Guitar Extended

A (possible) future of guitar

Make Python and Pure Data communicate on the Raspberry Pi

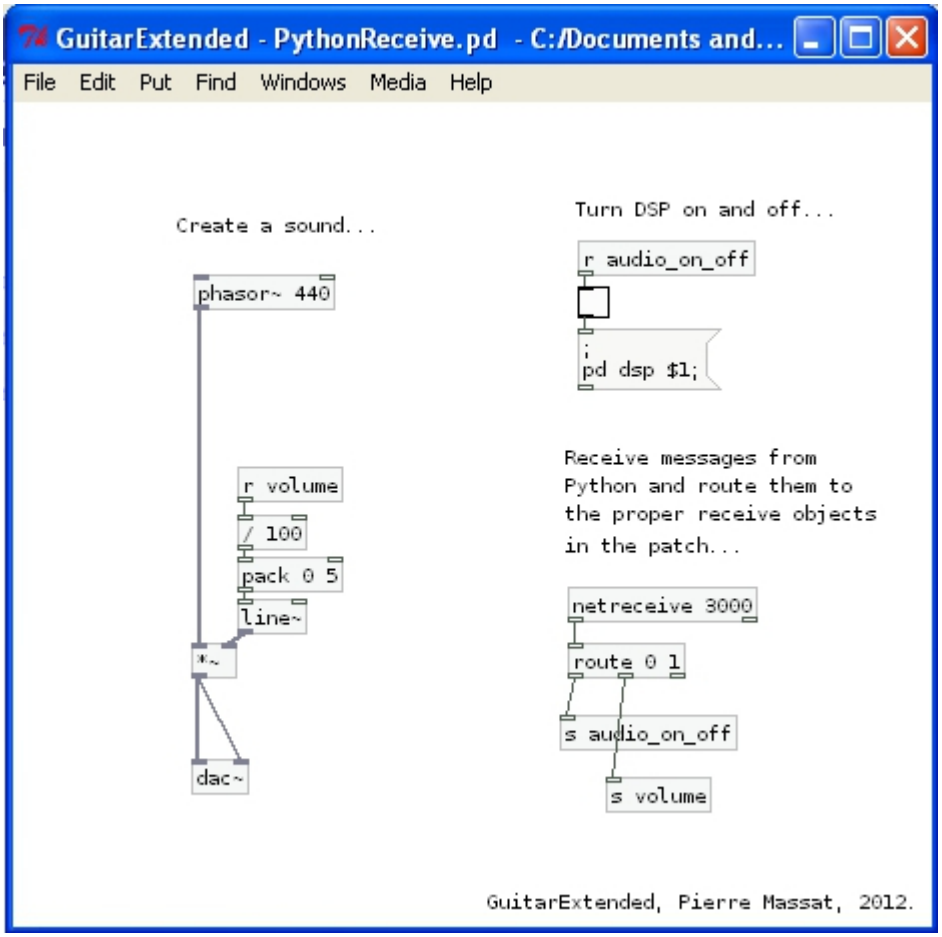
Posted on 11/03/2012 by guitarextended

When working with the Raspberry Pi, it may be useful to make Python and Pure Data communicate. You may want to use Python to setup a webserver for instance, and need to command Pd from within your Python script.

When you install Pd on the RPi ("sudo apt-get install puredata" if you're running the Raspbian distro), additional programs are installed with the core. Two of these programs (included in puredata-utils) are there to help you interface between Pd and another program : pdsend and pdreceive.

Let's look at a simple one-way communcation between Python and Pd, using pdsend. To use it, you'll need to have a [netreceive] object in your patch. As you can see in the patch below, [netreceive] must be created with a specific port number (3000 for instance). Python will have to send messages to the same port. I used a [route] object immediatly after [netreceive] in order to handle different

types of messages, identified by a number (0 for toggling DSP on and off, and 1 for volume).



In your Python script, you can create a specific function for communicating with Pd. In the script below, I have called it “send2Pd”. Other functions in your script can call it by asking it to send a specific message to Pd. “send2Pd” simulates the command line (pdsend was designed to be used with the command line) with the help of the ‘os’ module (please note that you need to import it in your script). The messages we send to Pd are all of the form “id value;”. We need the id because the Pd patch was made to handle more than one incoming message. Note the mandatory semicolon at the end of the message.

```
import os

# (Your own Python script that does whatever you need)


def send2Pd(message=''):
    # Send a message to Pd
    os.system("echo '" + message + "' | pdsend 3000")

def audioOn():
    message = '0 1;' # Id=0 (DSP), message=1 (turn it on)
    send2Pd(message)

def setVolume():
    vol = 80          # Set volume value (0-100)
    message = '1 ' + str(vol) + ';'  # make a string for use with pdsend
    send2Pd(message)
```

Note that you could also use pdreceive (and the [netsend] object in Pd) to get data from Pd into your Python script.

 guitarextended says:

10/30/2014 at 2:16 pm

Bonjour,

Si Python “voit” les messages alors il y a de l’espoir. Difficile de répondre sans voir le code.
Sinon il y a la possibilité d’utiliser OSC au lieu de netsend/netreceive :
http://en.flossmanuals.net/pure-data/network-data/osc/
https://pypi.python.org/pypi/python-osc