Want to Be A Maestro? A Simulation System Implemented on FPGA

Zong-Min Lin*, Chun-Hao Hsieh#, Dong-Yu Gao*, and Yin Chang*

#Department of Dentistry, National Yang-Ming University

* Department of Biomedical Engineering, National Yang-Ming University

zongminlin@livemail.tw, z2567989@hotmail.com

tonygaogao@hotmail.com, yichang@ym.edu.tw

Abstract— **With a change of gesture, a conductor can control every division of the concert band in our system. By wearing a glove with infrared emitters in one hand, and holding the baton equipped with the ultrasonic transmitter in the other hand, the system is made to detect the changes of the gestures and the movement of the baton. Our two simple notions: an infrared emitting and receiving mechanism, and the Doppler effect. After sensing these stimuli, the system transmits these data to FPGA for analysis. The Musical Instrument Digital Interface (MIDI) processor then reads these analysed data, and modifies the speed and loudness of the music synchronously. Eventually, the music is played by the electric piano. By means of changed gestures, the conductor can alter the characteristics of music at his/her will, and hears the sound of modified music synchronously.**

Keyword— Doppler effect; Infrared; MIDI; Conductor; Ultrasound

I. INTRODUCTION

It's such a wonderful and elegant experience for us to listen to incredible and stunning music performances in our daily lives. Slowly immersed in the beautiful, and intermittent musical sound, we can see a great conductor dressed in a black tuxedo, standing on stage, and making diverse gestures, which shows the speed, and intensity of the music, in order to enable the concert band to perform melodious music. This terrific scene inspires us to simulate conducting a concert band where various gestures can be used to control the division of the concert band.

Generally, a good conductor has several works to do during the musical concert:

A. Performance Instruction

The duration of a musical concert is often so long that it is difficult for performers to remember all the music details. Therefore, performers in different divisions of the concert band require the instruction from the conductor.

B. The Determination of the Characteristics of Music

A wonderful musical performance filled with the mixture of different musical characteristics – the musical speed and volume – is indispensable to a conductor.

C. Improvisation

Sometimes a subtle improvisation added into the music performance is necessary for a conductor.

By using infrared technique and the Doppler effect, we can simply sense the changes of the gestures and the movement of the baton respectively. Then, after receiving these data from sensors, they are transferred to FPGA for further analysis. Besides, music file stored in the SD card are also read to the FPGA. Once the analysis of sensed signals in FPGA is completed, the result will reflect the music. And eventually, the combination of analysed data and music data is transferred to the electric piano to play the modified music. With this simple notion, we can create a simulated symphony, in which by changing gestures, the conductor, the user, can alter the characteristics of music at his/her wills, and hears the sound of modified music synchronously.

Musical Instrument Digital Interface (MIDI), is a technical standard depicting the way in which digital interfaces and connectors allow a wide variety of electronic musical instruments, computers and other related devices to connect to and communicate with one another. MIDI has the capability to carry event messages that specify notation, pitch and velocity, control signals for parameters such as volume, vibrato, audio panning, cues, and clock signals that set and synchronize tempo between multiple devices.

II. SYSTEM FRAMEWORK

In our system, we act as a conductor to operate the simulative symphony. A conductor masters the tempo and volume of the different divisions of the concert band through changes of gestures and the baton. The signal processor and command generator block in the system receives and processes the stimuli, or namely orders from changes of the gesture, and adjusts the musical speed and sequences of the music played by the MIDI synthesizer, and also creates the simulative musical performance.

The system design diagram is shown in Figure 1. The conductor stands at the center of the system, and is surrounded by boxes, which are represented as different divisions of the concert band. Every box is embedded with two infrared receivers, and oriented to the conductor. The arrangement of the musical division is in a semicircle shape

for even effect of the system. There are also two microphones in the left and right-hand side of the conductor for receiving the baton's movements.

When acting as a conductor in our system, a conductor is required to wear a pair of gloves, and then holds a baton with the ultrasonic transmitter in the right hand. Infrared emitters are installed in both the center of the palm and the top of the left index finger. The former is for controlling the playing of any division of the concert band, but, on the contrary, the latter is to stop it.

The operation in our system can also be divided into two systems: the symphonic division control system and the speed control system. In the former system, we need two infrared receiver and emitter pairs to operate the start and stop of the performance in every symphonic division; the latter one is based on the Doppler effect to control the speed of the music through microphones and a baton with ultrasonic transmitters.

