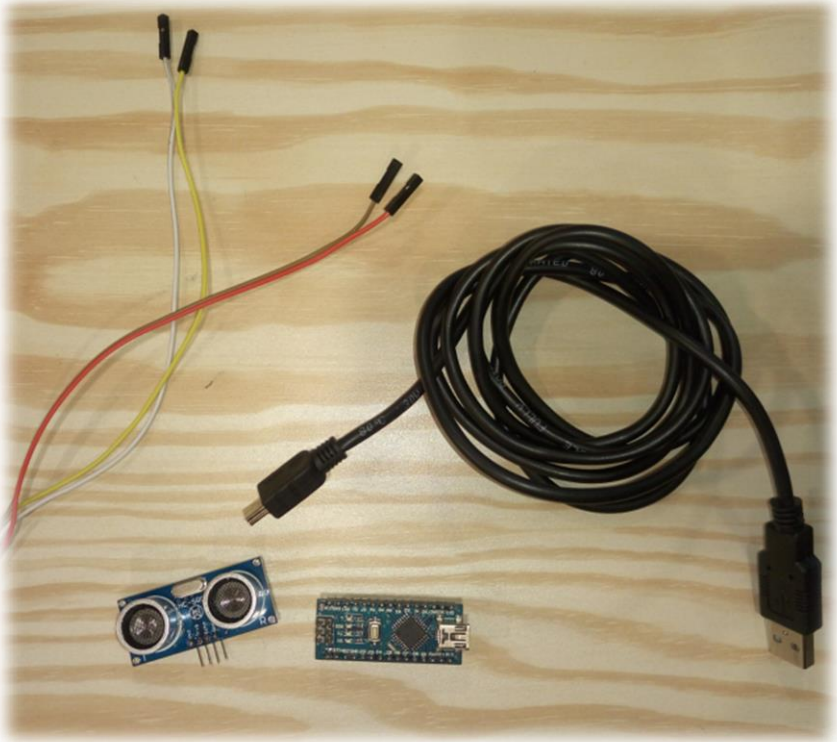
Requirements

To implement this project you need:

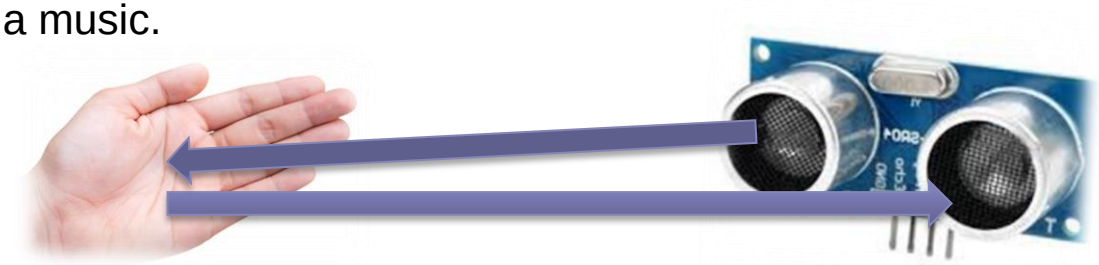
1. Arduino controller (simplest Nano or any other type)
2. Ultrasonic sensor HS-SR04
3. Some set of standard Arduino wires and proper USB cable
4. PC with Windows installed. OS X or Linux also suitable, but these OS require different software and drivers.



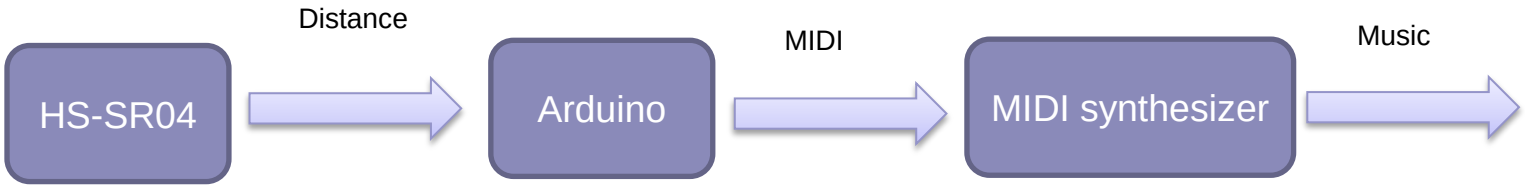
Project concept

The HC-SR04 ultrasonic sensor uses sonar to determine distance to an object.

