LBNL-PhotoFlute

CMLS Homework N.3

Group: **Last but not Least**

Professor: **Fabio Antonacci**
Assistant: **Marco Olivieri**

A.Y. 2021/2022

POLITECNICO
MILANO 1863

Contents

1 Introduction

LBNL-PF is a half-hardware, half-software instrument which can be easilly played with just seven photoresistors and an *enhances* conductivity sensor. The sound is generated by Supercollider with the help of a GUI in Juce that is able to customize the software instrument. LBNL-PF has been developed with three powerful software instruments: Arduino, Supercollider and Juce.

2 Data Fetching - Arduino

A simple Arduino sketch fetches data from the seven photoresistor and one aluminum foil-enhanced conductivity sensor. The data is sent onto the USB serial-port only when the Arduino board program detects changes inside the current state of the circuit, thus sparing computational power on both the Arduino and Supercollider sides.

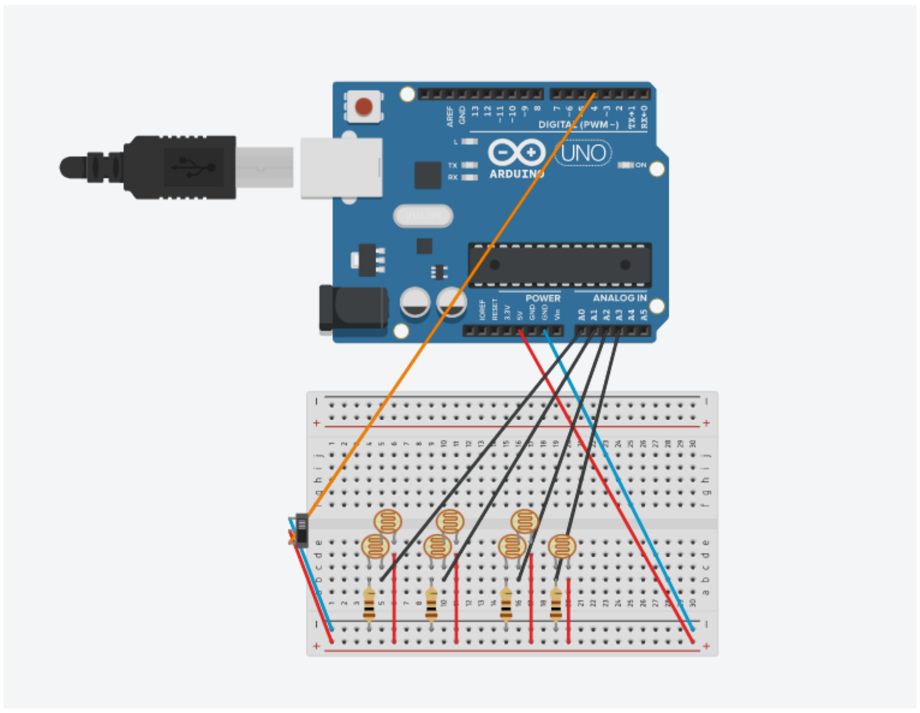

Figure 1: Circuit Scheme

2.1 The Sensors

There are a total of 8 sensors used by LBNL-PF:

- **7 Photoresistors**
- **1 Home-made sensor** constructed with aluminum foil

The photoresistors Each photoresistor controls one note of the flute, with values that are carefully calibrated to obtain a switch-like behaviour (on/off).

The conductivity sensor The other sensor is used to switch between two distinct modes of the PhotoFlute: a standard mode and a mode which enables sharp notes.

3 Audio Software Backbone - Supercollider

The heart and mind of the application is a Supercollider file which controls each synthesizer and each parameter which modifies the overall sound.

3.1 The Notes

Each note is a synthesizer of its own which is initialized with a gate value of zero to keep it from playing sounds. Each time the user puts his finger on top of a photoresistor, the gate of the synth which corresponds to that note rises and the sound plays. When the finger is no longer on the photoresistor, the gate is put back to zero and the sound stops.

3.2 Synth Definitions

Through the GUI, the user can choose between three different synthesizer types which are pre-defined as `SynthDef`: a flute, a sax and an organ. These three synths work in different ways.

