

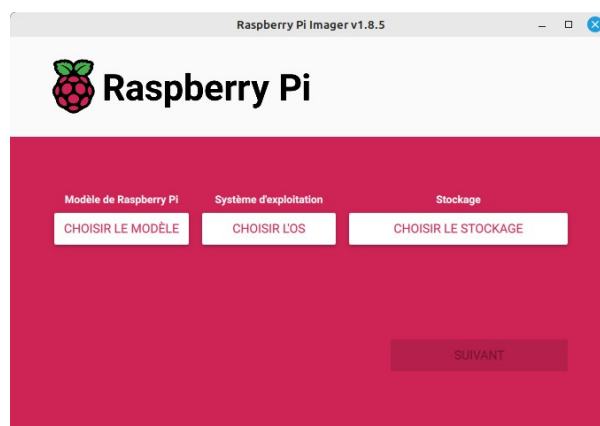
Installation OS sur Raspberry pi zéro 2-W

Table des matières

Installation OS sur Raspberry pi zéro 2-W.....	1
1. Installer Pi Imager.....	1
2. Connexion en SSH.....	3
3. Mise à jour de l'OS &.....	4
Vérifier la version de l'OS.....	4
4. Installation de git.....	4
5. Installation de wiringPi.....	5
6. Bus I2C et SPI.....	5
Installation des outils I2C.....	6
7. Utiliser la camera.....	6
Vérifier la présence de la caméra (en tant que root).....	6
Prendre une photo.....	7
Transférer la photo sur le PC.....	7
8. Incruster du texte dans une image.....	8
9. Installer rpitx.....	9
Test de l'installation.....	9
Test de la transmission SSTV.....	9
10. Test de la compilation en C++.....	10
11. Communication file IPC.....	10
12. Monter un système de fichiers en mémoire RAM (ramfs).....	11
13. Lancer le script start_ballon.sh au démarrage.....	11

