APN IOT

- Accès distant à un Raspberry Pi via Modem LTE (IoT)
- Mise en place de l'APN IoT
- How to Install Ubuntu in Raspberry PI

Accès distant à un Raspberry Pi via Modem LTE (IoT)

Accès distant à un Raspberry Pi via Modem LTE (IoT)

---

1. Module LTE utilisé : SIM8230G-M2 RedCap

Le modem utilisé dans ce projet est le module **SIM8230G-M2**, un module 5G RedCap basé sur la norme 3GPP Release 17.

Compatibilité avec d'autres modems

Le script reste compatible avec la majorité des modems LTE/5G utilisant des commandes AT standards.

Installation du module

Documentation officielle : https://www.waveshare.com/wiki/SIM8230G-M2

---

2. Objectif

Ce projet vise à établir un accès distant fiable à un Raspberry Pi connecté via modem LTE, en contournant les contraintes NAT habituelles des réseaux mobiles. Les points couverts sont :

- connexion LTE via APN dédié
- tunnel SSH inverse vers un serveur intermédiaire
- accès distant malgré le NAT opérateur
- reconnexion automatique en cas de perte réseau

3. Contexte réseau

Dans un réseau LTE, le Raspberry Pi est placé derrière un **NAT opérateur** : le modem obtient une adresse IP côté réseau, mais le Raspberry est en réalité derrière une adresse privée non accessible depuis l'extérieur.

Conséquences directes :

- impossible de se connecter directement en SSH depuis l'extérieur
- impossible de joindre le Raspberry depuis Internet

Solution retenue : reverse SSH (tunnel SSH inverse).

---

4. Accès via serveur de logs

L'accès au Raspberry ne se fait **pas directement**, mais via un serveur intermédiaire appelé serveur de logs. Ce serveur est accessible à la fois depuis le Raspberry (via le réseau LTE) et depuis les postes d'administration.

Le principe est le suivant :

- le Raspberry initie lui-même une connexion SSH vers le serveur de logs
- il expose un port local redirigé sur le serveur
- l'administrateur se connecte ensuite à ce port depuis le serveur

```
ssh localhost -p 2222
```

Cette approche présente plusieurs avantages :

- tous les membres de l'équipe ont accès au serveur de logs
- les connexions directes vers les équipements sont évitées
- l'accès aux équipements IoT est centralisé
- la gestion des accès SSH est simplifiée

> **autossh** est utilisé pour maintenir le tunnel actif : il relance automatiquement la connexion SSH en cas de déconnexion, sans intervention manuelle.

---

5. Préparation du Raspberry Pi

Ces étapes sont à réaliser en amont, directement sur le Raspberry Pi, avant de déployer le script.

Création d'un utilisateur dédié

```
# Création d'un utilisateur dédié
sudo adduser iot

# Ajout aux droits sudo
sudo usermod -aG sudo iot
```

Génération de la clé SSH

```
ssh-keygen -t ed25519 -f ~/.ssh/modem_tunnel -N ""
```