Using own hand or any other kind of obstacle we can get digital information that can be used to generate a music.



The distance information after properly processing will be sent to MIDI synthesizer software running on PC.

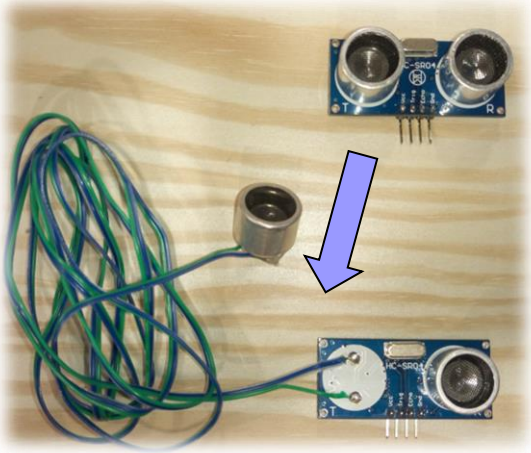


MIDI synthesizer will generate audio stream using MIDI messages from Arduino (including note pitches, velocity, etc) so music will be played by PC audio system.

Sensor rework

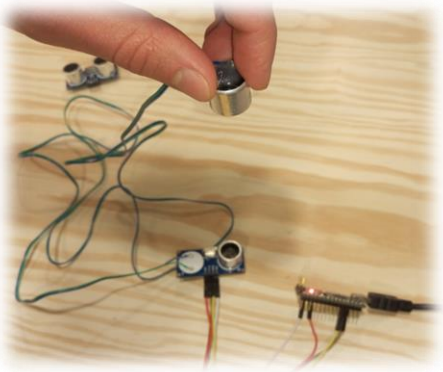
To play you can use standard HS-SR04 sensor without any changes.

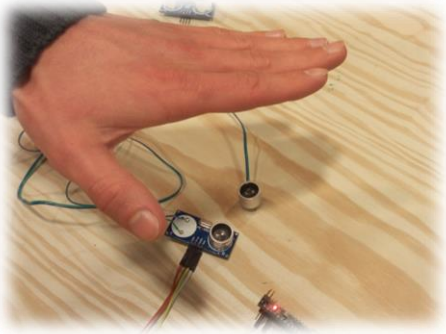
But the best performance and accuracy can be achieved if direct ultrasonic link will be used between US transvers and receiver instead of echo back signal.



So if you want to get high efficiency of playing you should to unsold transmitter and use two wires to connect it to PCBA.

In this case you should change the position of transceiver holding it in your hands to change the distance to source of ultrasonic sound.





But, in any way, you can still use echo back to play, just put the both transmitter and receiver under your hand.

Assembling and programming

Wiring process documentation (it's very simple!!) and Arduino code sketch you can find in repository https://github.com/rnd64/ToySon

Sketch uses MIDI library from FortySevenEffects, so install this library before start compiling of sketch using Arduino Library manager or from sources: https://github.com/FortySevenEffects/arduino_midi_library

Pay your attention that Sketch has some defines that will help you to tuning Arduino software to match your imagination.

For example, you can change music note range to achieve your own music preferences.

Please, see the definition section in .INO sketch file.

