Plug-and-play MIDI controller for expressive chord playing

Emil Sønderskov Hansen
Aalborg University CPH
esha19@student.aau.dk

Simon Andersen
Aalborg University CPH
sand22@student.aau.dk

ABSTRACT

For an amateur music producer, there is a significant learning curve to producing music with chords. To add richness and harmony to music, the producer needs extensive knowledge of music theory. Several products have been made to reduce this learning curve, including software and midi controllers. However, most solutions have many other features and lack ways for the producer to customize the chords intuitively. Therefore, the producer may not be able to express themselves in the way they want. This paper presents a plug-and-play MIDI controller to trigger chords in a desired scale easily. Furthermore, parameters for expressiveness have been added, such as strumming, random velocity, inversions, etc. The final prototype does have several shortcomings, and further implementation is needed before it is ready for the final evaluation. When ready, it should be assessed whether the controller intuitively enables amateur producers to express themselves through chords.

1. MOTIVATION

The beginning stages of music production require much learning as a newcomer with little experience in the field. One way of starting could be messing around with virtual instruments in a piano roll in their digital audio workstation (DAW). A way of increasing the richness and bringing harmony to a musical piece would be the addition of chords [1]. But beginners and amateurs might not have obtained the knowledge of music theory to sketch these chord structures or progressions through MIDI, let alone a keyboard.

Others have tried solving this issue by adding software in DAWs, MIDI instruments, or -controllers. In many cases, these implementations solve the issue differently by refining the user experience. E.g., the ability to group the matching chords together and assign the chords to a specific button. If a user does not have music theory experience, how would they be able to play chords that match a certain scale? On top of that, other implementations do not have musical expression components, like adding inversions, velocity, and strumming. Additionally, users should be able to customize their chords to make them unique and add their expressiveness to their music[2]. Many commercially available controllers that are solving this issue are also equipped with many other features. The user should learn how to interact with the device, which might cause *technology overload*[3] because there are so many features, knobs, and buttons available. A beginner to midi-controllers might have a tough time learning when there is a lot of unknown functionality in front of them.

Hence, we want to create a plug-and-play MIDI controller that provides the previously mentioned aspects. The user should be able to connect the controller to their favorite DAW and start playing and customizing chords right away.

2. RELATED WORK

This section introduces related work that has been released commercially and open-source.

2.1 Maschine MK3

The Machine MK3 [1] is a commercialized MIDI controller using silicon pads and knobs that provide a lot of functionality for producer enthusiasts. The device contains a feature called 'Chord Mode' where users can play chords in a set scale controlled by the available 'Chord Sets'[4]. Included in this feature is 'Note Repeat mode,' which is an arpeggio mode that plays the selected chords in the sequence where the user can modify the speed of the repetition[4]. Another component available is 'variation,' which can be applied after a chord progression has been recorded and will randomize the velocity in the MIDI information[5].

2.2 Ableton Push

The Ableton Push [2] is a commercialized midi-controller using silicon pads and knobs. This has a chord mode incorporated where the user can choose a specific scale, and the device highlights the notes that can be played in the scale [6]. The user then needs to know which notes match together in order to i.e. play a C major chord [7]. This device also includes a step sequencer that can arpeggiate the chords that are recorded [6].

2.3 Mini UNTZtrument

The Mini UNTZtrument [3] is a non-commercialized project published by the Ruiz Brothers on the Adafruit [4] website. In this project, they created a small drum pad controller

[1] https://www.native-instruments.com/en/products/maschine/production-systems/maschine/
[2] https://www.ableton.com/en/push/
[3] https://learn.adafruit.com/mini-untztrument-3d-printed-midi-controller
[4] https://www.adafruit.com/

that uses MIDI information to play notes [8]. It included four adjustable knobs that control modulation, expression, and filter cut-off [8]. Using this controller, the wanted drum sounds have to be assigned to the MIDI input and can be used with a virtual instrument [9].

3. FIRST ITERATION

From the related work found in section 2 we conceptualized the first iteration of what we wanted to implement.

3.1 Concept

The initial concept started as a compact chord controller using silicon pads or buttons as triggers. The main focus for the controller is being able to play chords that exist on a given scale, alongside the ability to be as expressive as possible when using the controller.