1. Installer Pi Imager

Sur un PC



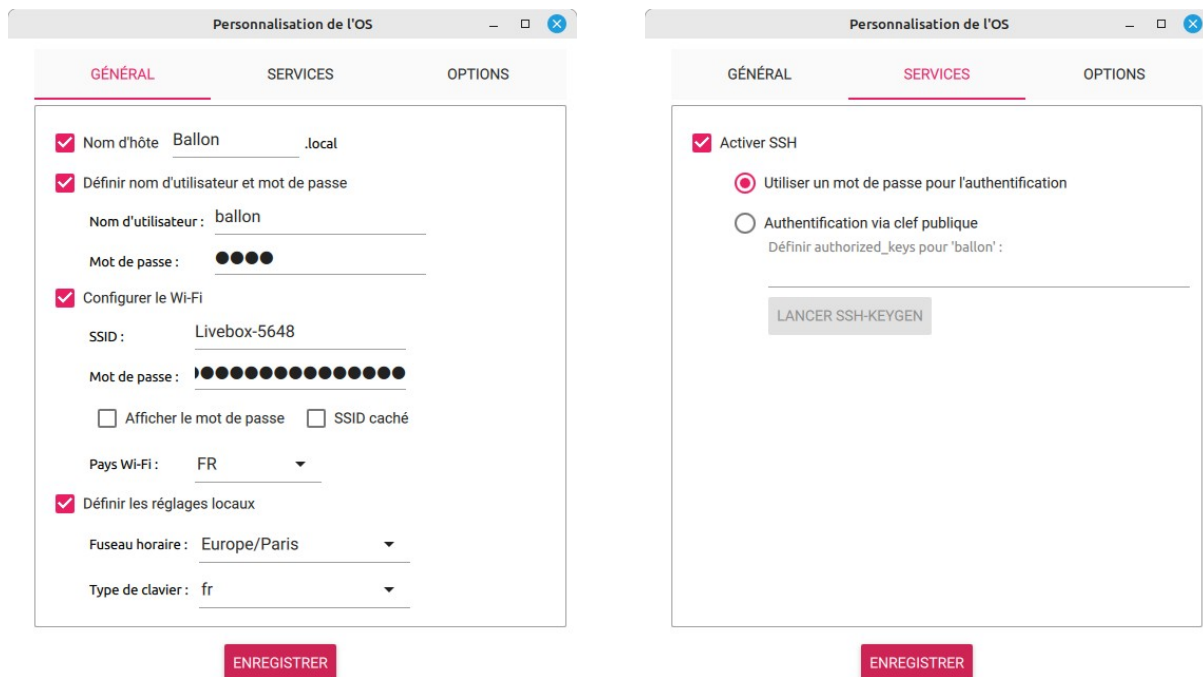
Configurer

1. le Modèle de Raspberry Pi,

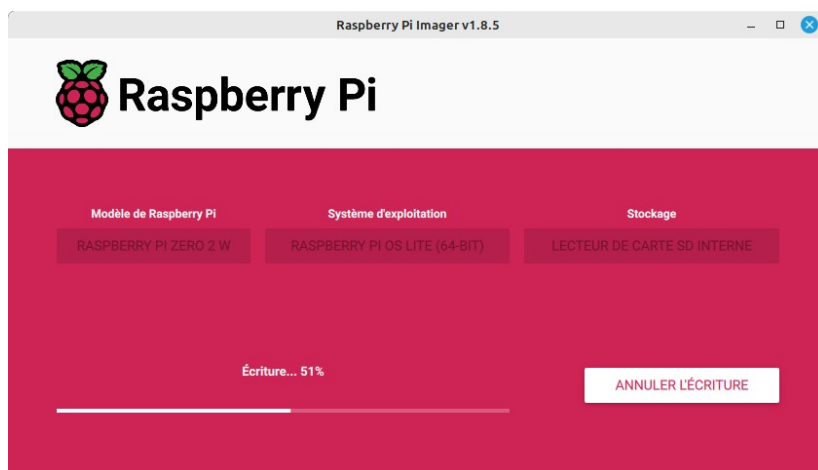
2. le système d'exploitation **Raspberry Pi OS Lite (64-bit)**
3. le stockage choisir le lecteur de carte SD



Utiliser la personnalisation de l'OS



puis lancer l'écriture



2. Connexion en SSH

Installer la carte SD, mettre sous tension, retrouver l'adresse IP sur la box

```
philippe@philippe-PC:~$ ssh ballon@192.168.1.47
The authenticity of host '192.168.1.47 (192.168.1.47)' can't be established.
ED25519 key fingerprint is
SHA256:SqFEsR5mTSXqJ9eqCS7cuBB66H5YHgQ3bqUHwg6pJ8.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? Yes
Warning: Permanently added '192.168.1.47' (ED25519) to the list of known hosts.
ballon@192.168.1.47's password:
```

Linux Ballon 6.12.25+rpt-rpi-v8 #1 SMP PREEMPT Debian 1:6.12.25-1+rpt1 (2025-04-30) aarch64

The programs included with the Debian GNU/Linux system are free software; the exact distribution terms for each program are described in the individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent permitted by applicable law.

3. Mise à jour de l'OS &

- `apt update` → rafraîchit la liste des paquets.
- `apt upgrade` → met à jour les paquets **sans suppression**.
- `apt full-upgrade` → met à jour **tout le système**, quitte à installer/supprimer des paquets.

```
apt update
apt full-upgrade
```

Fin Aout 2025 un problème est apparu, depuis les dernières mises à jour de Raspberry Pi OS Sur un **Raspberry Pi**, après un `apt upgrade` ou `full-upgrade`, la commande `libcamera-hello --list-cameras` n'existe plus, elle est remplacée par `rpigam-hello --list-camera`