After downloading of program just configure PC software to grub of MIDI stream from Arduino as described below.

```
/*
 *  Simple ultrasonic MIDI controller
 *  Version 0.1
 *  http://RnD64.com, 2016
 */

//
// Set MIDI_ENABLED to generate MIDI output instead of Serial debug messages
// This program uses hardware Serial port as output MIDI (115200, 8, n, 1)
// See for reference HS-SR04 http://www.micropik.com/PDF/HCSR04.pdf

#define MIDI_ENABLED 1

#if (MIDI_ENABLED)
  #include <MIDI.h>
#endif

//
// Global constants and parameters
//

const byte echoPin = 8; // Digital input pin to received echo from HS-SR04
const byte trigPin = 9; // Digital output pin to set trigger signal to HS-SR04
const byte TRIG_DURATION = 10; // usec, duration of control trigger signal in accordingly with HS-SR
const int ECHO_TIMEOUT = 8000; // usec, timeout to wait of echo back pulse
const unsigned int MIN_ECHO_DURATION = 200; // echo from closer objects will be ignored
const unsigned int MAX_ECHO_DURATION = 2200; // echo from more far objects will be ignored. In gene
const byte MEASUREMENT_COUNT = 10; // count of echo probes for each position to encrease accuracy o

//
// MIDI parameters and configuration
//

#if (MIDI_ENABLED)
MIDI_CREATE_DEFAULT_INSTANCE();
const byte MIDI_CHANNEL = 1; // MIDI channel to send all notes to PC
const byte MAX_MIDI_VELOCITY = 127; // Default velocity (loudness) for notes pitch
#endif
const unsigned int LOW_NOTE = 44; // Lowest note pitch to generate, 64 = E3 (1 quitar open string),
const unsigned int HIGH_NOTE = 64+12; // Highest note pitch to generate, first string, 12th quitar f
const unsigned long MIN_NOTE_INTERVAL = 5000; //usec, minimum interval between two notes folowing e
```

PC software setup

To provide compatibility with all Arduino controllers this project uses MIDI over Serial transport. So, some middleware is required to grab MIDI messages from Serial stream and translate them to MIDI interface.

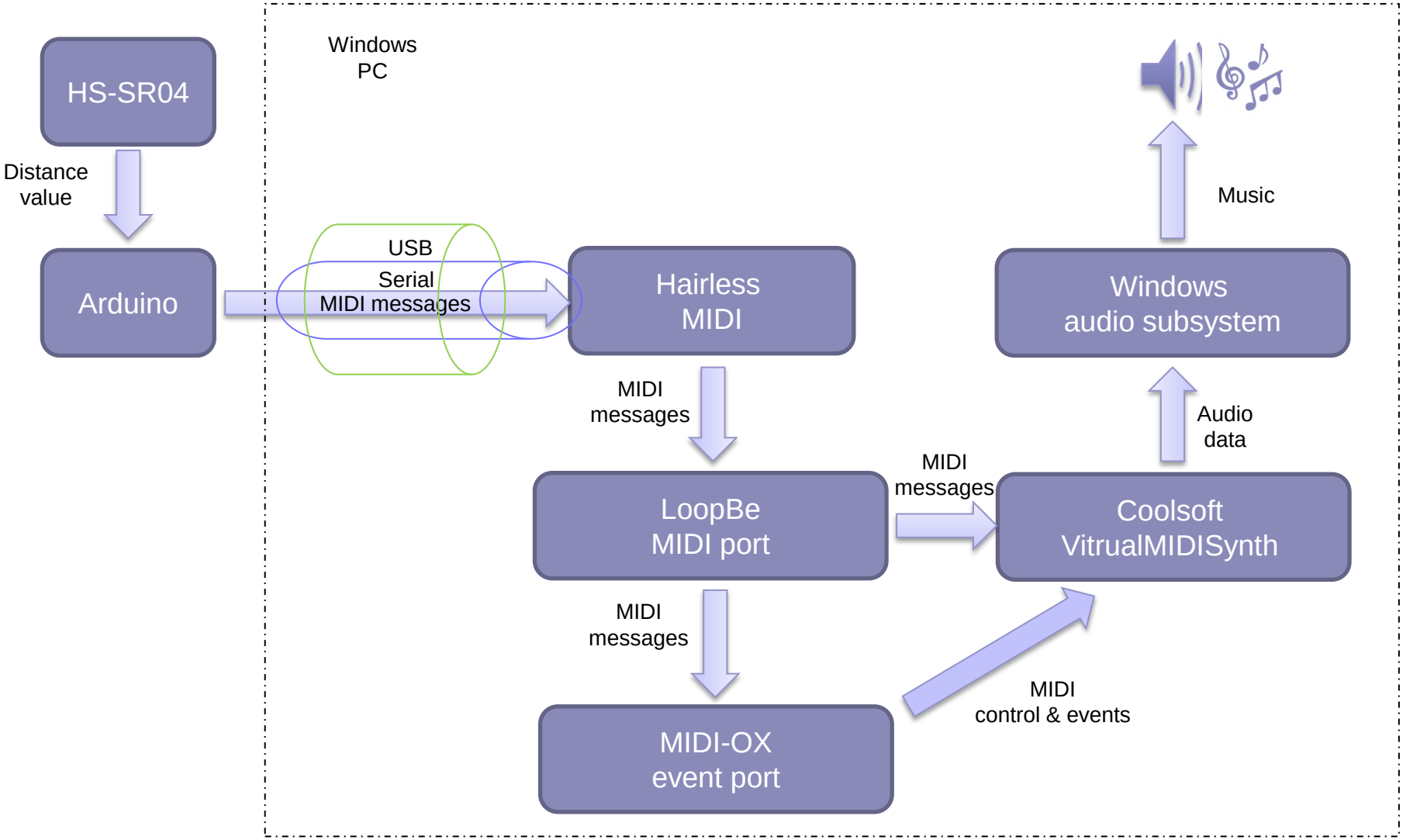
To achieve these targets you need use some additional software:

