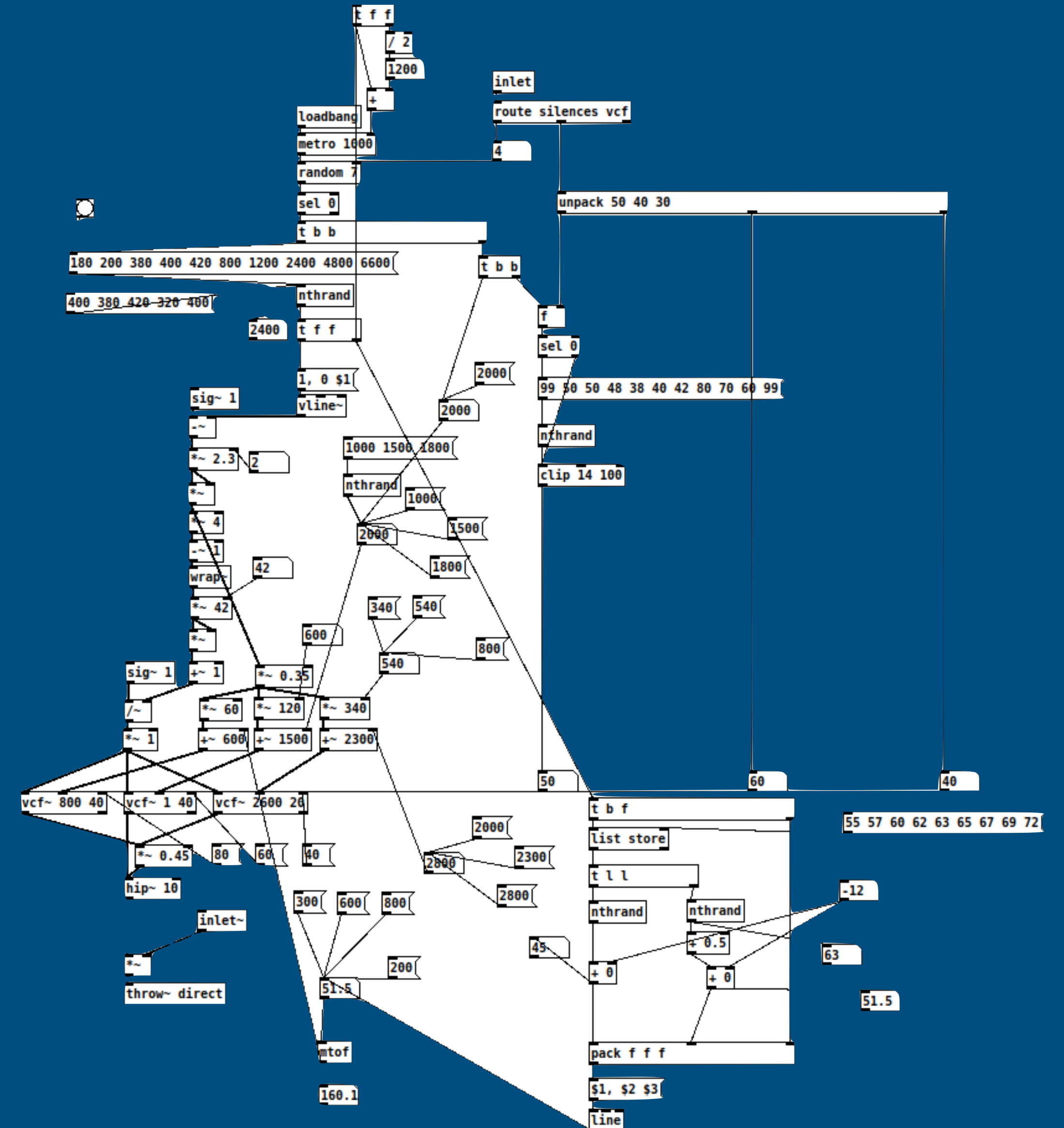


Fonocanal

*Balise sonore près les berges du canal
de Bourgogne en pays tonnerrois*

DOCUMENTATION

Auteur : Fred Voisin, 2020



Auteur

Frédéric Voisin

12 rue du dessus, 21540 Saint Mesmin

Téléphone : 09 54 16 10 40

Email : fred@fredvoisin.com

Site Web : www.fredvoisin.com

Commanditaire

Ici & Là - une poétique du canal de Bourgogne

28 grande rue basse, 89430 Tanlay

Téléphone : 09 53 67 45 02

Fax : 09 53 67 45 02

Email : ici.et.la.canal@gmail.com

Site Web : www.ici-la-canaldebourgogne.fr

Sommaire

Principe général	p. 4
Avertissement	p. 5
Matériel	p. 7
Installation	p. 9
Partition	p. 11
Exemples sonores	p. 36
Ressources techniques	p. 39

Avertissement

Le Fonocanal est une œuvre multimedia régie par le droit d'auteur, de même que la partition musicale et la présente documentation qui en permettent l'exploitation artistique. Si les programmes informatiques qui en assurent le fonctionnement son libres de droits, l'exploitation publique ou privée du Fonocanal fait l'objet de contrats de cession de droits d'auteur.

Les données statistiques (fréquentation, météo) sont accessibles sur le site web du fonocanal, à la page <https://fonocanal.net/stats>

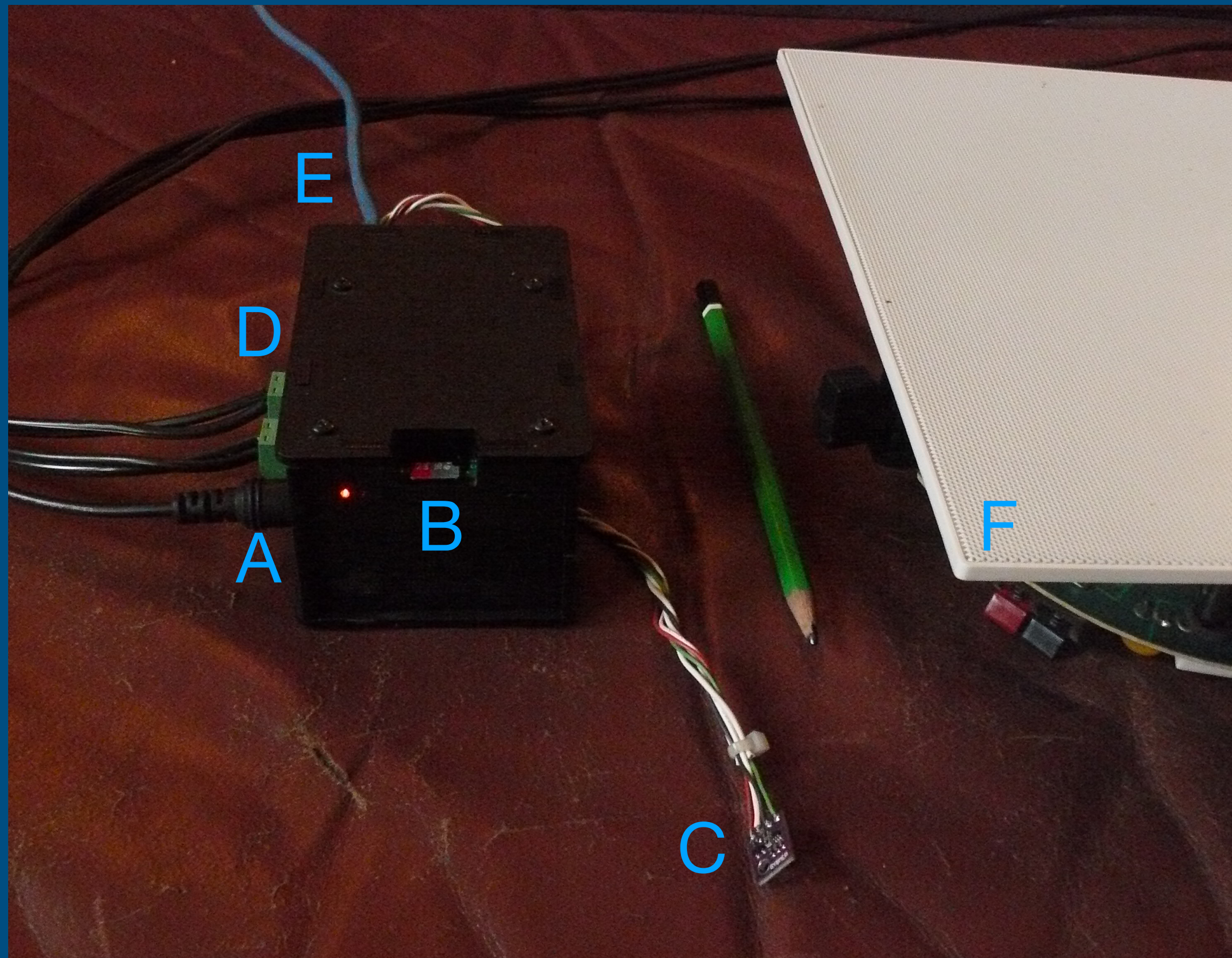
Pour toute information, merci de contacter l'auteur, Frédéric Voisin, ou l'association commanditaire de l'œuvre, *Ici et Là, une poétique du canal de Bourgogne*.

Principe général

Le Fonocanal est une balise sonore autopoïétique près les berges du canal de Bourgogne en pays tonnerrois. Il émet spontanément des sons et des sonorités qui varient selon le cycle des saisons, des journées et les conditions météorologiques. Il joue en temps-réel une partition ouverte qui intègre ces aléas dans la conduite de processus stochastiques préalablement composés et exécutés par de petits synthétiseurs modulaires audionumériques spécialement conçus. Les sons qu'il émet surprennent et interpellent le passant. Ils évoquent tantôt l'environnement naturel, tantôt le patrimoine culturel du tonnerrois.

Un bouton-poussoir permet au public d'écouter des productions réalisées au cours d'ateliers de création artistique et pédagogiques.

Matériel



Matériel de Fonocanal du Fonocanal en fonctionnement (Photo : Fred Voisin)

L'unité centrale consiste en un boîtier qui accueille une carte Raspberry Pi modèle 3B+ flanquée d'un amplificateur audio stéréo 2 x 45 W RMS (IQAudio DigiAMP+)

A - alimentation 19 V - 2A (photovoltaïque)

B - Mémoire SD (partition et système. d'expl.)