A. Symphonic Division Control System

The concert band always needs instruction in a perfect musical performance, for instance, the proper time to play music, the musical volume, the mood of the performers, and so on. By means of the infrared technique, we can control the performance at our will.

When wearing the left-handed glove with an infrared emitter installed at the tip of the index finger, we can easily point to any division of the concert band. Such will make that division start to play music as shown in Figure 2 (A) to (B). However, if we move our left palm towards any division of the concert band and then fist, it will result in stopping playing of this division shown in Figure 2 (C) to (D).

There are two infrared emitters in our left-handed glove, but their emitting frequencies are significantly different. The infrared emitter at the tip of the index finger stands for the start of the performance, while the one in the left palm stands for the stop of the performance. When a conductor wants any of the divisions to perform music, he/she needs to point at the division with his left index finger, and holds on for a few seconds. Such can make the infrared receiver sense the lasting signal, and the control system will order the division to start playing music. On the contrary, if a conductor wants to stop the playing of one of the symphony divisions, he needs to move his left palm, orient it to the division and halt in order to let the infrared receiver sense the lasting signal. Once the conductor clasps his left palm, the infrared receiver cannot sense any further signal, making the control system mute the music of the division.

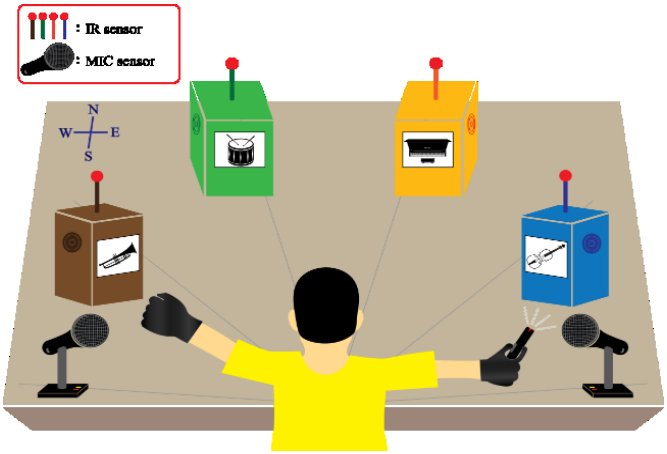

Figure 1. System Design Diagram

B. Speed Control System

Controlling the performance speed in conducting plays a crucial role in the symphony. For this reason, we installed an ultrasonic transmitter sounding a sinusoidal wave with the specific and high frequency in the top of the baton. Once the user waves the baton, both sides of microphones will receive the signal from the ultrasonic transmitter when the music still continues playing, causing a relative motion between these two units, and eventually, generating the Doppler effect shown in Figure 2 (E) to (F).

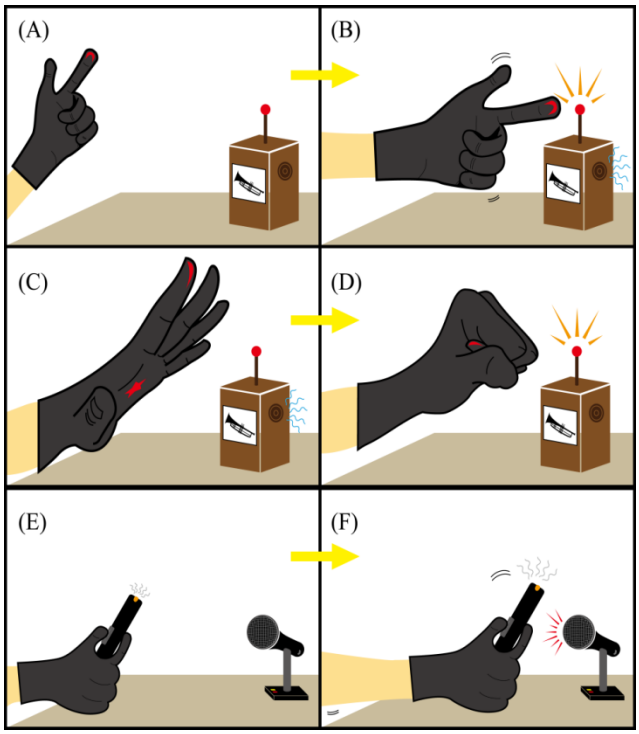

Figure 2. The gesture diagram. (A) to (B) illustrates the start of the performance to the division of the concert band by the infrared emitter at the tip of the left index finger; (C) to (D), the stop of the performance to the division of the concert band by the infrared emitter on the palm of the left hand; (E) to (F), the behavior between the microphone and the baton held by the right hand.