1. Hairless MIDI<->Serial Bridge http://projectgus.github.io/hairless-midiserial/
2. LoopBe1 MIDI driver http://nerds.de/en/loopbe1.html
3. Any suitable MIDI synthesizer. You can use built-in Microsoft GS Wavetable Synth or you can install Coolsoft VitrualMIDISynth http://coolsoft.altervista.org/en/virtualmidisynth Also you need for any suitable Soundfonts file http://coolsoft.altervista.org/en/virtualmidisynth#soundfonts

In addition, we recommend use MIDI-OX tool http://www.midiox.com/ to easy visual configure, setup and monitoring MIDI devices, instrument selection, etc.

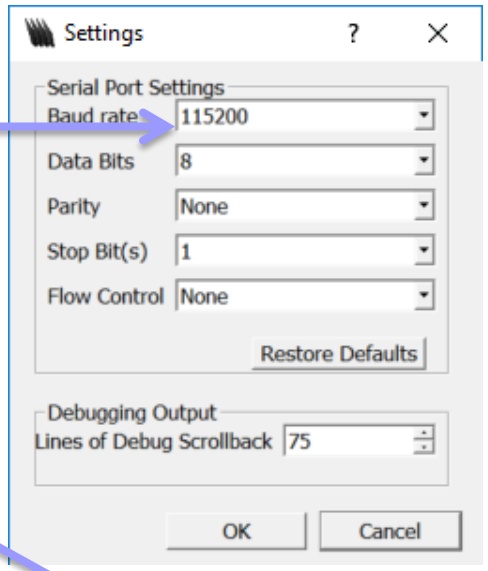
PC software setup

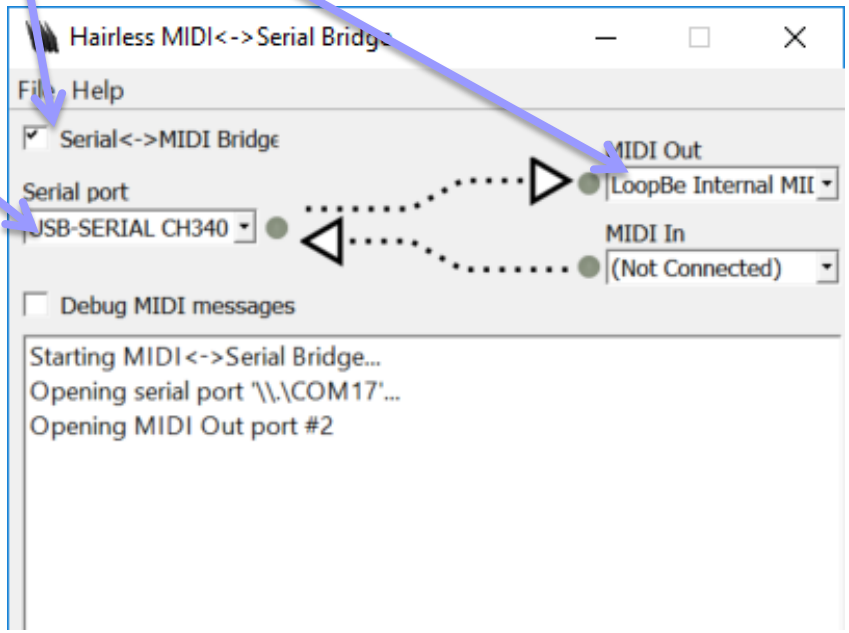
So, the environment to play showed below:



