

OscAlgoPi

Project: OscAlgoPi
Author: Staffan Melin, staffan.melin@oscillator.se
License: GNU General Public License v3.0
Version: 2.0 (20200831)
Project site: <http://www.oscillator.se/opensource>

Introduction

You plug a small box into a pair of speakers and it plays music forever. Music that changes and evolves.

That is the goal of this project: To play algorithmic music using the Raspberry Pi with good quality audio that you can connect to your stereo amplifier (or active speakers or headphones). The system is controlled using a web interface, so you can use any device connected to your Wi-Fi as a controller (computer or smartphone) and select different styles, tempo and degree of change.

This guide assumes you know how to operate a command prompt and use some basic text commands.

This project uses these great additional components:

- General User soundfont from Christian Collins (<http://schristiancollins.com/generaluser.php>)
- RtMidi API (<http://www.music.mcgill.ca/~gary/rtmidi/index.html>)
- TinyXML2 (<https://github.com/leethomason/tinyxml2>)

Thank you all!

This system has been developed on a Core i5 laptop running GNU/Linux Debian 10 Buster and a Raspberry Pi 4 with 4GB RAM. I used the Code::Blocks IDE to develop the C++ software.

Table of Contents

Introduction.....	1
How to use it.....	3
2. Connect to the Raspberry Pi.....	3
3. Start the Engine.....	5
4. Select a style.....	5
5. Change the degree of change (permutation).....	5
6. Change the tempo.....	5
7. Shut down the Raspberry Pi.....	5
How to build it.....	6
Equipment.....	7
Prepare the Raspberry Pi.....	7
Install Raspberry Pi OS.....	7
Enable SSH.....	7
Enable WIFI.....	7
Boot the Raspberry Pi.....	7
Configure the Raspberry Pi.....	8
SSH communication.....	8
Additional settings.....	8
Prepare files.....	8
Install Fluidsynth, RtMidi and TinyXML2.....	9
Compile and build the C++ code.....	9
Building oap - the engine.....	9
Building oapsignal - the signaller.....	10
Install the web server.....	10
Copy programs and files and fix the communication.....	11
Set up Fluidsynth.....	12
Test.....	13
Appendix. About the code.....	14
Appendix. Styles.....	15
Appendix. Fluidsynth configuration.....	20
Appendix. General Midi drums.....	21
Appendix. GeneralUser patches.....	22

How to use it

1. Power up your Raspberry Pi and let it boot (it should take less than a minute).
2. With another device (smartphone, laptop) connected to the same wifi, start a browser and visit the homepage on the Raspberry Pi.
3. Start the Engine.
4. Select a style
5. Change the degree of change (permutation).
6. Change the tempo
7. Shut down the Raspberry Pi

The following is a more detailed description.

2. Connect to the Raspberry Pi

To connect you will have to lookup the ip (eg 192.168.1.85) of the Raspberry Pi.

As I run GNU/Linux, I used the command

```
nmap -sL 192.168.1.0/24
```

in a terminal to list all connected devices.

It will show a line something like this

```
Nmap scan report for raspberrypi.lan (192.168.1.85)
```

which means that your Raspberry Pi is connected to the ip 192.168.1.85.

My wifi router always seem to assign the same ip to my Raspberry Pi, so I don't have to look it up every time.

If you have an Android phone you can use Network Discovery (on F-Droid) or for example Network Scanner (on Play Store).

Type this ip number into your browser and you will see the OscAlgoPi homepage:

OscAlgoPi

Server ip: ::1

Server: Linux debian6430 4.19.0-10-amd64 #1 SMP Debian 4.19.132-1 (2020-07-24) x86_64

Start Engine
Style: EP
Style: DR
Style (XML): SimpleFree
Style (XML): SimpleSync
Style: Upload
Style: Delete
Permutation 00
Permutation 10
Permutation 30
Permutation 40
Permutation 50
Permutation 60
Permutation 70
Permutation 80
Permutation 90
Permutation 99
Tempo x10
Tempo x2
Tempo x1
Tempo /2
Tempo /5
Shut down

3. Start the Engine

Click on "Start Engine" to start the system. The OscAlgoPi is ready to play some music.

4. Select a style

Click on the style buttons to select what kind of music the system is playing.

The OscAlgoPi comes with two pre-defined styles that are hard-coded into the system: EP (ElectroPop) and DR (DRone).

In addition you can download and install new styles that are defined in XML text files:

1. Download an OscAlgoPi style file to the device you are running the web browser on.
2. Click "Style: Upload", select the file and upload the file.

Click "Style: Delete" to remove installed styles.

For more information on styles and how to create your own see "Appendix. Styles".

5. Change the degree of change (permutation).

Click on the permutation buttons to change how much the system changes the music (larger values = more changes). Music can be changed in the following ways, where the changes are applied to the whole or only some notes of the sequence of a track.

- SWAP: Note pitches are shuffled.
- RHYTHM: Pitches are kept but their lengths are shuffled.
- SIMPLIFY: Some notes are changed into rests (ie removed).
- ADD: Some notes are added, the pitches for these are taken from already existing notes.
- SHIFT: Notes are shifted left or right.
- TRANSPOSE: Notes are shifted up and down in pitch, where the pitch "steps" are taken from already existing notes.
- ORIGINAL: The notes are restored to their original values.