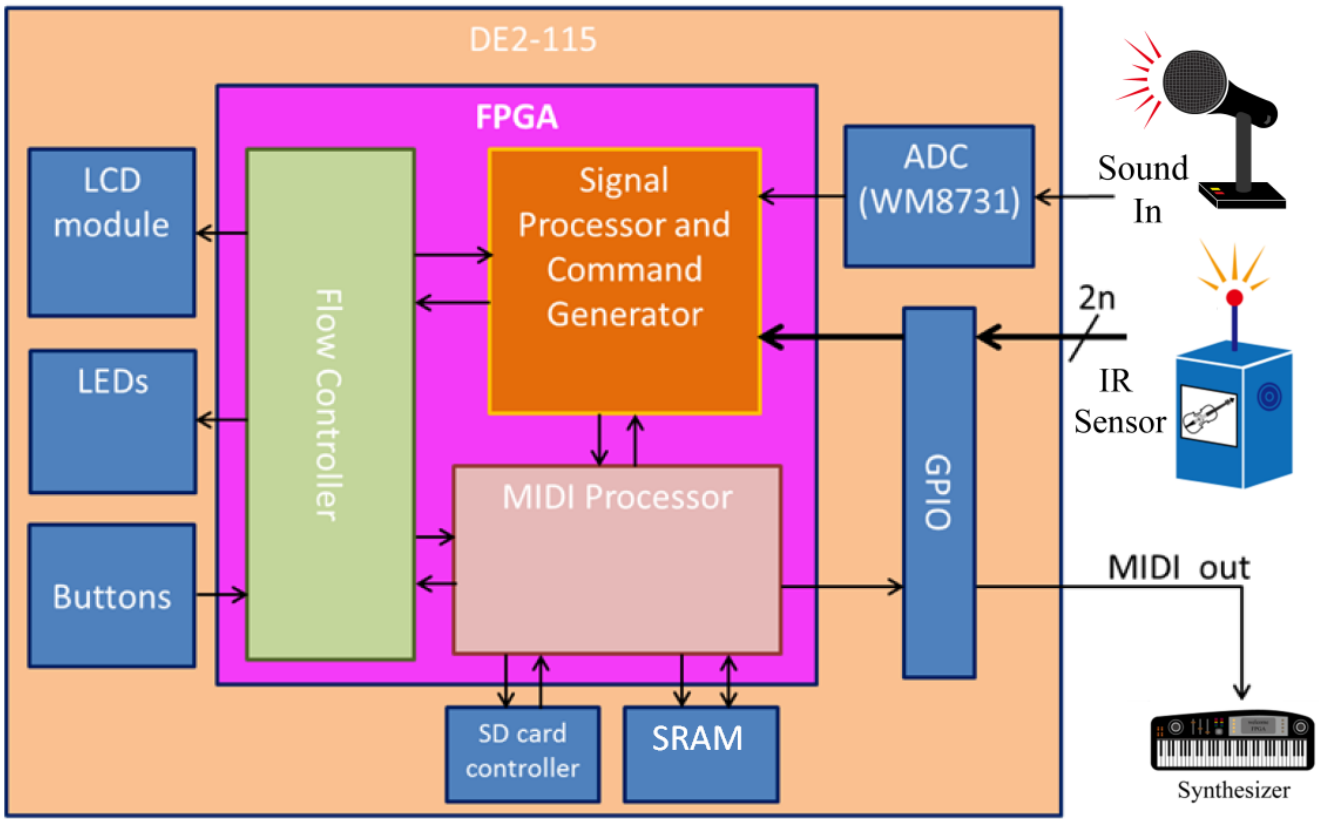

Figure 3. Block diagram of the system architecture

III. SYSTEM DESIGN

As shown in Figure 3, the whole system, implemented on FPGA, EP4CE115F29C7N, is divided into three main blocks:

- Flow Controller Block
- Signal Processor and Command Generator Block
- MIDI Processor Block

The Flow Controller Block has a finite state machine (FSM) architecture. Based on user input and some other information, it controls the process of the system by changing state registers. The Signal Processor and Command Generator block (SPCG) receives IR signals from peripheral general purpose IO pins (GPIOs) and ultrasound signals from a sound card. Then, SPCG analyses signals and generates commands to control the MIDI Processor. The MIDI Processor block is in charge of playing all of the MIDI messages, including decoding MIDI files, loading MIDI messages, retrieving MIDI messages and sending MIDI messages. More details are as follows.