Vérifier la version de l'OS

```
ballon@ballon:~$ cat /etc/os-release
PRETTY_NAME="Debian GNU/Linux 12 (bookworm)"
NAME="Debian GNU/Linux"
VERSION_ID="12"
VERSION="12 (bookworm)"
VERSION_CODENAME=bookworm
ID=debian
HOME_URL="https://www.debian.org/"
SUPPORT_URL="https://www.debian.org/support"
BUG_REPORT_URL="https://bugs.debian.org/"
```

4. Installation de git

```
root@ballon:/home/ballon# apt install git
root@ballon:/home/ballon# git --version
git version 2.39.5
```

5. Installation de wiringPi

```
git clone https://github.com/WiringPi/WiringPi.git
cd WiringPi/
~/WiringPi $ ./build debian
~/WiringPi $ mv debian-template/wiringpi_3.16_arm64.deb .
~/WiringPi $ sudo apt install ./wiringpi_3.16_arm64.deb
```

test de l'installation

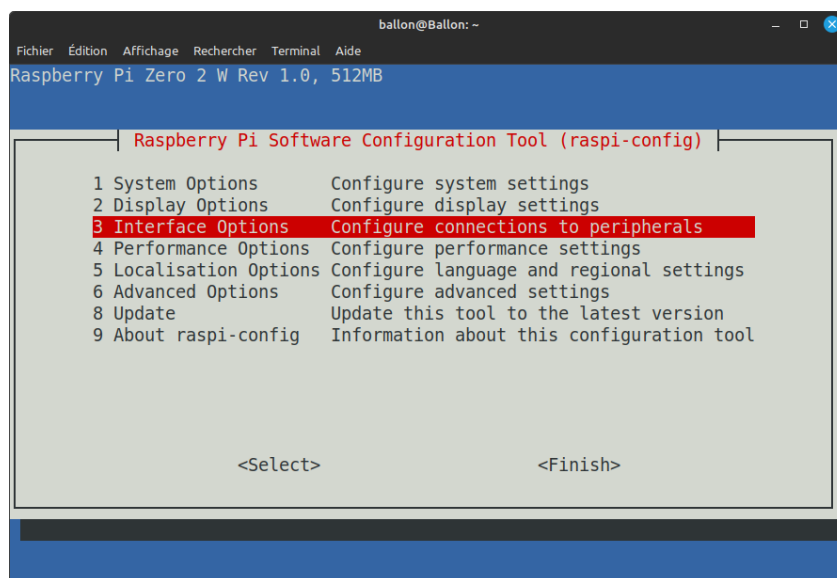
```
ballon@ballon:~/WiringPi $ gpio -v
gpio version: 3.16
Copyright (c) 2012-2025 Gordon Henderson and contributors
This is free software with ABSOLUTELY NO WARRANTY.
For details type: gpio -warranty

Hardware details:
Type: Pi Zero2-W, Revision: 00, Memory: 512MB, Maker: Sony UK

System details:
* Device tree present.
  Model: Raspberry Pi Zero 2 W Rev 1.0
* Supports full user-level GPIO access via memory.
* Supports basic user-level GPIO access via /dev/gpiomem.
* Supports basic user-level GPIO access via /dev/gpiochip (slow).
```

6. Bus I2C et SPI

Activer les bus i2c et spi avec raspi-config.



Installation des outils I2C

```
apt install i2c-tools
root@Ballon:/home/ballon# i2cdetect -y 1
 0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:  -- -- -- -- -- -- -- -- -- -- -- -- -- --
10:  -- -- -- -- -- -- -- -- -- -- -- -- -- --
20:  -- -- -- -- -- -- -- -- -- -- -- -- -- --
30:  -- -- -- -- -- -- -- -- -- -- -- -- -- --
40:  -- -- -- -- -- 48 -- -- -- -- -- -- -- --
50:  -- -- -- -- -- -- -- -- -- -- -- -- -- --
60:  -- -- -- -- -- 68 69 -- -- -- -- -- -- --
70:  -- -- -- -- -- 77
```