C - sonde atmosphérique

D - sortie audio amplifiée vers hauts-parleurs

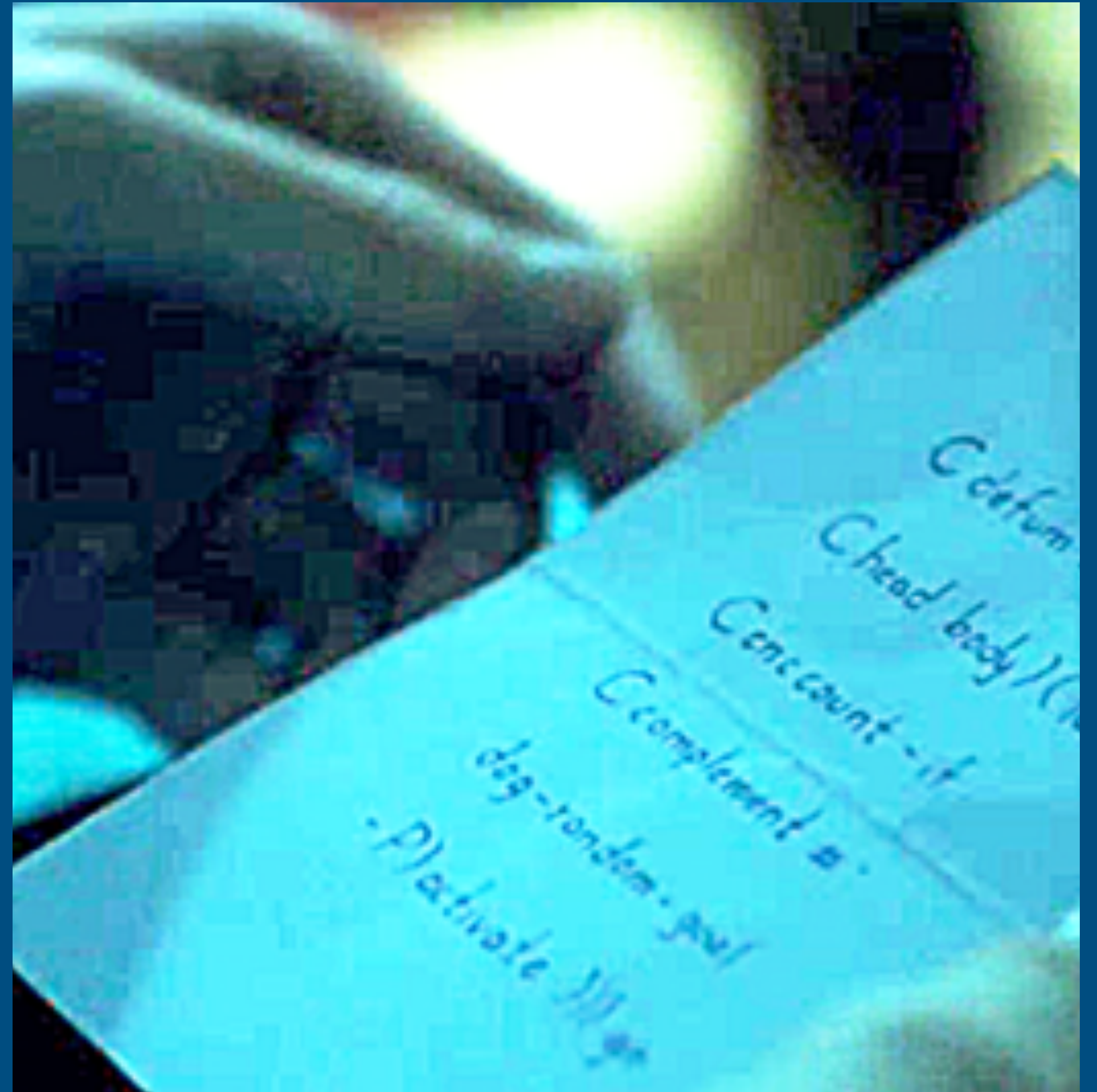
E - câble bouton-poussoir public

F - 2 haut-parleurs étanches

Installation

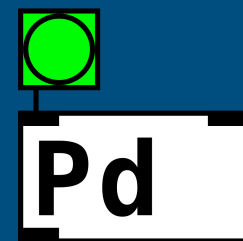
Au titre d'œuvre multimedia, le programme informatique constituant *Fonocanal* est distribué par l'auteur et ses représentants par le truchement d'un support de mémoire "flash SD" d'un volume de 8 Go, ce exclusivement pour son fonctionnement, ainsi que pour les productions artistiques et ateliers pédagogiques auquel il donne lieu.

L'installation s'effectue préalablement au démarrage de l'Unité Centrale en insérant cette cartemémoire dans l'habitacle prévu de recevoir son système d'exploitation.



Environnement logiciel requis

- Puredata Vanilla



- Bourne-Again Shell



- Lisp



- Python



- Apache, PHP

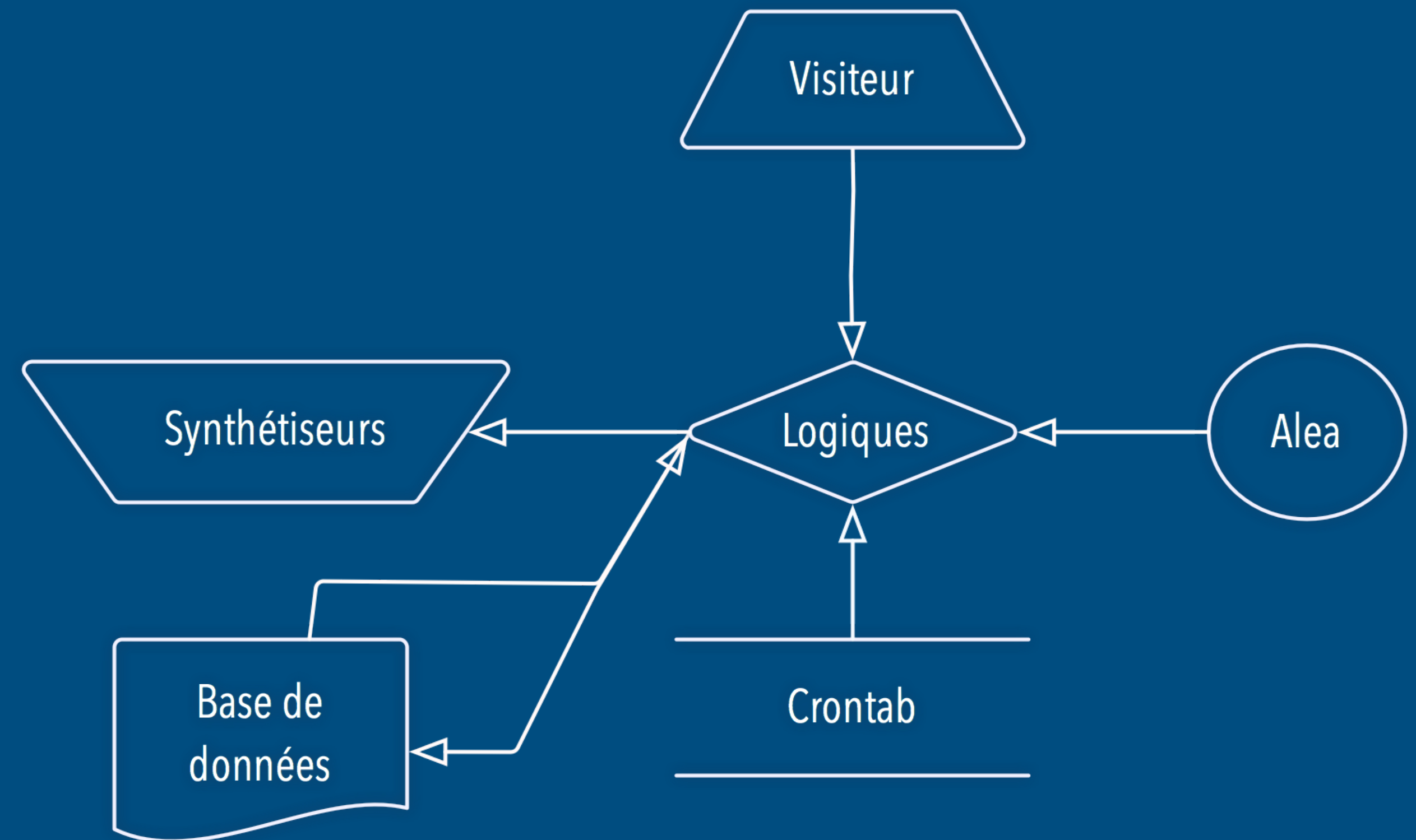


- Linux Raspbian



Les langages, formats et environnements informatiques auxquels font appel la partition (le code source) de *Fonocanal* figurent parmi les plus pérennes. Le langage Lisp et l'environnement Puredata sont des environnements fondateurs de l'informatique musicale. Ils sont ici exécutés au moyen de logiciels libres et de langages informatiques populaires (Python, PHP, Bash...).

Partition



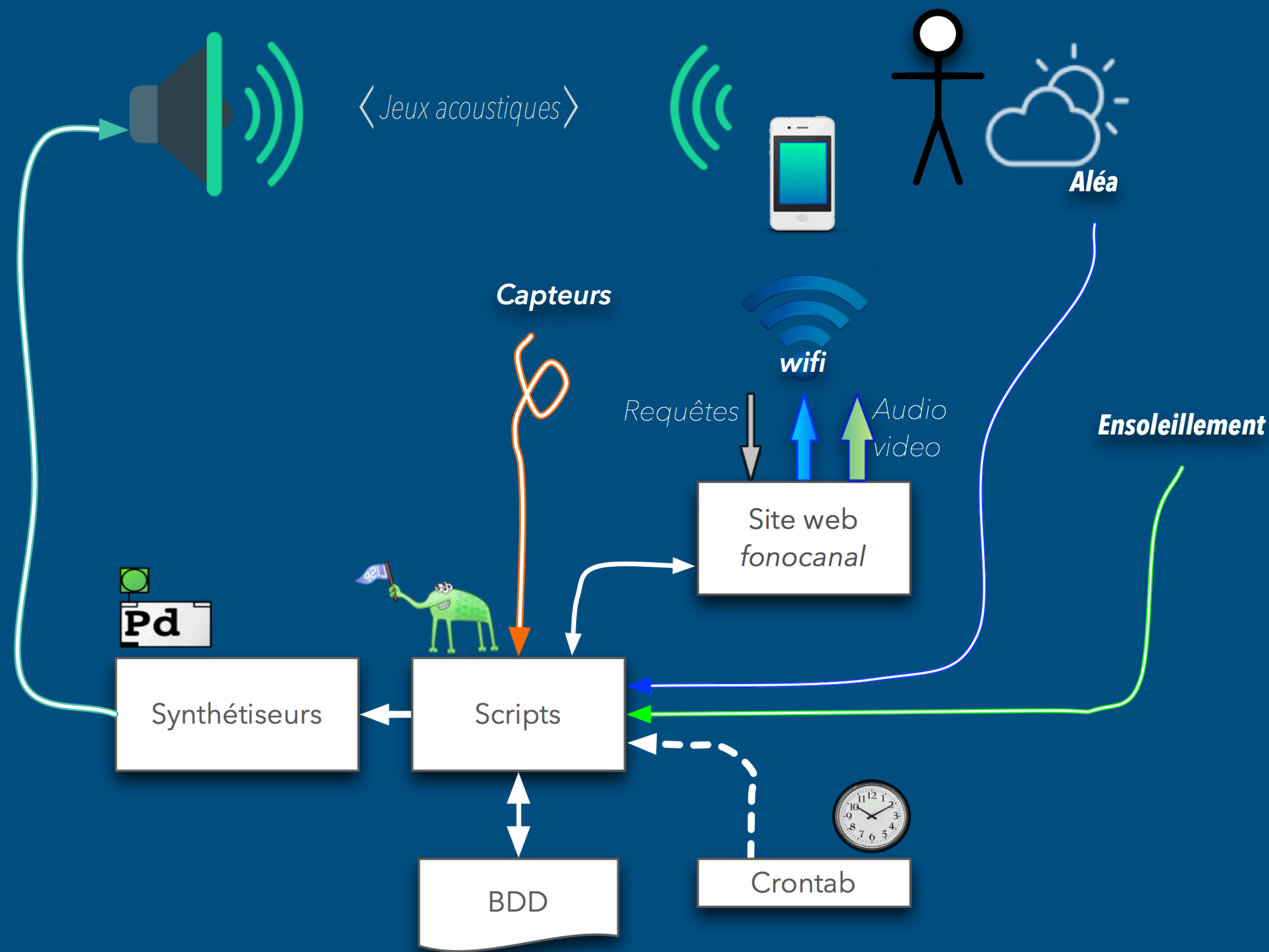
La partition de Fonocanal règle et contrôle l'ensemble de sa production sonore pour une durée infinie. **La partition est "ouverte"** :

