MICROTUNING MIDI STREAM USING MICROCONTROLLER

Riku Itäpuro

Tampere University of Technology
riku.itapuro@iki.fi

ABSTRACT

We present a simple microcontrol device for modifying a stream of MIDI data in real-time to achieve a better music perception via an enhanced tuning. Using MIDI is ideal, because it transmits only the metadata of music, not actual timbre. The pitch only is important here. Musical scales and their tuning are inherently static on equal tempered (ET) instruments like keyboards, but the tuning in pure style sounds better for some musicians. With a dynamic tuning, we can make an electrical instrument superior to a static ET-tuned instrument. There exist few such thing on market, but not on free software. Earlier methods have used multiple MIDI channels and Pitch Bend to achieve polyphonical tuning while our approach utilizes polyphonic aftertouch (PAT) controlling messages to save MIDI channels.

1. INTRODUCTION

This paper presents a microcontrol device which filters Musical Instrument Digital Interface (MIDI) data between controller (MIDI-keyboard) and sound generator (MIDI-synthesizer). MIDI defines both hardware communication interface and protocol for musical metadata. MIDI has been a universal connector between musical instruments for almost 30 years.

The main goal is to study how changing tone's pitch can result sometimes better sounding result. Solution can also be used as microtonal fixed tuning for alternate scales. Some music theory background will be presented here. In this study term microtuning means altering individual note's pitch while tuning means shifting master tune of an instrument.

Musical scales and tuning in western music are inherently static since 1800's when equally tempered tuned (ET) pianoforte's success over harpsichords' and claviers' was evident [1]. ET means, that every pair of adjacent notes has an identical frequency ratio. In western music frequency ratio of 2 is called an octave and it is divided to 12 tone equal semitones (12-TET). On other tuning systems notes are not always tuned to same pitch relations, but the tuning depends either from division of scale or from other together sounding notes which makes it dynamic. Electronic instruments and acoustic piano suffer a lot from 12-TET, even when the western ear is used to it. Yet musicians can hear difference between pure and ET tuning. Pure tuning can be heard for example at barbershop or trombone quartets, where musicians microtune their voice continuosly according to placement of together sounding notes. So problem is that tuning can not be defined beforehand. With dynamic tuning, we can make electrical instruments superior to static (and ET) tuned instruments and lessen the distance to acoustic world. This is an important problem also because for author's knowledge there exist only few similar products on market, but they all are patented. On spirit of open-source, everyone has freedom to learn how to make things for themselves despite patents.

Research methods include studying mathematical and acoustical theories, programming, prototyping device and measuring humans perception to final product. We propose to implement solution and give final product to both civilians and musicians to judge difference between normal and resulting output.

When succeeding, device has broad availability for users, because it will be cheap (less than 20 EUR) and uses open hardware and software. Because the solution is still work in progress, there exist no yet absolute numbers for efficiency. It will be fast enough for what it tries to accomplish, which is glimpse of better sounding tomorrow.

2. RELATED WORK

There exists computer music software algorithms for offline, online tuning and static mapping. Motivation for our research work comes from Hermode tuning [2] (1989), which describes algorithm used on few commercial synthesizers and computer sequencer programs, which are both expensive and closed to user.

Almost similar to our method is Moussa's patented invention to use Aftertouch [3]. His product also considers network consisting of multiple instrument networks where different instruments use different tuning methods.

Table 1. Comparison of ET and Pure Tuning

Key	$[C^1 - C^3]$	C^1	D^1	E^1	F^1	G^1	A^1	B^1	C^2	$D^2 \dots$
ET Tuning	[cents]	0	200	**400**	500	700	900	1100	1200	1400 ...
Pure Tuning	[cents]	0	204	**386**	498	702	884	1088	1200	1404 ...
ET Tuning	[fraction]	1:1	$\sqrt[6]{2}$	$\sqrt[3]{2}$	$\sqrt[12]{2^5}$	$\sqrt[12]{2^7}$	$\sqrt[12]{2^9}$	$\sqrt[12]{2^{11}}$	$\sqrt[12]{2^{12}} = 2$	...
Pure Tuning	[fraction]	1:1	9:8	5:4	4:3	3:2	5:3	15:8	2:1	17:8 ...

