

MIDI is used everyday around the world by musicians, DJs, producers, educators, artists and hobbyists to create, perform, learn and share music and artistic works. Advanced and innovative interfaces to make music through MIDI are constantly under development, as shown during the Annual General Meeting of the MMA occurred at the 2016 Winter NAMM Show. The importance of MIDI as a commonly-accepted and widely-adopted standard is also demonstrated by the libraries available for the main programming languages, such as the *C# MIDI Toolkit* for C#, the package *javax.sound.midi* for Java or the *MIDI Toolbox* for Matlab [27].

The MIDI data stream is a unidirectional asynchronous bit stream at 31.25 kbps. The interface on a MIDI instrument will generally include three different connectors, labeled MIDI IN, MIDI OUT, and MIDI THRU. The data stream is usually originated by a MIDI controller or by a MIDI sequencer. A *MIDI controller* is a device which is played as an instrument, and it translates the performance into a MIDI data stream in real time as it is played. Examples include not only keyboards, but also electronic drums, wind controllers and guitar-like MIDI devices. A *MIDI sequencer* is a device which allows MIDI data sequences to be captured, stored, edited, combined, and re-played. The MIDI data output from a MIDI controller or sequencer is transmitted via the devices' MIDI OUT connector. The final recipient of the data stream is commonly a MIDI sound generator or sound module, which will receive MIDI messages at its MIDI IN connector, and respond to these messages by playing sounds. Further information can be retrieved from the official documentation.

It is worth underlining that MIDI does not transmit audio signals; instead, it sends event messages about musical notes, controller signals for parameters, such as volume, vibrato and panning, cues and clock signals to set the tempo, and system-specific MIDI communications. In other terms, MIDI itself does not make sound, rather it encodes the exchange of messages generated by a MIDI chain and to be interpreted by synthesizers in order to produce sound. The pros and cons of this key aspect of MIDI will be discussed in Section V.

A MIDI device can be a piece of hardware (controllers, synthesizers, etc.), a virtual/software tool, or a part of a software environment. From a logical point of view, the simplest MIDI chain presents a message generator – like a keyboard controller – and a message consumer – like a synthesizer. There are devices that integrate both functions in a unique product, as shown in Figure 1. Obviously, more complex layouts can be created through the concatenation of multiple devices, e.g., controllers, sequencers and sound modules, as shown in Figure 2.

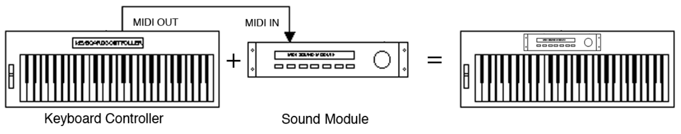


Figure 1. An example of minimal MIDI chain: a keyboard controller connected to a sound module.

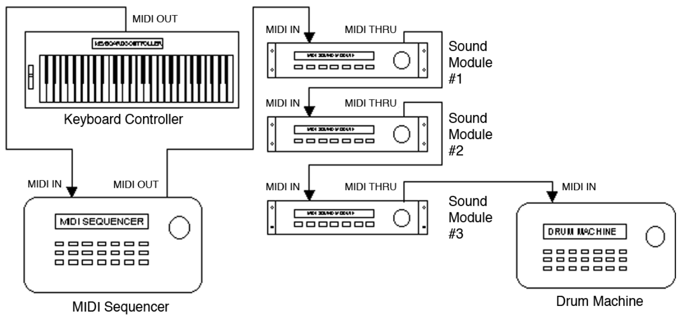


Figure 2. An example of non-trivial MIDI setup in daisy-chain configuration.

Thanks to the huge support offered by most keyboard controllers and to the availability of software counterparts for physical equipment, MIDI can be easily adopted in any educational environment. For instance, in the proposals detailed below, a network-attached computer will be sufficient to experience MIDI-based activities oriented to music education. According to some experts, MIDI can even bring a paradigm shift in music teaching and learning (see [28] and [29]). Among a number of didactic applications of the protocol, it is worth mentioning:

- the production of easy, low-cost, on-the-fly audio renderings of potentially complex music pieces, in contexts of instrument performance [30], vocal training [31], and assisted composition [32];
- the accompaniment for music lessons and individual instrumental practice [33];
- the support offered to real-time distributed music performances, mainly due to the lightweight exchange of MIDI messages, as documented in [34] and [35];
- new possibilities of music expression and interaction for people affected by various kinds of disability, thanks to ad-hoc MIDI-compatible controllers. A relevant application of MIDI to therapy and special education can be found in [36], which presents the use of MIDI devices in order to enable students with physical and learning disabilities to participate to a festival of popular music.

