International Journal of Electrical, Electronic and Communication Sciences
ISSN: 2517-9438
Vol:3, No:2, 2009

A Real-Time Signal Processing Technique for MIDI Generation

Farshad Arvin, and Shyamala Doraisamy

***Abstract*—**This paper presents a new hardware interface using a microcontroller which processes audio music signals to standard MIDI data. A technique for processing music signals by extracting note parameters from music signals is described. An algorithm to convert the voice samples for real-time processing without complex calculations is proposed. A high frequency microcontroller as the main processor is deployed to execute the outlined algorithm. The MIDI data generated is transmitted using the EIA-232 protocol. The analyses of data generated show the feasibility of using microcontrollers for real-time MIDI generation hardware interface.

***Keywords*—** Signal processing, MIDI, Microcontroller, EIA-232.

I. Introduction

BEFORE the invention of digital based music notation systems, paper-based musical notations and scores have been used for communicating musical ideas and compositions. Digital based music encoding an able simple editing, processing, and communication of musical scores. Music data are multi-dimensional; musical sounds are commonly described by their pitch, duration, dynamics and timbre. Most music database and data management systems use one or two dimensions and these vary based on types of users and queries.

There are many formats in which music data can be digitally encoded. These formats are generally categorized into a) highly structured formats such as Humdrum [1] where every piece of musical information on a piece of musical score is encoded, b) semi-structured formats such as MIDI in which sound event information is encoded and c) highly unstructured raw audio which encodes only the sound energy level over time. Most current digital musical data management systems adopt a particular format and therefore queries and indexing techniques are based upon the dimensions of music information that can be extracted or inferred from that particular encoding method.

Systems such as voice to MIDI, whereby format conversion and transcription of real-time audio signals to MIDI data stream are required, are currently been research and developed widely [2-4]. MIDI is a popular format in computer based music production such as composition, transcription, and so on. Size of MIDI files is very smaller than other music formats, because MIDI data consists of text messages that are defined instructions only and not sounds signal representation. It is suitable data format to be utilized in software application development. With Voice to MIDI system, we will be able to generate MIDI data of analog input rather easily. This system enables to convert a melody with a microphone to digital scores like MIDI. It extracts acoustical characteristics such as pitch, volume, and duration by intelligent algorithms and converts them into a sequence of notes for producing music scores. Thus, melodies will be translated into chromatic pitches without human intervention [5].

Challenges faced commonly in voice to MIDI systems include the clarity of input data for acquiring suitable results. In some systems such as [6], we must sing music simply with “ta ta ta …” expressively singing to prevent many inaccurate outputs. So, in those methods we are being forced to sing unnaturally.

The quality of MDI transcribed would also depend on the hardware capabilities. Also, some systems are based on software that needs intelligent algorithms for providing better quality music transcriptions. However, most of these systems use Digital Signal Processor (DSP) to process audio signals. A microcontroller as the main processor to process real-time audio signals is investigated in this study. The proposed technique is implemented completely with this microcontroller without a need for much complex calculations.

In this paper, we introduce an audio to MIDI transcription module encompassing a microcontroller and a pitch tracking algorithm. A hardware based real-time converter is offered which uses a real-time algorithm for implementing medium quality MIDI generator. The main aim is to extract music information from the voice signals to convert to MIDI representation. This hardware must be able to estimate some parameters such as pitch, note onset time, and duration, from the audio signals and generates MIDI messages.

The rest part of this paper is organized as follows. Section II describes the MIDI encoding algorithm for real-time signal processing. Section III describes the hardware architecture that is proposed for implementing the algorithm. Section IV describes the interface hardware modules. Experimental results are presented in section V.

Farshad Arvin and Shyamala Doraisamy are with the University of Putra Malaysia, Faculty of Computer Science and Information Technology; Department of Multimedia ; 43400 UPM Serdang, Selangor, Malaysia (e-mails: gs21875@ mutiara.upm.edu.my, shyamala@fsktm.upm.edu.my).

II. MIDI Encoding Algorithm

The encoding algorithm implemented with the hardware circuit converts real-time sampled audio signals to standard MIDI data. This allows real-time voice processing without using complex calculations such as FFT. A high frequency microcontroller is used as the main processor to implement the real-time digitisation algorithm.