- elle n'est pas déterministe et n'a pas de fin préétablie
- les données météorologiques participent de son exécution
- elle peut intégrer dynamiquement lors de son exécution de nouvelles instructions (scripts) et variantes de synthétiseurs provenant de différents auteurs et inspirées de la partition originale qui, autant que possible, est écrite avec un souci didactique
- elle se présente sous la forme d'un code source constitué de scripts et de patches et de données au format texte exclusivement, sous le répertoire ~/phonocanal/

La partition et ses développement ultérieurs respectent des principes :

- **instrumentaux** : *les sons de Fonocanal sont produits en temps-réel par des synthétiseurs spécialement conçus dans l'environnement PureData de telle manière qu'ils peuvent jouer de manière autonome :*
 - *selon les commandes qu'exécute la partition à travers le système d'exploitation (crontab)*
 - *et/ou lorsque le bouton poussoir est activé (message "bouton bang;", dans Puredata)*
 - *et/ou des requêtes Php via le site web local accessible par le via le réseau wifi local 'Fonocanal'.*
- **bio-acoustiques** : *les sons sont produits de telle manière qu'il se fondent aussi "naturellement" que possible, par imitation, par variation ou par camouflage, dans l'environnement immédiat d'une berge du canal de Bourgogne tout en étant de nature à attirer l'attention d'un passant avec discrétion.*
- **de simplicité** : *les synthétiseurs qui produisent les sons respectent une économie du calcul et d'énergie, outre une fin didactique.*
- **de robustesse** : *pour être joué et contrôlé de tout temps, sans maintenance.*

Synopsis



La partition initiale est constituée :

1. de **scripts** *Bash originaux qui établissent et régissent tous les processus - les logiques - de fonocanal, des capteurs (mécaniques, sondes, interface wifi) aux synthétiseurs.*

Il s'agit de programmes originaux exécutés par le système d'exploitation. Au nombre d'une centaine, ils sont exécutés automatiquement pour mettre en œuvre et régir *fonocanal* en relation avec le système d'exploitation, l'environnement (atmosphère) et les visiteurs (bouton-poussoir). Ils sont réunis au sein du répertoire `phonocanal/scripts`



- Les données atmosphériques sont mises à jour automatiquement dans l'environnement Puredata via le script `sonde.sh` toutes les 10 minutes. Ces mêmes données sont enregistrées et horodatées dans le fichier `~/phonocanal/data/atmosphere`.
- Les activations du bouton-poussoir sont transmises par le script `bouton.sh` dans l'environnement Puredata via le message 'bouton bang;'. Elles sont enregistrées et horodatées dans le fichier `~/phonocanal/logs/bouton`.

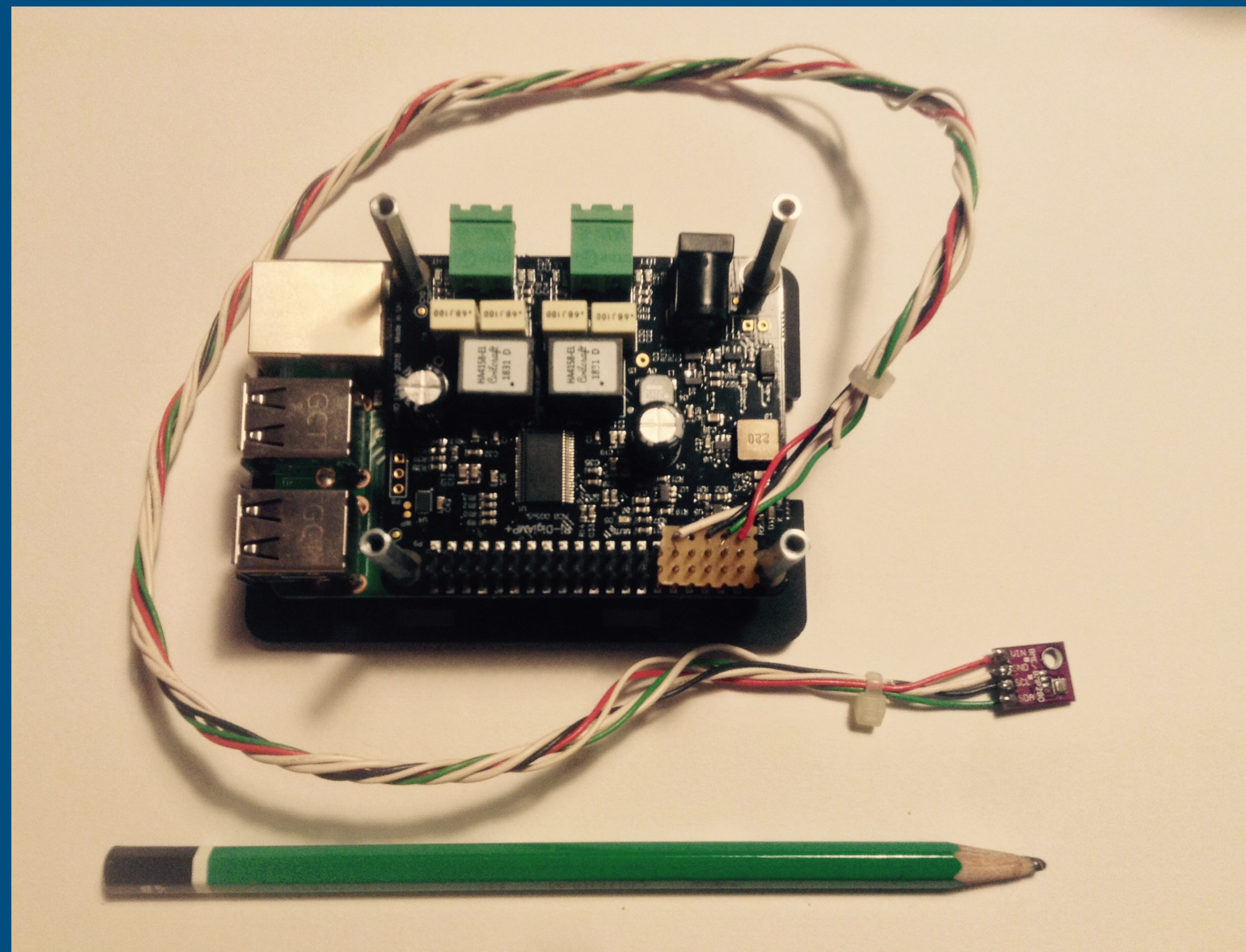
Ces données sont périodiquement mises à jour sur le site <https://fonocanal.net/stats>

```

pi@phonocanal0A: ~/phonocanal/scripts
pi@phonocanal0A:~/phonocanal/scripts $ sh sonde.sh
BMx280_HW_ID : 0x60
BME280 found. Humidity available.
Temperature: 16.57 C
Pression: 141484.27 Pascals
Humidite : 53.43 %
Temperature: 16.59 C
Pression: 141419.26 Pascals
Humidite : 53.24 %
Temperature: 16.57 C
Pression: 141474.63 Pascals
Humidite : 53.13 %
Temperature: 16.58 C
Pression: 141467.09 Pascals
Humidite : 53.07 %
Temperature: 16.58 C
Pression: 141457.46 Pascals
Humidite : 53.13 %
Temperature: 16.57 C
Pression: 141481.05 Pascals
Humidite : 53.04 %
Temperature: 16.58 C
Pression: 141453.64 Pascals
Humidite : 53.17 %
Temperature: 16.58 C
Pression: 141447.21 Pascals
Humidite : 53.00 %

```

Sondage automatique des conditions atmosphériques (script sonde.sh)



Sonde atmosphérique Bosch bme280, UC Raspberry Pi 4B+ flanquée d'un amplificateur audio Digi-AMP+ constituant, avec la partition, le cœur de Fonocanal

(Photo : Fred Voisin)


```

pi@phonocanal0A: ~/phonocanal/scripts

pi@phonocanal0A:~/phonocanal/scripts $ cat ./play
#!/bin/bash
# play
# charge et active le synthétiseur (arg2) dans une voix (arg2)
# s'etale sur plusieurs secondes

# test ici si le cpu le permet
# si non, libere le synthe le plus gourmand et prend sa place

pd="$1"
voix="$2"

dir="/home/pi/phonocanal/scripts"
dirsynthe="/home/pi/phonocanal/phonocanal-pd"
synthe=$(basename $pd .pd)
ok=0
cpu=$(($dir/askcpu.sh))

if [ "$cpu" -gt 75 ] then
    echo "%cpu CPU : too high !"
    pid=$(ps aux |grep "pd -schedlib" |awk '{print $2,$3}' | \
        sort -Vr -k 2 |head -n1 |awk '{print $1}')
    echo "killing $pid..."
    kill -9 "$pid"
    else
        ok=1
fi

if [ "$ok" -gt 0 ] then
    echo "on charge $pd..."
    $dir/tellpd "synthe $voix load $pd"
    sleep 1
    $dir/tellpd "synthe $voix start $pd"
    sleep 1
    $dir/tellpd "synthe $synthe dsp 1"
    echo "$synthe ok."
fi
pi@phonocanal0A:~/phonocanal/scripts $

```

Extrait du script play.sh

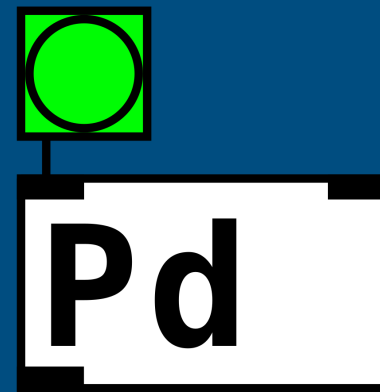
Le script ci-contre est chargé de convoquer un synthétiseur décrit dans un *patch* au format .pd.