---

6. Script complet

```
#!/bin/bash

# Arrêt du script en cas d'erreur
set -euo pipefail

# Port série du modem
PORT="/dev/ttyUSB2"

# Nombre max d'essais modem
max_tries=100
try=0

# IP de test Internet
ping_ip="8.8.8.8"

# Interface réseau modem
iface="usb0"
```

```
# Nombre max d'essais réseau
max_ping_tries=100

# Paramètres SSH
SSH_HOST="192.108.119.79"
SSH_USER="procom"
SSH_KEY="/home/iot/.ssh/modem_tunnel"

# Demande une IP via DHCP
sudo dhclient -4 -v usb0 || true

# Ajout des IDs USB modem
bind_ids() {
  sudo sh -c 'echo "1e0e 9073" > /sys/bus/usb-serial/drivers/option1/new_id'
  sudo sh -c 'echo "1e0e 9071" > /sys/bus/usb-serial/drivers/option1/new_id'
  sudo sh -c 'echo "1e0e 9072" > /sys/bus/usb-serial/drivers/option1/new_id'
  sudo sh -c 'echo "1e0e 9078" > /sys/bus/usb-serial/drivers/option1/new_id'
  sudo sh -c 'echo "1e0e 907b" > /sys/bus/usb-serial/drivers/option1/new_id'
}

# Attente du modem
wait_for_port() {
  while [ ! -e "$PORT" ]; do
    try=$((try+1))

    # Reboot si modem absent trop longtemps
    if [ "$try" -ge "$max_tries" ]; then
      sudo reboot
    fi

    bind_ids
    sleep 5
  done
}

# Envoi d'une commande AT au modem
send_at() {
  local cmd="$1"
  echo -e "${cmd}\r" | sudo tee "$PORT" >/dev/null
}
```

```
wait_for_port

# Initialisation modem
send_at "AT"
send_at "AT+CFUN=0"
sleep 5
send_at "AT+CFUN=1"
sleep 5

# Configuration APN
send_at 'AT+CGDCONT=1,"IP","iot"'
sleep 2

# Sélection opérateur
send_at 'AT+COPS=1,2,"99970"'
sleep 5

# Test réseau
ping_ok() {
  ping -I "$iface" -c 1 -W 2 "$ping_ip" >/dev/null 2>&1
}

# Reset modem
restart_module() {
  send_at "AT+CFUN=0"
  sleep 10
  send_at "AT+CFUN=1"
  sleep 5
  send_at 'AT+CGDCONT=1,"IP","iot"'
  sleep 2
  send_at 'AT+COPS=1,2,"99970"'
  sleep 10
}

# Attente de l'attachement réseau
sleep 20

# Boucle de connexion
for i in $(seq 1 "$max_ping_tries"); do
```

```
  if ping_ok; then
    break
  fi
  restart_module
done

# Reboot si pas de réseau après toutes les tentatives
if ! ping_ok; then
  sudo reboot
fi

# Tunnel SSH permanent
while true; do

  autossh -M 0 \
    -i "$SSH_KEY" \
    -o StrictHostKeyChecking=no \
    -o ServerAliveInterval=30 \
    -o ServerAliveCountMax=3 \
    -N \
    -R 2222:localhost:22 \
    ${SSH_USER}@${SSH_HOST}

  # Si le tunnel tombe et que le réseau est absent → reboot
  if ! ping_ok; then
    sudo reboot
  fi

  sleep 10
done
```

---

7. Démarrage automatique

Le script est déclaré comme un service systemd pour être lancé automatiquement au démarrage du Raspberry Pi.

Créer le fichier `/etc/systemd/system/modem.service` :

```
[Unit]
Description=Modem LTE launcher
After=network-online.target

[Service]
ExecStart=/usr/local/bin/modem-launch.sh
Restart=always
RestartSec=10

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

Puis activer le service :