3. ACOUSTIC AND MUSIC THEORY BACKGROUND

Frequency (f) ratio of 2 is called an octave and it is divided to 12 semitones (also called halftones) or 1200 cents. Computing with cents is easier than with frequencies. Relationship between frequency-ratios and cents is

$$\frac{f_1}{f_2} = 2^{\frac{cents}{1200}}. \tag{1}$$

Intervals on Pure tuning derive from lowest possible nominators ie. when intervals' frequencies are related by a small integer ratios they are said to be in consonance. It has been shown that 12-tone tuning can not satisfy perfect chords, because there are so wide variations on pitch for example on Major 3rds on Table 1, column E^1 showing comparison between ET and Pure C major scales.

On ET, notes are evenly distributed, so each note is multiple of 100 cents away from base note. Intervals are building blocks for chords. Comparison between Semitone, Whole Tone, and Major Third on ET and Pure Tuning in cents is presented on Table 2.

Human perception of pitch difference is about 5 cents. "You can hear about a nickel's worth of difference" has been said, but that depends also on sound level, duration, change time and musicality of listener. Alternation of G-note from 700 to 702 cents might go unnoticed to many, but changing note E from 400 cents to 386 can not go unnoticed.

Without hearing the difference, it can also be visualized using Lissajous figures. Figure 1 and Figure 2 show the difference between Pure and ET-tuned major third interval. Notice less variation and more symmetry on Pure interval. Lissajous figures can be made also with mechanical harmonograph [4]. Figures here have been drawn with Wolfram Alpha with commands below figures.

Table 2. Characteristics of ET and Pure Tune intervals

Interval name	Pitch [cents]	Remarks
HalfTone ET	100	Always
HalfTone Pure	~100	Depends on scale
Whole Tone ET	200	(100+100)
Whole Tone Pure	204	(114+90)
Major 3rd ET	400	Always
Major 3rd Pure	386	Always

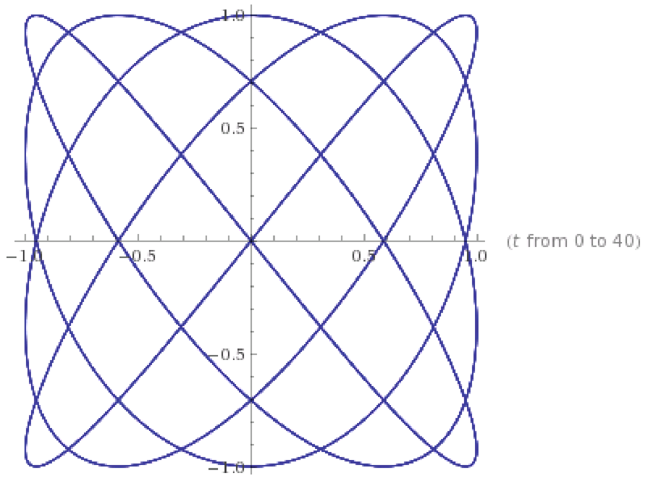

Fig. 1. Lissajous presentation of Pure major 3rd interval: $plot(x, y) = \sin(t), \cos(t * \sqrt[3]{2}f), t = 0, 20$

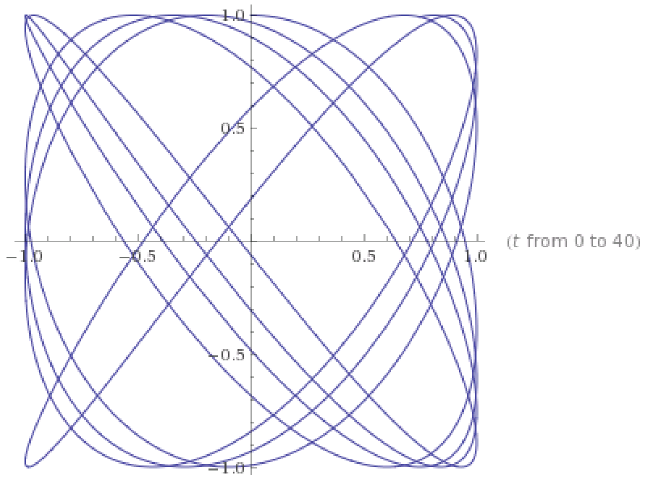

Fig. 2. Lissajous presentation of ET major 3rd interval: $plot(x, y) = \sin(t), \cos(t * 5/4), t = 0, 20$

4. ON IMPLEMENTATION

Existing parts are MIDI keyboard for controlling and MIDI sound generator for sound producing. A well-documented and open Arduino Duemilanove ATMega 328 microcontroller board was programmed to alter MIDI data stream.

