International Journal of Engineering and Science Invention
ISSN (Online): 2319 – 6734, ISSN (Print): 2319 – 6726
www.ijesi.org ||Volume 5 Issue 9|| September 2016 || PP.01-07

Micro-MIDI: A Real Time, Dynamic Microtonal MIDI Application

Serdar Celik[1]
[1]*(Music Technology/ Cumhuriyet University, Turkey)*

Abstract: *In this study, a software design which can make real time changes on MIDI tuning in order microtones to be used and played back on MIDI network was aimed. For this reason, a patch which works under the Max/MSP programming language and uses patch bend message with different MIDI channels in order to play, record, change and edit the microtonal tones, and a patch object were programmed. Patch object had the range to make changes between the -99 and + 99 cent values on all the chromatic intervals of an octave within MIDI tuning system and also with the Bach External (Automated Composer's Helper) imbedded afterwards, patch gained the function of real time notation which was able to write the data coming from MIDI controller with their microtonal signs and notes. In order to test the patch, polyphonic music vocalization was performed. For this reason, ten different notes were struck simultaneously. Within the test, any data loss and connection detach didn't come out. For this reason, this study proves that multı channel pitch bend usage upon MIDI network is confidential and appropriate for playing the monophonic and polyphonic music's. Together with the exe and dmg files which can work alone, open source code shared maxpat and txt files of programmed patch and patch object have been shared on the internet so that the proficiency of the patch to play and write the microtonal frets have been opened to discussion.*

Keywords: *Music Technology, MIDI Programming, Microtonality, Max/MSP, Pitch Bend*

I. Introduction

MIDI (Musical Instruments Digital Interface) is an indispensable communication protocol especially for it provides distance access, arranging the music and also for the note software's and the communication between their plug-ins. Sequencers allows MIDI data to be recorded, arranged and played back again. A song performed with any sound card or synthesizer can be recorded by Standard MIDI File (SMF) data format and thousands of MIDI files to be downloaded free which were made in this way are in access at present. It also has a system which allows numerous performance techniques via synthesizers and controllers and also has a sound library consist of 128 different instruments General MIDI (GM) performances to be played back after channel messages. Moreover, all the sound card producers have been in harmony with this rules sequence called GM [1, 2].

While MIDI are in use for free with its all mentioned or non mentioned opportunities above, MIDI tuning which is arranged according to Western music temperament tuning system and describes the key cent values of MIDI, turns out to be an extra function with some extra cost when it is exchanged with the cent values of microtonal pitches (*microtone*). In some developed and expensive keyboard workstations or synthesizers, each half interval can be changed between -99 and +99 cent values. However, this high cost solution can be realized with System Exclusive (SysEx) system message and system ID number given to exclusive synthesizer producers. For this reason, any SysExmessage formed by synthesizer produced by any firm cannot be understood by another firm's synthesizer. [3]. Even though SysEx message data was given a universal Device ID number after GM2 update which was developed by MIDI ManufacturersAssociation, its usage has not become practical like a channel message in MIDI hardware and software's due to the fact that it is a system message [1]. Another alternative solution was updating the MIDI Tuning Standard (MTS) and a lot of software (Scala, Alt-Tuner, TiMidity++, Xen- Arts Microtonal MIDI Software etc.) was updated by using this massage but it hasn't taken enough attention in music software industry. Reason for this can be shown that MTS update is a system message also and it doesn't allow real time change in all MIDI tuning.