```
sudo systemctl daemon-reload
sudo systemctl enable modem.service
```

---

8. Paramètres à adapter

Paramètre	Valeur par défaut	Description
`PORT`	`/dev/ttyUSB2`	Port série du modem
`APN`	`iot`	APN déclaré dans la configuration réseau
Opérateur	`99970`	Code PLMN de l'opérateur
`iface`	`usb0`	Interface réseau du modem
`SSH_USER`	`procom`	Utilisateur SSH sur le serveur de logs
`SSH_HOST`	`192.108.119.79`	IP du serveur de logs
Port tunnel	`2222`	Port exposé sur le serveur pour rebondir
`SSH_KEY`	`/home/iot/.ssh/modem_tunnel`	Chemin vers la clé SSH privée

Mise en place de l'APN IoT

Mise en place de l'APN IoT

Architecture réseau, gestion du cœur LTE, séparation de trafic par VLAN et accès aux équipements

---

1. Introduction et objectifs

1.1 Contexte

Ce rapport documente la mise en place d'une infrastructure LTE complète reposant sur la solution logicielle **Amarisoft**. Ce type de déploiement, appelé réseau mobile privé, permet de disposer d'un cœur de réseau LTE maîtrisé de bout en bout, depuis l'antenne radio jusqu'à la sortie vers Internet ou un réseau local d'entreprise.

L'infrastructure s'appuie sur une antenne SDR (Software Defined Radio), pilotée entièrement par logiciel, ce qui offre une grande flexibilité de configuration. Les équipements utilisateurs (UE) — téléphones ou modems LTE — se connectent à ce réseau exactement comme sur un réseau opérateur classique, sans modification de leur part.

Un point central de ce projet est la création d'un **APN dédié** (Access Point Name), nommé APN IoT, qui permet d'identifier et d'isoler un type de trafic spécifique dès la connexion de l'UE. L'APN constitue le point d'entrée logique par lequel un équipement accède au réseau : en déclarant un APN distinct, on peut lui appliquer des règles de routage, d'adressage et de sortie réseau différentes de celles du trafic standard.

1.2 Objectifs

Les exigences définies pour ce projet sont les suivantes :

- Ajouter un APN IoT au cœur LTE Amarisoft, distinct de l'APN par défaut.
- Un équipement utilisateur (UE) utilisant cet APN obtient une adresse IP privée dans la plage `10.52.0.x` lors de son attachement au réseau LTE.
- Le trafic transite par le cœur Amarisoft (MME / EPC) avant d'être acheminé vers le réseau via un VLAN dédié.
- Les UE disposent d'un accès à Internet et au réseau local (`10.51.x.x`).

- Le mode sans NAT est privilégié côté Amarisoft, afin de conserver l'adresse IP réelle des UE sur l'ensemble du chemin réseau.
- L'accès aux UE depuis l'extérieur doit être possible malgré les contraintes de routage inhérentes à ce type d'architecture.

---

2. Architecture globale

2.1 Composants principaux

L'infrastructure LTE déployée repose sur quatre composants logiciels fournis par Amarisoft, chacun remplissant un rôle précis dans la chaîne de traitement du trafic mobile.

Composant	Rôle
eNB (eNodeB)	Gestion de la couche radio LTE via l'antenne SDR. Point d'entrée physique des connexions UE.
MME (Mobility Management Entity)	Authentification des UE et gestion des sessions IP. Composant central du cœur réseau (EPC).
IMS	Gestion des services voix sur LTE (VoLTE).
MBMSGW	Diffusion multicast. Composant optionnel, non activé dans la configuration de base.

2.2 Rôle de l'APN

Un APN (Access Point Name) est l'identifiant que l'UE présente au cœur LTE lors de sa connexion pour indiquer quel type de session IP il souhaite établir. Dans une infrastructure Amarisoft, chaque APN peut être associé à un adressage IP, une interface de sortie et des règles réseau distincts.

Dans ce projet, un APN spécifique — l'APN IoT — est ajouté à la configuration du MME. Tout UE se connectant via cet APN se voit attribuer une adresse dans le sous-réseau `10.52.0.64/27` et son trafic est acheminé via le VLAN 52 en mode sans NAT, indépendamment du trafic des autres UE utilisant l'APN par défaut.

2.3 Séparation du trafic par VLAN

La décision architecturale centrale est de différencier deux types de trafic via deux VLANs distincts. Chacun dispose de son propre comportement réseau, ce qui permet un contrôle précis de l'acheminement des paquets selon l'interface de sortie empruntée.

VLAN	Adressage	Mode	Usage

VLAN 5	réseau principal	NAT actif	Accès Internet classique pour les UE (APN par défaut)
VLAN 52	`10.52.0.0/24`	Sans NAT	Trafic IoT : les UE conservent leur IP réelle et sont vus comme de vraies machines du réseau

> Le VLAN 52 en mode sans NAT est le point différenciant de cette architecture. Contrairement au VLAN 5 où les UE sont masqués derrière l'adresse IP du serveur, ici les UE utilisant l'APN IoT conservent leur propre adresse IP sur tout le trajet réseau et peuvent être adressés directement.

2.4 Plan d'adressage