Tips: Hairless MIDI Bridge

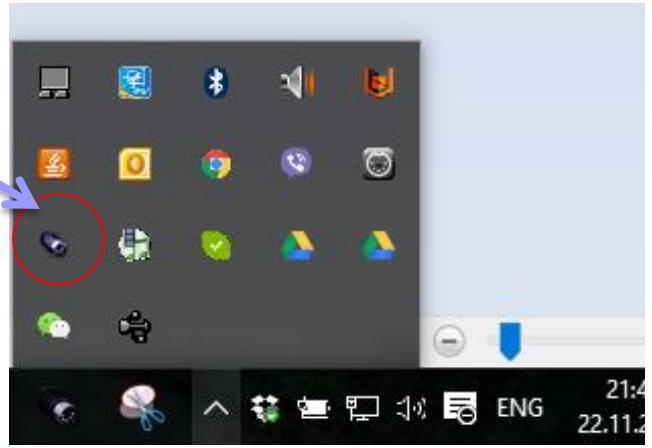
1. Set correct COM-port speed in preferences
2. Don’f forget switch-off before and switch-on bridge after Arduino code uploading
3. Select proper input COM-port for connected Arduino
4. Choose Loopbe Internal MIDI port as output
5. Input is not required in this project

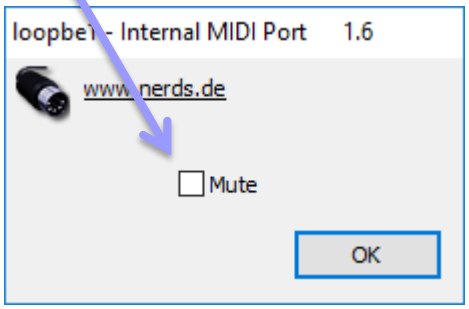




Tips: LoopBe Internal MIDI port

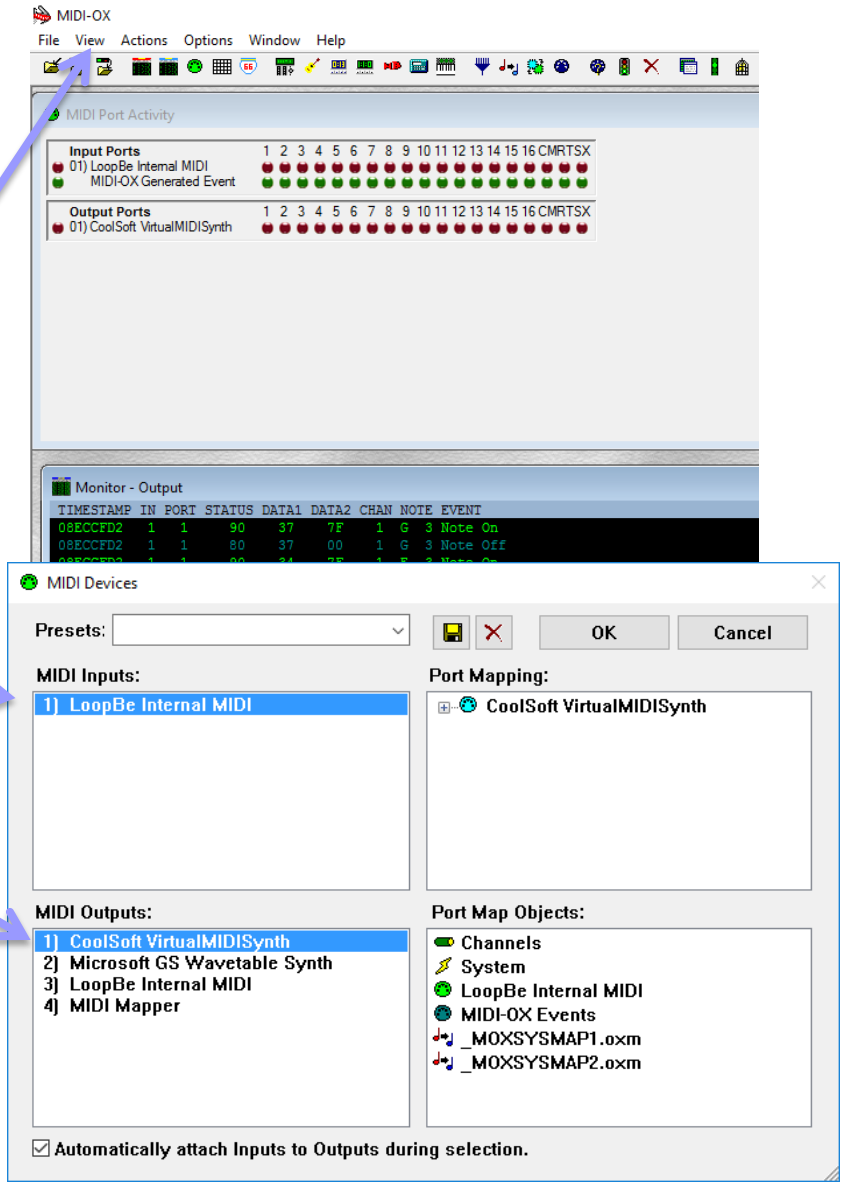
1. Be sure that Internal loopback port is not muted during testing