3.2 Interactions

Interactions are critical when creating New Interfaces for Musical Expression (NIME). There are many possible types of interaction between a performer and a system. [10]. Interactions are often a process of giving input and receiving feedback [10]. The system takes the input from the real world (e.g., through a sensor) and translates this to an electrical signal understandable by a computer. The system then gives the user feedback based on the input [10]. Based on the related projects, the different interactions with the controller were chosen to rely on physical interactions (such as button presses, turning knobs, etc.). The visual feedback is given through LED lighting to show the user that a button has been touched. The auditory feedback comes from a separate system (e.g., a computer) communicated through the MIDI protocol. From initial brainstorming about the desired functionality of the prototype based on the motivation and related work, the following physical interactions were chosen:

1. **Changing scale**

   The device should be able to choose a scale in the standardized western major and minor by pressing the buttons. Conceptually, the selection presented on the buttons would start from the C scale followed by C# or D♭, then every note in the octave.

2. **Triggering chords**

   The pads will use the information from the selected scale and send out the MIDI information from the respective chord chosen on the pad.

3. **Major & minor**

   Another selector or button would then switch between major and minor. When a scale is selected, the buttons will react accordingly and map out the available chords in the given scale.

4. **Expressiveness**

   Furthermore, the controller should have different sensors and buttons that will modify how the chords will be structured and behave. This includes:
   (1) Turning knob that controls strumming of the chord that will arpeggiate the beginning.
   (2) Turning knob that controls low- or high-pass filters.
   (3) Turning knob that controls the randomization of the velocity used in the MIDI information.
   (4) Slider for volume control
   (5) Distance sensor for more expressive filtering.
   (6) Button press for applying inversion of the chord

5. **LED/LCD visualiser**

   The interactions made by the user should be visualized by a small LED/LCD screen to make the user follow the current status of the adjusted sensors and scale.

3.3 Design

From defining the concept of the controller, a 3D draft sculpture was made, seen in figure 1. This shows the initial idea come to life by showing the chord trigger pads taking up most controllers' enclosures. The top components consist of the adjustable knobs (left), followed by the LED screen (middle) and the distance sensor (right).

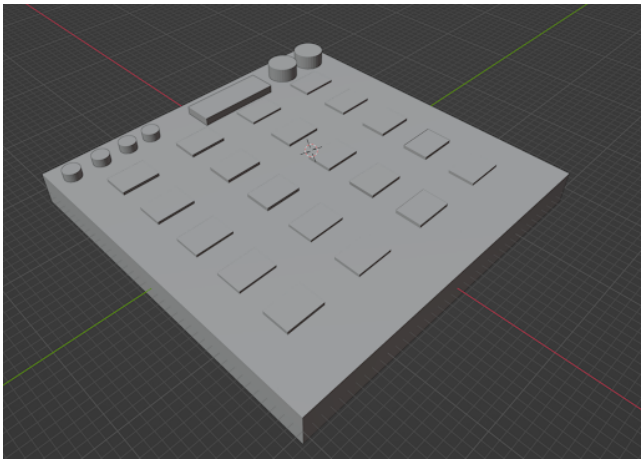

Figure 1. The 3D model of the low-fidelity prototype.

4. FINAL PROTOTYPE

4.1 Design

The design for the final prototype ended up being a group of silicon pads as buttons along with 4 knobs as seen in figure 2. This was heavily inspired by the UNTZtrument (see section 2.3). This meant the distance sensor from the initial prototype was scratched as well as the LED screen that would visualize the status of the controllers' adjustments.

4.2 Interactions

In the final prototype, the interactions for the available features were implemented through the silicone pads and knobs.

Figure 2. Final prototype design. The controller is currently in its chord mode selecting which scale to be played.

1. **Triggering chords**

   The component in use had 16 buttons available, and since the standard amount of chords in a major or minor scale is 7 we chose to make 12 of the buttons trigger chords. The first 7 pads is used to trigger the every chord in the selected scale in the MIDI protocol. The following 5 pads on top of the previous is then mapped to the next octave. For example if C Major is selected the mapped chords starts from 'C3' in the MIDI protocol. When the first 7 chords is mapped the next will change octave meaning continuing from C4. The remaining 4 buttons were used in other interactions.

   To explain how the chords is mapped in correlation to the MIDI protocol, pads and structure when a scale has been selected we refer to the list on table 1. The table shows the available chords in the C major scale.