The MIDI coding method that is used for processing real-time sounds, extracts single track MIDI events. This real-time conversion algorithm consists of two fundamental steps: a) peak detection using some analog components for converting voice to digital data and fundamental functions for peak detection. b) extract pitches and notes from detected peaks which employs several functions to extract MIDI parameters and output sequences. In Fig. 1 the proposed algorithm that is used to generate MIDI codes is shown. The process starts with capturing analog sound signals and then converts to digital form. The sound will be captured with a microphone and after filtering it will eliminate unwanted frequencies. It will then be sent to the peak detector module. The output of the peak detector module is peak streams which are used in the next module for extracting pitches. The note retrieval module processes the input peaks and generates notes. Also, note start times and delta times are estimated in this retrieval module.

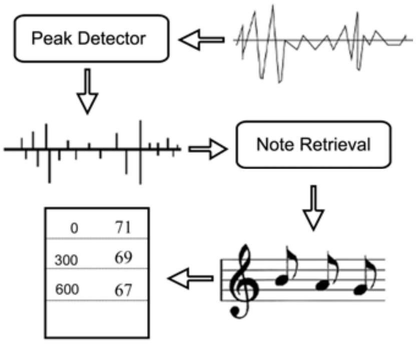

Fig. 1 MIDI encoding sequence diagram

A. Peaks Detection

In musical transcription, fundamental frequency is the significant part of a wave for conversion. The musical sound is a composite of many harmonics. While the fundamental frequency gives the sound its pitch, the harmonics give the sound its characteristic timbre. The sounds of a violin and piano are different even if they are all playing the same pitch. The difference is caused by the complex mixture of harmonics from each instrument as shown in Fig. 2.

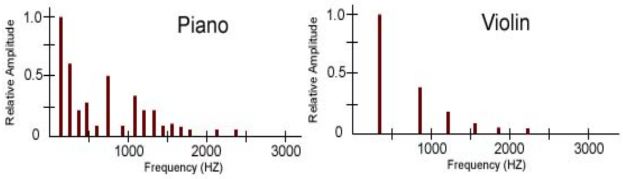

Fig. 2 Harmonics of similar notes played with Violin and Piano

Harmonics, which occur at integral multiples of frequency, often confuse pitch extracting and make it more difficult to decide the fundamental. Hence, many harmonics are eliminated with input filters.

Hardware receives digitised signal from input. The first row of Fig. 3 shows the analog signal input and the second row is the digital state of the input. The peak detector unit uses the digital sate of the sound wave for detecting fundamental frequencies.

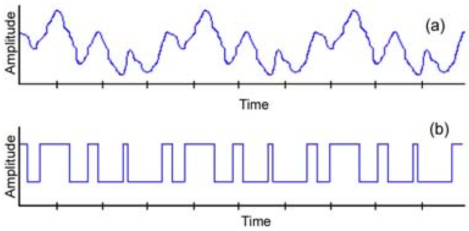

Fig. 3 (a) captured waveform signal with microphone, (b) converted input signals to digital pulses

Signal section is extracted by amplitude slicing a group of sequential significant peaks. All peaks that their absolute levels $|V(p)|$ are more than the specified threshold value ,will be selected. Moreover, every peak of each section that has a similar fundamental frequency is subdivided by a fundamental frequency of each peak. This subdivision allows finding the difference between note numbers of each pair of peaks in section [7]. Calculated fundamental frequency will be send to Note Retrieval unit for note extraction.

B. Note retrieval:

The note number $N(p)$ is calculated by fundamental frequency $f(p)$ with the following formula:

$$N(p) = 40\log\left(f(p)/261.6\right)+60 \tag{1}$$

The MIDI note number 60 is the musical note name C_4 with frequency value 261.6 Hz, or middle C note in musical instruments. This formula indicates, if the value $f(p)$ is increased to 2 times, the value of 12 which is an octave interval will be added to the $N(p)$.

Each calculated section is equivalent to one musical note, and will generate MIDI encoding based on the standard MIDI file formats [1]. Basically, there are two important event messages, which are a Note-On and a Note-Off. So, the Delta Time value will be calculated before each command as follows:

Delta Time 1, Note-On, Note Number 1, Velocity 1
Delta Time 2, Note-Off, Note Number 2, Velocity 2