In order microtones to be played back in MIDI tuning, another solution can be re-program the current MIDI messages again for different aims and duties apart from the attributed duties at present. However, verylittle percent of this kind of studies have taken part in scientific literature [4]. They send the information of pressure change on MIDI clavier keys, which they got by using polyphonic aftertouch message with the interface they programmed via SysEx message and use it to change MIDI tuning. Mus2 (www.mus2.com.tr) which is a commercial software has made the microtones of the Turkish Maqam Music played by MIDI and written in musical notes and also a huge database (SymbTr) for many of the maqam used in Turkish Music consisting of MIDI and MusicXML files has been prepared. In numerous studies on Mus2 microtonality and MIDI [5-7] it has the feature of the most developed commercial software to be used to write notes and play with microtoneaccidental signs. Moreover, [7] transformed his Turkish Music performance analysis into MIDI format by Matlab and offered a method presenting the microtonal keys of Turkish Music performances with PB values. Çelik[8] programmed and an interface with MM base which can be triggered with clavier, can playback the microtones consist of theorical and practical cent values and can also record these played notes in MIDI format. In this interface, pitch bend (PB) message was used multi channeled and with this method four different microtone for 10 different maqam were able to be played back successfully.

In this study, it is aimed that Çelik's method for microtonal play and his interface he programmed is to be developed more and patch and patch object are to be programmed with not Çelik's four microtones but with Max/MSP programming language which can change all the key values called chromatic between -99 and +99 units. It is aimed that programmed patch together with an outside MIDI with its In/Out ports will connect to controllers and exchange the cent value of MIDI keys, which it took from MIDI controller or recorded MIDI file with the cent values determined on the software. It is also expected to play these values with GM instruments, present the cent values it has changed on the software, on staff with proper accidental signs, and also give proper export format (MIDI, MusicXML, Open Music, etc.) which will open in note writing programs like Finale, Sibelius after recording the mentioned cent values and accidental signs.

Another aim of this study is to explain the method to be used patch programming and patch algorithm. The source codes of patch and the patch object in this study will be released in a format (maxpat or txt) which can be seen clearly connections among patches and will be a database for the future studies about microtonality and MIDI.

II. Method

2. 1. Programming Pitch Bend Message

In order microtones to be played, the unique channel message which can create change, even if it is just for a second, in MIDI tuning is the PB message which is supported by GM. Moreover, PB message has become an industrial standard for almost all MIDI software and equipment and it is a kind of message which can be carried on MIDI network easily. This flexibility and easy conveyable feature of PB message has been the main reason of its usage in this study to play the microtones.

A PB message consists of one status and two data bytes, and also two nibbles whose each MIDI consist of 4 bites. The first nibble in status byte determines the PB message number and the second byte determines the PB channel number. There can be 16 different channel numbers between values of 0 and 15. In this case, even if the second number gives the value of 0, this means 1. MIDI channel will be used. Data byte which follows the PB status byteconsist of PB data which gives the extend of the sliding of the sound to be done or has already been done. PB data bytes which can produce 14 bites of data in total has a high resolution between 0 and 16383 integer values. If PB value is 8192 units, this means there hasn't been any sliding in the sound however GM determines which the parts of the pitches, the highest and lowest values of PB values to be reached will change. According to GM, when PB value rises from 8192 to 16383, pitch value is one full note towards high and when it decreases from 8192 to 0 unit it is one note towards low so the value is two full intervals in total. With this information, how many PB units is equal to one cent value can be easily calculated.

$$V_{PB} = \frac{X_{PB}}{X_C}$$

Where V_{PB} is PB values of a cent, X_{PB} is range of the PB values (8192) and X_C is a cent number of one whole note. It is known that one whole note has a value of 200 cents, a cent value is8192/200 = 40.96 unit PB value.