6. Change the tempo

Click on the tempo buttons to change the tempo of the music. Tempo changes are applied after the sequence has finished playing. For styles where the tracks are synchronized (for example Style: EP) tempo changes become active when all currently playing sequences (parts of music) have finished. This is to keep all tracks synchronized.

7. Shut down the Raspberry Pi

Click on "Shut down" and wait half a minute to shut down your Raspberry Pi before you disconnect it from power.

How to build it

This is an overview of what we are going to do.

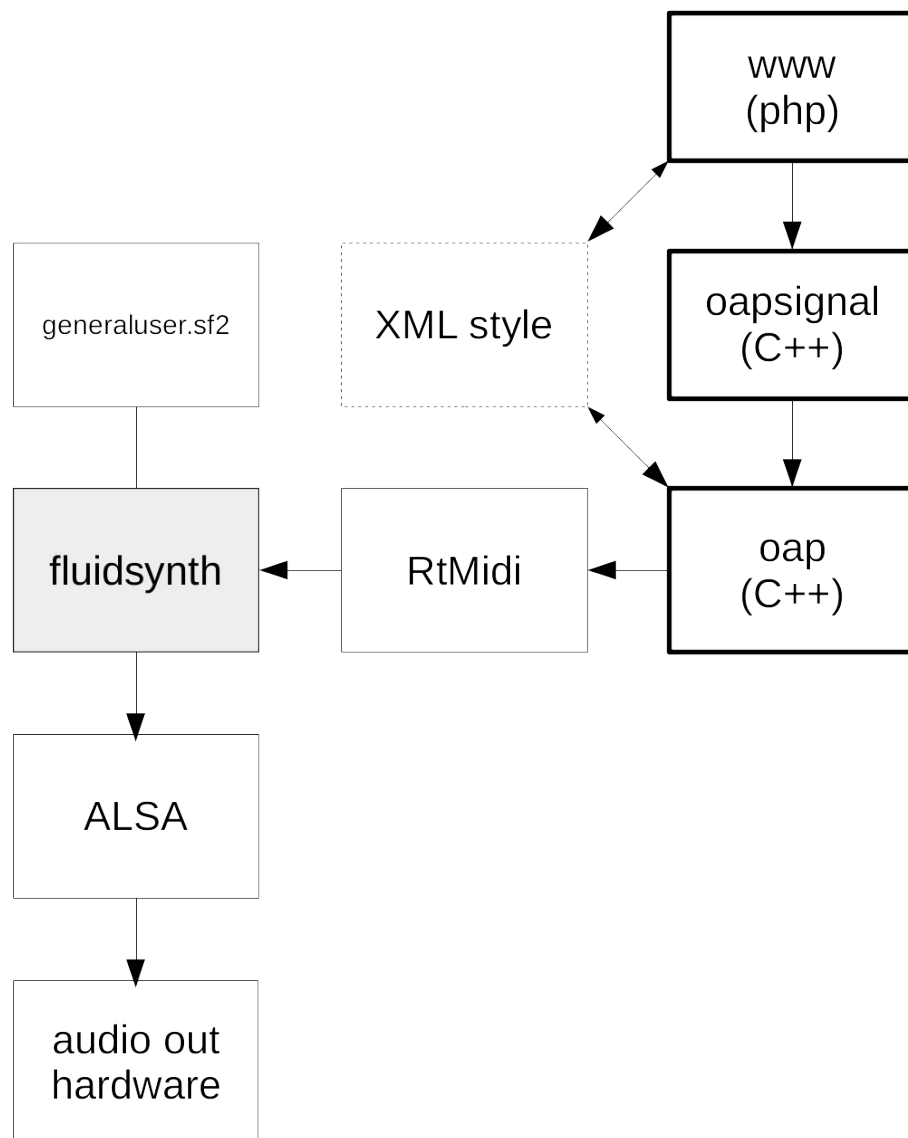
The sound is generated by the Fluidsynth softsynth using a soundfont (.sf2). Fluidsynth outputs the audio to the Raspberry Pi hardware using ALSA, the Linux sound system.

Our program is divided into three parts.

The main engine, **oap**, that controls the softsynth using MIDI transmitted using the RtMidi library.

The **website**, running on a webserver on the Raspberry Pi, that controls the system.

The "glue" between the site and the engine, **oapsignal**, that lets the site communicate with the engine using signals.



Equipment

- a computer running GNU/Linux (you can also use a computer with another OS but then my instructions can't be followed exactly)
- Raspberry Pi (project tested on a Raspberry Pi 4)
- power adapter for the Raspberry Pi
- SD-card and a card reader
- local network (wifi router) and a wifi connection
- headphones or an amplifier

Prepare the Raspberry Pi

As we are not going to connect the Raspberry Pi to a monitor, keyboard and mouse, the desktop software is not necessary. We are going to run our RPI headless.