7. Utiliser la camera

picam est la bibliothèque qui permet d'accéder et de contrôler la caméra. Elle offre une API C++ aux applications qui permet la configuration de la caméra et de la récupération des images. **Elle est installée par défaut sur PI OS version Bookworm.**

Vérifier la présence de la caméra (en tant que root)

```
root@Ballon:/home/ballon# picam-hello --list-camera

Available cameras

-----

0 : imx708 [4608x2592 10-bit RGGB] (/base/soc/i2c0mux/i2c@1/imx708@1a)
  Modes: 'SRGGB10_CSI2P' : 1536x864 [120.13 fps - (768, 432)/3072x1728 crop]
        2304x1296 [56.03 fps - (0, 0)/4608x2592 crop]
        4608x2592 [14.35 fps - (0, 0)/4608x2592 crop]
```



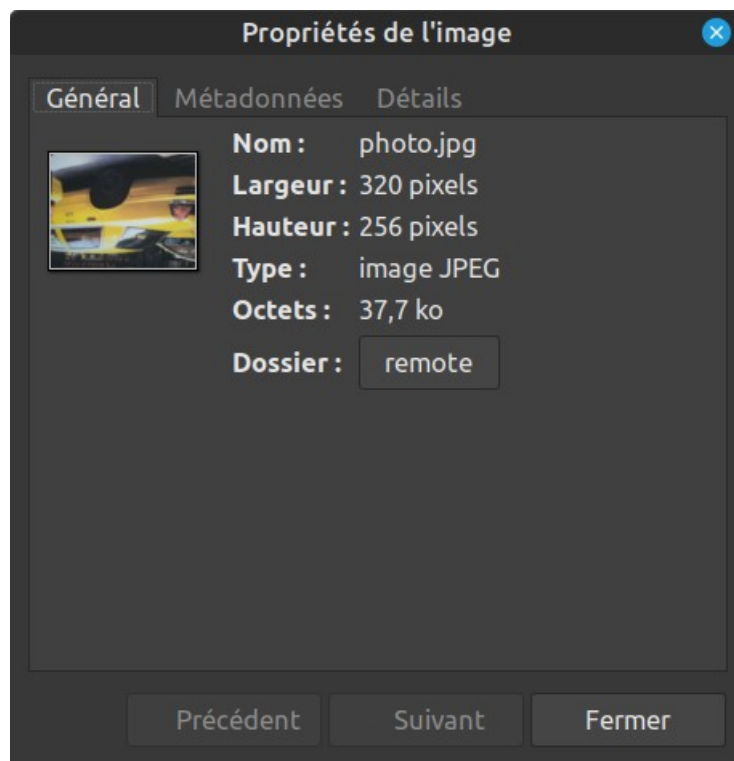
Prendre une photo

```
rpicam-still --width 320 --height 256 -o photo.jpg
```

Transférer la photo sur le PC

```
sshfs ballon@192.168.1.47:/home/ballon remote/
```

Cette commande **permet d'accéder en local (dans remote/) au contenu du répertoire /home/ballon de la machine 192.168.1.47**, comme s'il faisait partie du système de fichiers local.



La suite `sudo apt install libcamera-dev`

```
sudo apt install libcamera-dev
cd /usr/include/libcamera
cd libcamera
sudo cp -r * ../
ballon@Ballon:/usr/include/libcamera $ ls
base          control_ids.h framebuffer_allocator.h libcamera  pixel_format.h transform.h
camera.h      controls.h   framebuffer.h         libcamera.h property_ids.h version.h
camera_manager.h fence.h    geometry.h           logging.h   request.h
color_space.h formats.h   ipa
```