```
UE connecté (APN IoT)  : 10.52.0.66
Interface tunnel tun1  : 10.52.0.65   (sous-réseau /27 : 10.52.0.64/27)
Interface vlan.52      : 10.52.0.2/24
Gateway réseau 52      : 10.52.0.1

LAN principal          : 10.51.4.0/22
Gateway LAN            : 10.51.4.2
Interface serveur      : enp0s31f6 = 10.51.5.210
```

Chaque UE utilisant l'APN IoT reçoit une adresse dans le sous-réseau `10.52.0.64/27`, géré par l'interface tunnel `tun1`. Ce sous-réseau peut accueillir jusqu'à 30 équipements simultanément.

---

3. Fonctionnement réseau

3.1 Rôle des interfaces clés

Interface tun1 — lien interne LTE

L'interface `tun1` est une interface point-à-point virtuelle, interne au serveur. Elle matérialise le lien entre le cœur LTE Amarisoft et la pile IP Linux : tous les paquets émis par les UE en transitent avant d'être acheminés vers l'extérieur. Elle porte l'adresse `10.52.0.65`, qui sert de passerelle locale aux UE.

Interface vlan.52 — sortie réseau

L'interface `vlan.52` est l'interface réseau réelle, rattachée au réseau physique `10.52.0.0/24`. C'est elle qui permet aux UE d'être visibles en dehors du serveur Amarisoft avec leur propre adresse IP. Elle porte l'adresse `10.52.0.2` et communique avec la gateway `10.52.0.1`.

3.2 Chemin du trafic

VLAN 5 — accès Internet avec NAT

Dans ce mode, le trafic des UE est masqué derrière l'adresse IP du serveur Amarisoft. Le réseau externe ne voit qu'une seule source, ce qui simplifie le routage mais empêche de distinguer ou d'adresser les UE individuellement.

```
UE --> eNB --> MME --> VLAN 5 --> NAT --> Internet
```

VLAN 52 — trafic APN IoT sans NAT

Dans ce mode, les UE utilisant l'APN IoT conservent leur propre adresse IP tout au long du chemin. Le réseau doit connaître le sous-réseau `10.52.0.64/27` et savoir y acheminer les réponses.

```
UE (APN IoT, 10.52.0.66)
      |
  tun1 (10.52.0.65)
      |
  serveur Amarisoft
      |
  vlan.52 (10.52.0.2)
      |
  gateway (10.52.0.1)
      |
  +-------------------+
  |                   |
Internet          LAN 10.51.x.x
```

3.3 Mode sans NAT — implications

L'option `--no-nat vlan.52` indique à Amarisoft de ne pas effectuer de translation d'adresse pour le trafic sortant par `vlan.52`. Concrètement :

- les UE conservent leur adresse IP source (`10.52.0.x`) sur tout le trajet réseau ;
- aucune réécriture de paquet n'est effectuée côté Amarisoft ;
- le réseau de destination doit être en mesure d'acheminer les réponses vers `10.52.0.64/27` .

Dans cette configuration, cela fonctionne naturellement car le réseau `10.52.0.0/24` existe déjà sur `vlan.52` et la gateway `10.52.0.1` peut router vers ce sous-réseau.

> **Point de vigilance :** pour que le trafic retour (LAN → UE) fonctionne, les équipements du LAN doivent disposer d'une route vers `10.52.0.64/27` pointant sur `10.52.0.2`. Sans cette route, les réponses ne parviennent pas aux UE.

3.4 Synthèse des flux réseau

Sens	Condition requise	État
UE → Internet	NAT actif sur VLAN 5, ou routage correct côté réseau	Opérationnel
UE → LAN (10.51.x.x)	Routage correct, pas de `rp_filter` bloquant	Opérationnel
LAN → UE	Route vers `10.52.0.64/27` ajoutée côté réseau	À configurer côté réseau

---

4. Mise en place technique

4.1 Script lte_init.sh

Amarisoft fournit un script `lte_init.sh` dont le rôle est de configurer la sortie des paquets du cœur LTE vers le réseau physique. Il ne gère pas les VLANs en tant que tels, mais met en place les conditions nécessaires pour que les paquets issus du cœur soient correctement acheminés vers l'interface de sortie désignée.

Ce script prend en charge :

- l'activation du **forwarding IP** (`ip_forward = 1`), qui permet au serveur d'agir comme routeur entre les interfaces ;
- la configuration des règles **iptables** nécessaires, notamment le masquage NAT si le mode NAT est actif ;
- l'activation du **proxy ARP**, qui rend les UE visibles sur le réseau physique sans nécessiter de routes explicites sur chaque équipement ;
- le nettoyage des règles qu'il a lui-même créées lors d'une exécution précédente.

Problème rencontré

>

> **Important :** le script `lte_init.sh` supprime l'intégralité de ses propres règles `iptables` à chaque lancement, sans distinction d'interface. Appelé deux fois de suite pour deux interfaces différentes, le second appel écrase entièrement la configuration établie par le premier.

Ce comportement est inhérent à la conception du script. La séquence problématique est la suivante :

