Signals and Communication Lab
DII University of Brescia - ITALY

Digital Audio Processing

Lab.4 - The MIDI standard - 2014

Alessio Degani, [alessio.degani@ing.unibs.it]
http://www.ing.unibs.it/alessio.degani

MIDI (Musical Instrument Digital Interface) is a technical standard that describes a protocol, digital interface and connectors and allows a wide variety of electronic musical instruments, computers and other related devices to connect and communicate with one another. A single MIDI link can carry up to 16 channels of information, each of which can be routed to a separate device.

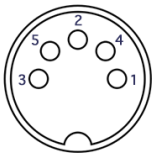


Figure 1: MIDI five-pin DIN connector. Standard applications use only three of the five conductors: a ground wire (pin 2), and a balanced pair of conductors (pin 4 and 5) that carry a +5 volt signal.

MIDI carries event messages that specify **notation**, **pitch** and **velocity**[1], **control signals** for parameters such as **volume**, **vibrato**, audio **panning**, cues, and **clock** signals that set and synchronize tempo between multiple devices. These messages are sent to other devices where they control sound generation and other features. This data can also be recorded into a hardware or software device called a sequencer or stored in a *.midi* or *.mid* file. In this Lab. class, we focus only on the **notes** messages (for example, **notes** messages can be generated by a electronic piano and can be sent to an expander[2]).

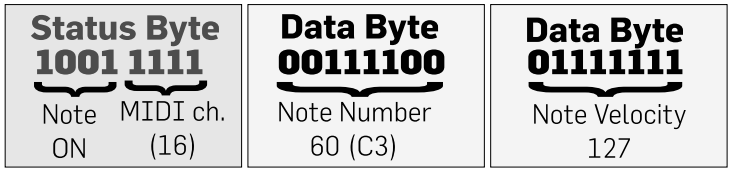


Figure 2: Example of MIDI noteON message.

The note number m is encoded using the **MTS** (MIDI Tuning Standard). The note's frequency f_m can be calculated from the midi note number m (and vice-versa) using the following formula (see the **Frequency chart** on the website):

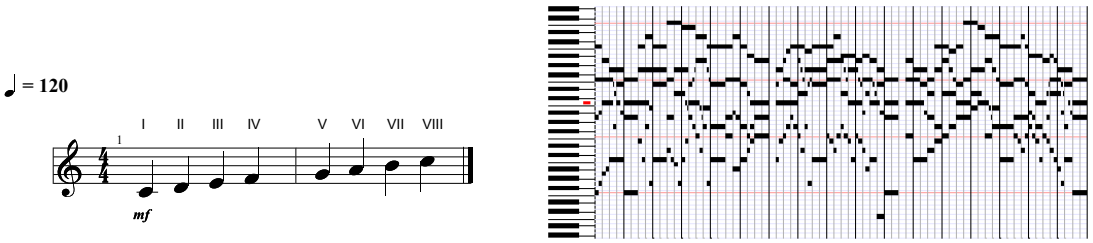
$$f_m = 440 \cdot 2^{\left(\frac{m-69}{12}\right)}. \tag{1}$$

A convenient way to show the midi note sequence is the **musical score** or the **piano roll** representation (Fig. 3).

1 Open a MIDI file in MATLAB

The file *test.mid* contains a sequence of MIDI **noteOn** and **noteOff** messages. Open this file with MATLAB using the attached scripts *readmidi.m* and *midiInfo.m*.

[1]The velocity is how forcefully a note was played (*piano*, *forte*, *fortissimo*, ...).
[2]Devices that synthesize the sound of a note.



(a) Music Score (b) Piano Roll

Figure 3: Different MIDI representation.

```
midi = readmidi( 'test.mid' );
[ Notes, endtime ] = midiInfo( midi, 0 );
```

The matrix *Notes* contains all the MIDI note messages in the file *test.mid* and *endtime* is the duration in seconds, of the entire song. Each row of *Notes* is a note and each column is a parameter of that note:

Track#	Channel#	Note# (m)	Velocity ($0-127$)	Start time (S)	End time (S)	ignore	ignore

The MIDI note number m (usually) goes from **21** (A0) to **108** (C8)

2 Plot and Play

- Create and plot the **piano roll** for the *test.mid* song
- Synthesize *test.mid* using a sinusoid for each note.

Some tips:

1. In MATLAB a piano roll can be seen as a matrix in which the rows are the MIDI notes and the columns are the time frames (i.e. a time frame is a time window of length, for example, 50 mSec). All the elements of this matrix is set to 0. If in a time frame τ, a midi note m is played, the element (τ, m) of the piano roll is set to the **velocity** value of that note. The piano roll matrix can be plotted with the following commands:

   ```
   imagesc( pianoRoll );
   axis( 'xy' );
   colormap( flipud(hot) );
   ```

2. For the synthesis of the MIDI file use a sampling frequency $f_s = 44100$ Hz. If you hear some *"clicks"* at each note's onset and offset, it can be useful to model the transient of each sinusoid as in Fig 4.

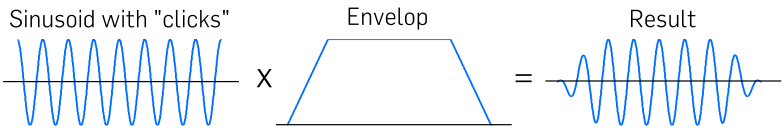


Figure 4: Envelop used for avoid *clicks*.