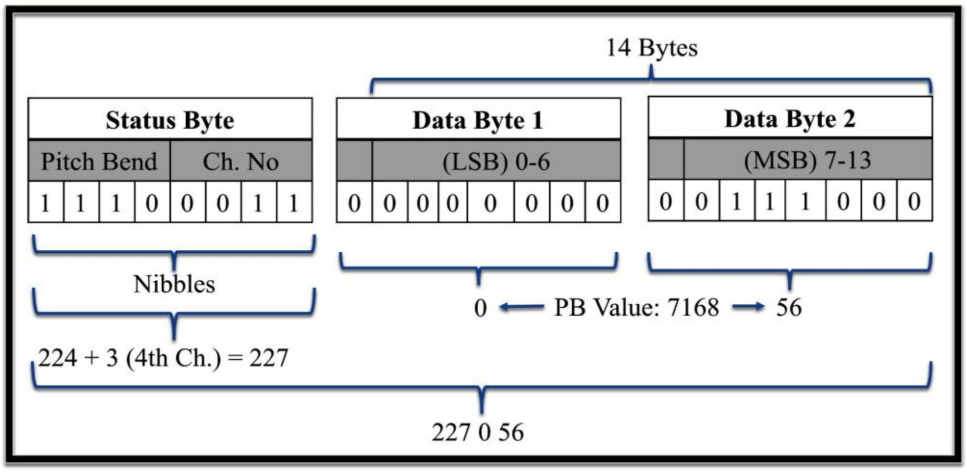


Figure 1: Pitch bend Message Status and Data Bytes

In Fig. 1. it is seen that when the status byte is transferred to number value in decimal base, second nibble PB is started (224) and the first nibble is to be carried via fourth channel (3). When data bytes are calculated in decimal base numbers, total PB value of the PB data bytes is 7168 units. Since this value is smaller than 8192 which is the balance value of PB message, this means the played key at that moment is going to be low-pitched and the PB value of this change is -1024 (7168-8192) units, its equivalence in cent value is -1024/40 = -25 units. PB message in the example in Fig. 1 has been formed to show that key sent with the 4th MIDI channel number will become 25 cents low-pitched. In the example, raw MIDI data of PB message which is 227 0 56 is re-coded according to dual counting system and sent to exit unit (MIDI- Out) via network.
The fact that PB message will be used for which MIDI key is determined by note message which includes the 'key was pressed' information and PB message always follows a note-on message. This message is expressed with one status byte and two data bytes. As it is in PB message, in the note-on message, while the first nibble in the status byte opens the note-on message, the other nibble is used to determine the 16 different MIDI channels. Unlike PB message, data bytes of note-on message are used to determine the variables with different data bytes. The key number of the notes to play the first data byte is determined by the daily value of this note. In Fig.2, in Note-on message example, data messages carrying the information that a note-on status message sent via 4th channel will be played at D#4 (63) key with ***ff*** (105) dynamic are shown. In Fig. 2, raw MIDI data of note-on message formed for D#4 key, which is desired to be played with 105 velocity value upon 4th MIDI channel, is found as "147 63 105" and in this message as it was the case for PB message is sent to exit unit.