For a more detailed discussion of the advantages and drawbacks of MIDI in music education, please refer to [37] and [38].

IV. THE WEB MIDI API

In Section II we have cited a number of heterogeneous web applications for music education, and in Section III we have explored the applicability of MIDI-based approaches to this field. Now the question is how to couple the advantages of on-line solutions with the potential of MIDI, and the answer lies in the Web MIDI API [7].

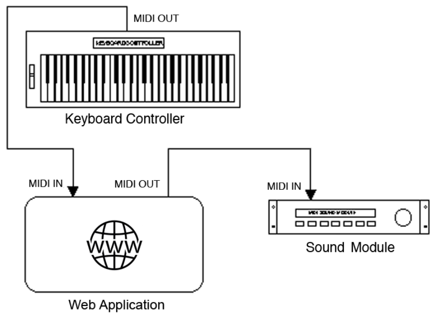


Figure 3. An example of MIDI chain formed by a web application that is piloted by a keyboard and controls a sound module.

This specification defines an API to support the MIDI protocol within web applications, so that applications can enumerate and select MIDI input and output devices on the client system and send and receive MIDI messages. In accordance with the MIDI philosophy, also the API is not designed to describe music or controller inputs semantically; rather, it is intended to reproduce the mechanisms of MIDI input and output interfaces, enabling direct access to devices that respond to MIDI (e.g., controllers, synthesizers, lighting equipment, other pieces of software, etc.). For the sake of clarity, please note that the Web MIDI API is not a way to play Standard MIDI files (SMFs) in a browser; this function should be performed by a future extension of the HTML5 `<audio>` tag instead.

The Web MIDI API puts the web application in communication with other parts of the physical or virtual MIDI chain by sending and receiving standard MIDI messages. In this way, the web application becomes a new MIDI-compatible actor connected to the MIDI chain, as intuitively shown in Figure 3. The API aims to enable a brand new class of applications on the web that can respond to MIDI controller inputs, even with no music purposes. Examples may range from MIDI-controlled video games to interfaces for the music expression by users with disabilities. All these approaches can result in the design and implementation of interactive browser-based educational products.

From a practical point of view, adopting the Web MIDI API in a web page requires to embed ad-hoc JavaScript code. The first steps is searching for MIDI available resources on the client system by invoking the `requestMIDIAccess()` method. Then, it is possible to select the input/output devices to be connected, if any. For an application that has to produce sound, this implies to pick at least the default MIDI synthesizer. Finally, the application can listen to MIDI messages as well as user actions in input, process them and finally send MIDI messages in output. The `send(data,timestamp)` method of the `MIDIOutput` interface enqueues the message(s) to be sent to the corresponding MIDI port, provided that the `data` parameter contains one or more valid and complete MIDI messages. It is also possible to specify when the data should be sent to the port; if the `timestamp` parameter is not present, is set to zero or to a time in the past, data are to be sent as soon as possible.

V. Critical Analysis

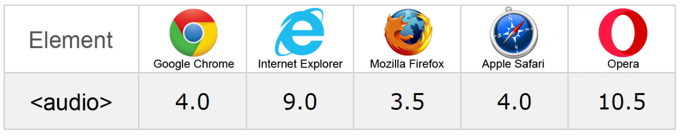
Before HTML5, the performance of audio files in a web page could occur only through a plug-in – e.g., Macromedia Flash or Windows Media Player – or non-standard HTML syntax such as `<bgsound>`, an Internet Explorer element associating a background sound with a page. These custom solutions caused incompatibilities between implementations and across different browsers, implying different behaviors of the same page when visited by different users and unwanted installations of third-party plug-ins.

The introduction of the `<audio>` element in HTML5 specification solved these issues. The idea was to provide web designers and programmers with a standard way to embed into web pages audio in MP3, Wav, and Ogg format. Nowadays, the `<audio>` element is largely supported by web browsers, as shown in Figure 4.

