A TouchOSC MIDI Bridge for Linux

Albert Gräf <aggraef@gmail.com>
Computer Music Research Group
Institute of Art History and Musicology (IKM)
Johannes Gutenberg University (JGU)

April 18, 2014

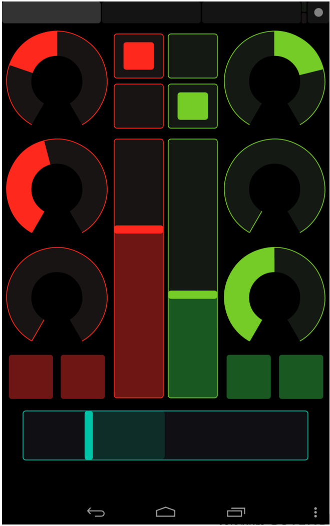
The Lemur (2005-2010)

Lemur
V2/More control
than ever before
the only multitouch and modular controller
for sequencers, synthesizers, virtual instruments,
vjing and lights, now even better
discover what's new

JOHANNES GUTENBERG UNIVERSITÄT MAINZ

TouchOSC

- configurable **multitouch** controller
- runs on **Android** and **iOS**
- TouchOSC **Editor**
- communicates via **OSC** (OpenSoundControl)
- **MIDI** support for “legacy” applications
- works via **RTP-MIDI** (not supported on Android) or proprietary **MIDI bridge** (Mac/Windows only)
- ⇒ **no MIDI support** on Linux :(



Pd-TouchOSC

- TouchOSC MIDI bridge in Pd
- reads MIDI mapping from TouchOSC layout
- translates between OSC and MIDI on the fly
- Zeroconf support (via Avahi on Linux, Bonjour on the Mac)
- written (mostly) in Pure (~ 500 lines Pure, ~ 130 lines C)
- Pd library `touchosc` with core externals `toosc`, `tomidi` and `oscbrowser`, compiles to native binary (`.so` on Linux)
- uses `mrpeach` externals for OSC connectivity
- `touchosc-bridge` patch glues everything together

Download

- You'll need:
  - Pd (+ cyclone, mrpeach)
  - Pure (+ pd-pure, pure-stldict, pure-xml)
  - TouchOSC
- Get it at Bitbucket:
  `https://bitbucket.org/agraef/pd-touchosc`
- Arch User Repositories:
  `https://aur.archlinux.org/packages/pd-touchosc-git`
- Binaries for Arch Linux and Mac OS X available
- Find my other projects on Bitbucket:
  `https://bitbucket.org/agraef/agraef.bitbucket.org`

TouchOSC Layouts

```
<?xml version=”1.0” encoding=”UTF-8”?>
<layout version=”13” mode=”0” orientation=”vertical”>
<tabpage name=”MQ==” scalef=”0.0” scalet=”1.0” >
<control name=”ZmFkZXIx” x=”44” y=”48” w=”50” h=”200”
     color=”red” scalef=”0.0” scalet=”1.0”
     type=”faderv” response=”absolute”
     inverted=”false” centered=”false” >
<midi var =”x” type=”0” channel=”1” data1=”1”
      data2f=”0” data2t=”127” />
</control>
...
</tabpage>
</layout>
```

TouchOSC Widgets

- Available widgets:
  - faders
  - rotary controls (knobs, encoders)
  - push and toggle buttons (0/1)
  - XY pads (x, y coordinates)
  - Multi-widgets: **arrays** of faders, buttons and XY pads
- Widgets can be arranged on **multiple pages** (tabbed interface) which have their own OSC addresses (`/1`, `/2`, etc.).
- Widget coordinates and dimensions are **absolute** (no automatic layout), but the editor provides various operations to align and arrange them on a page.

Control Variables

- Variable types:
  - *input* variables: change widget state when received on device
  - *output* variables: sent to host when operated on device
- Available variables:
  - x: primary control value (input/output)
  - y: secondary control value (XY pads; input/output)
  - z: touch value (0/1; output only)
  - c: color (0..8; input only)

TouchOSC Messages

OSC Message	Meaning
`/1`	first page
`/1/fader1 0.1`	*x* value of `fader1`
`/1/fader1/color red`	color (input only)
`/1/fader1/z 1`	touch value (output only)
`/1/xy1 0.1 0.7`	*x*, *y* values of a XY pad
`/1/multifader1/1 0.1`	1st subcontrol of a multi-fader
`/1/multifader1/1/z 1`	touch value of subcontrol
`/1/multixy1/1 0.1 0.7`	1st subcontrol of a multi-XY pad
`/1/multipush1/2/3 0.1`	subcontrol in column 2, row 3

MIDI Mapping

- Voice messages generally map the *last* data byte only.
- Pitch bend messages map the entire 14 bit value (MSB+LSB).
- Realtime sequencer messages (start/stop/continue) may be mapped to 0/1 variables (buttons, touch).
- Linear mapping from OSC $(x_1 - x_2)$ to MIDI $(y_1 - y_2)$:

$$y = y_1 + \frac{y_2 - y_1}{x_2 - x_1}(x - x_1)$$

MIDI Mapping

Type	Channel	Fixed Data	Mapped Data
control change	1-16	controller	value (0-127)
note	1-16	note number	velocity (0-127)
program change	1-16	-	program (0-127)
start	-	-	-
stop	-	-	-
continue	-	-	-
key pressure	1-16	note number	velocity (0-127)
channel pressure	1-16	-	velocity (0-127)
pitch bend	1-16	-	bend (0-16383)