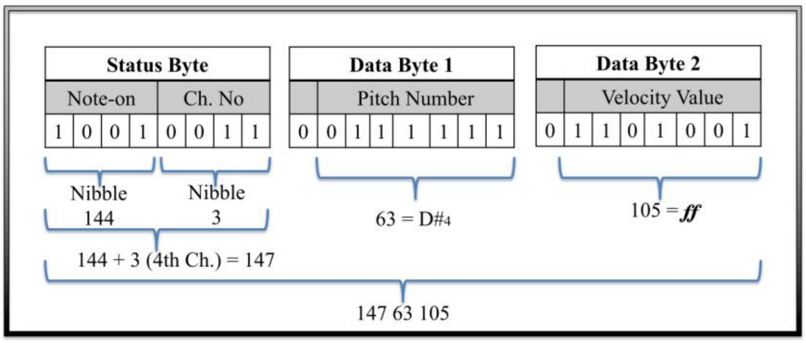


Figure 2: Status and Data Bytes of Note-on Message

Although it has features mentioned above and its high resolution PB message was designed to work PB wheel in an integrated way. So, PB message doesn't provide stable changes on MIDI tuning but it is rather used to play the instant oscillations among the cent values of the keys, reminding us the string stretching method in guitar. Each sliding movement towards the low or high key by using PB message is coded as an information of change in PB message and is transformed into MIDI data bytes which are formed sequencing successively. This continuity in PB message results from its continuous control structure and although it has a big extend to covey data to other MIDI software and hardwares connected to MIDI chain without any loss and also it can be carried within MIDI network easily, this continuity situation will be a problem when it is attempted to be used permanent change in MIDI tuning without PB message sliding effect. In that case, PB values of each microtone should be carried via different MIDI channels and this will cause the *running status* feature, which economizes upon MIDI bandwidth, to be out of the system and will cause multi channeled PB message to be an excessive burden on MIDI network. This data load which exceeds the MIDI bandwidth limit (31250 bytes/sn) will have data loss in the system. This view was supported in different studies [9-11] Maybe for this reason the studies about the usage of PB messages for the permanent tuning change has stayed limited.
In this study, in order to provide permanent MIDI tuning change PB message was programmed again. The continuous control feature of the PB message was limited and the unique PB message which can give the desired certain value was related to the key number of the note-on message. PB value for all the chromatic intervals within the one octave sound area and each PB and note-on message with 12 different MIDI channel number (except the10th channel since it belongs to percussion group) were sent to exit unit. The diagram below was prepared to explain the data way to follow for the only microtonal pitch. The numeric values on the diagram were given as an example and the cent value of D#4 key was wished to be lowered for 25 units. In order to play the D#4 key note-on message and to make it lower PB message were used. In lowering the sound, in order to prevent sliding effect and to make permanent changes on tuning, only one PB message was sent to exit unit before the note-on message. As it can be understood from the diagram, process priority is of PB message which stands for the green colored process levels and note-on message follows it. For this reason, channel number of PB and note-on messages are equaled so that PB message sent beforehand can change the pitch cent value of note on message.

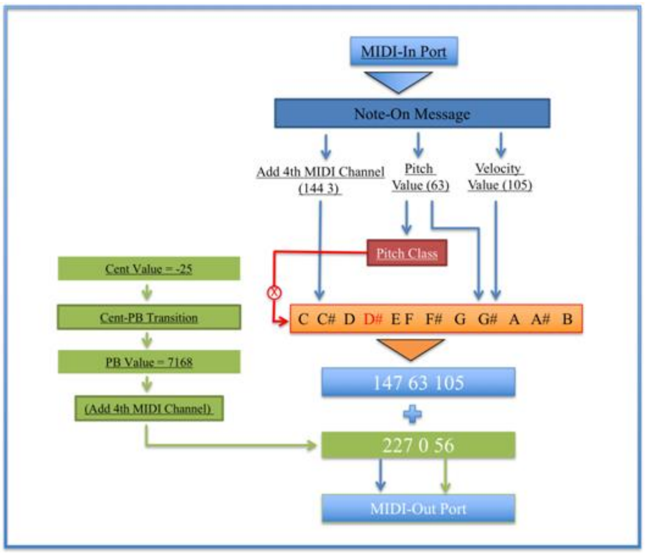
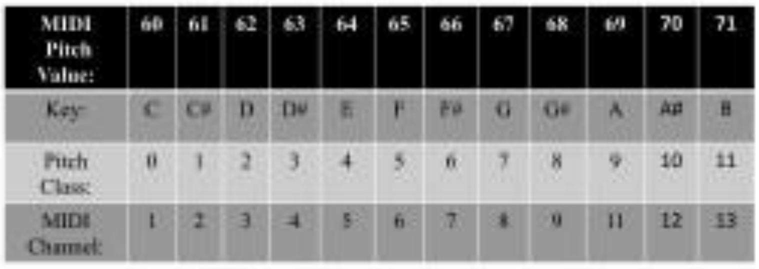


Diagram 1: Data Flow Diagram for D#4 Key to Be Lowered for 25 Cents.