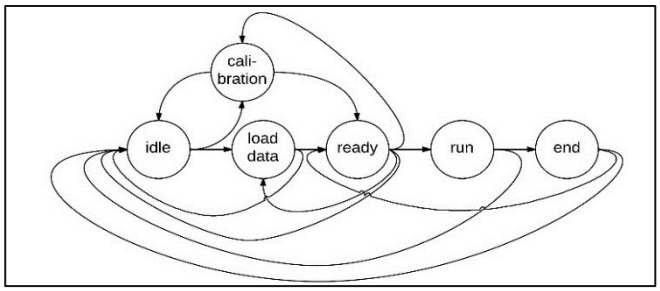

Figure 4. State flow of system

A. Flow Control – FSM

The system flow shown in Figure 4 includes six states, i.e. idle state, load-data state, calibration state, ready state, run state, end state. After reset, the system will be in idle state, waiting for the user to input a 'load-data' command and then it progresses to load-data state. When all data has been loaded and saved into memory, the system gets into ready state. In this state, the system is idle and waits for a 'start' command to progress to run state, in which the system starts playing music. In the end of the playing, the system turns into end state, then back to ready state, completing one playing process. In addition, the user can input a 'calibration' command to make the system turn into calibration state for alignment and calibration when it is in idle state or in ready state.

1) idle state: The state of initialization.

2) load-data state: In this state, the SMF data will be loaded from SD card and be decoded. All essential channel messages will be saved into SRAM in order, while tempo information will be saved in specific registers.

3) ready state: It's the idle state after data being saved in SRAM, waiting for user input to turn into next state.

4) run state: The system is playing. MIDI processor retrieves and transmits channel messages at specific speed, while SPCG is enable to receive IR signals and ultrasound signals and analyse those to generate commands.

5) end state: The state indicates the end of playing. Lasting a short moment, then it turns into ready state.

6) calibration state: The state undertaking calibration of some parameters in SPCG block for better analysis.

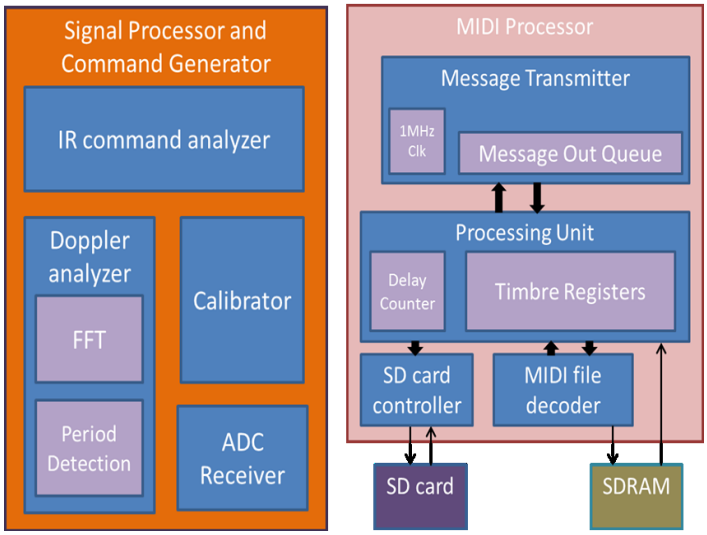

Figure 5. Block diagrams. Signal Processor and Command Generator (left) and MIDI Processor (right)

B. Signal Processor and Command Generator

Shown in Figure 5, there is an IR-command analyser, a Doppler-effect analyser, an ADC receiver and a calibration sub-block in this block. To expand, the Doppler-effect analyser is in charge of analysing ultrasound signals and detecting the period, while the ADC receiver block is responsible for connecting WM8731 audio codec as well as receiving audio data.

C. MIDI Processor Block

Including SMF-file loading and decoding, channel-message processing and transmitting, a core processor architecture, shown in Figure 5, is implemented in this block for all functions about MIDI data processing. MIDI channel message is saved in SRAM because the size of SRAM on DE2-115 is large enough to save a big amount of channel messages.

IV. TECHNIQUES

A. Playing and Muting the Instrument

The infrared modules are used to achieve these functions. Two IR transmitters with different wavelengths are fixed on the glove. One is for the function of playing an instrument, and the other is for that of muting an instrument. Each simulated instrument contains two IR receivers corresponding to transmitters. For simplification, the circuit of the transmitter on the gloves only contains two individual resistors to limit the current of the circuit. The most important issue of the receiver is 'debounce', so we implement 1 counter for debounce of the function of playing an instrument and 2 counters for debounce and delay of the function of muting an instrument.