C Major chord triggers

Pad	Chord	MIDI #	Notes
1	C Major	48, 52, 55	C3, E3, A3
2	D Minor	50, 53, 57	D4, F4, A4
3	E Minor	52, 55, 59	E4, G4, B4
4	F Major	53, 57, 60	F3, A3, C4
5	G Major	55, 59, 62	G3, B3, D4
6	A Minor	57, 60, 64	A3, C4, E4
7	B Diminished	59, 62, 65	B3, D4, F4
8	C Major	60, 64, 67	C4, E4, G4
9	D Minor	62, 65, 69	D4, F4, A4
10	E Minor	64, 67, 71	E4, G4, B4
11	F Major	65, 69, 72	F4, A4, C5
12	G Major	67, 71, 74	G4, B4, D5

Table 1. Adafruit Trellis chord mapping.

   Seen on figure 3 is how a chord is triggered through the 3 lower rows on the controller whereas the top row is the modification interactions.

2. **Changing scale**

   When a user wants to change the scale of the controller the 13th pad is pressed. After this, the LEDs will light up visualizing which buttons the user can press. The highlighted pads after the initial press on the 13th pad are a representation of the scales that can be selected as seen in figure 4. When one of the highlighted pads is pressed the respective scale will be mapped accordingly across the controller. The variety of scales can be seen in table 2.

Figure 3. Prototype triggering chords.

Chord selection mode

Pad	Scale
1	C
2	C# / D♭
3	D
4	D# / E♭
5	E
6	F
7	F# / G♭
8	G
9	G# / A♭
10	A
11	A# / B♭
12	B

Table 2. Scale selector.

3. **Switching between Major/minor**

Top pads

Pad	Functionality
13	Chord Mode
14	Major/Minor
15	Complexity Mode
16	Inversion Mode

Table 3. List of modes available on the top pads of the controllers.

   The interaction for changing the selected scale is available on the 14th pad. Here the only interaction is just one press and the selected scale will convert

Figure 4. Chord mode activated on the controller and selection is made by pressing a button on the 3 lower rows.

into either major or minor. All references to the top pad functionality can be seen in table 3.

4. **Complexity of chord**

   Another element we wanted to implement that ties into the expressiveness and customizability of the controller were the ability to add additional notes to the chords played. This was made available by pressing on the 15th pad. When complexity mode is activated, three buttons on the lowest row are highlighted as seen in figure 5. The different modes are listed in table 4. If the second complexity mode is selected every chord played will be the suspended 7th version of the chord. Meaning if e.g. a C major chord is triggered it will be converted to a CMaj7 and sent through the MIDI information.

Figure 5. Selecting which complexity mode to choose on the 3 highlighted buttons.

5. **Inversions**

Complexity modes

Complexity Mode	**Functionality**
1	Default chord without suspended notes
2	Suspended 7th
3	Suspended 7th & 9th

Table 4. List of different modes of complexity.

When pressed on the 16th pad inversion mode will be activated. This mode has 4 different outcomes. The default inversion selection will play the standard chord without any inversions. The other modes will start transposing the different notes used in the chord. 2nd press will transpose the root note one octave up, 3rd will do the same with the major third. Lastly the 4th will transpose the perfect fifth down one octave. This can be seen in table 5.

Inversion modes

Inversion Mode	**Functionality**
1	Default chord without inversions
2	Transpose the root note one octave up
3	Transpose the major third one octave up
4	Transpose the perfect fifth one octave down

Table 5. List of different modes of inversions.

6. **Expressiveness (knobs)**

Knobs

Knob	**Functionality**
1	Strumming / Delay
2	Velocity randomisation
3	Octave selection
4	Not implemented

Table 6. List knob functionality left from right.

The available functionality from the interactions with the knobs on the controller can be seen in table 6. The simple interaction consisted of turning the knobs. The more they are turned the more the functionality is affected. Meaning while playing the chords and turning the delay/strumming knob each note is delayed a tiny bit. The same concept goes for the velocity functionality as seen in figure 6. The octave selection is accessed through the third knob. The user is able to choose between 3 octaves, and the more the knob is turned the octave is changed.

The chord selection was also a new addition in comparison to the previous prototype.

Figure 6. Turning the randomization of the velocity.

4.3 Technical description

The following section will present the technical aspects of building the final design. This includes which components and sensors were used, the circuit design, the 3d design of the enclosure, and the code ported to the microcontroller.

