1 Quickly and easy install of the A2R tool

The following instructions have been written during the installation process of the A2R tool, in a Debian Squeeze host, in order to help those who face a new installation.

First thing, install node.js from source:

```
wget http://nodejs.org/dist/v0.8.17/node-v0.8.17.tar.gz
tar xzvf node-v0.8.17.tar.gz
cd node-v0.8.17/
ls
./configure
make
make install
```

Notice that the last command (*make install*) must be executed with root privileges.

Once we have node.js installed and running we can start installing the A2R index and proxy servers.

```
apt-get install mongodb

https://github.com/Terminal21/a2r_index.git
cd a2r_index/
npm install cjson express jade mongoose node-syslog optimist
node a2r_index.js
```

Leave the index server running and switch to another terminal :

```
https://github.com/Terminal21/a2r_proxy.git
cd a2r_proxy/
npm install cjson node-syslog optimist osc-min request
```

Before running the script, we must edit the file *config/proxy.config* to match the IP address of the index server. In this case, for my local host, the values are :

```
  "index_server_address": "192.168.1.39",  // never ever set this to 127.0.0.1
  "index_server_port": 7000
```

Then we can start the proxy:

```
node a2r_proxy.js
```

Oce we have both index and proxy servers running without errors we can install the python client. So, in another terminal:

```
apt-get install python-cherrypy3

git clone https://github.com/hangar-A2R/a2r_python_client.git
cd a2r_python_client/
```

Before running the python client we must edit the file *a2r_python_client.py* and change lines 13, 14, 17,and 18 to match the IP address and port where the python will be accesible, and where the index server is listening. In my case:

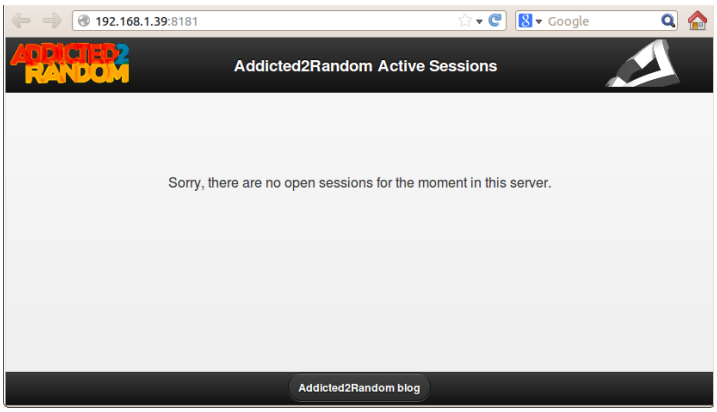
```
# The IP address and port where this GUI server will be available
gui_server_ip   = "192.168.1.39"
gui_server_port = 8181

# The IP address and port where A2R_index is listening
index_ip   = "192.168.1.39"
index_port = 7000
```

Then we can start the client:

```
python a2r_python_client.py
```

If we have done the installation correctly the python client should be accessible via any web browser in the IP address and port specified:



However, as shown in the previous figure, the default behaviour of the web client doesn't allow to do nothing until there are non active sessions playing. So, the next thing to try is to start a new session. For that we can make use of the A2R PureData demo patches:

```
https://github.com/hangar-A2R/a2r_pd_patches.git
cd a2r_pd_patches
```

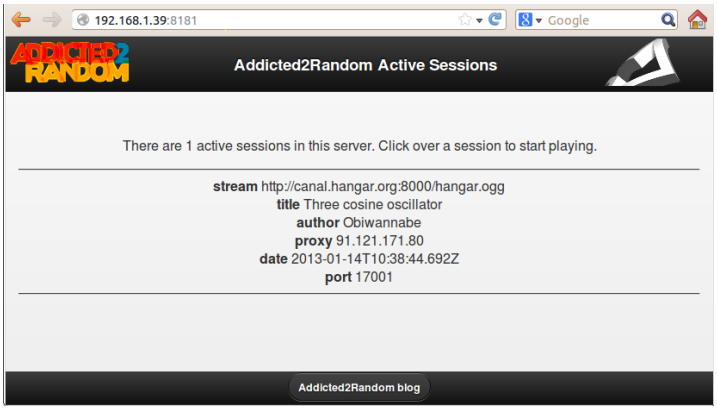
In order to start a session in the correct way you must edit the example patch a2r_threecososcillator.pd and modify the OSC bundle announcement (inside the

a2r_announcement subpatch) and the login and mountpoint information for your audio stream (inside the a2r_streamer subpatch). The a2r_announcement subpatch is where your session communicate with the A2R proxy and index server, announcing that a new session is available. The a2r_streamer subpatch is an audio streaming module allowing remote users to listen the session. If you don't understand what are the values to modify refer to sections 3 and 4 of this document.