Install Raspberry Pi OS

Download Raspberry PI OS Lite and install it:

- <https://www.raspberrypi.org/downloads/raspberry-pi-os/>
- <https://www.raspberrypi.org/documentation/installation/installing-images/README.md>

Enable SSH

While you have the SD-card still inserted in the computer, create a file named

```
ssh
```

on the boot partition to enable SSH.

Enable WIFI

You must also make sure that the Raspberry Pi is connected to your lan/network.

For WIFI to work you must create a file on the boot partition named

```
wpa_supplicant.conf
```

and fill it with:

```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
country=<Insert 2 letter ISO 3166-1 country code here>

network={
    ssid=<Name of your wireless LAN>
    psk=<Password for your wireless LAN>
}
```

Boot the Raspberry Pi

Unmount the sdcard. Insert it into your Raspberry Pi and connect it to power.

Useful links:

- <https://www.raspberrypi.org/documentation/configuration/wireless/headless.md>
- <https://www.raspberrypi.org/documentation/remote-access/ssh/README.md>

Configure the Raspberry Pi

SSH communication

You communicate with the headless Raspberry Pi using SSH.

As I run GNU/Linux, I use the command

```
nmap -sL 192.168.1.0/24
```

in a terminal to list all connected devices.

It will show a line something like this

```
Nmap scan report for raspberrypi.lan (192.168.1.85)
```

which means that your Raspberry Pi is connected to the ip 192.168.1.85.

If you have an Android phone you can use Network Discovery (on F-Droid) or for example Network Scanner (on Play Store).

Start a terminal window on your computer (not your Raspberry Pi) and enter:

```
ssh pi@192.168.1.85
```

Login with your password (the user is usually "pi"). Per default the password is "raspberry".

Additional settings

The password can be changed by;

```
sudo raspi-config
```

You can also use raspi-config to:

- Expand the file system to fill the SD card: 7 Advanced Options > A1 Expand Filesystem
- Force the audio output to the audio jack instead of the monitor: 7 Advanced Options > A4 Audio

You should also make sure your Raspberry Pi is updated:

```
sudo apt-get update  
sudo apt-get upgrade
```

Now reboot your Raspberry Pi:

```
sudo reboot
```

Now we have a Raspberry Pi running Raspbian, connected to your Wi-Fi, and accessible using SSH.

Prepare files

We are going to put the project files in

```
cd /home/pi
```


Download the project files:

```
wget
https://oscillator.se/sites/default/files/opensource/oscalgopi/oscalgopi.zip
```

Unpack them

```
unzip oscalgopi.zip
```

You will now have a directory structure looking like this

```
/home
  /pi
    /oscalgopi
      /code_oap
      /code_oapsignal
      /code_web
      /doc
      /rtmidi
      /styles
      /tinyxml2
      fluidconfig.txt
      generaluser.sf2
      generaluser_license.txt
      runfluid.sh
```

Install Fluidsynth, RtMidi and TinyXML2

All sound generation is done by Fluidsynth so we have to install it:

```
sudo apt-get install fluidsynth
```

Fluidsynth is also called ROMpler, ie it plays small samples of sounds (and modifies them in the process). They are stored in a format called .sf2, soundfonts.

OscAlgoPi works with a special soundfont called General User made by Christian Collins. The sounds are very well made in general, of small size, and are mostly mono sounds, which means we can pan them any way we like. The General User soundfont is included in the downloaded project.

We also need the RtMidi C++ MIDI library and the TinyXml2 library. They are both included in the download.

Useful links to these excellent resources:

- <http://schristiancollins.com/generaluser.php>:
- <http://www.music.mcgill.ca/~gary/rtmidi/index.html>
- <https://github.com/leethomason/tinyxml2>

Compile and build the C++ code

Now we have to build the two C++ programs used by OscAlgoPi.

First we have to install a package that is used for the ALSA sound system:

```
sudo apt-get install libasound2-dev
```

Building oap - the engine

Enter the directory where the code is located:

```
cd /home/pi/oscalgopi/code_oap
```

Next run the 9 compilations:

```
g++ -Wall -D__LINUX_ALSA__ -g -c /home/pi/oscalgopi/rtmidi/RtMidi.cpp -o  
obj/RtMidi.o  
g++ -Wall -D__LINUX_ALSA__ -g -c /home/pi/oscalgopi/tinyxml2/tinyxml2.cpp -  
o obj/tinyxml2.o  
g++ -Wall -D__LINUX_ALSA__ -g -c /home/pi/oscalgopi/code_oap/main.cpp -o  
obj/main.o  
g++ -Wall -D__LINUX_ALSA__ -g -c /home/pi/oscalgopi/code_oap/midi.cpp -o  
obj/midi.o  
g++ -Wall -D__LINUX_ALSA__ -g -c /home/pi/oscalgopi/code_oap/style_dr.cpp -  
o obj/style_dr.o  
g++ -Wall -D__LINUX_ALSA__ -g -c /home/pi/oscalgopi/code_oap/style_ep.cpp -  
o obj/style_ep.o  
g++ -Wall -D__LINUX_ALSA__ -g -c /home/pi/oscalgopi/code_oap/style_xml.cpp  
-o obj/style_xml.o  
g++ -Wall -D__LINUX_ALSA__ -g -c /home/pi/oscalgopi/code_oap/track.cpp -o  
obj/track.o  
g++ -Wall -D__LINUX_ALSA__ -g -c /home/pi/oscalgopi/code_oap/util.cpp -o  
obj/util.o
```