PB message can be formed by any cent values determined by the user. If we go on with the sample in the diagram, cent value given as 25 (since it is to be lowered it is written as -25) is first transferred to PB values of 14 bites. When it is calculated with the scale prepared for the transformation (PB Value = (40.69 x cent Number) + 8192) PB value correspondence of -25 cent value is found as 7168 units. If PB value is known and wished to directly sent to the exit port via 4th channel, the status byte to be added in front of the data bites should be 227 (224 + 3). For this reason, PB message which will make changes on 4th channel will be prepared before the note-on message (227 0 56).
Note-on message is triggered by an MIDI controller or software and raw MIDI data formed is called from the exit unit and this data is transformed to the data bytes of note-on message. So, the pitch and velocity values of note-on message are decomposed. In the sample in diagram, pitch value and velocity value is given as 63 (D#4) and 105, respectively. PB message waiting ready on 4th channel will be processed when the note-on message is sent via the same MIDI channel. For this reason, 4th channel status byte is added on the pitch and velocity values which consist of the data bytes of the note-on message and then sent to the exit unit. (147 63 105). However, the note- on messages should pass through the pitch class (PC) rotation filter before they are sent to the exit unit.

Table I. Classification of Pitch Class and Channel Number According to the Pitch Values



MIDI Pitch Value:	60	61	62	63	64	65	66	67	68	69	70	71
Key	C	C#	D	D#	E	F	F#	G	G#	A	A#	B
Pitch Class:	0	1	2	3	4	5	6	7	8	9	10	11
MIDI Channel:	1	2	3	4	5	6	7	8	9	11	12	13

PC, is the value gained after the MIDI pitch numbers between the 0 and 127 units in the base are transformed into the base of 12. With this way, all the pitch between 0 and 11 happens to be classified with the figures. As it is seen in the table above, MIDI pitch value is either 48 or 60, this will not change the PC value, whichever octave the C note stands in, PC value is 0, channel number will be 1. If we go back to our example, for the D#4 key whose pitch value is 63, PC value will be 3. With the PC value added in front of the note-on message, note-on message happens to be addressed to the PB message with the same channel number. As it is seen in the diagram, D#4 key with number 63 will be sent together with the PC value with number 3 via 4th channel. For this reason, note-on and PB messages happen to be sent to the exit via the same channel and MIDI pitch with number 63 is vocalized with -25 cent value. In order to monitor all the MIDI pitches, this method can be applied on all the MIDI pitches. So, all the cent values of the keys named chromatic can be controlled whichever octave they are in and these values can be used for the permanent MIDI tuning change.

2. 2. Programming with Max/MSP

Max/MSP (www.cycling74.com) is a graphic based programming language and it owns object libraries to process MIDI, audio, video and make user interface. Max/MSP (MM) objects are the functions with certain process and limits. Objects which communicate with each other by binding with patch cables, are re-bound by MM software developers in order to perform certain purposes and this algorithmic formed with these bound objects is called as a whole patch. Patches come together and becomes the subpatches of the bigger patches. Patch which reaches to a certain process power can be recorded as a patch object again. However, MM can give a 'source code open' print which shows the patch objects used in the patch and the bindings. All the sub patches forming the whole patch, patch objects and patch cable connections can be recorded by maxpat txt or JASON format. In literature, there exist more detailed information about patch, patch object and MM programming [12-14].

This study, in which Max/ MSP 7.1. update is used, aims to explain the patch sample prepared only for one key (D#4) instead of patch object or subpatches programmed in this search so it is expected that the reader will form a parallelism between diagram and the figures and data flow diagram explained above. For this reason, the colors used in diagram are chosen similar with those of cables and patch object.