Consequently, at present there are standard approaches to sonify web pages through background music and to play audio content on demand. Nevertheless, in some cases a solution based on pre-recorded sound files is neither efficient nor effective. Let us consider those applications where audio is produced in response to a specific user action, and sound parameters (pitch, loudness, duration, timbre, etc.) should change accordingly. For the sake of clarity, the sonification of a button click can be easily implemented by loading a default sound file, whereas a web interface to explore multiple music-scale models performed by different instruments would require a considerable number of recordings; and the situation would become even more critical in the case of an interactive application (e.g., a touch-sensitive keyboard with multi-timbral synthesis) where user actions are unpredictable. In those music-oriented applications where sound parameters may change considerably and are difficult to predict since they depend on users' choices and behaviors, MIDI can play an important role.

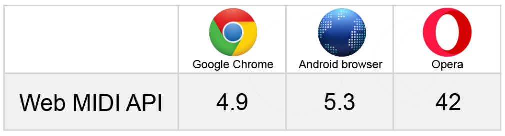
Among the advantages offered by a MIDI-based approach, we can mention:

- A production of sound samples completely demanded to a MIDI synthesizer, with no need to pre-record audio fragments with all the variants required (e.g., different pitches, durations, timbres, etc). This implies also the possibility to use the full range of General MIDI patches for multi-timbral synthesizers, and to modify sounds through all the supported Control Change commands;
- A very light and compact client-server exchange of music data, since MIDI does not transmit audio signals, rather it sends commands to trigger audio events;
- The possibility to embed web applications into MIDI chains, by connecting them with other MIDI-compatible real (hardware) or virtual (software) devices;
- Provided a basic knowledge of the MIDI protocol, the easiness of use by programmers, even if the adoption of the HTML5 `<audio>` element is very straightforward as well.



Element	Google Chrome	Internet Explorer	Mozilla Firefox	Apple Safari	Opera
<audio>	4.0	9.0	3.5	4.0	10.5

Figure 4. The main browsers supporting the HTML5 `<audio>` element. The numbers below specify the first browser version that offers full support.



	Google Chrome	Android browser	Opera
Web MIDI API	4.9	5.3	42

Figure 5. Browsers supporting the Web MIDI API. The numbers below specify the first browser version that offers full support.

Of course, there are also some annoying drawbacks to take into account:

- MIDI is not an audio format, consequently the resulting sound quality on different systems is unpredictable since it largely depends on the characteristics of the sound module in use;
- MIDI is not a music-notation format, so it can be employed to encode scores only in non-professional applications or in simplified contexts, such as most children-oriented music games;
- The Web MIDI API requires a MIDI chain to produce sounds, namely the presence of either a hardware or a software synthesizer;
- The Web MIDI API has not reached the status of standard yet, and at the moment of writing it is supported only by the browsers listed in Figure 5.

The first two drawbacks listed above are connected to the choice of MIDI as a way to represent music symbols and produce sounds; conversely, the last two are directly linked to the current characteristics of the Web MIDI API. Experience shows that the success of web applications resides in multi-platform support, cross-browser compatibility and no need to perform ad hoc installations, whereas at present the Web MIDI API seems to go the opposite way: it requires the adoption of a specific subset of browsers and the availability of physical MIDI devices attached to the client – an uncommon situation for most users – or, more likely, the installation of software emulators.

Some of these problems will be hopefully solved when the API becomes a standard. In the meanwhile, to get round the problem it is possible to distribute applications wrapped in ready-to-use software packages that embed all the required installations, resources, and configurations. For example, Docker [39] is an open-source project wrapping up in a so-called *container* all the resources needed in order to run one or more processes. This isolation concerns both hardware (CPU, memory, file system, etc.) and software (libraries, tools, code, and so on). After the public deployment, the package launched on a client behaves like a virtual machine.