And link them together to create our OscAlgoPi engine program:

```
g++ -o oap obj/RtMidi.o obj/tinyxml2.o obj/main.o obj/midi.o obj/style_dr.o  
obj/style_ep.o obj/style_xml.o obj/track.o obj/util.o -lasound -lpthread
```

Building oapsignal - the signaller

Enter the directory where the code is located:

```
cd /home/pi/oscalgopi/code_oapsignal
```

Next run the compilation:

```
g++ -Wall -fexceptions -O2 -c /home/pi/oscalgopi/code_oapsignal/main.cpp -o  
obj/main.o
```

And link it to create our OscAlgoPi signal program:

```
g++ -o oapsignal obj/main.o -s
```

Install the web server

Our user interface will be provided by a web server on the Raspberry Pi. The web server we will use is called Apache. Install it using the following commands:

```
sudo apt install apache2
```

Enter these commands to set up the directories:

```
sudo mkdir /var/www/html/styles  
sudo chown -R pi:www-data /var/www/html/  
sudo chmod -R 770 /var/www/html/
```

You also need to install PHP, the language our user interface is written in, as well as some additional components:

```
sudo apt install php php-mbstring php-xml
```

You don't need to install a database engine.

If you want to you can test that your web server is up and running by opening a web reader on another computer on the same lan/network and enter the ip of the Raspberry Pi. You should get the "Apache2 Debian Default Page".

As the web interface should be able to shut down the Raspberry Pi, you have to add the following line

```
www-data ALL=(ALL) NOPASSWD: /sbin/shutdown
```

to the file /etc/sudoers using the command

```
sudo nano /etc/sudoers
```

which launches a simple text editor. Insert the line after the line that begins with "%sudo", press CTRL+O to save and CTRL+X to exit.

Useful links:

- <https://raspberrypi.stackexchange.com/questions/26357/shutting-down-raspberry-pi-with-php>
- <https://linuxize.com/post/how-to-install-apache-on-debian-10/>

Copy programs and files and fix the communication

Remove the default home page

```
sudo rm /var/www/html/index.html
```

and copy the new one

```
sudo cp /home/pi/oscalgopi/code_web/*.php /var/www/html
```

Now copy the two compiled C++ program files so the web server can access them

```
sudo cp /home/pi/oscalgopi/code_oap/oap /var/www/html
sudo cp /home/pi/oscalgopi/code_oapsignal/oapsignal /var/www/html
```

To enable the different components to speak to one another we have to setup some things.

First we have to tell the web server (Apache 2) that it should use the same file system for temporary files as our programs. Edit this file:

```
sudo nano /lib/systemd/system/apache2.service
```

and set PrivateTmp to false.

We also have to check that the web server is prepared to accept file uploads:

Check where the PHP server is storing its settings (the location of php.ini):

```
php --ini
```

On my installation it is

```
/etc/php/7.3/cli/php.ini
```

Check that this setting is

```
file_uploads = On
```

Reload and restart apache:

```
sudo systemctl daemon-reload
sudo systemctl restart apache2.service
```

The program oapsignal has to know which process id (pid) the main engine oap has. So oap writes this in a file. Without the change to PrivateTmp they will not be able to see this.

To set the right permissions you should create the file

```
sudo touch /var/tmp/oapsignal.txt
```

and set write permission for all

```
sudo chmod a+w /var/tmp/oapsignal.txt
```

This file contains two things on line 1 and 2: the id of the main process (a number) and the name of the XML style that is chosen by the web interface.

Open the file using for example the nano file editor

```
nano /var/tmp/oapsignal.txt
```

and enter the following default information:

```
0
simple
```

Copy the example styles so the web server will be able to access them:

```
sudo cp /home/pi/oscalgopi/styles/*.xml /var/www/html/styles
```

The program oapsignal uses signals to talk to the engine, oap. We have to give oapsignal permission to send these signals:

```
sudo setcap cap_kill+eip /var/www/html/oapsignal
```

I am not sure if this next step is necessary, but we will do it anyway! :)

```
sudo usermod -a -G audio www-data
```

Now is a good time to reboot the Raspberry Pi.

```
sudo reboot
```

Set up Fluidsynth

Fluidsynth has to be started for the oap engine to work.

It is started by a script

```
/home/pi/oscalgopi/runfluid.sh
```

which uses

```
/home/pi/oscalgopi/fluidconfig.txt
```

to set up Fluidsynth.

We have to make this script run at Raspberry Pi startup.

First we make it executable (able to run):

```
chmod a+x /home/pi/oscalgopi/runfluid.sh
```

Then we edit the startup file to make the script run on startup:

```
sudo nano /etc/rc.local
```

and add the following line before the "exit 0":

```
sudo /home/pi/oscalgopi/runfluid.sh & > /home/pi/oscalgopi/log.txt 2>&1
```

Now is a good time to reboot!

```
sudo reboot
```

Useful links:

- <https://www.dexterindustries.com/howto/run-a-program-on-your-raspberry-pi-at-startup/>
- <https://lucidbeaming.com/blog/running-fluidsynth-on-a-raspberry-pi-zero-w/>
- <https://raspberrypi.stackexchange.com/questions/8734/execute-script-on-start-up>

Test

To test your Raspberry Pi, follow the "How to use it" section above.