\\`flute` The first synthesizer emulates the sound of a simple flute via means of adding oscillators with generated noise that replicates that of air coming out of the instrument. Here is how it works:

- Three envelopes for the generated noise, the overall sound and the vibrato effect are generated with `EnvGen.kr`
- Noise is generated with `LFClipNoise.ar`
- The vibrato effect is created as a `SinOsc.ar` with a frequency of 5Hz
- Multiple delays are added to mimic the resonance of the flute
- Sound is panned and outputted using both `LocalOut.ar` and `Out.ar`

\\`sax` The second synthesizer emulates the sound of a saxophone again by means of interpolating and adding different oscillator signals. Here is how it works:

- Values are generated by multiplying an arithmetic serie with random exponential values in a narrow range
- A signal is generated as a `SinOsc.ar` whose argument frequency is modulated with a `SinOsc.kr` that has random frequency and amplitude
- The signal is filtered with a band-pass filter and a ADSR envelope is applied
- The obtained signal is finally panned and outputted with `Out.ar`

`\organ` Last but not least, the third synthesizer emulates the sound of an organ by applying a vibrato effect to a pulse oscillator. Here is how it works:

- An ADSR envelope is generated
- The vibrato effect is created with a

  `SinOsc.ar` modulated by a tuning factor
- A signal is created as the sum of to impulse oscillators
- The signal is filtered with a resonant low-pass filter and multiplied by the envelope
- DC bias of the signal is removed with `LeakDC.ar`
- The obtained signal is finally panned and outputted with `Out.ar`

3.3 Sharp Mode

Sharp Mode (that is, the mode that lets you play sharp notes) is enabled with the conductivity sensor which basically acts as an on/off switch. Let's say we have an array **of length 14**:

NotesArray = [Synths with non-sharp freqs,   Synths with sharp freqs]

What the sensor does is setting an offset variable to the value 7 when the Sharp Mode is on, so that Supercollider accesses the synthesizer with an index offsetted by 7, turning the flute's photoresistors notes to sharp notes.

3.4 Synth Customization

There are many global variables which are controlled via OSC Messages coming from the Juce GUI. These are used to change some aspects of the sound:

- Overall Volume
- Panning
- Synthesizer Type
- Octave of the Notes

4 Graphical User Interface - JUCE

The graphical user interface takes its cue from mobile applications. This makes it very simple and intuitive.

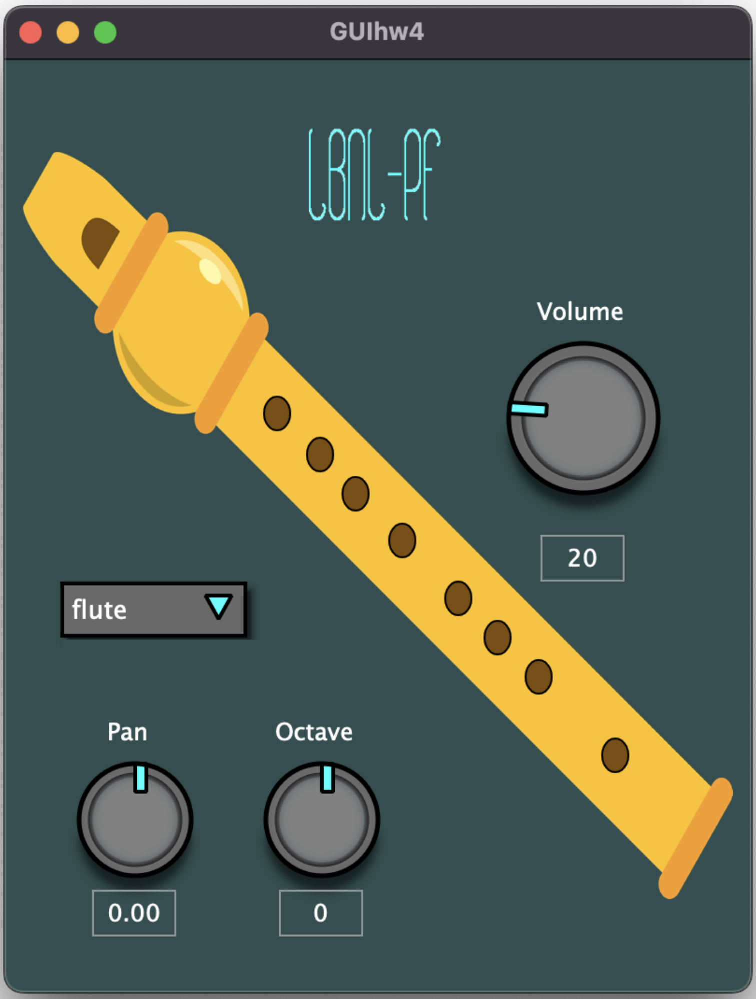

Figure 2: GUI

4.1 Components

The juce components that are inside the GUI are the following:

- `juce::ImageComponent`: for the logo and flute images.
- `juce::TextButton`: for the flute's holes.
- `juce::Slider`: for the volume, the pan and the octave knob.