VI. Case Study: a Web Application for Music Coding

In order to foster artistic creativity and analytical skills in young students, we designed, implemented and released a publicly-available web environment for *music coding*. Music

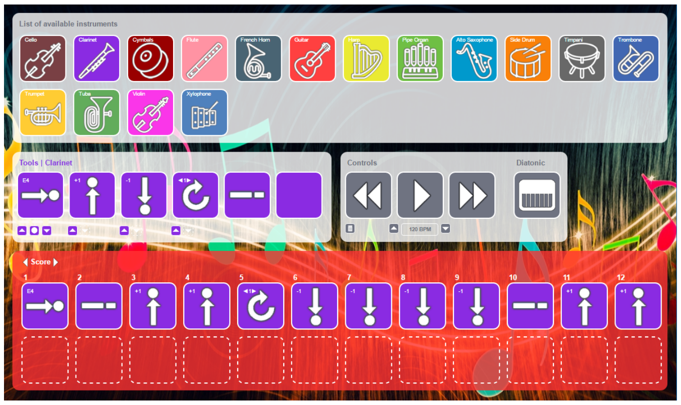


Figure 6. A web interface for music coding where music is generated through the Web MIDI API.

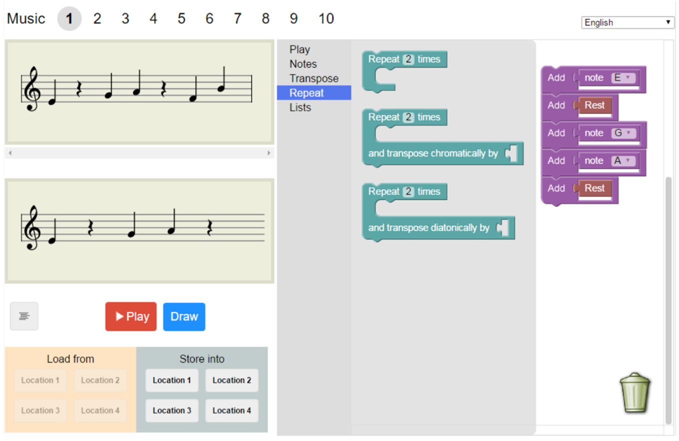


Figure 7. An evolution of the web application for music coding, based on Google Blockly and deployed as a Docker package.

coding can be seen as an evolution of *creative coding* [40], where the latter is applied to music education and its goals embrace not only personal creativity, but also the analysis and comprehension of music processes.

The web environment – to this end – proposes a simplified subset of music operators involving mainly rhythmic and melodic aspects of music. Examples include `Play(p)`, i.e., “Play a given pitch”, `Transpose(v)`, i.e., “Modify the previous pitch according to a number of ascending/descending steps”, and `Tie()`, i.e., “Extend the duration of a note”.

In addition to a short list of music operators, the interface also presents a palette of musical instruments to choose from. The reasons that led to multi-timbral support are manifold: making music creation and reproduction more engaging, supporting the study of harmony and the listening of polyphony through different timbres, introducing young students to the characteristics of orchestral instruments, etc.

Due to the young age of expected users, music operators and musical instruments are represented through playful and colored icons, as shown in Figure 6.

This is not the place to provide further details about the user interface, nor to discuss the validity of its pedagogical approach. For those interested in deepening these topics, a detailed discussion can be found in [41] and [42]. Rather, in this context we are interested in the way music is produced in response to user choices: in fact, sound performance has been demanded to the Web MIDI API.

The choice of the Web MIDI API against other standard solutions, such as sampled sounds, is motivated by a number of factors. First, the multi-timbral support combined with the possibility to play a wide range of pitches would have required the generation of approximatively 2000 sound samples. MIDI syntax easily solves this issue thanks to:

- the parametric values of Note On events, which make pitch (and even velocity) easy to set;
- the use of different MIDI channels properly configured through Program Change messages in order to support different timbres.

Furthermore, the MIDI numeric representation of music parameters allows to easily compute a music tune coming from an algorithmic process. Let us recall that the interface has been conceived to foster music coding, thus notes are not necessarily expressed in an imperative way (e.g., “Play C4”), but they can result from the application of a number of music operators (e.g., “Play C4”, “Transpose a major third up and play”, “Repeat the last note”). In this sense, the MIDI approach intrinsically simplifies the computation of music parameters that drive sound performance.

Such a web application for music coding is publicly available at http://coding.lim.di.unimi.it. Please note that the Web MIDI API requires a compatible browser – e.g., Google Chrome – and the connection to a MIDI sound module to work.