Adjust the volume:

```
alsamixer
```

Appendix. About the code

I have modelled much of the system on my previous Arduino programming, with `setup()` and `loop()`. The web server talks to the oap program using the helper program `oapsignal`, which sends Linux signals to oap.

The main loop is in `main()` in the `main.cpp`.

The `setup()` and `loop()` functions select the right functions with a simple select statement:

```
void setup(OscTime aTime)
{
    switch (gStyle) {
        case STYLE_EP:
            ep_setup(aTime);
            break;
        case STYLE_DR:
            dr_setup(aTime);
            break;
        case STYLE_XML:
            xml_setup(aTime);
            break;
    }
    gTempoNew = gTempo;
}

void loop(OscTime aTime)
{
    switch (gStyle) {
        case STYLE_EP:
            ep_loop(aTime);
            break;
        case STYLE_DR:
            dr_loop(aTime);
            break;
        case STYLE_XML:
            xml_loop(aTime);
            break;
    }
}
```

The `*_setup()` and `*_loop()` functions are defined in their own C++ files.

Appendix. Styles

The styles are defined in the style/*.xml files. The two exceptions are the Style EP (Electro Pop) and Style DR (Drone) styles that are hardcoded.

To make you own style you have to create a style XML and upload it. This section describes the format of these files.

Styles are made up of Tracks - think of them as a sound, for example piano.

Each Track can have one or more patterns, consisting of one or more notes and rests.

Styles also contain information on how to move from one pattern to another.

This transition works in one of two ways. This is defined at the start of the style file.

- Synced: When the patterns have finished, all tracks shift to new patterns (or all stay on the same pattern). This means that all patterns with the same number have to be of the same length.
- Free: When the pattern of a track has finished, only this track shifts to a new pattern. The tracks are not synchronized. This can be useful for drone music, where you can make patterns of different length for different tracks so the music doesn't repeat (as often).

If you want a rest, MIDI note number should be 0.

Note length is given as

- 1 = 1 (whole)
- 2 = 1/2 (half)
- 4 = 1/4 (quarter)
- 8 = 1/8 (eighth)
- 16 = 1/16
- 32 = 1/32
- 64 = 1/64
- 128 = 1/128
- 256 = 1/256

You can also add a multiplier, eg 2*3 (1/2 * 3).

Here the style format is explained using the two demo files. Comments are colored.

First is an example of a Synced style.

```
<style>
  <name>SimpleSync</name> Name displayed in web interface
  <synced>1</synced> Are tracks synchronized to each other (1) or not
  (0)
  <tempo>2</tempo> Note lengths are multiplied by this factor (so 2 is
  half tempo of base tempo that is 120 bpm)
  <tracks>2</tracks> Number of tracks in this style (max: 50)
```

```

<sequences>2</sequences> Number of sequences defined later on

<track> First track, number 0
  <number>0</number>
  <channel>5</channel> Which MIDI channel to play this on (0-15)
  <type>4</type> Indicates what kind of changes/permutation are
allowed on this track 1 (STABLE) Allowed permutations: None; 2 (DRUM)
Allowed permutations: RHYTHM, SIMPLIFY, ADD, ORIGINAL; 3 (RHYTHM) Allowed
permutations: None; 4 (PITCH) Allowed permutations: SWAP, RHYTHM, SHIFT,
TRANSPOSE, SIMPLIFY, ADD, ORIGINAL.
  <program>49</program> Program number of GeneralUser patch
  <bank>0</bank> Bank number of GeneralUser patch
  <volume>100</volume> Volume of track (0-127)
  <pan>30</pan> Pan of track (0-127, L: 0, C: 64, R: 127)
  <reverb>60</reverb> Reverb level (0-127)
  <chorus>20</chorus> Chorus level (0-127)
  <velocity>100</velocity> MIDI velocity of notes of track
  <patterncount>2</patterncount> Number of patterns
  <pattern number="0"> Start of pattern 0 (as this is a synced
track it must have the same length as all other pattern 0 on all tracks)
    <note>36,4</note> MIDI number number, note length
    <note>41,4</note>
    <note>43,4</note>
    <note>41,4</note>
    <note>36,4</note>
    <note>41,4</note>
    <note>43,4</note>
    <note>41,4</note>
  </pattern>
  <pattern number="1"> Start of pattern 1
    <note>48,4</note>
    <note>41,4</note>
    <note>43,4</note>
    <note>48,4</note>
    <note>43,4</note>
    <note>41,4</note>
    <note>43,4</note>
    <note>41,4</note>
  </pattern>
</track>

<track>
  <number>1</number>
  <channel>6</channel>
  <type>4</type>
  <program>97</program>
  <bank>0</bank>
  <volume>100</volume>
  <pan>95</pan>
  <reverb>100</reverb>
  <chorus>40</chorus>
  <velocity>100</velocity>
  <patterncount>2</patterncount>
  <pattern number="0">
    <note>60,2</note>
    <note>62,2</note>
    <note>65,2</note>
    <note>62,2</note>
  </pattern>
  <pattern number="1">
    <note>55,2</note>
    <note>57,2</note>
    <note>60,2</note>
    <note>57,2</note>