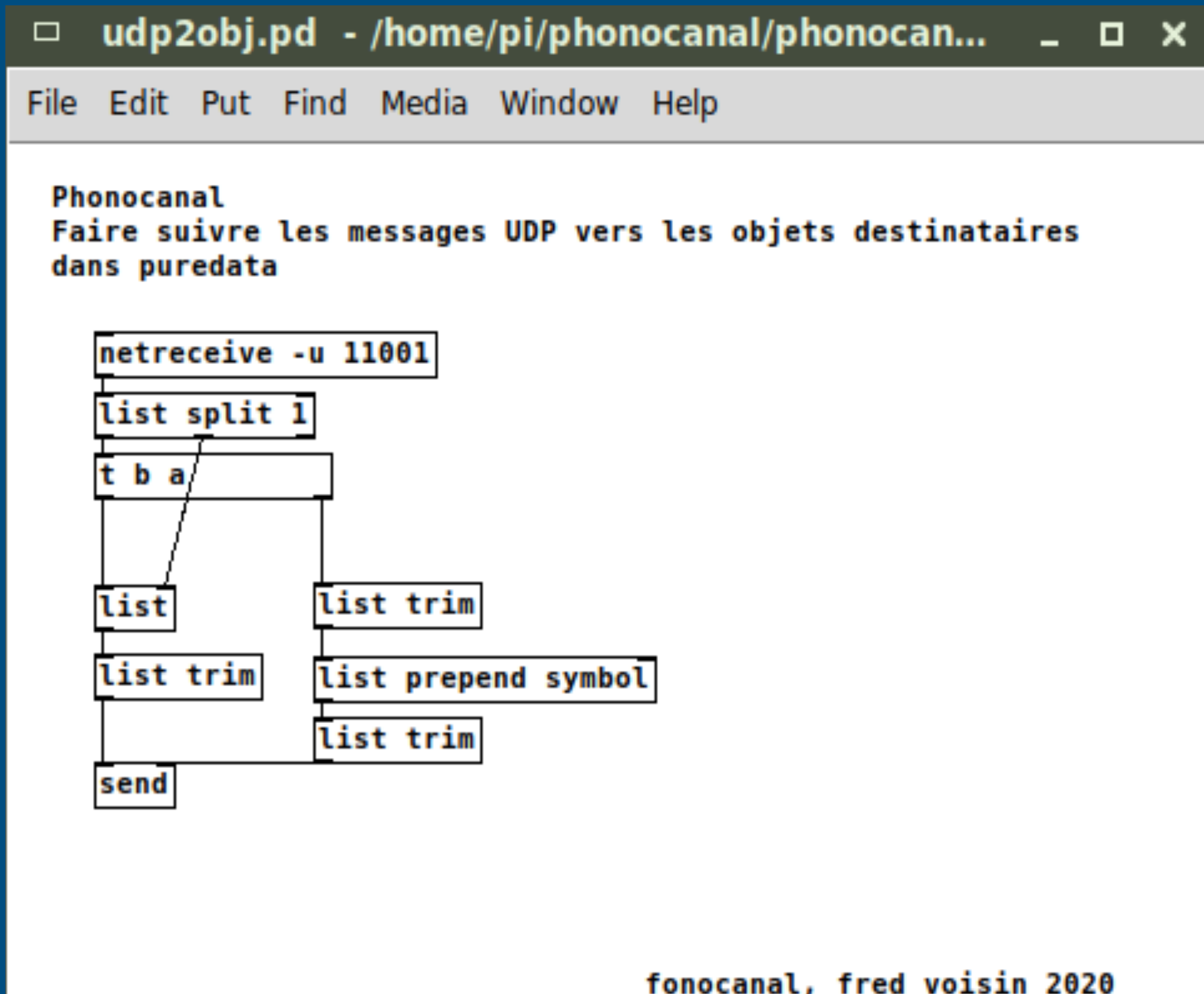
Si la réserve en CPU est trop faible, le synthétiseur le plus gourmand est chassé.

Ce script es appelé par d'autres scripts qui régissent la distribution des synthétiseurs dans la production sonore dans le temps. Ainsi par exemple, les synthétiseurs sont plus nombreux et variés l'été que l'hiver.

2. de ***patches***, qui décrivent les synthétiseurs et leur logique de contrôle particulière.

Il s'agit de programmes originaux créés dans l'environnement Puredata (.pd) sous la forme de représentations graphiques et fonctionnelles des logiques musicales et de traitement du signal des synthétiseurs. Leur nombre est amené à évoluer en production. Trois patches : `scene.pd`, `outputs.pd` et `udp2pd.pd` assurent le contrôle et la diffusion de tous les synthétiseurs, lesquels exécutent les instructions de jeu que leur envoient périodiquement les ***scripts***. Ils sont réunis au sein du répertoire unique `phonocanal/patches`.





Exemple de patch Puredata : `udp2pd.pd`

Ici le patch `udp2pd.pd` ci-contre assure l'acheminement des commandes adressées aux synthétiseurs par les différents scripts Bash.

Ces commandes sont des messages formés d'une ligne de mots et de nombres dont en premier lieu le nom (l'adresse) du synthétiseur.

Par ex.: "criquet nuit période 10;"

Elles sont acheminées avec le protocole TCP/IP en mode 'asynchrone' UDP (port UDP 11001), préféré pour sa fiabilité.

3. d'une **base de données** qui renseigne les scripts et les patches de Fonocanal.

Elle est constituée de données éthologiques, de listes de paramètres de sonification, de transcriptions de mouvements, de synthétiseurs, de fichiers audio indexés qui participent aux productions et interactions sonores, tous réunis au sein du répertoire unique `~/phonocanal/data`. Des *scripts* et des *patches* l'interrogent et mettent à jour continuellement les informations qu'apporte l'environnement de *Fonocanal*.




```

pi@phonocanal0A: ~/phonocanal/data
(setq *oiseaux*
 '(
  ((chouette epeiche) (0 0 10 30 60 80 90 80 50 15 0 0)
   (ville village parc jardin) t)
  ((chouette cheveche) (0 10 50 100 60 10 0 0 0 0 0 0)
   (champs bocage friche) t)
  ((chouette hulotte) (100 100 100 60 30 0 0 0 20 40 80 100)
   (champs bocage friche ville village parc jardin) t)
  ((hibou moyen duc) (0 20 80 100 80 20 0 0 0 0 0 0)
   (champs bocage friche bois foret) t)
  (choucas (10 50 90 100 100 80 50 40 60 100 80 20)
   (ville village parc jardin) nil)
  (martinet (0 0 0 10 40 80 100 90 20 0 0 0)
   (ville village parc jardin) nil)
  (moineau (50 75 100 100 100 75 50 50 50 50 50)
   (ville village parc jardin champs friche bocage) nil)
  (pie (100 100 90 60 50 50 50 60 70 70 60 75)
   (ville village parc jardin bocage champs friche littoral rivière bois foret) nil)
  (sitelle (25 80 100 75 75 60 60 0 60 50 40 30)
   (ville village parc jardin bois foret) nil)
  (faucon (0 20 40 90 100 90 50 20 0 0 0 0)
   (ville village parc jardin bocage champs friche bocage littoral etang lac rivière) nil)
  (tourterelle (10 30 60 100 75 45 45 50 55 60 30 10)
   (ville village parc jardin) nil)
  (pigeon (20 30 60 100 100 100 100 80 40 30 20)
   (ville village parc jardin bois foret bocage champs friche bocage) nil)
  ((mesange charbonniere) (100 100 100 100 75 50 25 0 10 20 10 50)
   (ville village parc jardin bois foret champs friche bocage) nil)
  ((mesange bleue) (50 100 100 100 75 50 35 20 0 0 0 20)
   (ville village parc jardin) nil)
  (rougequeue (0 0 10 50 100 100 60 40 30 70 70 10)
   (ville village parc jardin littoral rivière etang lac falaise montagne) nil)
  (verdier (0 40 100 100 100 100 50 0 10 20 10 0)
   (ville village parc jardin bois foret bocage friche champs bocage) nil)
  ))
--More--(48%)

```

Extrait du fichier oiseaux-db.lisp

Par exemple, ce fichier oiseaux-db.lisp réunit des données relatives à la présence de différents oiseaux, selon lieux et période de l'année.

Elles informent, au moyen des scripts, le calcul de la probabilité du jeu des synthétiseurs dont le comportement général s'aligne sur celui d'oiseaux.

- ▶ Des éléments de la partition sont portés au sein de la *table chronologique* de planification du système d'exploitation (**crontab**) ;
- ▶ Le **site web** local "Fonocanal" peut participer du jeu, au moyen de fonctionnalités et scripts PHP qui font appel aux *scripts* de la partition ;
- ▶ Outre la partition, le système d'exploitation et les programmes informatiques requis sont libres de droit dans le présent usage artistique (GNU-Linux).

"crontab"

```
pi@phonocanal0A:~ $ uname -a
Linux phonocanal0A 4.19.66-v7+ #1253 SMP Thu Aug 15 11:49:46 BST 2019 armv7l GNU/Linux
pi@phonocanal0A:~ $ sudo lshw |grep product |head -n1
    product: Raspberry Pi 3 Model B Plus Rev 1.3
pi@phonocanal0A:~ $ sudo crontab -l
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command
### phonocanal cron
### fred voisin 2020
# For more information see the manual pages of crontab(5) and cron(8)
# and the documentation of fonocanal

## ASTRONOMIE
2 0 * * * /home/phonocanal/scripts/astro.sh
## METEO
3 * * * * /home/phonocanal/scripts/actuel.sh
16 * * * * /home/phonocanal/scripts/actuel.sh
26 * * * * /home/phonocanal/scripts/actuel.sh
36 * * * * /home/phonocanal/scripts/actuel.sh
46 * * * * /home/phonocanal/scripts/actuel.sh
56 * * * * /home/phonocanal/scripts/actuel.sh
56 * * * * /home/phonocanal/scripts/actuel.sh
56 * * * * /home/phonocanal/scripts/actuel.sh

## CONFIGURATION ESPACE - TEMPS
# reinitialisation quotidienne à 3h03 (reboot général)
3 3 * * * /home/phonocanal/scripts/fadeout-all.sh 0 120
3 7 * * 1 /home/phonocanal/scripts/reload-fonocanal.sh
```

Extrait du crontab du système d'exploitation (/etc/crontab)

Chaque jour,

- à 2h00 : calcul des éphémérides : astro.sh
- à 3h03, initialisation automatique de la configuration des synthétiseurs selon les à conditions atmosphériques et la réserve d'énergie : choix de l'effectif, effets, rythme et allure (cf. *Base de données*).

Le choix des synthétiseurs est adapté à la réserve d'énergie, la pression atmosphérique, la saison et le jour de la semaine.

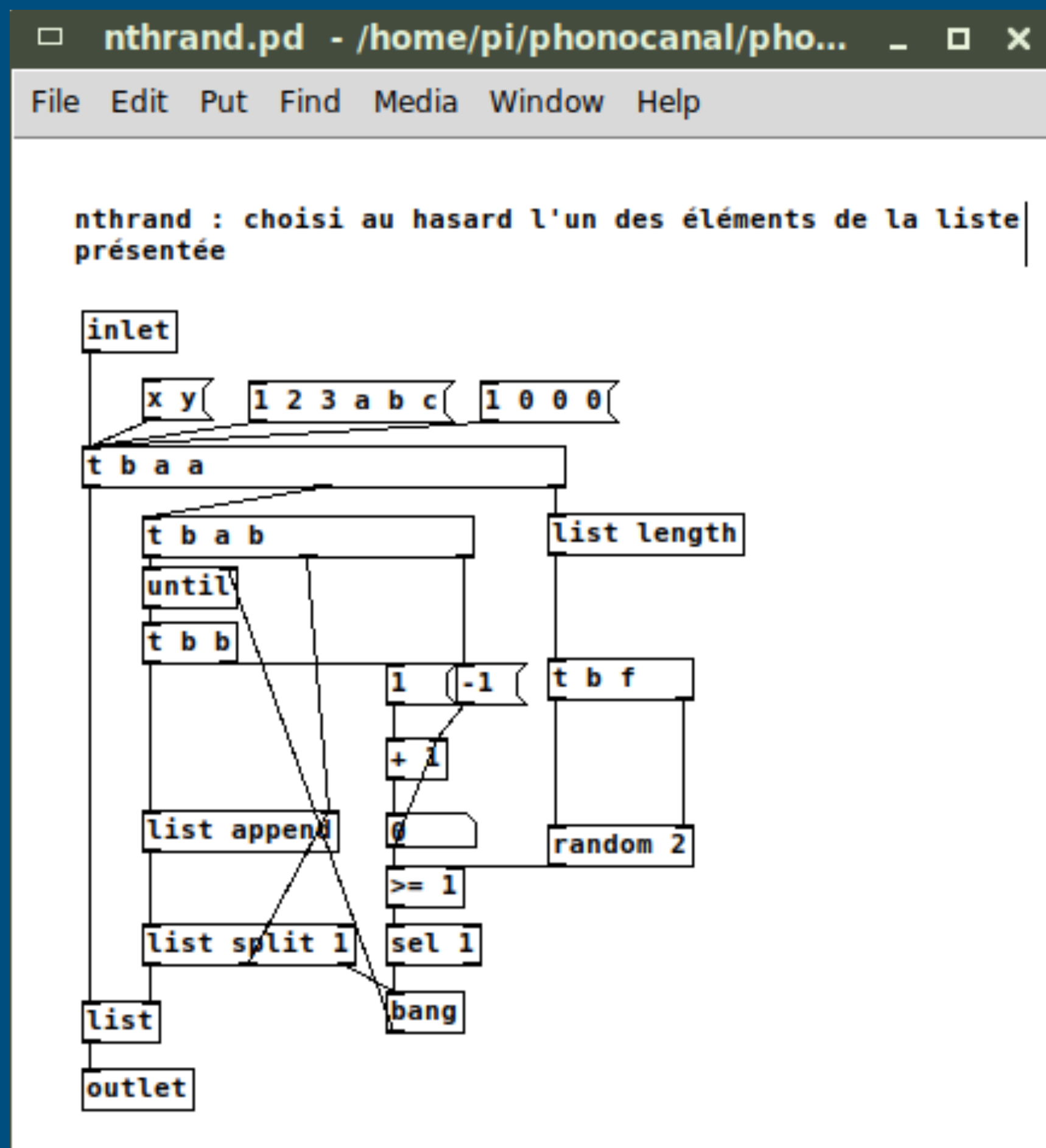
Toutes les 10 minutes

- mise à jour des données météorologiques (actuel.sh)

Et toutes les 20 minutes :

- mise à jour des synthétiseurs
- ...