4.3.1 Components, sensors and circuit design

When choosing the appropriate components or sensors for the controller, it is important to base the choice on which actions of the user will be important for the application [11]. In the case of this controller, two main interactions need to be considered; (1) triggering chords and (2) manipulating parameters. Based on the related projects in section 2, the preferred interaction for chord triggering is pushing a button. For controlling parameters such as strumming and velocity, the user needs more control than a button can provide. In this case, a knob can provide the user with much more control in an intuitive way. Based on this, the following components were chosen for the controller. For changing scale and choosing complexity, the buttons provide enough control, along with the ability to give feedback through the LEDs.

- **Adafruit Trellis** [5]

  The Adafruit Trellis is a driver system with 16 keypads with LED backlighting. Each tile on the component has a keypad reader and a LED sequencer, all controllable with the I2C communication protocol. The Adafruit Trellis is ideal for the controller presented in this paper, as most of the functionality for using buttons/keypads to trigger chords is already implemented.

- **Rotary potentiometers**

  Four different rotary potentiometers control the parameters for the expressions strumming, velocity randomization, and changing octaves (see section 4.2). The value of the potentiometers will be mapped to match each parameter.

- **Teensy 4.0** [6]

  The microcontroller Teensy 4.0 is used as the main controller for the controller. Teensy 4.0 includes easy implementation of the MIDI protocol and pins for communication through I2C, which is needed to write and read data to the Adafruit Trellis. Making it a great fit as a microcontroller for the controller.

The final circuit design can be seen in figure 7.

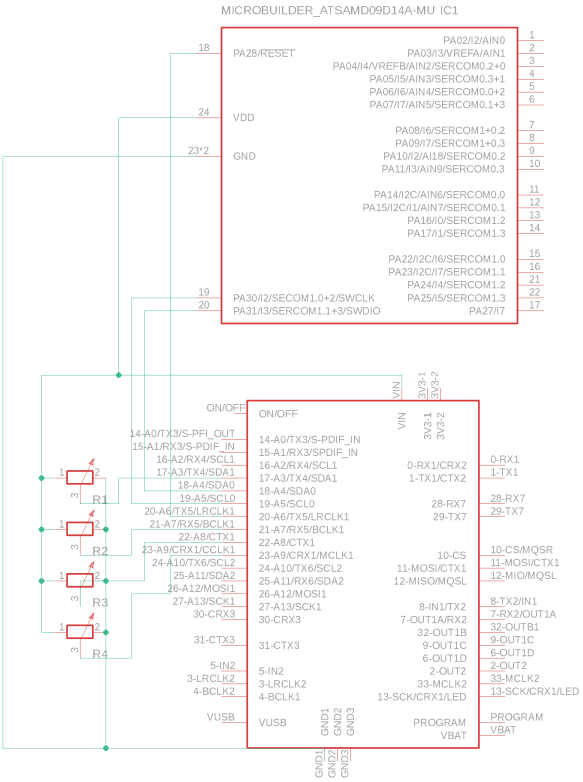

Figure 7. Curcuit Diagram. One Adafruit Trellis and four rotary potentiometers connected to a Teensy 4.0

4.3.2 Enclosure design

The enclosure for the controller was designed in 3D, heavily inspired by the enclosure from the UNTZtrument (see section 2.3). The UNTZtrument uses the same components (Adafruit Trellis and potentiometers), and with a few alterations to make the enclosure fit the Teensy 4.0, the enclosure was constructed. The enclosure was separated into four different 3D objects; a bottom, a frame, a cover, and a tray for holding the Adafruit Trellis. The four other objects can be seen in figure 8.

The four objects were printed using a 3D printer and assembled with the components afterward.

