puredata, raspberry-pi, carnabal, audio, installation, em

Installation d'ambiance sonore pour le Carnabal des Monstres 2026

Installation sonore stéreo de sons d'ambiance. Les sons ont été réalisé dans un atelier de bruitages pour le Carnabal des Monstres 2026

Avec Pure data, Raspberry Pi (abrégé en rpi ci-dessous)

Installation matérielle

Préparation du rpi

Insérer la carte micro-SD dans l'ordinateur puis lancer le logiciel dédié pour installer l'OS

```
rpi-imager &
```

OS choisi : RPI OS Debian Trixie, compte utilisateur : xor (même mot de passe que d'habitude), accès SSH autorisé, nom de l'hôte : carnabal.local
nb : ça n'est peut-être pas idéal, la version lite de raspberrypi OS aurait pu suffire... (des notes sur une install headless complète ici)

Insérer la carte micro-SD dans le rpi, brancher alim, HDMI, clavier et souris et c'est parti

D'abord installation de quelques paquets par le terminal

```
sudo apt update
sudo apt install terminator micro xclip puredata-core puredata
```

(puredata est installé en version 0.55.2)

Démarrage avec interface graphique

Dans cette étape, on vérifie que le patch fonctionne correctement en ALSA...
Sur ces modèles de rpi, il y a 3 sorties sonores possibles : 2 HDMI et 1 sortie analogique minijack. Pour que le patch fonctionne dans pure data, il faut régler la carte son adéquate : sortie sonore analogique sur mini-jack, pour cela :
Dans le menu «media», choisir paramètres audio
Décocher «périphérique d'entrée» (on ne s'en sert pas)
Cocher «périphérique de sortie» et choisir «bcm2835 Headphones (hardware)»
Ça devrait fonctionner corectement

Démarrage sans interface graphique

Dans cette étape, on vérifie que le patch peut démarrer en ligne de commande par un script.
Tout d'abord il faut trouver le bon chiffre pour identifier la sortie audio

```
aplay -l               # donne des infos sur le type de cartes
pd -alsa -listdev      # renvoie les numéros d'identification
```

On peut lire

```
audio input devices:
0. vc4-hdmi-0 (hardware)
1. vc4-hdmi-0 (plug-in)
2. vc4-hdmi-1 (hardware)
3. vc4-hdmi-1 (plug-in)
4. bcm2835 Headphones (hardware)
5. bcm2835 Headphones (plug-in)
audio output devices:
0. vc4-hdmi-0 (hardware)
1. vc4-hdmi-0 (plug-in)
```

```
2. vc4-hdmi-1 (hardware)
3. vc4-hdmi-1 (plug-in)
4. bcm2835 Headphones (hardware)
5. bcm2835 Headphones (plug-in)
API number 1
```

Différence entre hardware et plugin
Le (hardware) est l'accès direct ALSA à la carte (hw:0,0) : accès exclusif, pas de conversion automatique, et il n'accepte que le format natif du matériel.
Le (plugin) passe par la couche plughw d'ALSA, qui convertit automatiquement le taux d'échantillonnage, le format et le nombre de canaux si besoin.

→ On utilisera la sortie 4, on peut démarrer le patch en ligne de commande par

```
pd -nogui -alsa -noadc -audiooutdev 4 -r 44100 -audiobuf 200 -open /home/xor/carnabal/mix-carnabal-2.pd
```

Cette commande lance pure data sans priorisation temps réel (option -rt), le «gros» buffer audio de 200 ms fonctionne bien dans ce cas d'un patch génératif, mais si on voulait utiliser cette option, il faudrait donner des droits particulier au groupe audio

```
sudo nano /etc/security/limits.d/audio.conf
```

Et y ajouter

```
@audio - rtprio 95
@audio - memlock unlimited
```

Configurer le lancement automatique au démarrage

On crée un «service» en utilisant systemd

```
sudo touch /etc/systemd/system/carnabal.service
sudo micro /etc/systemd/system/carnabal.service    # éditer le fichier avec micro (copier-coller possible par l'install de xclip)
```

Y mettre la configuration ci-dessous, pure data est démarré avec la priorisation temps réel qui le fait passer devant les autres processus (lignes LimitRTPRIO et LimitMEMLOCK, et l'option -rt dans la ligne ExecStart)

```
[Unit]
Description=Patch Pure Data Carnabal des Monstres 2026
After=sound.target

[Service]
User=xor
Group=audio
LimitRTPRIO=95
LimitMEMLOCK=infinity
ExecStart=/usr/bin/pd -rt -nogui -alsa -noadc -audiooutdev 4 -r 44100 -audiobuf 200 -open /home/xor/carnabal/mix-carnabal-2.pd
Restart=always
RestartSec=5

[Install]
WantedBy=multi-user.target
```

/!\ attention au nom du *User*

Tester le script automatique de lancement

```
sudo systemctl daemon-reload
sudo systemctl enable --now carnabal.service
```

On peut lire le journal du service (et les erreurs éventuelles) avec :

```
journalctl -u carnabal -f
```