Defined MIDI code for Note-On command is the hexadecimal value “9x”. Also “8x” value is defined for Note-Off command. In these commands, x shows channel number of each note. Note Number 1 and 2 is calculated by the formula (1), and Velocity 1 and 2 is also the same value that are given by (2) as follows:

$$sqrt\left(V_{\max}(s)\right)\cdot 127 \tag{2}$$

III. Implemented Hardware Architecture

This section describes a designed decoder hardware that generates MIDI commands such as notes and control messages. An ATMEGA168 microcontroller is used in described algorithm implementation. The microcontroller performance of 20 MIPS throughput at 20 MHz supplies enough rates for implementing encoding algorithm. Also, each IO of this microcontroller works in both digital and analog modes simultaneously. This chip is employed as the main processor which provides serial MIDI data from the received microphone analog signals. The serial data is sent to PC to be saved in the MIDI format. Moreover, external memories are used for caching outputs and buffering. These memories are utilized in offline mode as well. The proposed method is implemented with C language and some critical functions of algorithm are implemented with assembly language. Experimental results show that, Assembly routines provide better results than C functions in some critical routines. Thus, timers and ADC (Analog to Digital Converter) interrupt routines are written in assembly instructions to prevent probable errors in extraction calculations. Fig. 4 is the architecture of the proposed system.

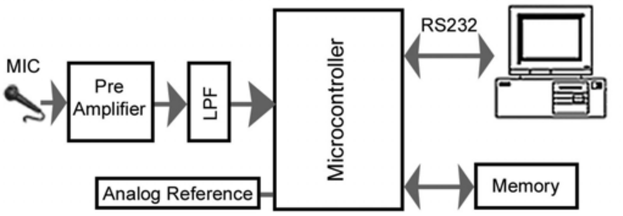

Fig. 4 Architecture of MIDI converter hardware

The Universal Synchronous and Asynchronous serial Receiver and Transmitter (USART) is a highly flexible serial communication device. This device is used for transmitting MIDI data to PC. The serial RS232 protocol is selected for communicating between the designed hardware and PC [8]. The hardware sends MIDI contents as serial bits in 38.4 Kbps. Therefore, a TXD interrupt should be enabled to prevent data loss. This interrupt routine helps the system to transmit MIDI streams. The implemented method does not need any DSP. Just one microcontroller with high frequency external oscillator is enough for getting good results, although DSP results have better quality in comparison with microcontrollers. However, the selected microcontroller executes all algorithms with limited error rates. The price of microcontrollers is also cheaper than low-cost DSPs. Thus, it is an important reason for choosing microcontrollers as a processor in a lot of projects. Also, microcontrollers are installed in simple boards and do not require complicated hardware.

The preamplifier unit is used between microphone and microcontroller that provides better wave form to process. The preamplifier output signals are detected with the analog unit of the microcontroller. The ADC's typical resolution is 10-bits and maximum ADC frequency is approximately 625 KHz. Signal components higher than the Nyquist frequency ($f_{ADC}/2$) should not be present for either kind of channels, to avoid distortion from unpredictable signal convolution. Hence, a low-pass filter is used for applying the signals as inputs to the ADC to remove high frequency components. The cut-off frequency is calculated with the following formula:

$$f_c = \frac{1}{2\pi\tau} = \frac{1}{2\pi.RC} \tag{3}$$

The resistor and the capacitor values are calculated for 3.40 KHz (R=470Ω and C=0.1µf). A simple filter circuit is employed for this implementation as shown in Fig. 5.

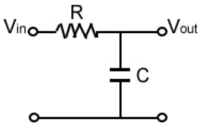

Fig. 5 Low-pass filter is used for input line filtering

A normal conversion takes 13 ADC clock cycles. Thus, ADC clock should be 572 KHz for providing 44 KHz sample rate. If a lower resolution than 10 bits is employed, the input clock frequency to the ADC can be higher values to get a higher sample rate. Consequently, 8-bits sampling size is selected for our implementation that each sample value is between 0 and 255. For single-ended conversion, the result is:

$$V_{ADC} = \frac{V_{IN} \cdot 256}{V_{REF}} \tag{4}$$

Where V_{IN} is the voltage of the input pin and V_{REF} is the selected voltage reference. An ADC interrupt routine is used for buffering samples. This routine will be called after each conversion value automatically.

As we stated, working in analog and digital modes simultaneously for each input pin is suitable feature of this microcontroller series. So, this feature is used for peak detecting unit of suggested algorithm.

IV. Interface Hardware Modules

The microcontroller is employed as the main processing unit which works with an external 20 MHz clock source. This frequency provides the hardware to execute defined tasks. Furthermore, two serial external memories are used in the buffer unit. Also, these memories will assist the processor in an offline mode generating MIDI codes. The reference voltage stability is a significant subject in using ADC unit, because its output will change if V_{REF} is not constant, as in (4). For this reason, XC6206 component is used in the presented implementation reference unit. The XC6206 series are precise, low power consumption, high voltage, positive voltage regulators manufactured using CMOS and laser trimming technologies. The series provides large currents with a significantly small dropout voltage as shown in Fig. 6.