Recently, we made the interface evolve releasing a new version in form of a Google Blockly game. Blockly [43] is a library for building visual programming editors which uses interlocking, graphical blocks to represent code concepts like variables, logical expressions, loops, and more. In our context, blocks have been linked to music concepts, e.g., music operators and basic score elements, as shown in Figure 7. For the reasons above, also this application adopts the Web MIDI API as a base for the sound engine. In order to solve the cross-platform and installation issues mentioned in Section V, we decided to make the Blockly-based implementation available to schools and other interested institutions as a Docker package.

VII. Conclusion

Thanks to its well-known characteristics, the web can be a great means to design, implement and distribute valid and engaging educational tools, and the field of music is no exception. In recent years we have seen the release of a large number of web applications focusing on music, including playful approaches to learn a musical instrument, professional ear-training apps, on-line viewers/players for music archives, and so on. Within this multi-faceted scenario, MIDI – a format dating back to the 80’s but still widely adopted by the community of professional musicians – has shown signs of great vitality.

In this paper we have analyzed the applications of MIDI to web frameworks for music-education, an intersection made possible by the Web MIDI API. At the moment of writing, this W3C initiative is still under development, and the support offered by web browsers is very limited. Nevertheless, we consider the Web MIDI API very promising, and many implications in education – e.g., by interfacing the web app with external MIDI controllers – have yet to be explored.

References

[1] K. W. Miller, J. Voas, and G. F. Hurlburt, "BYOD: Security and privacy considerations," It Professional, vol. 14, no. 5, 2012, pp. 53–55.

[2] R. Afreen, "Bring your own device (BYOD) in higher education: opportunities and challenges," International Journal of Emerging Trends & Technology in Computer Science, vol. 3, no. 1, 2014, pp. 233–236.

[3] P. Bruder, "Gadgets go to school: The benefits and risks of BYOD (bring your own device)," The Education Digest, vol. 80, no. 3, 2014, p. 15.

[4] J. Offutt, "Quality attributes of web software applications," IEEE software, vol. 19, no. 2, 2002, pp. 25–32.

[5] "MIDI.org," http://www.midi.org/, accessed: 2017-02-22.

[6] J. Flanagan, "Sensitive piano keys let pianists create new sounds," New Scientist, vol. 219, no. 2925, 2013, p. 22.

[7] C. Wilson and J. Kalliokoski, "Web MIDI API," W3C, working draft, Dec. 2016, https://webaudio.github.io/web-midi-api/.

[8] P. R. Webster, Computer-based Technology and Music Teaching and Learning: 2000–2005. Dordrecht: Springer Netherlands, 2007, pp. 1311–1330.

[9] "Bach digital," https://www.bach-digital.de/, accessed: 2017-02-22.

[10] "BEIC - Biblioteca musicale," http://www.beic.it/it/articoli/biblioteca-musicale/, accessed: 2017-02-22.

[11] "Archivio La Scala," http://www.teatroallascala.org/archivio/, accessed: 2017-02-22.

[12] U. Wolf, "Bach-autographen online. kooperationsprojekt bach-digital angelaufen urz liefert kompetenz und rechenpower," Universität Leipzig, Journal, vol. 4/2008, 2008, p. 24.

[13] A. Baratè, L. A. Ludovico, and G. Haus, "Integration of audio resources into a digital library: The BEIC case study," International Journal of Digital Curation, vol. 10, no. 2, 2015, pp. 48–57.

[14] G. Haus and L. A. Ludovico, "The digital opera house: an architecture for multimedia databases," Journal of Cultural Heritage, vol. 7, no. 2, 2006, pp. 92–97.

[15] "Internet Archive: Digital Library of Free Books, Movies, Music & Wayback Machine," https://archive.org/, accessed: 2017-02-22.

[16] "IMSLP/Petrucci Music Library: Free Public Domain Sheet Music," http://imslp.org/, accessed: 2017-02-22.

[17] P. M. Hash, "Internet resources for historical research in music education," Journal of Historical Research in Music Education, vol. 31, no. 1, 2009, pp. 3–5.

[18] N. Barry, "Integrating web based learning and instruction into a graduate music education research course: An exploratory study," Journal of technology in Music Learning, vol. 2, no. 1, 2003, pp. 2–8.

[19] "DREAM — Digital Resource Exchange About Music," http://dreammusictool.ca/, accessed: 2017-02-22.