S'il faut relancer

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

Test grandeur nature
Éteindre le pi, débrancher HDMI, clavier, souris et relancer le Pi
→ ok, ça fonctionne :)

Régler le volume sonore
Dans un terminal, avec alsamixer, on fixe le niveau de sortie de la carte son BCM2835, puis on enregistre le réglage

```
alsamixer
sudo alsactl store
```

Passer le rpi en read-only
Cela permet de démarrer / éteindre en le branchant/débranchant

```
sudo raspi-config
```

Choisir «performance options» puis «Overlay File System» : activer l'OFS et valider que la partition passe en lecture seule

Stopper le démarrage automatique

Remettre le rpi en lecture-écriture

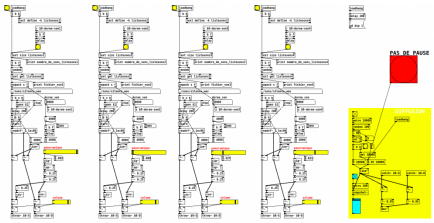
Comment désactiver le démarrage automatique du service ?
Pour désactiver le lancement automatique au démarrage et arrêter le script immédiatement (il reste quand même disponible pour être relancé) :

```
sudo systemctl disable --now carnabal.service
```

Pour connaître l'état du service :

```
systemctl is-enabled carnabal   # enabled / disabled
systemctl status carnabal       # actif ou non
```

Patch et code utile



mix-carnabal-2.pd

Dans les grandes lignes : 4 pistes de sons, dont on peut régler le volume et la panoramisation indépendamment. Sur chaque piste un son est lu au hasard, puis un autre, certains sons sont des silence pour ajouter des pauses dans le montage. Chaque liste de sons est d'un type différent. Un silence global se met en place de temps en temps (décidé par le hasard). Beaucoup de hasard en somme!

Notes et améliorations :

- les objets [random] sont pseudo-aléatoires… Les séquences sont toujours les mêmes à chaque redémarrage!
- une pause de silence est ajoutée pour l'expo (avec des durées plus longues que celles du patch), le gros bouton rouge sert à l'annuler pour les tests

Aussi un script python qui fait des listes de fichiers (.wav ou .WAV), avec leur durée en millisecondes, utilisés par les objets [text define -k listesons]

liste_sons_avec_duree.py (cliquer pour afficher le code)

liste_sons_avec_duree.py

```
#!/usr/bin/env python3
"""
Script sans dépendances externes (bibliothèque standard uniquement).
Nécessite que la commande `sox` (ou `soxi`) soit installée sur le système.

Fonctionnement :
- Liste tous les fichiers .wav / .WAV du répertoire courant, triés par ordre alphabétique.
```

```
- Pour chacun, récupère la durée en millisecondes via sox.
- Écrit dans un fichier texte, une ligne par fichier :
      {chemin/nom_de_fichier_complet} {durée_en_millisecondes};

Utilisation :
- à utiliser avec un patch pure data et l'objet [text]
- placer ce script dans le répertoire du patch pure data
- indiquer le chemin dans la variable chemin ci-dessous
- copier le contenu de la liste créée dans l'objet [text define -k liste]
"""

import os
import subprocess
import sys

# Chemin du dossier à traiter (à adapter si besoin)
chemin = "./sons"

# Nom du fichier texte de sortie
fichier_sortie = "listesons.txt"


def get_duree_ms(chemin_fichier):
    """
    Récupère la durée d'un fichier wav en millisecondes en utilisant sox (soxi).
    Retourne un int (ms) ou None en cas d'erreur.
    """
    try:
        # soxi -D renvoie la durée en secondes (float) sur stdout
        resultat = subprocess.run(
            ["soxi", "-D", chemin_fichier],
            capture_output=True,
            text=True,
            check=True,
        )
        duree_secondes = float(resultat.stdout.strip())
        return round(duree_secondes * 1000)
    except (subprocess.CalledProcessError, FileNotFoundError, ValueError) as e:
        print(f"Erreur lors du traitement de {chemin_fichier}: {e}", file=sys.stderr)
        return None


def main():
    # Liste des fichiers .wav / .WAV dans le répertoire "chemin"
    fichiers = [
        f for f in os.listdir(chemin)
        if f.lower().endswith(".wav") and os.path.isfile(os.path.join(chemin, f))
    ]

    # Tri par ordre alphabétique
    fichiers.sort()

    with open(fichier_sortie, "w", encoding="utf-8") as sortie:
        for nom_fichier in fichiers:
            chemin_complet = os.path.join(chemin, nom_fichier)
            duree_ms = get_duree_ms(chemin_complet)
            if duree_ms is not None:
                sortie.write(f"{chemin_complet} {duree_ms};\n")

    print(f"Terminé. Résultats écrits dans {fichier_sortie}")

if __name__ == "__main__":
    main()
```

Article extrait de : http://www.lesporteslogiques.net/wiki/ - **WIKI Les Portes Logiques**
Adresse : **http://www.lesporteslogiques.net/wiki/projets/carnabal_2026/ambiance_sonore_carnabal**
Article mis à jour: **2026/10/19 12:20**