Logique stochastique

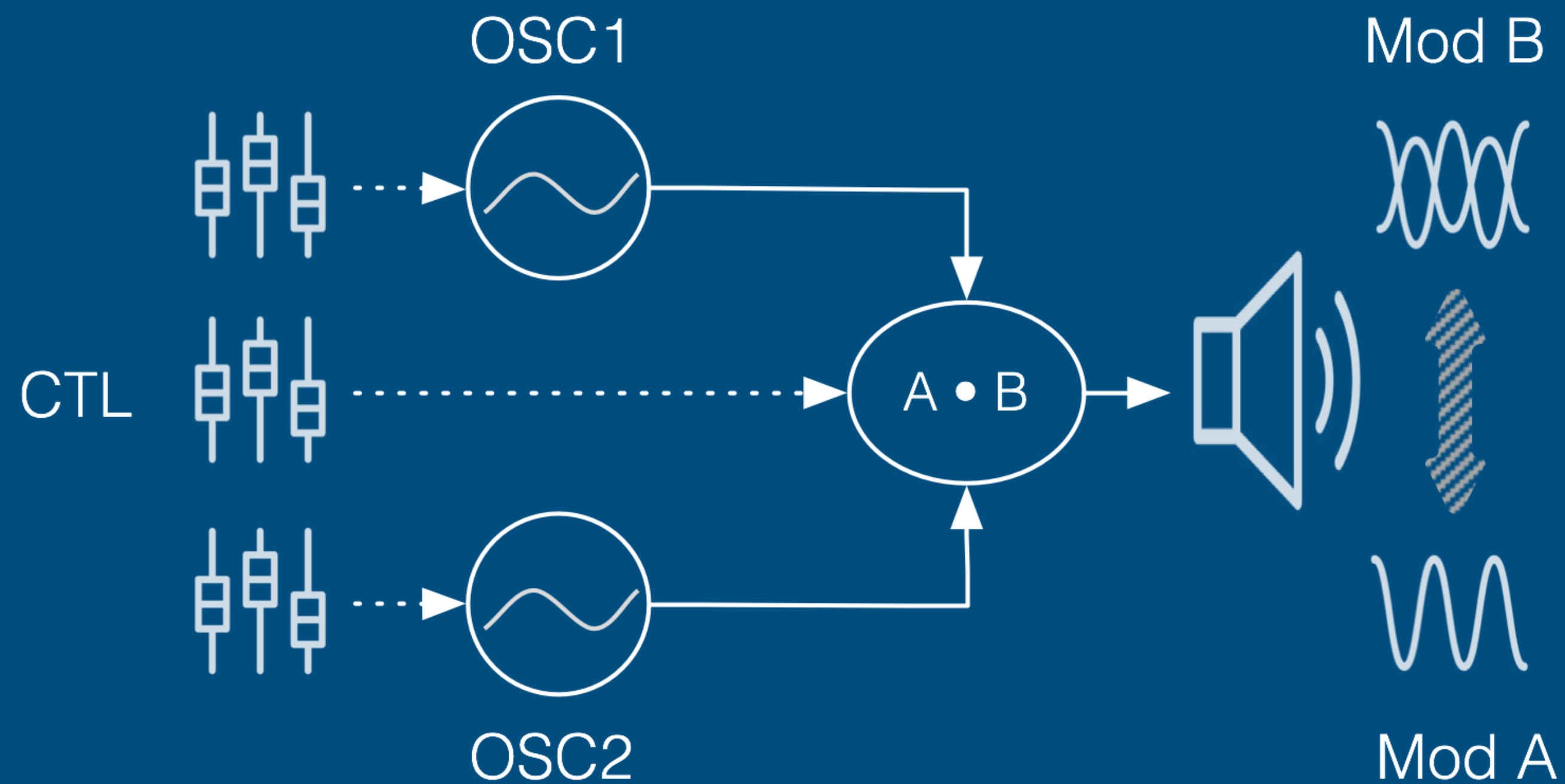


Des **processus sobres** permettent de générer des séquences qui répondant à des indications précises en termes stochastiques et prédéfinies par l'auteur.

Dans cet exemple simple (nthrand.pd), l'instruction consiste en une liste qui représente de manière triviale les valeurs souhaitées et leur **loi de distribution**. Ici la présentation de la liste "1 0 0 0" donnera lieu à l'output "1" avec une probabilité d'1/4. Les listes peuvent recevoir jusqu'à 256 valeurs. L'espace des possibles est de préférence strié de modes ou de gammes pentatoniques voire diatoniques, ou de durées plus ou moins proportionnelles.

Modèles "organiques"

A différentes manières d'agencer ou de moduler des oscillateurs simples correspondent des variétés de sonorités qui **imitent ou évoquent machines, fluides, animaux...**



Un même synthétiseur peut, en alternant des valeurs contrastées de ses intensités de contrôle (régimes), alterner des sons qui semblent être tantôt d'origine animale, tantôt d'origine humaine, ou mécanique, ou électronique (Mod A / Mod B).

Par soucis d'économie de la réserve d'énergie, les synthétiseurs font appel aux processus audionumériques les plus simples, notamment oscillateurs et filtres, en nombre moindre. Bien qu'automatique, leur contrôle est, à l'inverse, plus sophistiqué.

Registres des synthétiseurs

Chacun des sons est toujours produit avec l'intention d'être apparenté à l'un au moins des 3 registres suivants :

- ▶ Géophonique (G) des sons naturels d'origine tellurique et atmosphérique (vent, ruisseau,...)
- ▶ Biophonique (B) des sons d'origine animale ou végétale (appels, chants, stridulations, grincements,...)
- ▶ Anthropophonique (A) des sons d'origine humaine et artificielle (parole, musique, machines,...)



“aube.sh”

```
pi@phonocanal0A: ~/phonocanal/scripts
pi@phonocanal0A:~/phonocanal/scripts $ cat aube.sh
#!/bin/bash
# fonocanal,fred voisin 2020

dir=/home/pi/phonocanal/scripts

# on attend l'aube
HEURE=`date |cut -d, -f2 |cut -d: -f1`
MINUTE=`date |cut -d, -f2 |cut -d: -f2`
HM=$( expr $HEURE \* 60 + $MINUTE )
START=`tail -n 1 $dir/../data/astro.csv |cut -d, -f2`
STARTH=$( echo $START |cut -d: -f1 )
STARTM=$( echo $START | cut -d: -f2 )
STARTHM=$( expr $STARTH \* 60 + $STARTM )
WAIT=$( expr $STARTHM - $HM )
DELAI=$(expr $WAIT \* 60 )
if [ $DELAI -lt 0 ]
then DELAI=0
fi

# un oiseau chante au lever du soleil
# selon meteo ce dont se charge le script oiseau-solo
sleep $DELAI
$dir/fonolog "aube..."
$dir/oiseau-solo 1 96 60

sleep 1200
# apres 20 minutes un autre oiseau plus calme le rejoint
# selon meteo
$dir/oiseau-solo 1 80 60

sleep 2400
# apres une heure un autre oiseau chante (toujours)

$dir/tellpd "synthe 3 load merle.pd"
sleep 1
$dir/tellpd "synthe 3 start merle.pd"
$dir/tellpd "synthe merle dsp 1"
$dir/tellpd "synthe merle play 5"
$dir/fonolog "aube terminée."

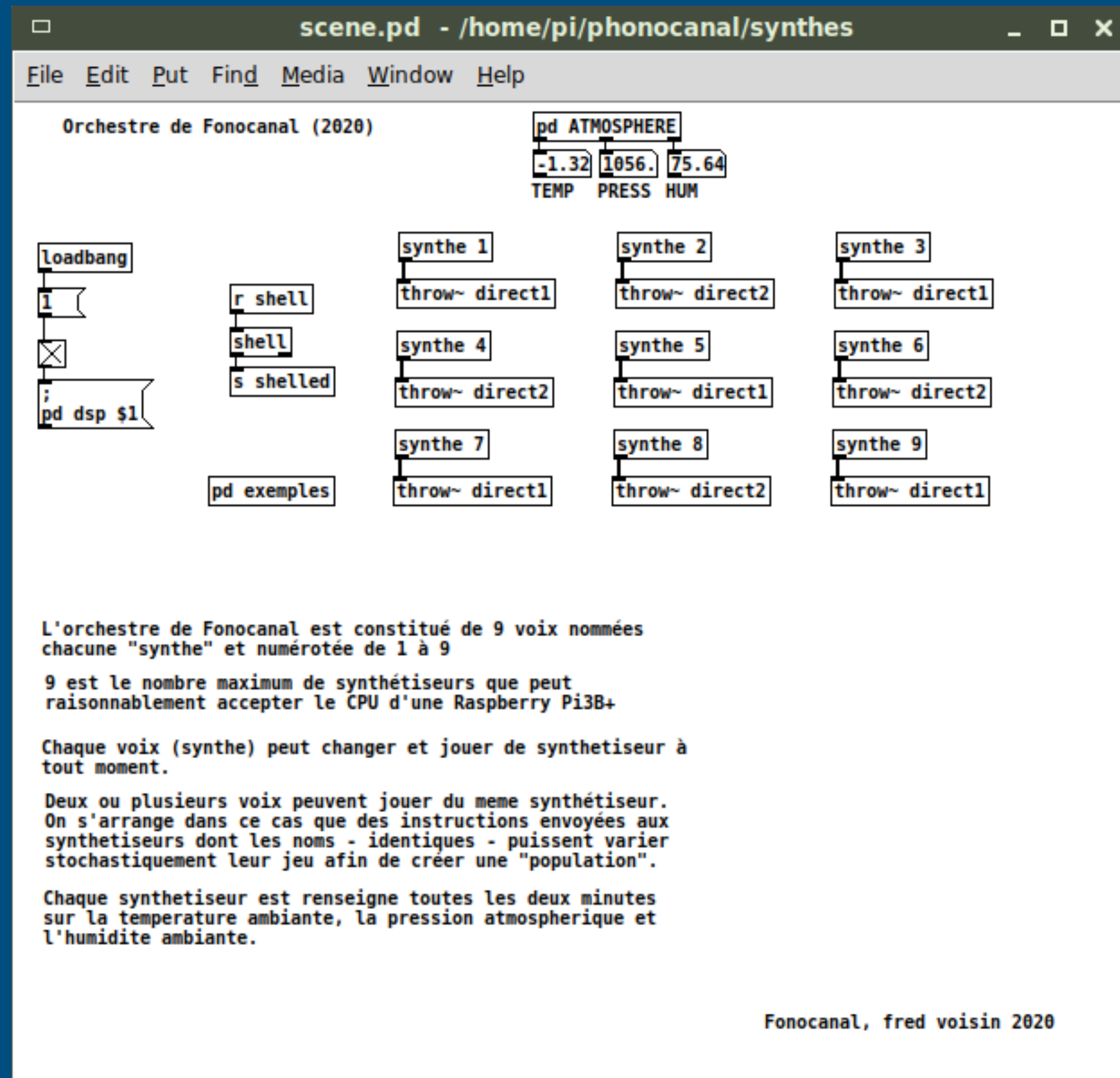
pi@phonocanal0A:~/phonocanal/scripts $
```