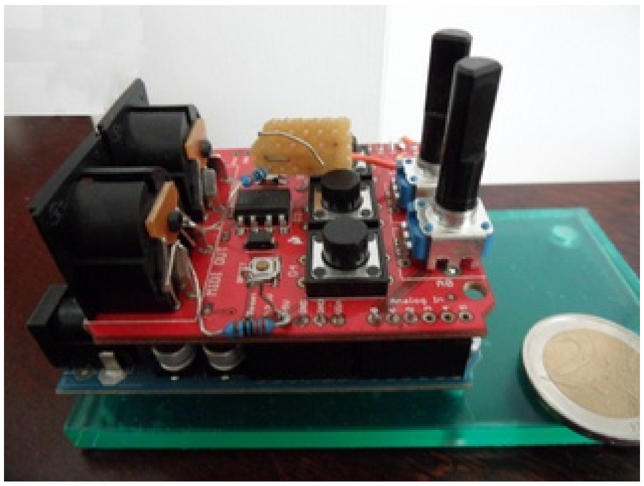

Fig. 3. Arduino Duemilanove micro-controller(bottom) with MIDI-shield(top). MIDI-connectors (IN & OUT) on left, buttons and controllers on right (available for programming)

It has an MIDI shield which connects Arduino's serial I/O to DIN-MIDI interface, which connects to both MIDI-device via MIDI cables. Some soldering was needed to attach buttons, potentiometers and DIN-type MIDI interfaces on board. Existing model had broken surface mounted component which was replaced with an old fashioned resistor. The picture of Arduino is shown in Figure 3.

Arduino board has small but sufficient 32KB flash memory for program itself, but only 2KB SRAM for variables, heap, and stack. That may be limitation for careless programmer. Current program code uses about 7KB of flash. MIDI protocol works on 31250 Bauds which is both low speed and incompatible with many existing serial protocol speeds. Arduino can be powered with 9V-battery and does not need external monitor or keyboard, so it is much better than even smallest computer or laptop. On development phase power was still drawn from computer's USB-connector.

Because MIDI sends only controlling elements of music and not the actual timbre, it takes less bandwidth. There is no need for pitch recognition, because pitch is already included in 3-byte MIDI NOTE-ON message. An adjustment based on relations between different notes can be computed fast. The solution allows notes to be microtuned by dividing semitones yet to another 128 smaller steps. Most people can not separate two pitches, which are less than 1 cent apart from each other, so resolution

$$\frac{100cents}{128} = 0.78125cents, \tag{2}$$

while less accurate than earlier Pitch Bend method's

$$\frac{100cents}{16384} = 0.0061cents, \tag{3}$$

is still sufficient for everyday use. Our solution uses ± 14 steps ($\pm 11cents$) for tuning one note.

The program uses Arduino's MIDI-library [5]. Software solution starts by reading incoming keyboard signals with help of the MIDI-library to a data structure. The distance between each note with respect the root note (that is the lowest note) can be mapped with help of the interval table. The workflow of this is shown in Figure 4. Later, also chord balancing is planned. Balancing means, that the chord is set in balance both on the vertical axis and on historical respective so, that earlier changes will smooth future changes. One can also in future control how strength tuning will be: 25-, 50-, 75- or 100-percent strict Pure Tuning.

Early tests show progress on first tasks. Played notes are tuned in almost not audible delay in real time. No optimizations have been made yet. Command 'All-notes-off' must still be implemented for panic, because small input buffer size can miss some NOTE OFF messages from keyboard. Without corresponding NOTE OFF message one note will stay audible forever.