```



```

        </pattern>

    </track>

From here, the info given is special to Synced styles.
First we connect patterns into sequences to describe how they fit together.
    <!--For this sequence, which patterns are active on each track -->
    <sequence number="0"> Sequence number (max: 50)
        <pattern>0</pattern> Track 0 will play pattern 0...
        <pattern>1</pattern> ...and track 1 will play pattern 1
    </sequence>
    <sequence number="1">
        <pattern>1</pattern> Track 0 will play pattern 1...
        <pattern>0</pattern> ...and track 1 will play pattern 0
    </sequence>

Transitions describe the chance/probability that the style will move to the
next next sequence (and the tracks will play those patterns).
    <!--For this sequence, what are the probabilities of moving to another
sequence-->
    <transition sequence="0">
        <chance>80</chance> 80% chance that we will stay in sequence 0
        <chance>20</chance> 20% chance that we will move to sequence 1
    </transition>
    <transition sequence="1">
        <chance>50</chance> 50% chance that we will move to sequence 0
        <chance>50</chance> 50% chance that we will stay in sequence 1
    </transition>

</style>

```

Next is as example of a Free style:

```

<style>
    <name>SimpleFree</name>
    <synced>0</synced> Are tracks synchronized to each other (1) or not
(0)
    <tempo>5</tempo>
    <tracks>3</tracks>
    <sequences>3</sequences> Currently not used for Free styles

    <track>
        <number>0</number>
        <channel>0</channel>
        <type>4</type>
        <program>4</program>
        <bank>8</bank>
        <volume>100</volume>
        <pan>100</pan>
        <reverb>80</reverb>
        <chorus>20</chorus>
        <velocity>100</velocity>
        <patterncount>2</patterncount>
        <pattern number="0">
            <note>36, 2</note>
            <note>41, 4</note>
            <note>43, 2</note>
            <note>41, 4</note>
            <note>36, 4</note>
            <note>41, 4</note>
            <note>43, 4</note>
        </pattern>
        <pattern number="1">
            <note>48, 2</note>

```

```

        <note>41,4</note>
        <note>43,2</note>
        <note>48,4</note>
        <note>43,4</note>
        <note>41,4</note>
        <note>43,4</note>
    </pattern>
</track>

<track>
    <number>1</number>
    <channel>6</channel>
    <type>2</type>
    <program>14</program>
    <bank>11</bank>
    <volume>40</volume>
    <pan>64</pan>
    <reverb>60</reverb>
    <chorus>10</chorus>
    <velocity>100</velocity>
    <patterncount>2</patterncount>
    <pattern number="0">
        <note>67,0</note>
        <note>60,2</note>
        <note>62,2</note>
        <note>65,2</note>
    </pattern>
    <pattern number="1">
        <note>59,0</note>
        <note>55,2</note>
        <note>57,2</note>
        <note>60,2</note>
    </pattern>
</track>

<track>
    <number>2</number>
    <channel>7</channel>
    <type>4</type>
    <program>89</program>
    <bank>0</bank>
    <volume>100</volume>
    <pan>25</pan>
    <reverb>100</reverb>
    <chorus>40</chorus>
    <velocity>100</velocity>
    <patterncount>2</patterncount>
    <pattern number="0">
        <note>36,1*3</note>
        <note>38,1*7</note>
    </pattern>
    <pattern number="1">
        <note>48,1*4</note>
        <note>50,1*5</note>
    </pattern>
</track>

```

From here, the info given is special to Free styles.

We describe, for each track, what the chances are for moving to another pattern.

```
<sequence number="0"> track number 0
```

```
<pattern number="0"> If we are just finished with pattern 0...
```

```

        <chance>60</chance> ...there is a 60% chance of staying in
pattern 0
        <chance>40</chance> ...and a 40% chance for this track to
move to pattern 1
    </pattern>
    <pattern number="1"> If we are just finished with pattern 1...
        <chance>80</chance> ...there is a 80% chance of moving to
pattern 0
        <chance>20</chance> ...and a 20% chance for this track to
stay in pattern 1
    </pattern>
</sequence>

<sequence number="1"> track number 1
    <pattern number="0">
        <chance>70</chance>
        <chance>30</chance>
    </pattern>
    <pattern number="1">
        <chance>30</chance>
        <chance>70</chance>
    </pattern>
</sequence>

<sequence number="2"> track number 2
    <pattern number="0">
        <chance>50</chance>
        <chance>50</chance>
    </pattern>
    <pattern number="1">
        <chance>80</chance>
        <chance>20</chance>
    </pattern>
</sequence>
</style>