Script aube.sh

Exemple de script exécuté chaque jour, à 4h

- le premier bloc définit l'heure de lever du soleil (qui a été mise à jour à 2 heures) dans le fichier astro.csv de la base de données
- à l'heure précise du lever du soleil, un synthétiseur est convoqué et joue selon les conditions atmosphériques actuelles dans les conditions du *script* oiseau-solo.
- Après 20 minutes, un autre synthétiseur de la même espèce se joint éventuellement au premier
- Une heure après lever du soleil, le synthétiseur du *patch* merle.pd joue, indépendamment des conditions.

Production sonore

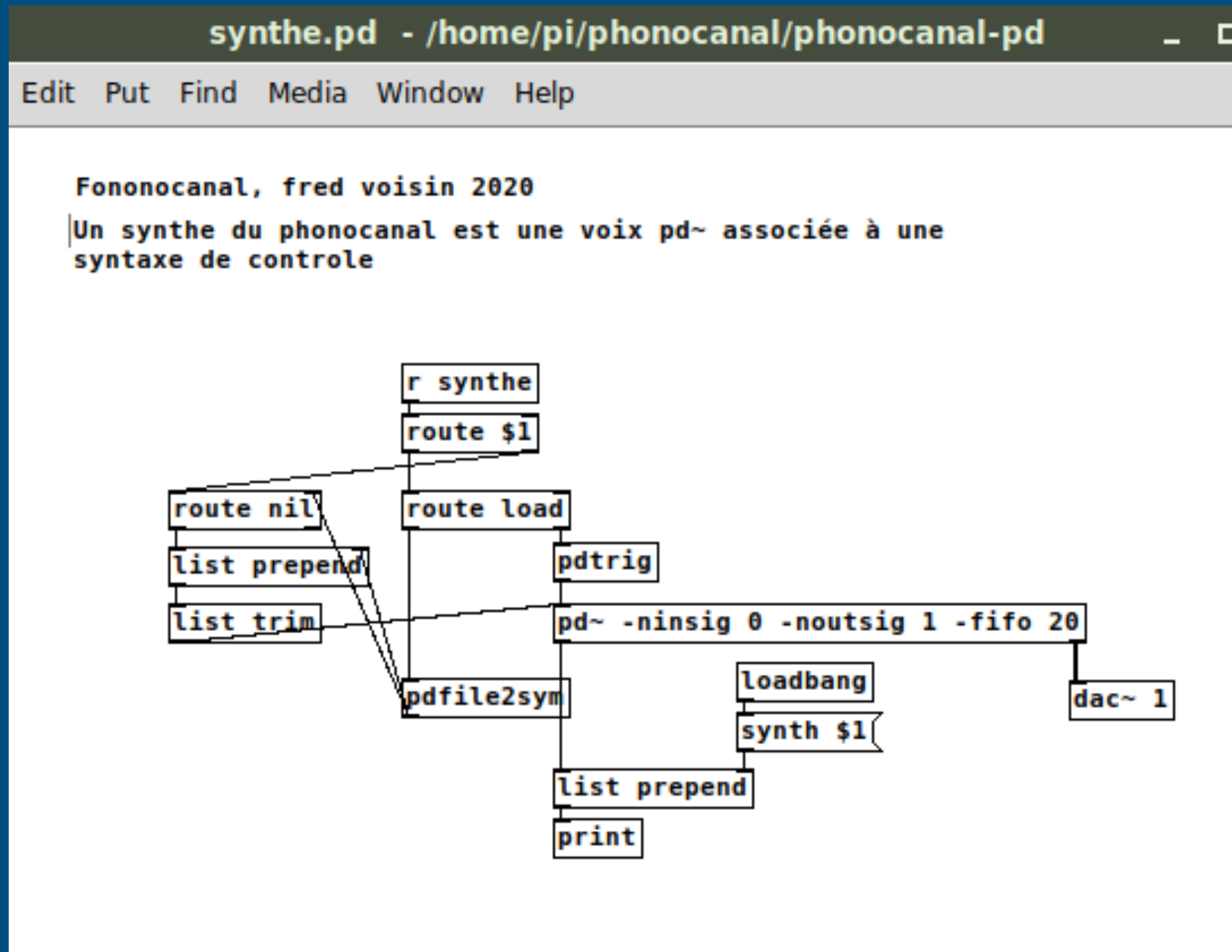


[Patch scene.pd](#)

La production sonore est assurée par un ensemble de synthétiseurs indépendants les uns des autres et qui répondent chacun aux instructions de la partition, au sein du patch scene.pd, ci-contre, lequel permet de jouer jusqu'à neuf synthétiseurs simultanément dans les limites des ressources en calcul disponibles (CPU).

La partition étant "ouverte", de nouveaux synthétiseurs peuvent être intégrés à tout moment.

Synthétiseurs

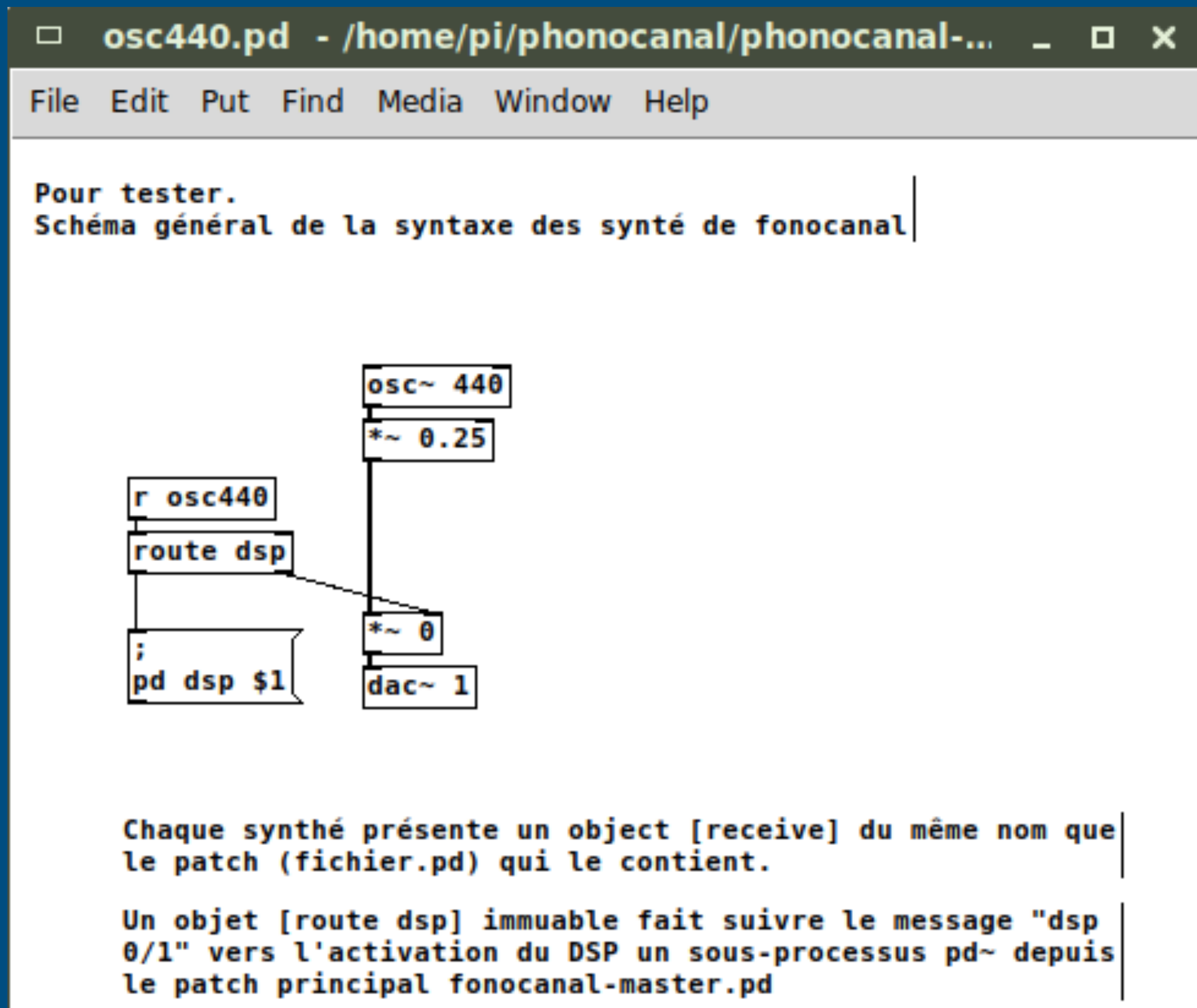


Patch synthe.pd permettant d'instancier dynamiquement un synthétiseur (voix) au format .pd

5 sortes de synthétiseurs assurent la production sonore du *fonocanal* en Temps-Réel, selon

- des modèles organiques (animaux, machines...)
- des modèles de fluides (eau, vent)
- des modèles de bruits (orages, bris)
- des modèles de résonances (cloches, cavités)
- des procédés de synthèse granulaire

Structure des synthétiseurs



[Patch osc440.pd](#)

Chaque synthétiseur est un module indépendant décrit dans un fichier unique au format Puredata (.pd) du nom du synthétiseur.

Il exécute uniquement les messages qui le désigne spécifiquement par ce même nom.

L'exemple ci-contre - osc440 - montre la structure - triviale - que doit respecter tout synthétiseur de fonocanal (ici pour la production d'un la pur).