B. Period Detection

Doppler effect, otherwise known as Doppler shift were first described by Christian Andreas Doppler in 1842. It explains the phenomenon whereby the relative motion between the observer and source of the wave results in a change in frequency of the wave.

Based on the equipment of DE2-115, we chose 40 kHz as the frequency of the ultrasound, which is still the standard frequency of ultrasound for range detection. On average, the range of BPM (beats per minute) of a song is from 30 to 180. Supposing that the range of motion of the conductor's hand is 50 cm or so, the frequency will shift to the range 40.030 kHz ~ 40.177 kHz.

Keniciro Aoki et al. described in an experiment that a pendulum source is giving out a 3520 Hz sound, through Discrete Fourier Transform (DFT) and plotting the peak frequency at each moment, the periodic variation of the peak frequency can be observed, shown in Figure 6. Similarly, supposing that the conductor's hand gesture moves like a pendulum, we can get the variation of peak frequency through DFT. Then using peak detection or regression, we can evaluate the period of the curve and further regulate the tempo of playing songs.

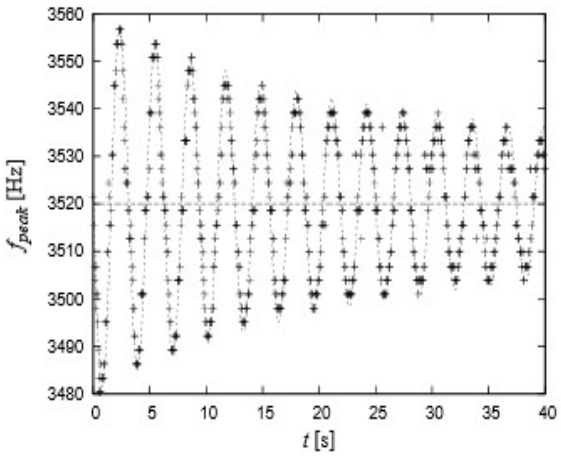

Figure 6. Simulation of peak frequencies measured from a sound source swinging as a pendulum

C. MIDI Files Decoding

The standard file format in MIDI system is SMF (Standard MIDI File, .mid files). MIDI files are organised into data chunks with each prefixed by a 4 byte header, as shown in Table 1. The header chunk prefixed as "MThd" contains information on the whole song including MIDI format type, number of tracks and timing division. 3 format types having a value of 0, 1 or 2, describe how the following track information is to be interpreted. A type 0 MIDI file contains all the MIDI events for the song, including the song title, time signature, tempo and music events.

Due to its simplicity, we chose the type 0 to simulate the symphonic band which contains 1 track and 16 channels to represent 16 divisions. There is only one "MTrk" truck or called track chunk which stores all the MIDI events in the file under type 0. Track chunks contain all of the information for an individual track including, track name and music events.

MIDI events are divided into three groups: channel messages, system messages and meta events. Most of the system messages and meta events are not used in the musical content so it will be ignored after decoding. In our system, SMF is first read sequentially from the SD card and the header trunk is recorded to determine the tempo for songs. Then, the channel messages and other messages regarding to musical content are stored in the SRAM and will be read by the processing unit while the music is playing.

TABLE 1

FORMAT OF CHUNKS IN SMF

	Types (presented in ASCII code)	Length	Data Format in Chunks
SMF	**0x4D546864 (MThd)**	**6 bytes**	***<format>,<tracks>, <division>***
	0x4D54726B (MTrk)	**variable**	***<delta_time>, <event>***

There are two factors controlling the tempo in the MIDI songs. It can be calculated in the formula below.

$$\boldsymbol{tick\ time(us)} = \frac{\boldsymbol{Tempo\ (us/beat)}}{\boldsymbol{division\ (ticks/beat)}}$$

The MIDI header chunk contains the time division used to decode the track event delta times into "real" time. This value represents either ticks per beat or frames per second. In our system, only the ticks per beat will be presented. The top bit of the word is 0 and the following 15 bits describe the time division in ticks per beat. Ticks per beat translate to the number of clock ticks or track delta positions in every quarter note of music.

The meta event, 0xFF5103, also plays a role in tempo control. It sets the sequence tempo in terms of microseconds per quarter-note which is encoded in three bytes. It is usually found in the first track chunk. If no set tempo event is present, 120 beats per minute is assumed.

D. MIDI Tracks and Volume Control

MIDI Channel Events are the most common type of track event and usually make up the bulk of a MIDI file. The Table 2 gives an overview of the five MIDI Channel Events, listing their numeric value and parameters.