```

Appendix. Fluidsynth configuration

/home/pi/oscalgopi/fluidconfig.txt

```
rev_setroomsize 0.8
rev_setwidth 5
rev_setlevel 0.5
rev_setdamp 0.1
cho_set_depth 0.8
cho_set_level 0.5
```

/home/pi/oscalgopi/runfluid.sh

```
#!/bin/bash

if pgrep -x "fluidsynth" > /dev/null
then
echo fluidsynth already flowing
else
fluidsynth -si -a alsa -j -K 16 -L 1 -g 0.6 -o synth.reverb.active=1 -o
synth.chorus.active=1 -o synth.polyphony=32 -o synth.midi-bank-select=gs -o
audio.period-size=1024 -o audio.periods=2 -f fluidconfig.txt
/home/pi/oscalgopi/generaluser.sf2 &
fi
```

Appendix. General Midi drums

The numbers listed correspond to the MIDI note number for that drum sound.

Drum sounds added in General MIDI Level 2 are tagged with (GM2).

27 High Q (GM2)	48 High Tom 2	69 Cabasa
28 Slap (GM2)	49 Crash Cymbal 1	70 Maracas
29 Scratch Push (GM2)	50 High Tom 1	71 Short Whistle
30 Scratch Pull (GM2)	51 Ride Cymbal 1	72 Long Whistle
31 Sticks (GM2)	52 Chinese Cymbal	73 Short Guiro
32 Square Click (GM2)	53 Ride Bell	74 Long Guiro
33 Metronome Click (GM2)	54 Tambourine	75 Claves
34 Metronome Bell (GM2)	55 Splash Cymbal	76 High Wood Block
35 Bass Drum 2	56 Cowbell	77 Low Wood Block
36 Bass Drum 1	57 Crash Cymbal 2	78 Mute Cuica
37 Side Stick	58 Vibra Slap	79 Open Cuica
38 Snare Drum 1	59 Ride Cymbal 2	80 Mute Triangle
39 Hand Clap	60 High Bongo	81 Open Triangle
40 Snare Drum 2	61 Low Bongo	82 Shaker (GM2)
41 Low Tom 2	62 Mute High Conga	83 Jingle Bell (GM2)
42 Closed Hi-hat	63 Open High Conga	84 Belltree (GM2)
43 Low Tom 1	64 Low Conga	85 Castanets (GM2)
44 Pedal Hi-hat	65 High Timbale	86 Mute Surdo (GM2)
45 Mid Tom 2	66 Low Timbale	87 Open Surdo (GM2)
46 Open Hi-hat	67 High Agogo	
47 Mid Tom 1	68 Low Agogo	

Appendix. GeneralUser patches

SF2 - Generaluser.sf2 (bank-program)

000-000 Stereo Grand	000-057 Trombone	000-114 Steel Drums
000-001 Bright Grand	000-058 Tuba	000-115 Wood Block
000-002 Electric Grand	000-059 Muted Trumpet	000-116 Taiko Drum
000-003 Honky-Tonk	000-060 French Horns	000-117 Melodic Tom
000-004 Tine Electric Piano	000-061 Brass Section	000-118 Synth Drum
000-005 FM Electric Piano	000-062 Synth Brass 1	000-119 Reverse Cymbal
000-006 Harpsichord	000-063 Synth Brass 2	000-120 Fret Noise
000-007 Clavinet	000-064 Soprano Sax	000-121 Breath Noise
000-008 Celeste	000-065 Alto Sax	000-122 Seashore
000-009 Glockenspiel	000-066 Tenor Sax	000-123 Birds
000-010 Music Box	000-067 Baritone Sax	000-124 Telephone 1
000-011 Vibraphone	000-068 Oboe	000-125 Helicopter
000-012 Marimba	000-069 English Horn	000-126 Applause
000-013 Xylophone	000-070 Bassoon	000-127 Gun Shot
000-014 Tubular Bells	000-071 Clarinet	
000-015 Dulcimer	000-072 Piccolo	001-038 Synth Bass 101
000-016 Tonewheel Organ	000-073 Flute	001-044 Mono Strings Trem
000-017 Percussive Organ	000-074 Recorder	001-048 Mono Strings Fast
000-018 Rock Organ	000-075 Pan Flute	001-049 Mono Strings Slow
000-019 Pipe Organ	000-076 Bottle Blow	001-052 Concert Choir Mono
000-020 Reed Organ	000-077 Shakuhachi	001-056 Trumpet 2
000-021 Accordion	000-078 Irish Tin Whistle	001-057 Trombone 2
000-022 Harmonica	000-079 Ocarina	001-059 Muted Trumpet 2
000-023 Bandoneon	000-080 Square Lead	001-060 Solo French Horn
000-024 Nylon Guitar	000-081 Saw Lead	001-061 Brass Section Mono
000-025 Steel Guitar	000-082 Synth Calliope	001-080 Square Wave
000-026 Jazz Guitar	000-083 Chiffer Lead	001-081 Saw Wave
000-027 Clean Guitar	000-084 Charang	001-098 Synth Mallet
000-028 Muted Guitar	000-085 Solo Vox	001-120 Cut Noise
000-029 Overdrive Guitar	000-086 5th Saw Wave	001-121 Fl. Key Click
000-030 Distortion Guitar	000-087 Bass & Lead	001-122 Rain
000-031 Guitar Harmonics	000-088 Fantasia	001-123 Dog
000-032 Acoustic Bass	000-089 Warm Pad	001-124 Telephone 2
000-033 Finger Bass	000-090 Polysynth	001-125 Car-Engine
000-034 Pick Bass	000-091 Space Voice	001-126 Laughing
000-035 Fretless Bass	000-092 Bowed Glass	001-127 Machine Gun
000-036 Slap Bass 1	000-093 Metal Pad	
000-037 Slap Bass 2	000-094 Halo Pad	002-102 Echo Pan
000-038 Synth Bass 1	000-095 Sweep Pad	002-120 String Slap
000-039 Synth Bass 2	000-096 Ice Rain	002-122 Thunder
000-040 Violin	000-097 Soundtrack	002-123 Horse Gallop
000-041 Viola	000-098 Crystal	002-124 Door Creaking
000-042 Cello	000-099 Atmosphere	002-125 Car-Stop
000-043 Double Bass	000-100 Brightness	002-126 Scream
000-044 Stereo Strings Trem	000-101 Goblin	002-127 Lasergun
000-045 Pizzicato Strings	000-102 Echo Drops	
000-046 Orchestral Harp	000-103 Star Theme	003-122 Howling Winds
000-047 Timpani	000-104 Sitar	003-123 Bird 2
000-048 Stereo Strings Fast	000-105 Banjo	003-124 Door
000-049 Stereo Strings Slow	000-106 Shamisen	003-125 Car-Pass
000-050 Synth Strings 1	000-107 Koto	003-126 Punch
000-051 Synth Strings 2	000-108 Kalimba	003-127 Explosion
000-052 Concert Choir	000-109 Bagpipes	
000-053 Voice Oohs	000-110 Fiddle	004-122 Stream
000-054 Synth Voice	000-111 Shenai	004-123 Scratch
000-055 Orchestra Hit	000-112 Tinker Bell	004-125 Car-Crash
000-056 Trumpet	000-113 Agogo	004-126 Heart Beat