Instanciation des synthétiseurs

```
pi@phonocal10A:~/phonocal/scripts $ cat play
#!/bin/bash
# Fonocal, fred voisin 2020
# play <patch> <synthe>
# charge et active le synthétiseur (arg2) dans une voix (arg2)
# s'etale sur plusieurs secondes
# test si le cpu le permet
# si non, libere le synthe le plus gourmand et prend sa place

pd="$1"
voix="$2"

dir="/home/pi/phonocal/scripts"
dirdsynthe="/home/pi/phonocal/phonocal-pd"
synthe=$(basename $pd .pd)
ok=0
cpu=$(($dir/askcpu.sh))

if (( $(echo "$cpu > 75" |bc -l ) ))
then
    echo "%cpu CPU : too high !"
    pid=$(ps aux |grep "pd -schedlib" |awk '{print $2,$3}' |\
    sort -Vr -k 2 |head -n1 |awk '{print $1}')
    echo "killing $pid..."
    kill -9 "$pid"
else
    ok=1
fi

if [ "$ok" -gt 0 ]
then
    echo "on charge $pd..."
    $dir/tellpd "synthe $voix load $pd"
    sleep 1
    $dir/tellpd "synthe $voix start $pd"
    sleep 1
    $dir/tellpd "synthe $synthe dsp 1"
    echo "$synthe ok."
fi
```

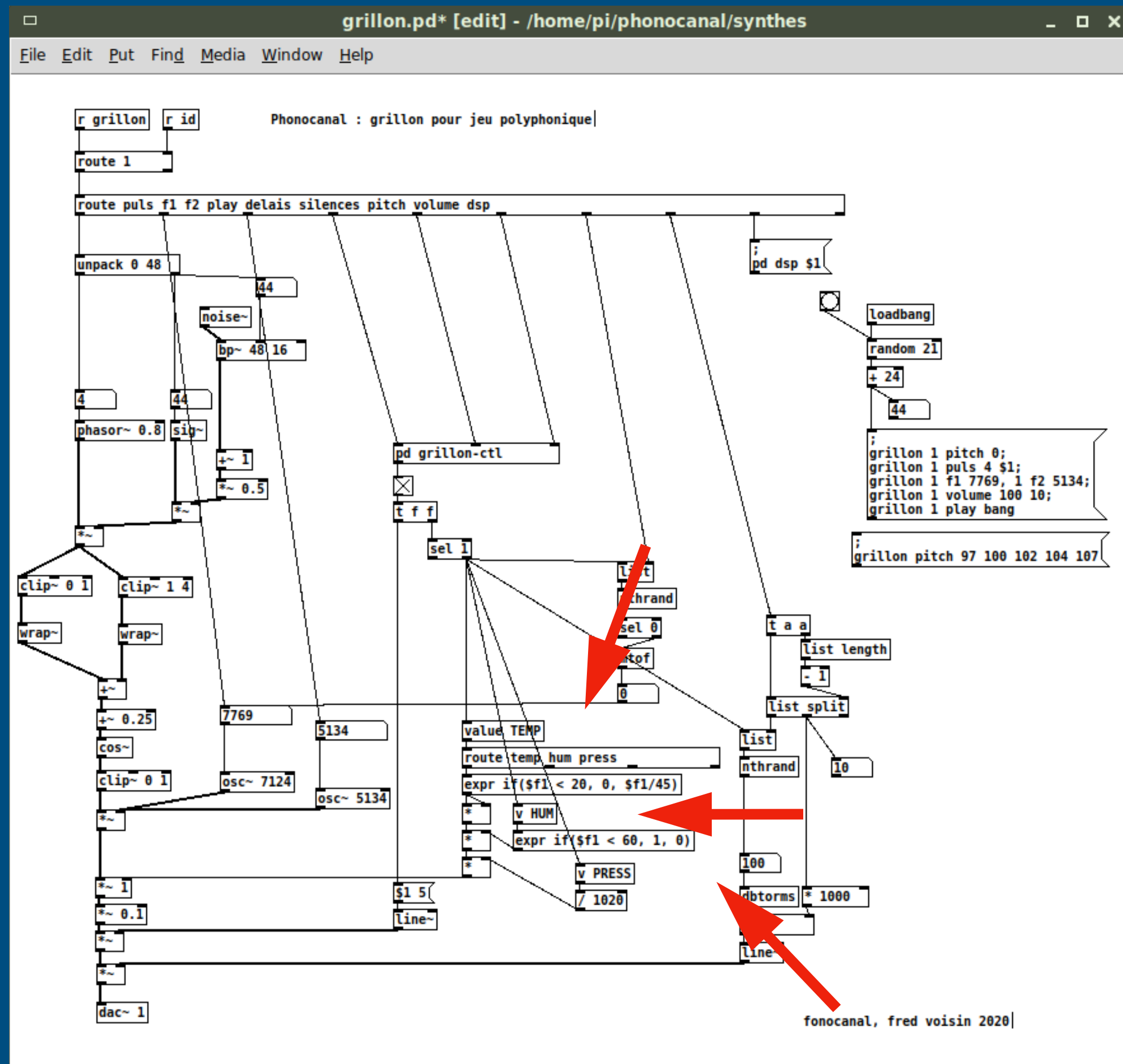
Script play.sh

Ce script permet au système d'instancier dynamiquement un synthétiseur dans l'environnement Puredata sans interrompre le jeu en cours avec la commande suivante à laquelle fait appel la partition :

`play <patch> <voix>`

Il s'assure que la charge du processeur en permet l'exécution, sans quoi le nouveau synthétiseur remplace l'actuel le plus gourmand en ressources CPU.

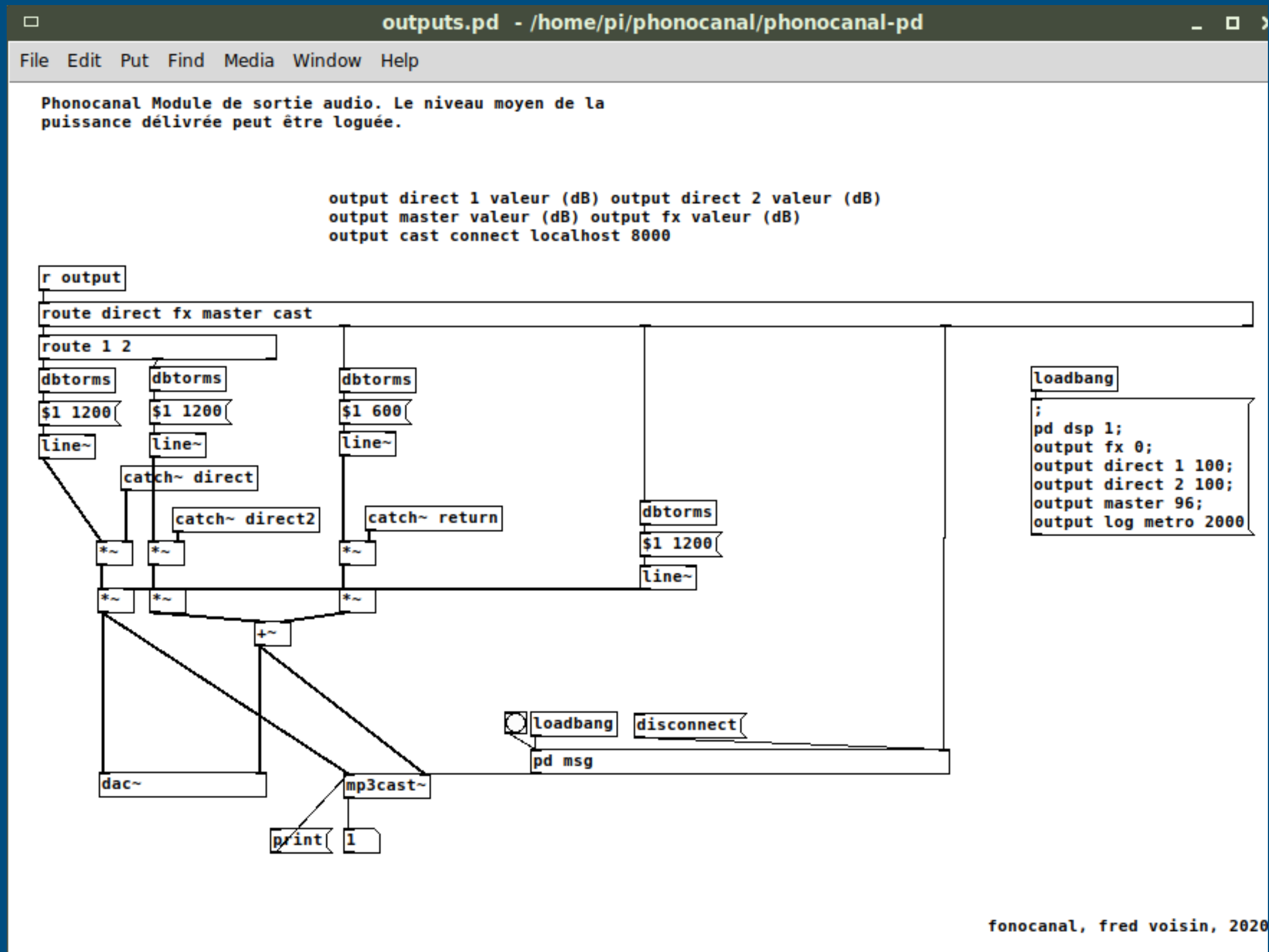
Exemple de synthétiseur : grillons



[Patch grillon.pd](#)

Ce programme effectue la synthèse de sons de grillon selon la structure présentée précédemment. Plusieurs instances de ce programme peuvent être exécutées en même temps, avec des valeurs paramétriques individualisées mais qui partagent néanmoins les valeurs communes de la température ("TEMP") de l'humidité ("HUM") et de la pression ("PRESS") atmosphériques (cf. flèches) qui sont automatiquement mises à jour.

Diffusion



[Patch outputs.pd](#)

La diffusion sonore s'effectue en stéréo au moyen des deux haut-parleurs intégrés et permanents et, éventuellement, via le réseau Wifi "Fonocanal" auquel peuvent se connecter tablettes et smartphones via le site web local "fonocanal".

Les latences observées différant de plusieurs secondes d'un équipement à l'autre, la diffusion sonore crée autant d'échos spatialisés par les visiteurs eux-mêmes.

Streaming audio

```
tried /home/pi/phonocanal/phonocanal-pd/externals/mp3cast~.pd_linux and succeeded
verbose(4): mp3cast~: mp3 streamer version 0.5, written by Yves Degoyon
warning: direct: multiply defined
warning: direct: multiply defined
warning: direct: multiply defined
warning: direct: multiply defined
warning: direct: multiply defined
warning: direct: multiply defined
warning: direct: multiply defined
warning: direct: multiply defined
warning: direct: multiply defined
warning: direct: multiply defined
warning: direct: multiply defined
warning: direct: multiply defined
warning: direct: multiply defined
warning: direct: multiply defined
warning: direct: multiply defined
saved to: /home/pi/phonocanal/phonocanal-pd/outputs.pd
verbose(4): mp3cast~: mp3 streamer version 0.5, written by Yves Degoyon
LAME mp3 settings:
  output sample rate: 44100 Hz
  bitrate: 224 kbit/s
  mode: joint stereo
  quality: 5
  mp3 chunk size: 0
server type is IceCast
verbose(4): mp3cast~: mp3 streamer version 0.5, written by Yves Degoyon
LAME mp3 settings:
  output sample rate: 44100 Hz
  bitrate: 224 kbit/s
  mode: joint stereo
  quality: 5
  mp3 chunk size: 0
server type is IceCast
mp3cast~: isPublic set to yes
mp3cast~: url set to http://localhost
mp3cast~: setting mp3 stream to 44100Hz, 128kbit/s, mode 1, quality 4
mp3cast~: name set to mystream
mp3cast~: name set to fonocanal-live
mp3cast~: mountpoint set to live.mp3
mp3cast~: set server type to IceCast 2
mp3cast~ : setting password to XXXXXXXXXX
mp3cast~: connecting to port 8000
mp3cast~: logged in to localhost
verbose(4): mp3cast~ : using lame version : 3.99.5
mp3cast~ : lame initialization done. (1)
mp3cast~: initialising buffers

pi@phonocanal10A:~/phonocanal/phonocanal-pd $ saved to: /home/pi/phonocanal/phonocanal-pd/outputs
```

Connection et démarrage du flux de streaming

Le flux de streaming est assuré au format mp3 par le programme libre Icecast 2.