8. Incruster du texte dans une image

```
apt install imagemagick
```

vérification de la version

```
root@Ballon:/home/ballon# convert --version
```

```
Version: ImageMagick 6.9.11-60 Q16 aarch64 2021-01-25 https://imagemagick.org
```

```
Copyright: (C) 1999-2021 ImageMagick Studio LLC
```

```
License: https://imagemagick.org/script/license.php
```

```
Features: Cipher DPC Modules OpenMP(4.5)
```

```
Delegates (built-in): bzlib djvu fftw fontconfig freetype heic jbig jng jp2 jpeg lcms lqr ltdl lzma  
openexr pangocairo png tiff webp wmf x xml zlib
```

commande pour incruster du texte

```
convert -pointsize 20 -draw "text 10,20 'F4KMN'" photo.jpg photo_texte.jpg
```


9. Installer rpitx

rpitx est un logiciel open-source qui permet à un Raspberry Pi de fonctionner comme un émetteur radio. Le nom "rpitx" est dérivé de "Raspberry Pi" et "transmitter" (émetteur en anglais). Il permet au Raspberry Pi de générer des signaux RF (Radio Fréquence) sur différentes gammes de fréquences.

```
git clone https://github.com/F5OEO/rpitx
cd rpitx
./install.sh
```

A la fin de l'installation un message demande une autorisation pour modifier le fichier /boot/firmware/config.txt

```
In order to run properly, rpitx need to modify /boot/config.txt. Are you sure (y/n) y
Set GPU to 250Mhz in order to be stable
Raspbian 12 detected using /boot/firmware/config.txt
Installation completed !
```

Si on édite ce fichier on retrouve à la fin :

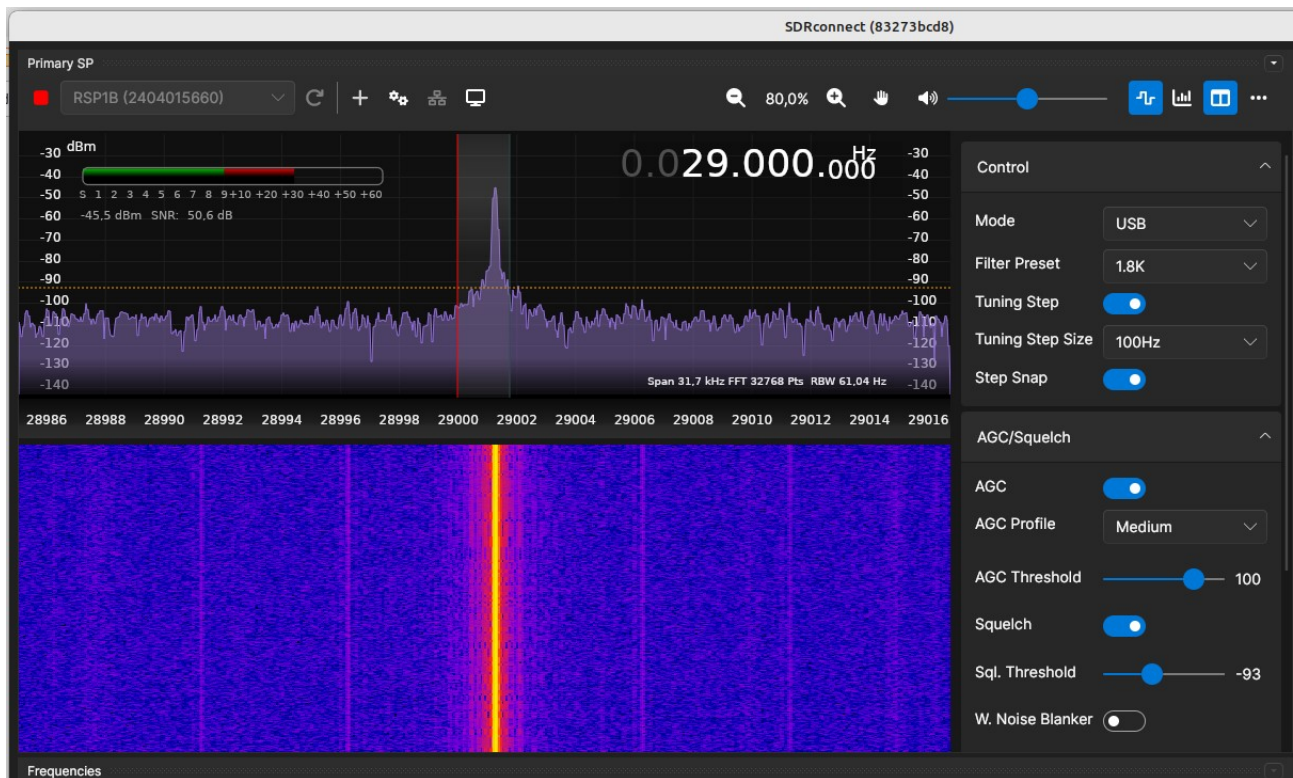
```
[all]
gpu_freq=250
force_turbo=1
```