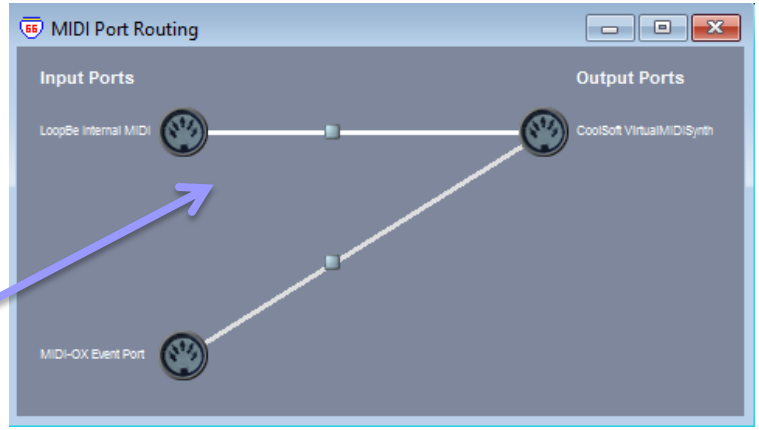
Tips: MIDI-OX

1. Use “Options->Midi devices..” to select proper input and Output devices to attach

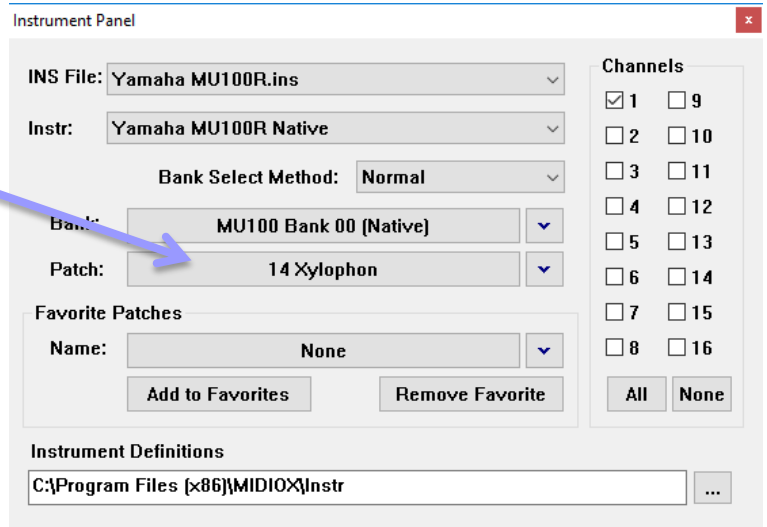


Tips: MIDI-OX

2. Use “View->Port routings” to check and define proper connectivity of input and output MIDI devices as showed on the right picture