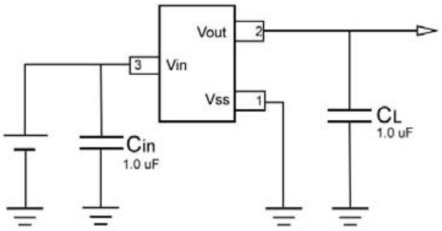

Fig. 6 Schematic of xc6202 series voltage regulator

SMT components are selected for the implementation of the designed main board. One of the SMT component benefits is the reduction in noise. This is primarily due to smaller electrical paths compared to leaded components. This feature is very useful in high frequency implementations where low noise contribution is mandatory and is a design feature.

In addition, simple software is written which saves received serial data in hard disk with '.mid' extension. This software is a suitable test bed for acknowledging implemented algorithm. It uses PC serial COM port for receiving MIDI data.

V. Experimental Results

The designed hardware was tested with several sound samples that were played with an electronic keyboard. This section describes the real-time signal processing technique for a simple sound wave that is played with piano sound. A piano tone constantly changes in timbre as it decays after it is sounded. Fig. 7 shows original sound waves that are piano sounds. The first row wave is captured by a simple low quality microphone and the second row wave is captured by a high quality microphone. Experimental results show that, microphone quality is a significant reason to provide suitable outputs.

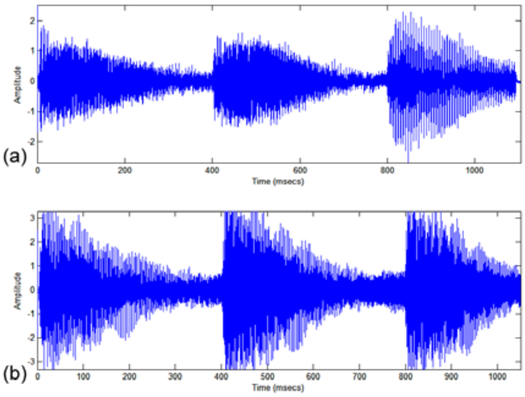

Fig. 7 Captured sample sound waves. (a) captured by low quality microphone. (b) captured by high quality microphone

Fig. 8 (a) is the energy diagram of the signal captured with the low quality microphone and Fig. 8 (b) is with the high quality microphone. The energy level of sound is a significant parameter to estimate exact pitch number from frequencies. Low energy sounds will be eliminated when the peak detector module processes the input signals.

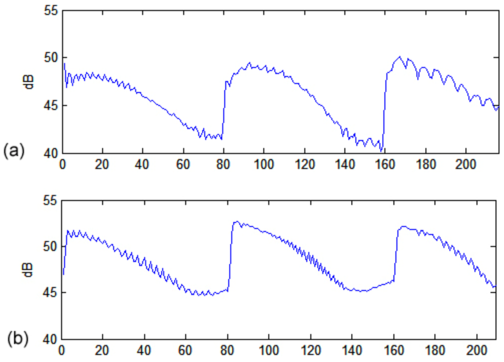

Fig. 8 Energy diagram of captured sample waves. (a) low quality microphone. (b) high quality microphone

Fig. 9 shows frequency diagram of sample wave. As the diagram illustrates, there are three levels of frequencies in the sample wave that are combined with some noises. C_4, D_4, E_4 were played with piano and were captured with our hardware interface.

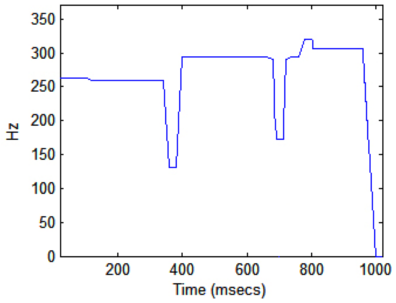

Fig. 9 Frequency diagram of sample wave

To implement this proposed hardware, real-time voice signals were divided in to several windows as shown in Fig. 10. In these windows, samples are captured and saved to an array in memory. After each sampling period, the microcontroller will process these samples to find the fundamental frequency. When the size of windows gets bigger, a lot of samples will be captured and output will be near to real frequency. But if the window size is large, the processor will not have enough time to process the captured data. In addition, selecting a big size window can result in losing notes with short durations.