Définit la fréquence du GPU (processeur graphique VideoCore). Ici, elle est forcée à **250 MHz**. Cela force le CPU et le GPU à tourner **toujours à la fréquence maximale**.

Test de l'installation

Le premier test est d'émettre une porteuse afin de vérifier avec une clé SDR et un récepteur logiciel la bonne réception du signal.

```
root@Ballon:/home/ballon/rpitx# ./tune -f 29001200
```



Comme on peut le voir sur la capture d'écran on obtient un trait pour la fréquence 29,001200 Mhz

Test de la transmission SSTV

/home/ballon/rpitx/pisstv /home/ballon/photo_date.rgb 29000000

La transmission de l'image se fait au format martin 1 sur une durée de **1 minute et 57 secondes**.

10. Test de la compilation en C++

Copier le répertoire **stop** qui se trouve sur mon dépôt github.

```
root@Ballon:/home/ballon# cd stop/  
root@Ballon:/home/ballon/stop# make all  
gcc -Wall -Wextra -Werror -O2 -c stop.c -o stop.o  
gcc -Wall -Wextra -Werror -O2 -o stop stop.o -lwiringPi
```

Tester l'exécution du programme

```
root@Ballon:/home/ballon/stop# ./stop &  
[1] 942  
root@Ballon:/home/ballon/stop#
```

Appuyer sur le bouton poussoir

```
root@Ballon:/home/ballon/stop# Détection d'un front descendant sur BP GPIO 25  
Broadcast message from root@Ballon on pts/1 (Thu 2025-07-03 11:25:03 CEST):  
The system will power off now!  
Connection to 192.168.1.47 closed by remote host.  
Connection to 192.168.1.47 closed.  
philippe@philippe-PC:~$
```

11. Communication file IPC

Copier le répertoire **lora-files** qui se trouve sur mon dépôt github.

Puis compiler avec la commande make all

Vérifier la présence des 2 files avec la commande ipcs -q

```
root@ballon:/home/ballon/lora_files# ipcs -q  
  
----- Message Queues -----  
  
key      msqid    owner    perms    used-bytes  messages  
0x0000162e 0      root     666      1320        5  
0x0000162f 1      root     666      0           0
```

12. Monter un système de fichiers en mémoire RAM (ramfs)

Montage automatique au démarrage (via `/etc/fstab`)

1 Ouvrir le fichier `/etc/fstab`

2 Ajouter cette ligne à la fin

```
tmpfs /ramfs tmpfs defaults,size=100M 0 0
```

3 Crée le dossier si ce n'est pas fait :

```
sudo mkdir /ramfs
```

après un redémarrage vérifier avec `mount`

```
ballon@ballon:~$ mount | grep /ramfs  
tmpfs on /ramfs type tmpfs (rw,relatime,size=102400k)
```

Tout contenu dans `/ramfs` est **perdu au redémarrage**.

13. Lancer le script `start_ballon.sh` au démarrage

En tant que root

`crontab -e`

ajouter la ligne suivante

```
@reboot /home/ballon/start_ballon.sh > /home/ballon/log/start_ballon.log 2>&1
```