005-122 Bubbles	008-118 808 Tom	012-089 Solar Wind 2
005-124 Windchime	008-125 Starship	012-119 Tambourine
005-125 Siren	009-014 Carillon	012-122 White Noise Wave
005-126 Footsteps	009-125 Burst Noise	012-127 Shooting Star
		013-048 Woodwind Choir
006-125 Train	011-000 Piano & Str.-Fade	013-080 Square Lead 3
	011-001 Piano & Str.-Sus	013-081 Saw Lead 3
007-125 Jet Plane	011-004 Tine & FM EPs	013-088 Night Vision
	011-005 Piano & FM EP	016-025 Mandolin
	011-008 Tinkling Bells	
008-004 Chorused Tine EP	011-014 Bell Tower	120-000 Standard Drums
008-005 Chorused FM EP	011-038 Techno Bass	120-001 Standard 2 Drums
008-006 Coupled Harpsichord	011-039 Pulse Bass	120-008 Room Drums
008-014 Church Bells	011-049 Stereo Strings Velo	120-016 Power Drums
008-016 Detuned Tnwl. Organ	011-050 Synth Strings 4	120-024 Electronic Drums
008-017 Detuned Perc. Organ	011-051 Synth Strings 5	120-025 808/909 Drums
008-019 Pipe Organ 2	011-061 Brass Section 3	120-026 Dance Drums
008-021 Italian Accordion	011-078 Whistlin'	120-032 Jazz Drums
008-024 Ukulele	011-081 Sawtooth Stab	120-040 Brush Drums
008-025 12-String Guitar	011-087 Doctor's Solo	120-048 Orchestral Perc.
008-026 Hawaiian Guitar	011-088 Harpsi Pad	120-056 SFX Kit
008-027 Chorused Clean Gt.	011-089 Solar Wind	
008-028 Funk Guitar	011-096 Mystery Pad	128-000 Standard
008-030 Feedback Guitar	011-098 Synth Chime	128-001 Standard 2
008-031 Guitar Feedback	011-100 Bright Saw Stack	128-008 Room
008-038 Synth Bass 3	011-119 Cymbal Crash	128-016 Power
008-039 Synth Bass 4	011-121 Filter Snap (KW tjoff)	128-024 Electronic
008-048 Orchestra Pad	011-127 Interference	128-025 808/909
008-050 Synth Strings 3	012-000 Bell Piano	128-026 Dance
008-061 Brass Section 2	012-004 Bell Tine EP	128-032 Jazz
008-062 Synth Brass 3	012-010 Christmas Bells	128-040 Brush
008-063 Synth Brass 4	012-027 Clean Guitar 2	128-048 Orchestral
008-080 Sine Wave	012-038 Mean Saw Bass	128-056 SFX
008-081 Doctor Solo	012-048 Full Orchestra	
008-107 Taisho Koto	012-049 Mono Strings Velo	
008-115 Castanets	012-080 Square Lead 2	
008-116 Concert Bass Drum	012-081 Saw Lead 2	
008-117 Melodic Tom 2	012-088 Fantasia 2	