```
lte_init.sh vlan.5           # configure la sortie pour VLAN 5
lte_init.sh --no-nat vlan.52 # écrase tout ce qui a été fait précédemment
```

Solution retenue

Pour contourner cette limitation, la solution retenue consiste à n'appeler `lte_init.sh` **qu'une seule fois** pour l'interface principale, puis à compléter manuellement la configuration de l'interface secondaire sans repasser par le script.

4.2 Script d'initialisation personnalisé

Amarisoft ne permettant qu'un seul point d'initialisation réseau natif, la gestion des deux interfaces de sortie est confiée à un script d'initialisation personnalisé. Ce script est appelé directement par Amarisoft au démarrage du service, ce qui garantit que la configuration réseau est toujours appliquée dans le bon ordre et sans dépendance à un service externe.

---

5. Gestion du driver SDR

5.1 Problème rencontré

L'antenne SDR est pilotée par un module noyau (`sdr.ko`) fourni par Amarisoft. Ce module doit impérativement être compilé pour la version exacte du noyau Linux en cours d'exécution.

Lors du déploiement, une incompatibilité a été identifiée entre la version du noyau pour laquelle le driver avait été compilé et la version effectivement active au démarrage du système :

```
Driver compilé pour  : 5.15.0-171
Noyau actif au boot  : 5.15.0-173
```

Conséquence directe : le fichier `/dev/sdr0` était absent au démarrage, rendant l'eNB incapable d'accéder à l'antenne et donc inutilisable.

5.2 Solution mise en œuvre

La résolution passe par trois actions complémentaires :

1. **Forcer le boot sur le noyau compatible** (`5.15.0-171`) via la configuration du chargeur GRUB.
2. **Rendre ce choix persistant** en définissant ce noyau comme entrée par défaut dans GRUB, de sorte qu'un redémarrage involontaire n'entraîne pas de régression.
3. **Bloquer les mises à jour automatiques du noyau** pour éviter qu'une nouvelle version s'installe et devienne la version par défaut.

> Cette contrainte est commune aux installations utilisant des drivers noyau compilés pour une version spécifique. Elle devra être prise en compte lors de toute opération de maintenance : une mise à jour du noyau nécessite une recompilation du module `sdr.ko`.

---

6. Accès aux équipements utilisateurs

6.1 Problématique

Une fois les UE attachés au réseau LTE et dotés d'une adresse IP dans le sous-réseau `10.52.0.64/27`, il s'est avéré impossible de s'y connecter directement depuis le réseau d'administration. En mode sans NAT, les UE possèdent une adresse IP routable, mais le retour du trafic depuis le LAN vers les UE nécessite une route spécifique que tous les équipements du réseau ne possèdent pas nécessairement. Par ailleurs, selon la configuration de l'UE, celui-ci peut ne pas accepter de connexions entrantes directes.

6.2 Solution : tunnel SSH inverse via un serveur tiers

Pour pallier cette limitation, une solution de tunnel SSH inverse a été mise en place. Le principe repose sur un **serveur tiers** — appelé ici serveur de log — accessible à la fois depuis les UE (via le réseau LTE) et depuis les postes d'administration.

Principe de fonctionnement

Plutôt que d'essayer de joindre l'UE directement, c'est l'UE lui-même qui initie la connexion SSH vers le serveur tiers et y ouvre un port de rebond. L'administrateur se connecte ensuite à ce port

depuis le serveur tiers, ce qui lui donne accès à l'UE sans avoir besoin d'une route directe vers lui.