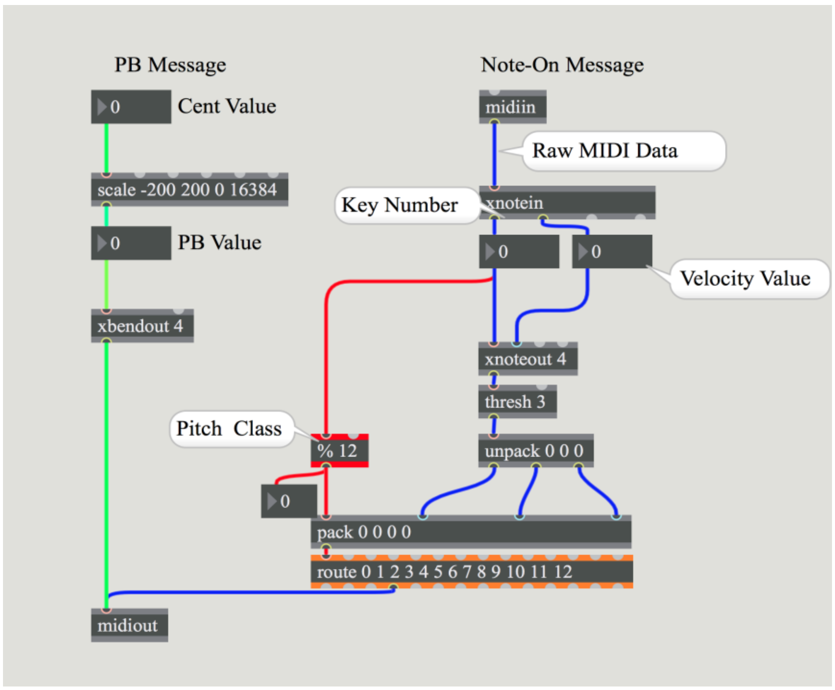


Figure. 3: Max MSP Patch Which Makes Key D# = 25 Cent Lower Pitch

In Fig. 3, only the patch sample which makes key D#25 cent more is shown. The fact that argument value of the *xbendout* object on the patch is chosen as 4 means that this MIDI object will communicate only via 4th channel. With the channel number information, status bite of the PB message is formed and it is sent to the *midiout*object which is a MIDI exit unit. On the right, under the note-on message heading, *midiin* object sends the raw data (144 63 105) which it accepted from MIDI equipments and software to the *xnotein* object and here Note-on message is separated into data bites. *xnoteon* 4 object adds the 4th channel status bite on the note- on message and it is sent to the route object which acts as a rotation filter and determined by PC. % object will give PC 3 value for the 63 key value. So, the note-on message for D# key comes out of the 4th exit of the route object as a raw MIDI data (147 63 105) and reaches the *midiout* object and there it combines with PB message. As a result, number 63 D# key becomes 25 cents lower. Vocalization process lasts when the note- off (147 63 0) message is sent following the note-on message. Patch prepared with this method is applied on other keys so, a patch object happens to be programmed for all the keys in the sequence in the chromatic sequence.

After patch object was programmed, the user interface of the patch was prepared. On the interface, in order for the user to control the cent value of each pitch, slider objects which provide the real time communication were added. In order for the cent value changed pitches to play back with 128 different instruments of GM, program change MIDI message was re-programmed as it became multi channeled. MIDI in and out ports for the communication with the outside equipment and software were formed. A cyber keyboard by programming ASCII codes was prepared. So, patch could be triggered by MIDI equipments outside. Qwerty computer keyboard was also added for the microtones to be played as an auxiliary tool.