5. LIMITATIONS

Study gave out following limitations: There was no metric available to measure the quality of the new tuning other than human ear. Measuring the processing delay was difficult, because microcontroller misses realtime clock. Used sound generator must recognize and map polyphonic aftertouch messages and apply depth of tuning, but this needs to be done only once. Many voices on sound bank were using a lot of unnecessary vibrato and other effects on their timbre, while pure sinusoidal or square wave tones were best on testing. Notes must now be played from highest to lowest order so that program recognizes lowest tone, but that limitation is easy to correct. Human perception for tone changes may set limitation for computations of single alternation. Only Pure Tuning is implemented. Least limitating components are on controlling MIDI device: it needs to be able to send NOTE ON messages.

6. CONCLUSION

Used study methods included empirical testing and implementation of basic functionality of tuning device but lacked some of the planned parts. Implementation of the micro-controlled device shows unique features, but is still a toy compared to the planned model. Yet the current limitations do not hinder continuous developing of the model.

Future work on this topic might consider continuing with software implementation, because prototype here showed that it is possible to make a difference on tuning with minimal hardware. One possibility would also be to use simulators instead of the real device for testing.

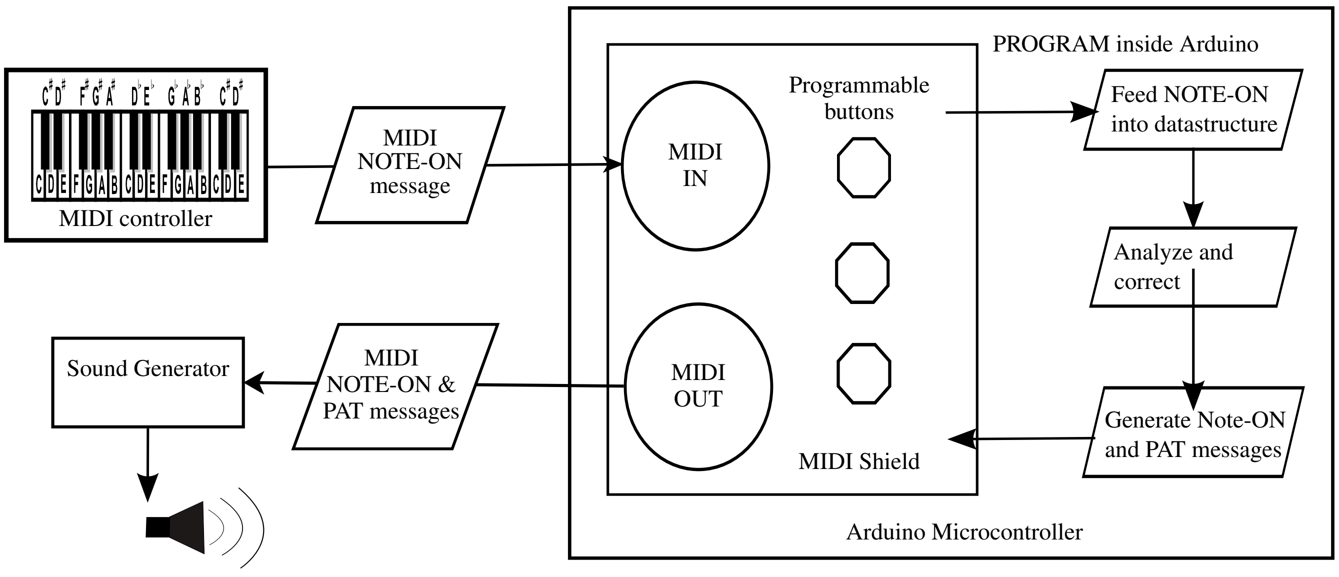

Fig. 4. Workflow of dynamic tuning.

REFERENCES

[1] G. Green, "History of Piano Tuning," Online. Cited 2013-12-08, 2004.

[2] W. Mohrlok. (1999) Original implementation. [Online]. Available: http://www.hermode.com

[3] A. S. Moussa, "Perception-Based Microtuning over MIDI Networks," *IEEE Multimedia*, vol. 13, no. 1, pp. 56–64, 2006.

[4] A. Ashton, *Harmonograph: A Visual Guide to the Mathematics of Music*, ser. Wooden Books. Walker & Company, apr 2003.

[5] Arduino MIDILibrary. Online. Cited 2013-12-02. [Online]. Available: http://playground.arduino.cc/Main/MIDILibrary