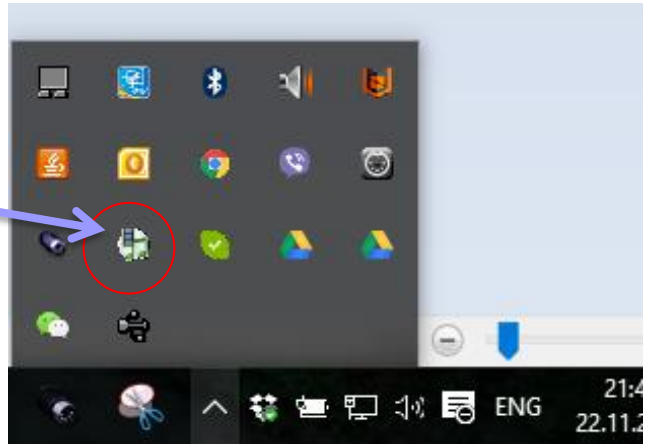


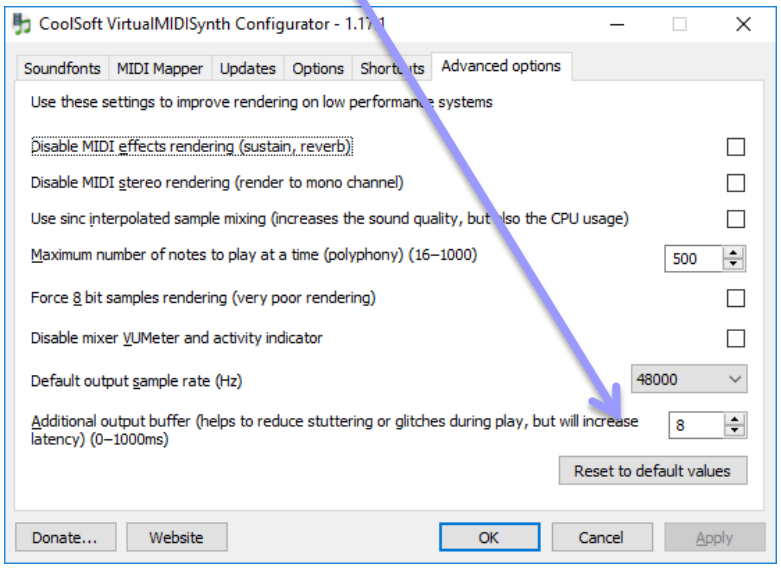
3. Use “View->Instrument panel..” to manage music instruments you want to play

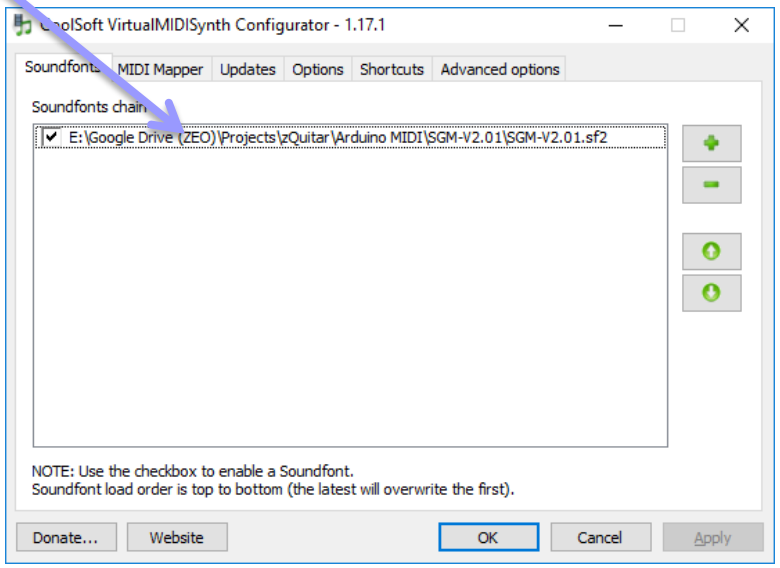


Tips: CoolSoft VirtualMIDISynth

1. Add wanted Soundfonts file
2. Configure buffers to optimal delays and sound quality







Enjoy!