For the notation support of the user's interface, score and roll objects and various human interface objects connected to these objects in Bach external (www.bachproject.net) were included in the patch. Upon transport object which provides support for the roll object was added, real time microtonal performance data were shown on the staff with proper accidental signs and rhythmic values. score object designs the MIDI data it took from roll object according to the quantization values which is determined by the user and numerous musical variable (unit, beat, speed, rhythm (balance), clef choose, etc.) and it can combine with the proper accidental signs. score object also supports MIDI, Music XML, OpenMusic and text export formats. Some microtones can be played by Bach. *ezmidiplay* objects usage of connected to role and score objects however, the cent solubility of this object is 25 units, almost ¼ of the half interval. Even if the recorded microtonal value is smaller or bigger than the 25 cent value Bach. *ezmidiplay* object will vocalize this note by rolling it up to either 0 or 25. As a solution to this problem, usage of Bach. *ezmidiplay* objectwas given up and patch object this was programmed in this study was used. So the patch object happened to load a two sided duty; One of its duties was when it was triggered by one of its connected MIDI equipment (for ex, MIDI keyboard), to play with the microtonal cent values determined by the user, and the other one was to re-address the MIDI data taken from the score object for the pitch values whose cent values were changed.

III. Conclusion

At the end of this study, programmed patch object plays the pitches having microtonal cent value by creating permanent change on MIDI tuning. The cent value of all MIDI keys in an octave together with the added objects on the user interface of the object can change between -99 and +99 units. With its this aspect, patch object has the feature of being the first software which was written with MM programming language and could be played by GM instruments by being able to make cent changes for all the chromatic intervals. Played microtonal pitches were able to be notated by means of Bach external used on the user interface of the patch and reversely, patch object made the Bach External gain the high solubility played feature.

Cent values were changed with the MIDI keyboard connected to the patch and microtonal intervals were tested for the possible problems. Bandwidth was forced in order to observe the the possible data loss due to the MIDI network, microtonal playing trials were done upon MIDI keyboard and as a result no data loss were observed. All the MIDI performance data played by using the MIDI keyboard were recorded with its note values. Music XML file taken from the patch was opened in Finale 2014 software and all the performance data with its accidental signs (Music XML format can allow up to the quarter tone) and cent values was played on this software.

Besides all these, this patch functions as a tonal-atonal converter also. It can call MIDI, MusicXML, Txt and Open Music formats and it can exchange the cent value of the MIDI keys which is arranged according to the temperament tuning system with the desired cent value. So, after the tonal-microtonal process of exchange, the cent values of the notes called into the patch are exchanged with the microtonal cent values and accidental signs are added, then recorded again. With the Bach External support added in the patch, the cent value of the microtonal pitches, 1/4 (50 cent), 1/8 (25 cent) and 1/100 (1 cent) is shown with different symbols which express the solubility. These symbols attached to the notes, show the microtonal cent value in fraction or whole number format while the 1/4 and 1/8 solubilities are symbolized with traditional accidental signs.