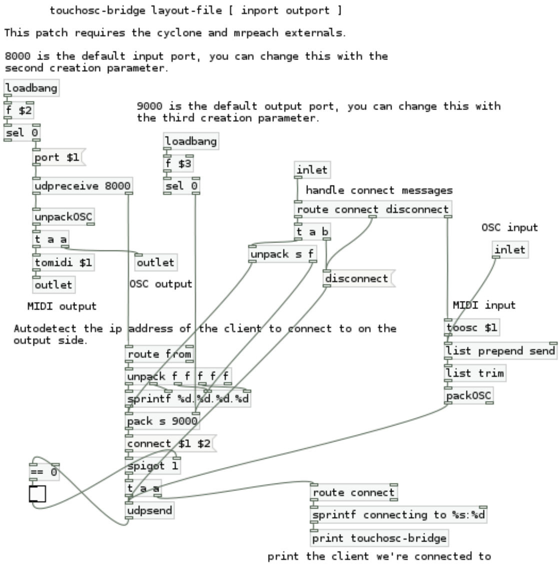
MIDI Encoding in Pd

- designed to facilitate interfacing with Pd's MIDI objects
- implemented by `midi-input` and `midi-output` patches

Message Type	Format	Message Type	Format
note	`note` n v c	pitch bend	`bend` v c
control change	`ctl` v n c	start	`start`
program change	`pgm` n c	stop	`stop`
key pressure	`polytouch` v n c	continue	`cont`
channel pressure	`touch` v c		

n = note/controller number, v = value/velocity, c = MIDI channel

TouchOSC Bridge Patch



touchosc-bridge layout-file [inport outport]

This patch requires the cyclone and mrpeach externals.

8000 is the default input port, you can change this with the second creation parameter.

9000 is the default output port, you can change this with the third creation parameter.

handle connect messages

OSC input

MIDI input

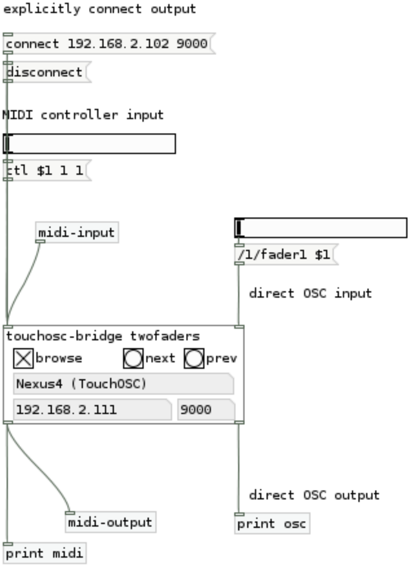
OSC output

MIDI output

Autodetect the ip address of the client to connect to on the output side.

print the client we're connected to

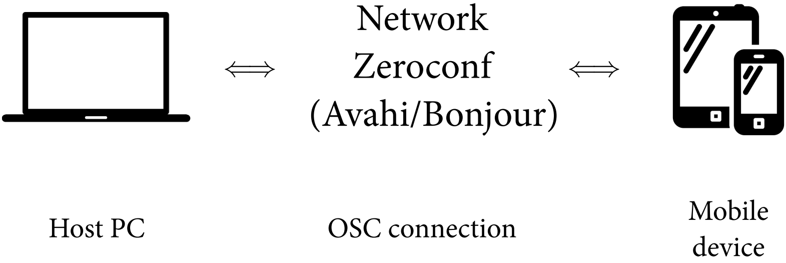
Simple Example



```
explicitly connect output
[connect 192.168.2.102 9000(
[disconnect(

MIDI controller input
[ctl $1 1 1(

[midi-input]
                                  [/1/fader1 $1(
                                  direct OSC input

[touchosc-bridge twofaders]
☒ browse     ○ next  ○ prev
Nexus4 (TouchOSC)
192.168.2.111        9000

                                  direct OSC output
[midi-output]                     [print osc]
[print midi]
```

Network Connections

- OSC via UDP (mrpeach or OSCx externals in Pd)
- use Zeroconf for negotiating network addresses (`_osc._udp`)

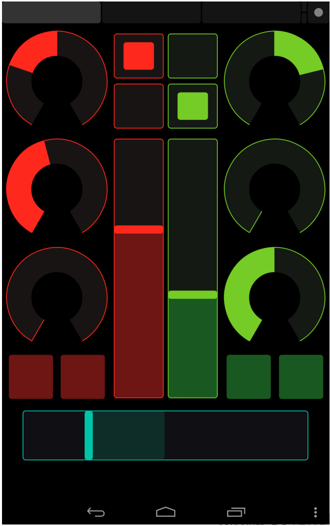


Host PC ⟺ Network Zeroconf (Avahi/Bonjour) ⟺ Mobile device

OSC connection

Demo

- TouchOSC Editor
- Transfer layout to device
- Load layout in Pd
- Establish OSC connection
- Connect Pd with MIDI application



Conclusion

- Pd-TouchOSC provides a **TouchOSC MIDI Bridge** for Linux
- Advantages:
  - open protocol (OSC)
  - open source (Pure source code, Pd patch)
  - can easily be customized for your own purposes
  - cross-platform
- Disadvantages:
  - requires an OSC connection to the device
  - layout file must be available on the host side
  - requires Pd right now (standalone version doable)

Future Work

- TouchOSC is affordable and works well, but also has some drawbacks:
  - *closed source*
  - no *automatic layouts*, requires manual editing
  - no *dynamic configuration* of control elements
  - no *customizable behaviours* of control elements
- Alternatives:
  - Liine's Lemur (closed source; expensive; iOS-only)
  - Charlie Roberts' Control (open-source; Android, iOS)
  - Hanjo Schumacher's Bydcontrol (open-source; any browser)
- *Dynamic interface generation* for Faust, LV2, …