```
UE (10.52.0.66)
    |
    |--(1) connexion SSH sortante vers serveur de log-->
    |
Serveur de log (accessible depuis l'UE et l'admin)
    |
    |<--(2) connexion SSH entrante depuis poste admin--
    |
    |--(3) rebond sur le port exposé par l'UE-->
    |
Accès à l'UE
```

Déroulement

1. L'UE établit un tunnel SSH vers le serveur de log en exposant un port local via une redirection de port distante.
2. L'administrateur se connecte au serveur de log.
3. Depuis le serveur de log, il rebondit sur le port exposé par l'UE pour obtenir une session sur l'équipement.

> Ce mécanisme de tunnel SSH inverse est une solution éprouvée pour accéder à des machines non joignables directement, que ce soit pour des raisons de routage, de pare-feu ou de NAT. Il ne nécessite aucune modification de l'infrastructure réseau existante et fonctionne dès lors que l'UE dispose d'un accès sortant vers le serveur tiers.

---

7. Bilan

7.1 Résultats obtenus

L'ensemble des objectifs définis en début de projet a été atteint. L'infrastructure LTE Amarisoft est opérationnelle avec les caractéristiques suivantes :

- APN IoT configuré et opérationnel au sein du cœur LTE.
- Réseau LTE fonctionnel, avec attachement et authentification des UE.
- Antenne SDR opérationnelle sur le noyau compatible verrouillé.

- Double réseau actif : VLAN 5 avec NAT et VLAN 52 sans NAT.
- Accès Internet et LAN disponibles pour les UE.
- Accès aux UE depuis l'extérieur assuré via tunnel SSH inverse.
- Architecture propre, intégrée nativement dans le service Amarisoft.

7.2 Bonnes pratiques retenues

La conception de cette infrastructure a été guidée par plusieurs principes :

- **Séparation nette des réseaux** : chaque flux dispose de son VLAN, de son comportement et de ses règles propres.
- **Intégration native** : la configuration réseau est gérée par les mécanismes internes d'Amarisoft plutôt que par des services externes, ce qui réduit les points de défaillance.
- **Idempotence** : les scripts d'initialisation sont conçus pour ne pas produire d'effets indésirables lorsqu'ils sont exécutés plusieurs fois.
- **Stabilité du noyau** : le kernel est verrouillé sur la version compatible avec le driver SDR, évitant toute régression silencieuse.

7.3 Conclusion