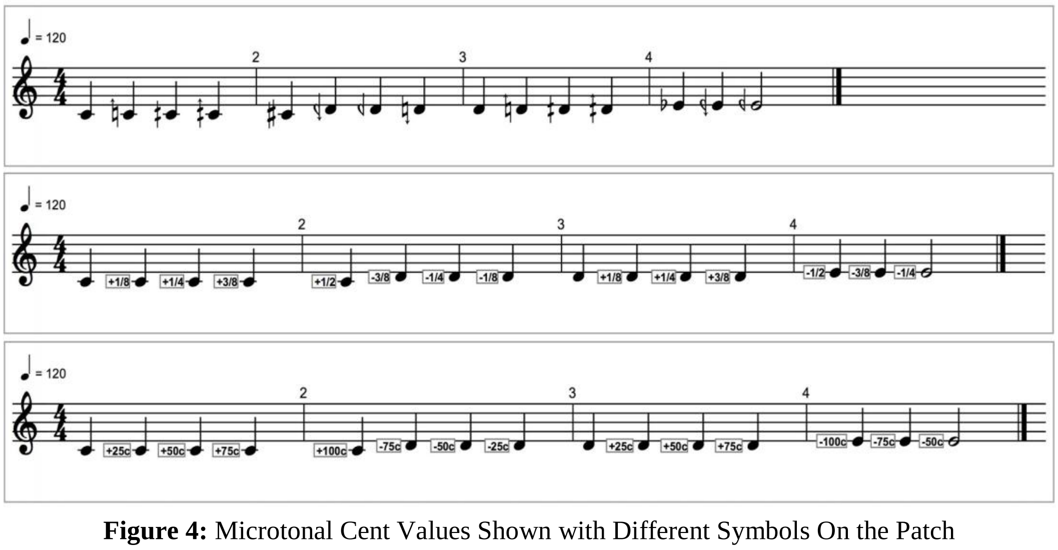


Figure 4: Microtonal Cent Values Shown with Different Symbols On the Patch

In order for the MM program developers to monitor the object relations and in the patch, output in the maxpat format was created and prepared patch Micro-MIDI was recorded with the name of object *microtune*. Moreover, in order for the prepared patch user interface to be used without the need of its loading into the system (MacOsX, Windows) exe and dmg, which works standalone, output were created. All the above mentioned files output was distributed through the http://www.serdarcelik.xyz and the programmed patch object and patch interface were opened to the discussion and with this it is aimed that the prepared patch and patch object in this study are to be a useful and free tool for the researchers, performers, programmers and everyone who is interested in microtonality.

Acknowledgements

This study has been supported with the Postdoctoral Abroad Research Fund by TÜBİTAK Scientist Support Division Head.

References

[1]. R. Guerin, *MIDI Power! The Comprehensive Guide (Power!)* (Course Technology Press, 2005)
[2]. A. Pejrolo, *Creative Sequencing Techniques for Music Production: A Practical Guide to Pro Tools, Logic, Digital Performer, and Cubase*. (Taylor & Francis, 2011)
[3]. D. Franz, *Recording and Producing in the Home Studio: A Complete Guide*. (Hal Leonard Corporation, 2004)
[4]. A. S. Moussa and T. P. Baker, A dynamic microtunable MIDI system: problems and
[5]. solutions. In *Automation Congress, 2002 Proceedings of the 5th Biannual World* (Vol. 13, pp. 339-344).
[6]. B. Bozkurt, M. K. Karaosmanoğlu, B. Karaçalı, and E. Ünal, Usul and makam driven automatic melodic segmentation for Turkish music. *Journal of New Music Research*, *43*(4), 2014, 375-389
[7]. B. Bozkurt and B. Karaçalı, A computational analysis of Turkish makam music based on a probabilistic characterization of segmented phrases. *Journal of Mathematics and Music*, *9*(1), 2015, 1-22.
[8]. A. C. Gedik, TürkMüziğiİcralarınınAnaliziiçinbir MIDI Projesi. *DokuzEylülÜniversitesiGüzelSanatlarFakültesiDergisi*, *10*(10), 2013, 81-92.
[9]. S. Çelik, A. Eden, G. Karşıcı and O. L. Öner, TürkMsksmMüziğiİçin Max/MSP TabanlıMikrotonalArayüzTasarımı, *Asos Journal of Academic Social Science,* Year:2, İssue: 1, 2014, 463-472.
[10]. A. S. Moussa, Perception-based microtuning over MIDI networks, IEEE *Multimedia*, 13(1), 2006, 56.
[11]. D. Keislar, History and principles of microtonal keyboards. *Computer Music Journal*, 1987, 18-28.
[12]. G. Hair, I. Pearson, A. Morisson., N. Bailey, D. Mcgilvray and R. Parncutt, The RosengardenCodicil:Rehearsing Music in Nineteen-Tone Egual Temperament. Scottish Music Review, Volume 1, No. 1, 2007, pp.99
[13]. A. Cipriani and M. Giri, *Electronic Music and Sound Design* (Contemponet, 2010).
[14]. E. Lyon, Designing *Audio Objects for Max/MSP and Pd*. (AR Editions, Inc., 2012)
[15]. V. J. Manzo, *Max/MSP/Jitter for Music: A Practical Guide to Developing Interactive Music Systems for Education and More*. (Oxford University Press, 2011)