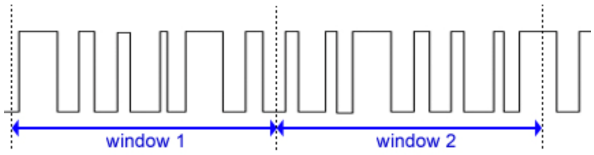

Fig. 10 Input signal is divided into several windows

In Table I, there are three different outputs from differing window sizes. In those experiments, duration times of 10, 100, and 500 ms were selected to process the same played notes. In each window size, 200 notes were played and results were obtained. The true extracted notes percents are shown in note durations. Middle size windows have average result in both notes size. Hence, 100 ms window size was selected for next step of experiments. Dynamic window sizes could possibly generate better results for differing note sizes. However, this will have to be further investigated. For implementing dynamic window sizes, we have to detect gaps between played notes and estimate optimal window size for getting improving the performance.

TABLE I
PERCENTAGE OF CORRECTLY TRANSCRIBED NOTES

Window Size	Short duration notes	Long duration notes
10 ms	30 %	40 %
100 ms	75 %	80 %
500 ms	60 %	70 %

After selecting window size, the next step in the experiments was obtaining output scores from playing three sequential notes. Proposed hardware produced similar notes with original played notes but they are not the exact same notes. Fig. 11 (a) is the original notes that are played with electronic keyboard. Fig. 11 (b),(c) are real-time transcribed notes with the proposed hardware in none silent environment. These outputs were not good results. Fig. 11 (d) transcribed notes with isolated hardware in silent environment. That hardware employs some filters in power supply unit and signal input. Also, used separated power supply for analog unit and digital unit with separated ground signals.

The filtering hardware provides better MIDI encodings from the sound waves. In the last experiment, sounds were captured in silent room and also a high quality fixed microphone was employed for getting suitable results. Results show that, real notes retrieval will be accessed in ideal environment. Therefore, some hardware units must be improved for getting suitable results in noisy environments.

VI. CONCLUSION

This paper presents a new designed hardware for MIDI standard that transcribes real-time voice to MIDI data. The implemented transcription algorithm with microcontroller processor is an aim of this paper. Using simple algorithm for real-time processing and hardware circuit for MIDI interface without require any DSP is a feature of this paper. The experimental results show that the proposed hardware generates good transcribed output in silent environment. Also, some hardware filters can assist to get excellent results. For future work, dynamic windowing approach will be implemented for getting better results in all note durations. This implementation can be improved to multi-track MIDI code generator.

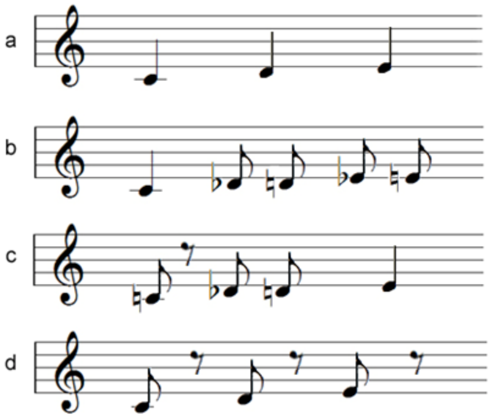

Fig. 11 (a) original played notes with electronic keyboard, (b),(c) transcribed notes in none silent environment, (d) transcribed notes when used some filters in hardware power supply and isolated analog and digital units

REFERENCES

[1] Eleanor Selfridge-Field., “Beyond MIDI”, The MIT Press, 1997.

[2] Mark Nelson “Getting Started in Computer Music”, Thomson Course Technology PTR, 2006

[3] Muramatsu T., Hai Q., Hashimoto S., “Sound database system retrieved by sound,” IPSJ Proceedings of 54-th National Conference, 1997.

[4] A. Ghias, J. Logan, D. Chamberlin, and B. C. Smith, “Query by humming: Musical information retrieval in an audio database.” Cornell University,1997

[5] N. Itou, K. Nishimoto, “A Voice-to-MIDI System for Singing Melodies with Lyrics” *ACE 2007*, Salzburg, Austria, 2007

[6] Jun, S., Takeshi, M., Masanobu, M. and Masuzo, Y., “Automatic Scoring of Melodies Sung by Humming” *Tech. Rep. Musical Acoust..* Soc. Jpn., Vol.23, No.5, pp.95-100, 2004.

[7] Toshio Modegi1, Shun-ichi Iisaku, “Proposals of MIDI Coding and its Application for Audio Authoring”, *MMCS, IEEE International Conference*, pp 305 – 314 , 1998

[8] Popa, M.; Popa, A.S.; Cretu, V.; Micea, M. “Monitoring Serial Communications in Microcontroller Based Embedded Systems” *ICCES,* pp 56 – 61 , 2006