[20] R. Upitis, K. Boese, and P. C. Abrami, "Demonstrating DREAM: A digital resource exchange about music." International Association for Development of the Information Society, 2015.

[21] "EMIPIU," http://emipiu.di.unimi.it/, accessed: 2017-02-22.

[22] A. Baratè, G. Haus, L. A. Ludovico, and G. Presti, "Advances and perspectives in web technologies for music representation," DigitCult - Scientific Journal on Digital Cultures, vol. 1, no. 2, 2016, pp. 1–8.

[23] S. Erotoli and V. Vacchi, C'è musica per tutti. Vol. A-B. Pearson, 2014.

[24] "Chrome Music Lab," https://musiclab.chromeexperiments.com/, accessed: 2017-02-22.

[25] P. Adenot, C. Wilson, and C. Rogers, "Web Audio API," W3C, working draft, Dec. 2015, https://www.w3.org/TR/webaudio/, accessed: 2017-02-22.

[26] M. M. Association, "The complete MIDI 1.0 detailed specification version 96.1," La Habra: MIDI Manufacturers Association, 1996.

[27] T. Eerola and P. Toiviainen, MIDI Toolbox: MATLAB Tools for Music Research. Jyväskylä, Finland: University of Jyväskylä, 2004. [Online]. Available: www.jyu.fi/musica/miditoolbox/

[28] G. Jordahl, "Teaching music in the age of MIDI," Classroom Computer Learning, vol. 9, no. 2, 1988, pp. 78–85.

[29] J. Zhu, "MIDI and music teaching in colleges of multimedia system application," in Key Engineering Materials, vol. 474. Trans Tech Publ, 2011, pp. 1926–1930.

[30] M. Ajero, The effects of computer-assisted keyboard technology and MIDI accompaniments on group piano students' performance accuracy and attitudes. ProQuest, 2007.

[31] C. Yun and W. Fu, "On the feasibility of employing MIDI in vocal music teaching," Journal of Mianyang Normal University, vol. 4, 2010, p. 038.

[32] S. Reese, "MIDI-assisted composing in your classroom," Teaching music, 1995, pp. 199–206.

[33] F. Kersten, "Using MIDI accompaniments for music learning at school and at home: MIDI and other computer technologies can help students build their musical skills in the classroom and at home," Music Educators Journal, vol. 90, no. 4, 2004, pp. 44–50.

[34] D. Gang, G. V. Chockler, T. Anker, A. Kremer, and T. Winkler, "TRANSmidi: a system for MIDI sessions over the network using transis," in Proceedings of the International Computer Music Conference. Citeseer, 1997, pp. 283–286.

[35] D. Akoumianakis, G. Vellis, I. Milolidakis, D. Kotsalis, and C. Alexandraki, "Distributed collective practices in collaborative music performance," in Proceedings of the 3rd International Conference on Digital Interactive Media in Entertainment and Arts, ser. DIMEA '08. New York, NY, USA: ACM, 2008, pp. 368–375. [Online]. Available: http://doi.acm.org/10.1145/1413634.1413700

[36] B. Cole, "MIDI and communality," Organised Sound, vol. 1, no. 01, 1996, pp. 51–54.

[37] T. E. Rudolph, Teaching music with technology. GIA Publications, 2004.

[38] X. Guo, "The application and research of computer MIDI technology in music education," in International Conference on Social Science and Technology Education (ICSSTE 2015), 2015, pp. 236–241.

[39] "Docker," https://www.docker.com/, accessed: 2017-02-22.

[40] K. Peppler and Y. Kafai, "Creative coding: Programming for personal expression," Retrieved August, vol. 30, no. 2008, 2005, p. 314.

[41] L. A. Ludovico and G. R. Mangione, "Music coding in primary school," in Smart Education and Smart e-Learning, ser. Smart Innovation, Systems and Technologies, R. J. Howlett, L. C. Jain, and V. Uskov, Eds. Springer International Publishing, 2015, pp. 449–458.

[42] A. Baratè, L. A. Ludovico, and G. R. Mangione, "A web framework to develop computational thinking through music coding," in Proceedings of the 2nd International Conference on New Music Concepts (ICNMC 2016), M. Della Ventura, Ed. Milano, Italy: ABEditore, 2016, pp. 157–167.

[43] "Google Blockly," https://developers.google.com/blockly/, accessed: 2017-02-22.