TABLE 2

FORMAT OF MIDI CHANNEL EVENT

Event Type	Value (4 bits)	Channel (4 bits)	Parameter 1 (1 byte)	Parameter 2 (1 byte)
Note Off	1000	0000-1111	note number	Velocity
Note On	1001	0000-1111	note number	Velocity
Aftertouch	1010	0000-1111	note number	Aftertouch value
Controller	1011	0000-1111	controller number	controller value
Program Change	1100	0000-1111	program number	not used

MIDI Channel Events usually composes of 3 bytes. The first byte describes the commands and corresponding channels. The following bytes are parameter 1 and parameter 2. Both formats depend on the event types. For example, Note Off Event is used to signal when a MIDI key is released. Note number specifies which of the 128 MIDI keys is being played and the velocity determines how fast/hard the key was released. In our design, when the volume off gesture is detected by the IR, the MIDI processor will generate a "Note Off" signal to mute the channel. The Note On event is similar to the Note Off event. The volume on gesture is also detected and will be responded by a Note On signal.

E. MIDI Command Transfer

Hardware interface of MIDI dataflow is an asynchronous serial interface with a Baud rate of 31.25kbps. Data contains 0 as start bit, 1 as end bit and 8 data bits, transferring in LSB first mode. Our system reads the data in the queue of message out under 1 MHz clock and sends out the data under 31.25 kHz after frequency dividing.

MIDI hardware implementation is as recommended as Figure 7. DIN 5-pin connector is used to connect the MIDI IN port of the synthesizer.

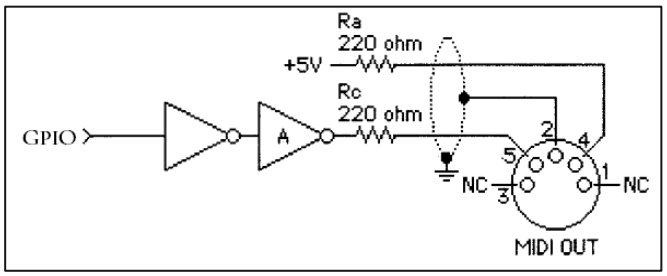

Figure 7. MIDI hardware design

ACKNOWLEDGEMENT

We acknowledge support of Altera corp. and Terasic Technologies corp. and thank prof. Chang Yin for encouragement as well as financial support. Also, we are indebted to prof. Tsao Jen Ho for helpful suggestions on ultrasonic techniques. Last but not least, Thank Stephanie Wen for proofreading.

REFERENCE

[1] Aoki, K., T. Mitsui, and Y. Yamamoto, Direct quantitative measurements of Doppler effects for sound sources with gravitational acceleration. arXiv preprint arXiv:0911.3819, 2009.

[2] Braem, P. and T. Bräm, A pilot study of the expressive gestures used by classical orchestra conductors. Journal of the Conductor's Guild, 2001. 22(1-2): p. 14-29.

[3] Chen, J.-m., Implementation of MIDI Synthesizer on SoC platforms. 2006.

[4] Roland Dictionary for Terminology. Available from: http://www.rolandtaiwan.com.tw/glossary/glossary.htm.

[5] MIDI Manufacturers Association. Available from: http://www.midi.org/.

[6] MIDI Messages Table. Available from: http://www.midi.org/techspecs/midimessages.php#3.

[7] Standard MIDI File. Available from: http://home.roadrunner.com/~jgglatt/tech/midifile.htm.

[8] General MIDI guide. Available from: http://www.midi.org/techspecs/gmguide2.pdf.

[9] Wikipedia - MIDI. Available from: http://zh.wikipedia.org/zh-tw/MIDI.

[10] MIDI 1.0 Detailed Specification, Revised September 1995. Available from: http://madamebutterface.com/assets/documents/MIDI%201.0%20Detailed%20Specification.pdf.

[11] WIKI Conductor. Available from: http://wikipps.hk/%E6%8C%87%E6%8F%AE/.

[12] The Platform of High-Scope Program - Doppler Effect. Available from: http://case.ntu.edu.tw/hs/wordpress/?p=2841.

[13] MIDI File Format. Available from: http://www.sonicspot.com/guide/midifiles.html.

[14] Wikipedia – Doppler Effect. Available from: http://zh.wikipedia.org/wiki/%E5%A4%9A%E6%99%AE%E5%8B%92%E6%95%88%E5%BA%94.