[5] Adafruit Trellis: https://www.adafruit.com/product/1616

[6] Teensy 4.0: https://www.pjrc.com/store/teensy40.html

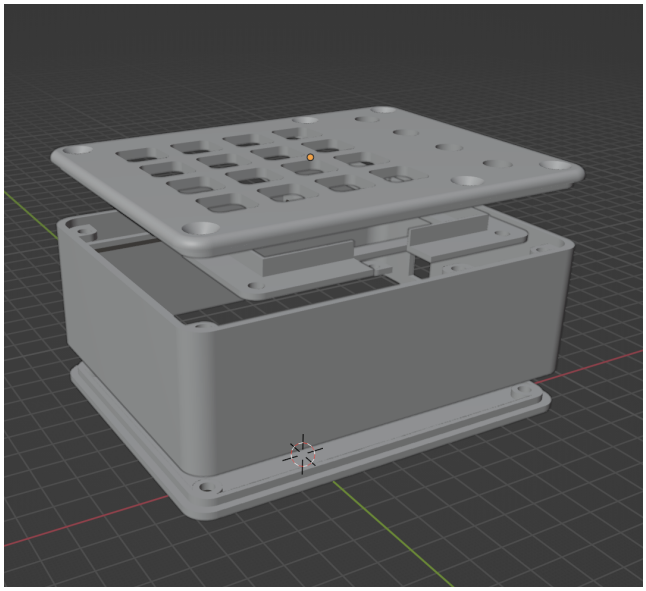

Figure 8. The 3D enclosure. Consisting of four different objects.

4.3.3 Microcontroller code

All the code is written using C++ with Arduino[7]. Arduino provides unique methods and addition to C++, which makes it easy to control a microcontroller. The code can be ported to a Teensy using the extension Teensyduino.

For the implementation of reading the different interactions using the buttons (see section 4.2) on the Adafruit Trellis, the *Adafruit_Trellis.h* library provided by Adafruit was included in the project. The library offers the method *trellis.justPressed()* for reading when a button is pressed, and *trellis.justReleased()* for reading whether the button is released. Furthermore, adding feedback through LEDs is also implementable through the *trellis.setLED()* method. For the implementation of the more expressive parameters using knobs, the built-in Arduino method *analogRead()* was used. The value of the potentiometer can then be mapped to match the parameter changed using the *map()* method. For example, the potentiometer controlling strumming is mapped to go from 0ms to 150ms.

For MIDI implementation, Teensyduino provides the usbMIDI library. The methods *usbMIDI.sendNoteOn()* and the method *usbMIDI.sendNoteOff()* can be used to control when a chord is played when matched with the *trellis.justPressed()* and *trellis.justReleased()* method. To know which MIDI notes to send to the computer, a separate document with arrays of integers representing the different notes in chords (as seen in section 4.2) to send through midi (see figure 9). The chords' notes are then sent through MIDI one by one with a random velocity depending on the value of the second knob and with a *delay()* method in between, creating the strumming effect. All of the code is attached in the appendix.

[7] Arduino: https://www.arduino.cc/

```
int Cmaj[4]   =  {NOTE_C3, NOTE_E3, NOTE_G3, 0};
int Csmaj[4]  =  {NOTE_Cs3, NOTE_F3, NOTE_Gs3, 0}
int Dmaj[4]   =  {NOTE_D3, NOTE_Fs3, NOTE_A3, 0};
int Dsmaj[4]  =  {NOTE_Ds3, NOTE_G3, NOTE_As3, 0}
int Emaj[4]   =  {NOTE_E3, NOTE_Gs3, NOTE_B3, 0};
int Fmaj[4]   =  {NOTE_F3, NOTE_A3, NOTE_C4, 0};
int Fsmaj[4]  =  {NOTE_Fs3, NOTE_As3, NOTE_Cs4, 0
int Gmaj[4]   =  {NOTE_G3, NOTE_B3, NOTE_D4, 0};
int Gsmaj[4]  =  {NOTE_Gs3, NOTE_C4, NOTE_Ds4, 0}
int Amaj[4]   =  {NOTE_A3, NOTE_Cs4, NOTE_E4, 0};
int Asmaj[4]  =  {NOTE_As3, NOTE_D4, NOTE_F4, 0};
int Bmaj[4]   =  {NOTE_B3, NOTE_G4, NOTE_Fs4, 0};
```

Figure 9. Example of major chords in an array of integers. Each variable is an integer representing a MIDI note

4.4 Methods for evaluation

When assessing and evaluating a new digital instrument, the most important stakeholder is the performer. If the performer can not successfully play the instrument in an intended way, the instrument has failed [12]. A study by J. Barbosa et al., explored what type of evaluation is used in all NIME papers [13]. This study found the performer to be the most evaluated stakeholder [13]. Furthermore, the most common target to evaluate was the whole instrument (i.e., not just the input, mapping, etc.). The study also found the most used goal of an evaluation to be to investigate how the target performs according to specific criteria. Based on this, the controller presented in this paper should have the performer/user as the evaluation target. Furthermore, it should investigate how the user performs with the controller according to criteria matching the specific goals of our controller. These criteria could be intuitiveness (i.e., if the user can easily understand and learn how to play chords) and expressiveness (i.e., if the users can express themselves as they want through the implemented parameters). Before the final evaluation, it could also be helpful to review some usability testing to identify and remove any usability flaws. The System Usability Scale (SUS) could be used to do this.