Ce projet a permis de déployer un réseau mobile privé complet, maîtrisé de la couche radio jusqu'à la sortie réseau. L'ajout de l'APN IoT offre une séparation nette du trafic dès la connexion de l'UE, sans impact sur les autres usages du réseau. La combinaison du mode sans NAT et du VLAN dédié donne aux équipements IoT une identité réseau à part entière, tandis que la solution de tunnel SSH inverse répond pragmatiquement à la contrainte d'accessibilité. L'ensemble constitue une base solide, documentée et extensible pour de futurs travaux sur les réseaux mobiles privés.

How to Install Ubuntu in Raspberry PI

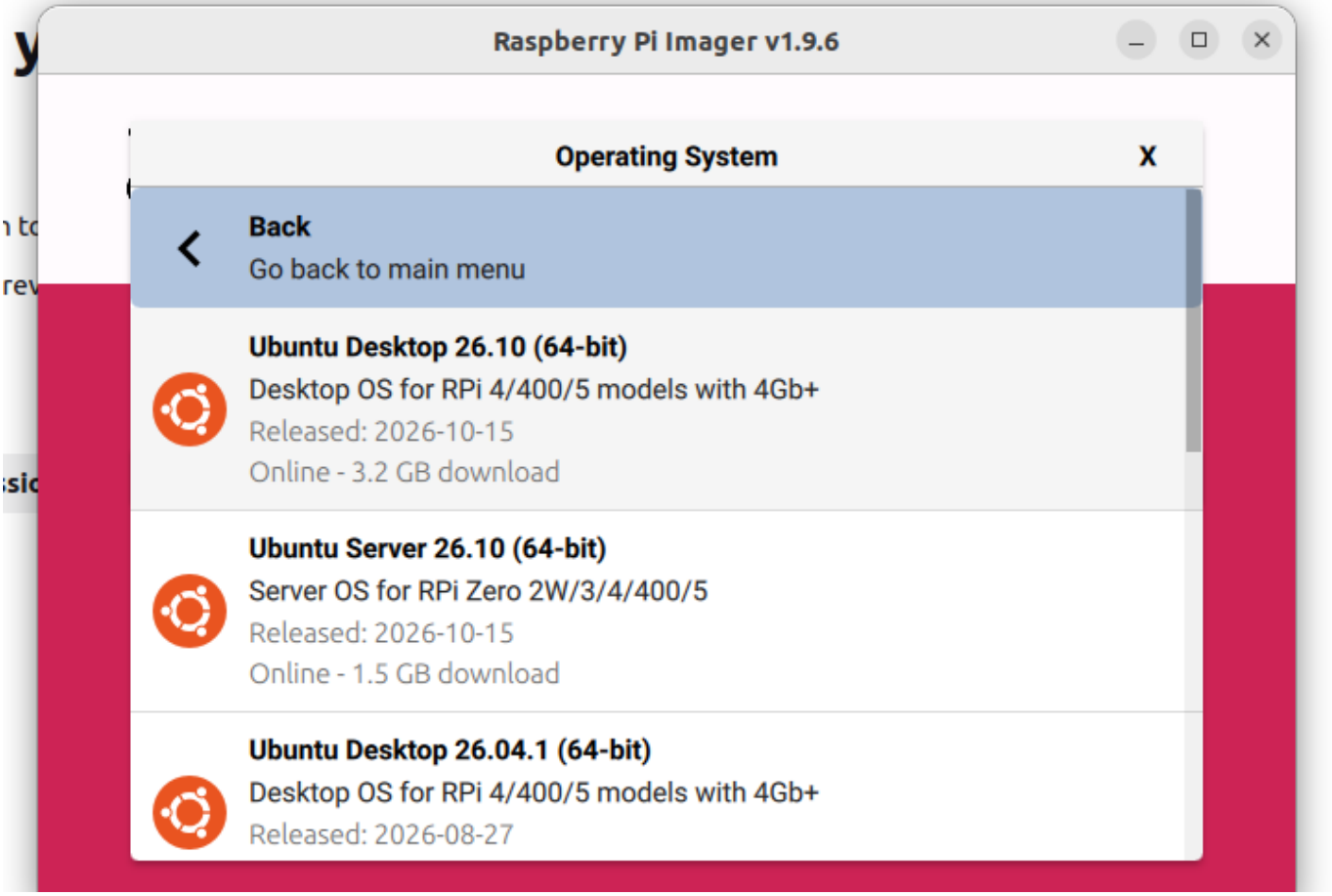
First you need to download Raspberry Pi Imager, for this purpose you need to execute:

sudo snap install rpi-imager

You can select the model of raspberry pi, in this case **Raspberry Pi 5**

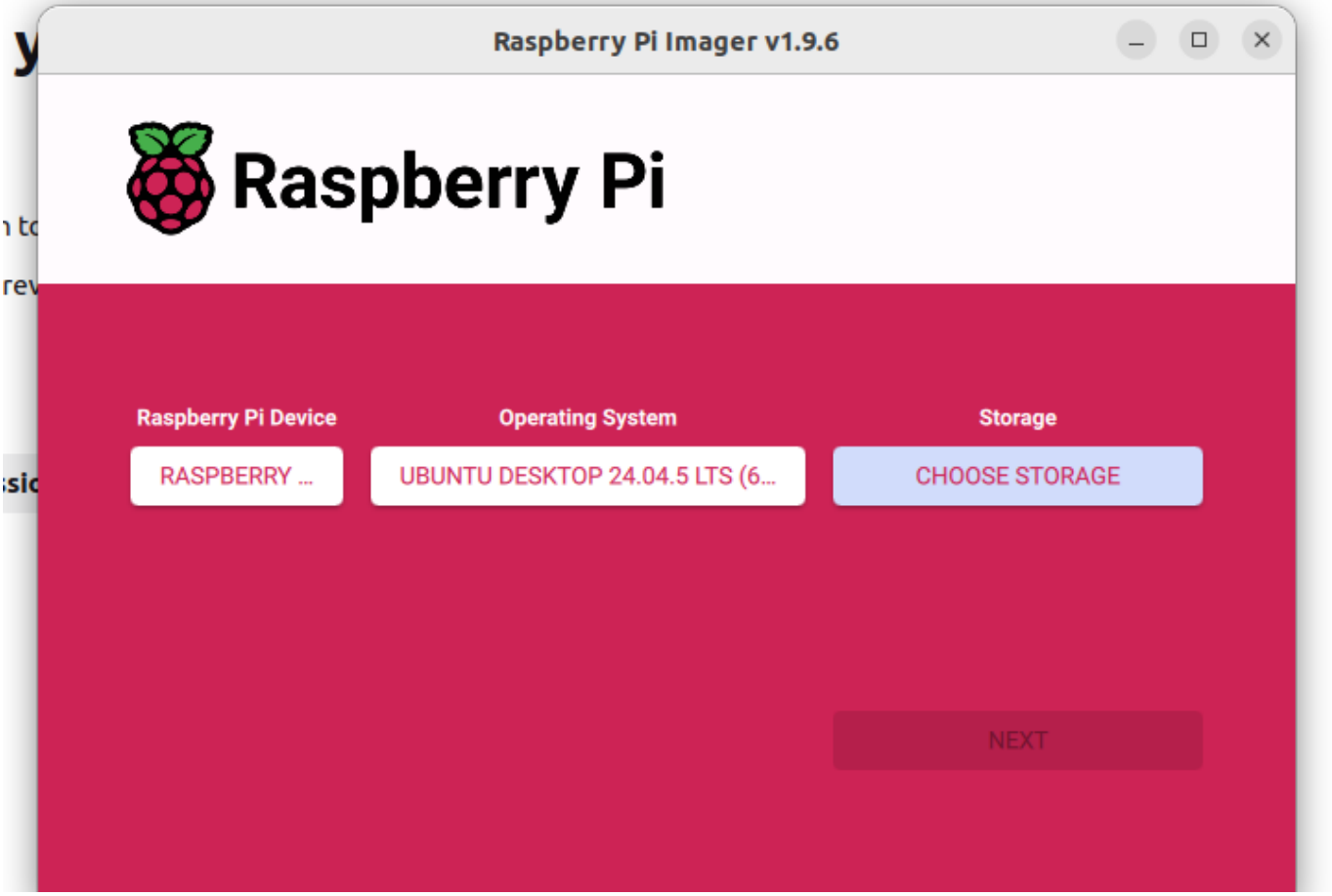


After you need to define the OS, in this case Ubuntu:



To finalize the procedure, you need to connect an SDCard (you need an Mini SDCard reader.

And this will appear in the menu of Storage.


Raspberry Pi Imager v1.9.6
Raspberry Pi
Raspberry Pi Device
Operating System
Storage
RASPBERRY ...
UBUNTU DESKTOP 24.04.5 LTS (6...
CHOOSE STORAGE
NEXT