Des synthétiseurs sont automatiquement et arbitrairement chassés si les ressources du processeur sont insuffisantes.

Charge CPU

pi@phonocanal0A: ~/phonocanal/scripts

pi@phonocanal0A: ~/phonocanal/synthes

pi@phonocanal0A: ~/pho

top - 13:44:18 up 1 day, 20:50, 4 users, load average: 0.51, 0.46, 0.46
Tasks: 141 total, 2 running, 90 sleeping, 0 stopped, 0 zombie
%Cpu(s): 8.3 us, 2.7 sy, 0.0 ni, 88.8 id, 0.2 wa, 0.0 hi, 0.1 si, 0.0 st
KiB Mem : 948300 total, 351768 free, 96788 used, 499744 buff/cache
KiB Swap: 102396 total, 102396 free, 0 used. 742708 avail Mem

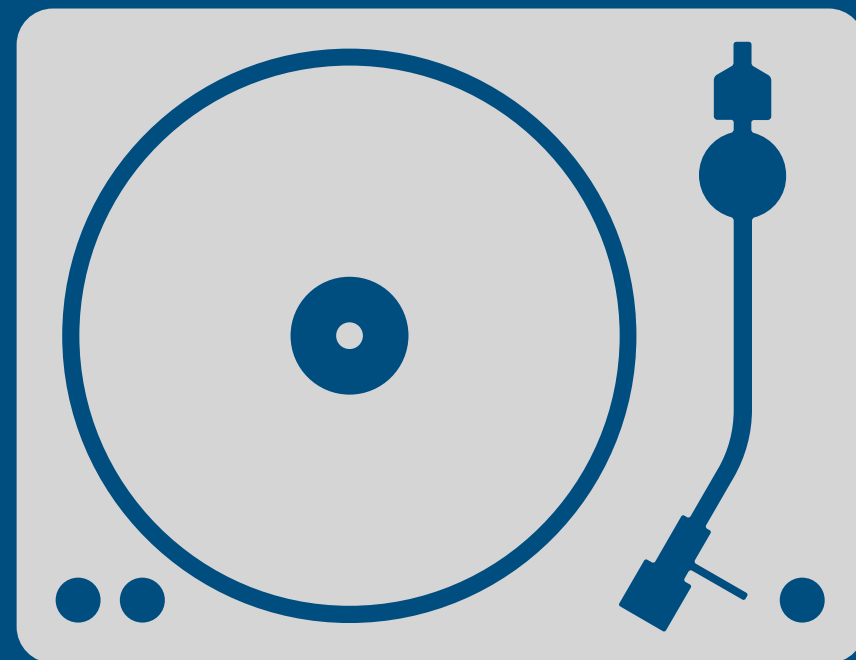
PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
20607	pi	20	0	38156	11036	4020	R	24.2	1.2	32:20.24	pd
27730	pi	20	0	5296	3268	2864	S	4.6	0.3	0:18.01	pd
27594	pi	20	0	5296	3308	2904	S	4.0	0.3	0:18.18	pd
28010	pi	20	0	5296	3404	2864	S	4.0	0.4	0:16.38	pd
28145	pi	20	0	5296	3380	2976	S	2.6	0.4	0:10.40	pd
28444	pi	20	0	8228	3400	2840	R	1.0	0.4	0:00.21	top
623	gkrellmd	20	0	14272	2396	2104	S	0.7	0.3	20:22.28	gkrellmd
105	root	20	0	35100	9704	9264	S	0.3	1.0	9:19.62	systemd-journal
660	icecast2	20	0	58656	8084	6860	S	0.3	0.9	1:45.96	icecast2
1	root	20	0	28168	6192	4940	S	0.0	0.7	0:07.47	systemd

Test de charge du processeur (commande 'top')

La liste des tâches exécutées par le processeur met ici en évidence 5 instances de l'environnement Puredata (cf. colonne "COMMAND" : pd) : celle consacrée à la diffusion locale et au streaming audio occupe 24,2% du CPU. Les 4 autres instances "pd" représentent 4 synthétiseurs jouant en même temps et occupent chacun 2,6 à 4,6% du CPU. Le jeu prend ici, au total, près de la moitié de la puissance dont dispose le processeur, streaming audio compris.

Lecteur de fichiers audio

Des productions sonores préalablement enregistrées au format .wav peuvent être jouées à l'aide de *patches* nommés lecteur1.pd, lecteur2.pd, etc. convoqués, commandés et diffusés au même titre que tout autre synthétiseur.



Exemples sonores

Les exemples sonores suivants (cf. liens internet) sont produits par le *fonocanal*. Ils illustrent quelques uns des procédés qu'il met en jeu selon les points de vue :

- de la production sonore, les synthétiseurs devant aussi économes que possible en terme de consommation d'énergie (CPU), condition inhérente à cette production ;
- de l'automation stochastique, déterminée par des calendriers, par différentes formes d'interaction et d'usages.

Ils doivent être envisagés, ici, comme de courts extraits de sessions beaucoup plus longues et variées, en relation à l'environnement des berges du canal de Bourgogne, à ses usagers, à la météo...

La variété des synthétiseurs et leurs modalités de jeu sont amenées à s'accroître au fur et à mesure des productions et des ateliers auxquels Fonocanal donne lieu, outre des œuvres "fixes, préfixées, spécialement réalisées.

Grenouilles et variantes

(01-grenouilles-variations.mp3, 1'36")

Trois mêmes synthétiseurs constitués de quelques oscillateurs imitent des batraciens : les sons semblent plus ou moins naturels, incongrus ou étranges selon l'attention que l'on y porte. Les choix des synthétiseurs, les imitations, les variations et leurs fréquences dépendent de la participation des visiteurs, de l'heure, des saisons...

Grenouilles en streaming

(02-grenouilles-stream.mp3, 1'50")

Les mêmes synthétiseurs sont diffusés par le Fonocanal au loin tandis que s'y superposent leurs échos déformés - plus proches - diffusés par les smartphones de deux visiteurs connectés au réseau wifi. Les visiteurs peuvent aussi choisir explicitement le style des variations sonore soit au moyen d'une page consacrée du site web local, soit sans connection wifi, selon leur position autour du *Fonocanal* telle que détectée par ses capteurs dissimulés.

Grillons nocturnes

(03-grillons-nocturnes.mp3, 2'02")

Deux mêmes "grillons" de synthèse jouent la nuit faisant émerger un jeu d'oscillations plus lentes. Un autre insecte de synthèse se joint à eux en faisant entendre une autre sonorité d'élytres. Ils se taisent à l'approche d'un visiteur dont la présence est détectée par les capteurs.

Grillons pentatoniques

(04-grillons-penta.mp3, 2'03")

Cinq "grillons" ou "cigales" de synthèse jouent chacun stochastiquement une note dans une gamme pentatonique. Leur intensité et leurs sonorités varient selon la température ambiante. Le fonocanal est entendu au loin tandis que leur écho est retransmis via deux smartphones, plus près. La variété des insectes de synthèse et leur nombre varient selon le temps...

Merloscillateur

(05-merloscillo.mp3, 1'19")

L'imitation du chant d'un Merle est réalisée à travers une transcription effectuée préalablement. Celle-ci est jouée par un simple oscillateur, avec de légères transpositions à l'unisson, à la seconde, à la tierce ou quarte. Une légère modulation de fréquence varie selon phrases. En situation, l'imitation n'est pas prévue d'être confrontée au chant original comme ici, à titre d'illustration. De multiples chants d'oiseaux locaux font l'objet d'imitations...

Vièle à vent

(06-viele-a-vent.mp3, 1'11")

Des résonateurs de synthèse, accordés selon la tradition locale d'accord des vièles à roue (à 5 cordes), sont mises en résonance par un vent de synthèse dont la force et les variations sont asservies aux conditions météorologiques.

Horloge parlante

(07-horloge-parlante.mp3, 3'09")

Le contrôle stochastique du tick-tack d'une horloge synthétisé au moyen de deux ou trois oscillateurs et de filtres crée tantôt des variations rythmiques, tantôt des variations qui évoquent des voyelles à la manière d'un vocoder vocal. La pulsation de cette horloge peut accompagner, pourquoi pas, une activité physique en plein air dont le tempo peut de varier selon l'heure, la saison, mais aussi selon les indications de visiteurs à travers le site web local de Fonocanal... Des interactions très simples, physiques ou par le site web, sont associées à différentes variations.

Droïde

(08-droide.mp3, 1'31")

Les plus simples procédés de synthèse sonore, articulés de manière aléatoire en même temps que par un, deux ou trois contrôles triviaux proposés aux jeunes et moins jeunes visiteurs peuvent revêtir un caractère ludique capable d'évoquer un univers populaire (par exemple *Star Wars*) ou l'histoire de la musique électronique (par exemple Ligeti, Nono...). Ce minimalisme qui convoque hasard, imitation et articulation est aussi emblématique de Fonocanal. Un jeu d'échantillons sonores, extraits de jeux électroniques ou d'animations peut aussi remplacer les oscillateurs...

Des récits relatifs au pays du Tonnerrois sont également prévus d'être diffusés soit automatiquement à certaines heures précises, au même titre que les synthétiseurs, soit selon le choix des visiteurs via un menu dédié sur le site web de Fonocanal.

Les exemples suivants, collectés entre 1970 et 1990, sont extraits de fonds mis gracieusement à notre disposition par la Maison du Patrimoine Oral de Bourgogne, les plus pertinents relativement aux fonds de l'INA ou de la Bibliothèque Nationale. Leur qualité a parfois nécessité un travail de restauration (cf. exemple "Fête-Dieu 1900").

Traditions culinaires

(09-MPOB-1.mp3, 1'52")

Traditions culinaires racontées par une femme élevée dans des fermes du Tonnerrois vers les années 1930 (extrait, fond radiophonique Jean-Pierre Marry).

Travail des champs

(10-MPOB-2.mp3, 1'34")

Un agriculteur du Tonnerrois né en 1921 raconte le travail des champs et des vignes dans les années 1930 (extrait, fond radiophonique Jean-Pierre Marry).

Fête-Dieu 1900

(11-MPOB-3.mp3, 1'42")

Un habitant de Sambourg né en 1888 raconte le déroulement des rogations de la Fête Dieu vers les années 1900 dans le Tonnerrois (extrait, fond "Les Bégas de Neuvy").

Résonances anciennes

(12-resonances-anciennes.mp3, 1'39")

La même histoire racontée à travers les résonances de la vièle et un léger écho donne le sentiment de venir de loin (en plus de masquer les imperfections de l'enregistrement)...

Ressources techniques

Unité centrale embarquée ARM	https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/?resellerType=home
Amplificateur audio	https://uk.pi-supply.com/products/iqaudio-pi-digi-amp-raspberry-pi-class-d-amplifier
Sonde numérique de température, d'hygrométrie et de pression atmosphériques I2C	https://www.bosch-sensortec.com/products/environmental-sensors/humidity-sensors-bme280/
Buttons électromécaniques	https://www.cl.cam.ac.uk/projects/raspberrypi/tutorials/robot/buttons_and_switches/
Réseau wifi	https://howtoraspberrypi.com/connect-wifi-raspberry-pi-3-others/
Site web HTML5	https://httpd.apache.org/ , https://www.php.net/
Système d'exploitation informatique Raspbian	https://www.raspbian.org/
Environnement de programmation multimedia	https://puredata.info/
Site web du Fonocanal	https://fonocanal.net



Le Fonocanal a été conçu avec le soutien
du dispositif DICRéAM d'aide à la création

