PEACHNOTE PIANO: MAKING MIDI INSTRUMENTS SOCIAL AND SMART USING ARDUINO, ANDROID AND NODE.JS

Joren Six
Royal Academy of Fine Arts & Royal Conservatory,
University College Ghent, Belgium
joren.six@hogent.be

Vladimir Viro
Ludwig-Maximilians-University,
Munich, Germany
vladimir@viro.name

EXTENDED ABSTRACT

Playing music instruments can bring a lot of joy and satisfaction, but not all apsects of music practice are always enjoyable. In this contribution we are addressing two such sometimes unwelcome aspects: the solitude of practicing and the ”dumbness” of instruments.

The process of practicing and mastering of music instruments often takes place behind closed doors. A student of piano spends most of her time alone with the piano. Sounds of her playing get lost, and she can't always get feedback from friends, teachers, or, most importantly, random Internet users. Analysing her practicing sessions is also not easy. The technical possibility to record herself and put the recordings online is there, but the needed effort is relatively high, and so one does it only occasionally, if at all.

Instruments themselves usually do not exhibit any signs of intelligence. They are practically mechanic devices, even when implemented digitally. Usually they react only to direct actions of a player, and the player is solely responsible for the music coming out of the insturment and its quality. There is no middle ground between passive listening to music recordings and active music making for someone who is alone with an instrument.

We have built a prototype of a system that strives to offer a practical solution to the above problems for digital pianos.

From ground up, we have built a system which is capable of transmitting MIDI data from a MIDI instrument to a web service and back, exposing it in real-time to the world and optionally enriching it.

To demonstrate the technical feasibility of the system we have implemented a *“Continue a melody*“-service, which works as follows: a user plays something on a keyboard, maybe just a few notes, and pauses for a few seconds. In

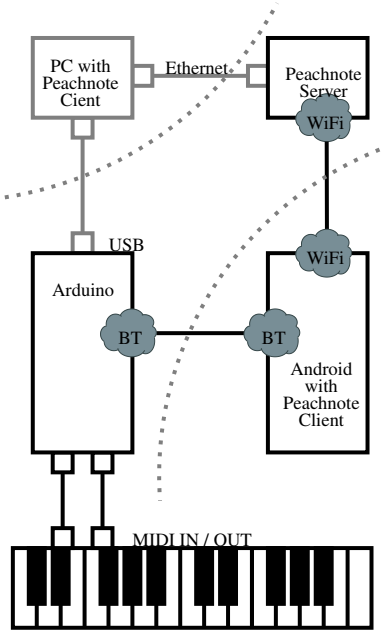

Figure 1. The Peachnote Piano components: a MIDI keyboard (below), a custom built MIDI to Bluetooth (BT) bridge (center left), a smartphone with a Peachnote Piano client (center right) and the Peachnote Piano server (top right). The greyed out part is a PC with a Peachnote Piano client.

the meantime, the server searches through a large database of MIDI piano recordings, finds the longest fuzzy match for the user's most recent input, and, after a short silence on the users part, starts streaming the continuation of the best matched performance from the database to the user. This mechanism, in fact, is way of browsing a music collection. Users may play a known leitmotiv or just improvise something, and the system continues playing a high quality recording, "replying" to the musical proposition of the user. If the user then starts playing again, the stream from the server is suspended and a new search request is prepared, considering not only the user's input, but also the recent notes received from the server as well.

This use case demonstrates bi-directional MIDI communication and data distribution by the server.

The melody matching is done on the server, which is implemented in Javascript in the Node.js framework. The whole dataset (about 350 hours of piano recordings) resides in memory in two representations: as a sequence of pitches, and as a sequence of "densities" at the corresponding places of the pitch sequence dataset. This second array is used to store the rough tempo information (number of notes per second) absent in the pitch sequence data. By combining the two search criteria we can achieve reasonable approximation of the tempo-aware search without its computational complexity.

The implementation of the hardware is based on the open-source electronic prototyping platform Arduino. Optocoupled MIDI ports (IN/OUT) and the BlueSMiRF Bluetooth module were attached to the main board, as can be seen in the middle left block of Figure 1. The BlueTooth module is configured to use the Serial Port Profile (SPP) which emulates RS-232. The software on the Arduino manages bi-directional, low latency message passing between three serial ports: USB (through an FTDI chip), BlueTooth and the hardware MIDI-IN and OUT port. The standard Arduino firmware has been replaced with firmware that implements the "Universal Serial Bus Device Class Definition for MIDI Devices": when attached to a computer via USB, the Arduino shows up as a standard MIDI device, which makes it compatible with all available MIDI software.

The software client currently works on the Android smartphone platform. It is represented using the middle right block in Figure 1. The client can send and receive MIDI events over its Bluetooth port. Pairing, connecting and communicating with the device is done using the Amarino library, available on `http://www.amarino-toolkit.net/`. The client communicates with the Peachnote Piano server using TCP sockets implemented on the Dalvik Java runtime .

This demo being the beginning, we are looking forward to implementing more social features on top of the presented platform, and extending it to other hardware platforms.