- `juce::Label`: for the knobs names.
- `juce::ComboBox`: for the instrument decision.

4.2 Connection with SuperCollider - sending messages

In order to make the connection between JUCE and SuperCollider working properly it is necessary to create an `juce::OSCSender` object and a `juce::DatagramSocket` object. Each time a message has to be sent the OSC-Sender object is called with the method `send`. This takes two arguments, the `OSCAddressPattern` and all the variables that have to be sent.

4.3 Connection with SuperCollider - receiving messages

Regarding the receiving part of JUCE, the class `OSCMessageReceived` is used which takes as input the `OSCMessage`. We receive data from Supercollider that tells us which of the notes are being currently played and if Sharp Mode is enabled.

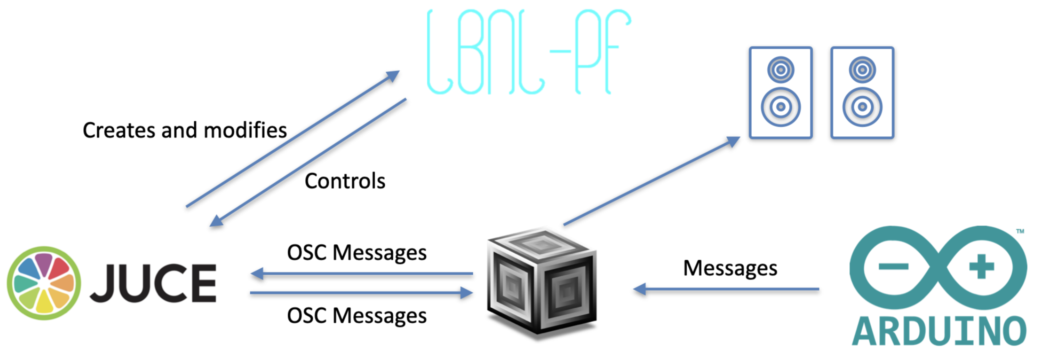

Figure 3: Overall Connection Scheme

5 How Does It Work

As previously explained, LBNL-PhotoFlute is easy to use.

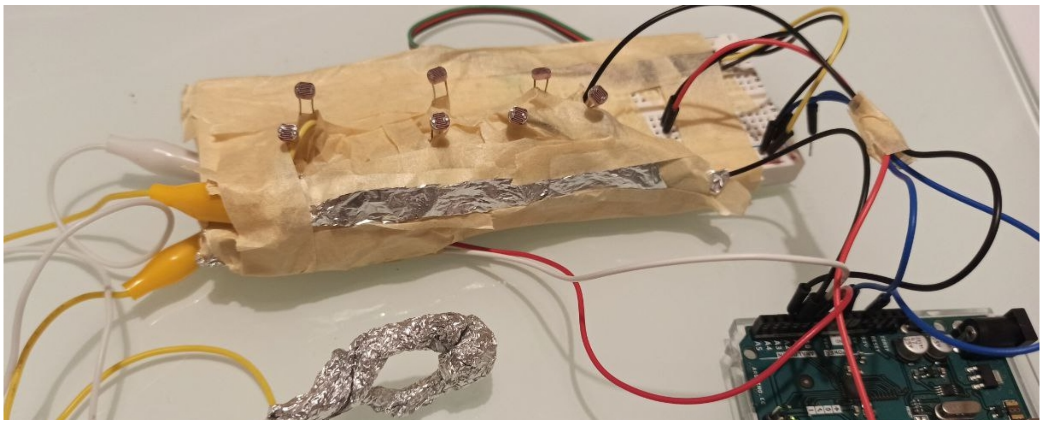

Figure 4: LBNL-PhotoFlute

Here are some general rules to keep in mind:

- Put your fingers on the photoresistors to play notes
- Use the aluminum foil sensor to toggle Sharp Mode
- Use the GUI to customize the sound
- Enjoy LBNL-PF