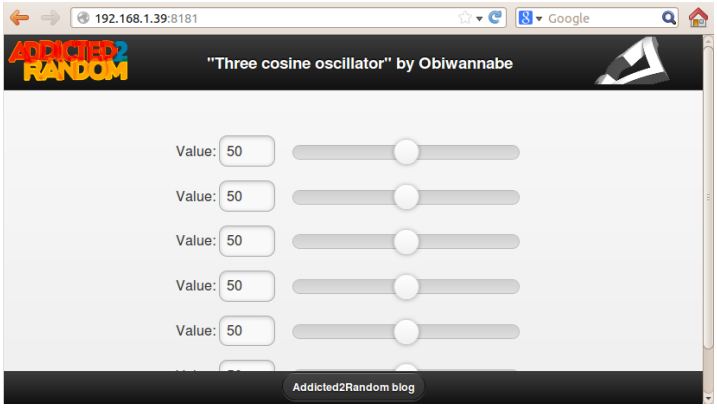
Once the demo patch is correctly configured you can start a new session just by opening it :

```
pd-extended -nogui a2r_threecososcillator.pd
```

If the patch initializes correctly, with all the mandatory fields in the OSC announcement and using the proper IP and port of the proxy server, we should be able to see our new session in the list of Addicted2Random Active Sessions of the python client interface:

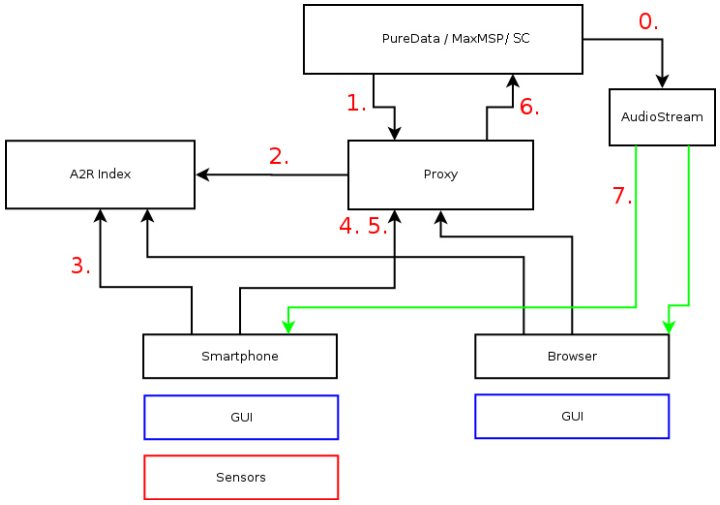


Now, from the browser we should be able to select the running session and start listening and interacting with it:



1.1 Troubleshoting

A good way of troubleshoting is by looking at the flow diagram of the A2R tool:



If you have all the pieces running just try to follow the flow shown in the figure. Start the pd patch and check if the streaming is running (0), then check if the proxy server has received the announcement (1) if not it's probable that your pd patch is sending the OSC announcement to a wrong IP or port, then check if the index is also informed (2) of the new session (try starting the index server in verbose mode with "node a2r_index.js -v"), again if the index doesn't receive nothing it's probable that you don't have a correct value for the index server IP or port in the proxy.conf file.

2 About the python client

The python client is intended to be a way for interacting with the A2R tool from any web browser. So it's an alternative to the android app. However, the way it works is exactly the same.

- First it connects to the A2R index server to get list of active sessions.
- The user selects a session to play.
- Start playing the audio stream and builds a GUI for interacting with the sound.

The whole application is exposed to the browsers using *Cherrypy* so it's basically a web server waiting for two different kinds of browser "GET" requests:

- When "GET /" is received the python client connects via TCP socket to the index server in order to build a web GUI interface with the information of all active sessions: metadata, proxy url and port, audio stream url, etc...
- When "GET /send_data" is received the python client send data to the proxy server via UDP in a compact packet (8 bits for sensor identifier, 16 bits for the value, and 8 bits for CRC checksum).

2.1 Browsers

The web interface is supposed to work in any HTML5 browser, (for now it's only tested in Firefox (v. 16.0 and 18.0) and Chrome (23.0)), both in desktop and mobile versions. However, We've seen a large difference between Firefox and Chrome audio buffering, being Chrome much more slow than Firefox in start playing audio, this means a lot of "lag" in our scenario. We are still trying to solve this issues in Chrome, so for the moment is better to use Firefox.

3 OSC announcement

4 Audio stream