5. CONCLUSION

The goal was to create a plug-and-play MIDI controller to easily trigger chords and allow performers to express themselves as they want. A MIDI controller has been successfully designed and implemented, with minor and major chord scales for any individual to play right away. Furthermore, several parameters for adding expressiveness have been implemented.

Nevertheless, the controller has some shortcomings, leaving room for further development. As of now, the user does not have any immediate way to know what the different controls (knobs, buttons) do. The top row of the buttons is for adjusting parameters, but the device does not afford the user to press these for control. Furthermore, the

user does not have any way to separate these from the rest of the buttons. The same problem is present for the knobs, which are separate from the buttons, but does not have any label for what they control. If the user has changed the scale, the user does not have any direct way to know which scale they are on. Furthermore, the exact value of the parameters is not present (e.g., the user does not know how many milliseconds the strumming is). These issues might affect the intuitiveness of the controller negatively.

To address these issues, the first thing to do would be to add labels to each button and a knob about what they control. This could be done in the 3D design of the enclosure. Furthermore, an LCD screen could be added to give the user exact knowledge of what they are controlling and what the value of that parameter is. An LCD was already conceptualized in the first iteration (see section 3.2), and should therefore be added to the design again. These optimizations should help the user to express themselves in the exact way they want. However, a learning curve is needed for the most advanced MIDI controllers (e.g., Maschine MK3 and Ableton Push), and is therefore also expected to present for out controller. Fixing these issues, along with usability testing should prepare the controller for final testing on the end target group. The goal of the final evaluation should be to asses whether this controller has a place for a beginner producer or a performer without extensive music knowledge with a focus on intuitiveness and expressiveness.

References

[1] M. Müller, *Fundamentals of Music Processing: Audio, Analysis, Algorithms, Applications*. Springer International Publishing, 2015. [Online]. Available: https://books.google.dk/books?id=HCI_CgAAQBAJ

[2] C. Dobrian and D. Koppelman, "The 'e'in nime: musical expression with new computer interfaces," 2006.

[3] P. Karr-Wisniewski and Y. Lu, "When more is too much: Operationalizing technology overload and exploring its impact on knowledge worker productivity," *Computers in Human Behavior*, vol. 26, no. 5, pp. 1061–1072, 2010.

[4] N. I. E. Staff, "Working with chords." [Online]. Available: https://www.native-instruments.com/en/maschine-mikro-quickstart/working-with-chords/

[5] J. Gibbons. Maschine: Chords mode masterclass! essential chords tips. Youtube. [Online]. Available: https://youtu.be/JdUqpvHjrwc?t=1196

[6] A. E. Staff, "Push playing notes and chords." [Online]. Available: https://www.ableton.com/en/push/playing-notes-and-chords/

[7] Ableton. Learn push 2: Playing chords. Youtube. [Online]. Available: https://www.youtube.com/watch?v=WdwnkcTjOuQ&t=1s

[8] R. Brothers. Mini untztrument: 3d printed midi controller. Adafruit. [Online]. Available: https://learn.adafruit.com/mini-untztrument-3d-printed-midi-controller

[9] A. Industries. Mini untztrument - 3dprinted midi controller raspberrypi. Youtube. [Online]. Available: https://youtu.be/bM7NN-fiMXs?t=31

[10] B. Bongers, "Physical interfaces in the electronic arts interaction theory and interfacing techniques for real-time performance," 2000.

[11] W. Putnam and R. B. Knap, "Input/data acquisition system design for human computer interfacing," 1996.

[12] S. O'Modhrain, "A framework for the evaluation of digital musical instruments," *Computer Music Journal*, vol. 35, no. 1, pp. 28–42, 2011.

[13] J. Barbosa, J. W. Malloch, M. M. Wanderley, and S. Huot, "What does 'evaluation' mean for the nime community?" in *New Interfaces for Musical Expression*, 2015.