

# MXQ Pro (S905W) – From Android TV Box to Linux Node

## Objectif

Recycler un boîtier TV Android basé sur un **Amlogic S905W** afin d'en faire un **nœud Linux headless**, destiné à des usages réseau tels que :

- VPN,
- accès distant,
- services légers.

## Contexte

Projet personnel d'étudiant, intégré à un **homelab**, documenté dans une démarche de **portfolio technique**.

L'objectif est autant fonctionnel que pédagogique, avec un fort accent mis sur la compréhension du matériel, du processus de boot et du diagnostic système.

---

## Présentation du matériel

### Boîtier

- Modèle : MXQ Pro (Android TV Box)
- SoC : Amlogic S905W
- CPU : Quad-core ARM Cortex-A53
- RAM : ~1 Go
- Stockage interne : eMMC (environ 8 à 16 Go selon le modèle)
- Stockage externe : carte micro-SD

- Réseau : Ethernet 100 Mbps
- Usage initial : Android TV (obsolète)



---

## Pourquoi le S905W est intéressant (et limité)

Le **S905W** est un SoC très répandu dans les TV box à bas coût.

Il n'est pas officiellement supporté par Armbian, ce qui implique une dépendance à :

- un u-boot communautaire,
- des Device Tree ([.dtb](#)) adaptés,
- une phase importante d'essais et d'erreurs.

Cela en fait une plateforme avec un **fort intérêt pédagogique**, malgré des limites claires en termes de support et de performances.

---

# Sources et choix de la build Armbian

Le système d'exploitation retenu pour ce projet est **Armbian**, basé sur **Debian Bookworm**, via une build communautaire adaptée aux boîtiers TV Amlogic (S905W).

Les images officielles Armbian ne prenant pas en charge ce type de matériel, le choix s'est porté sur une **build communautaire maintenue activement**, spécifiquement conçue pour les SoC Amlogic S9xxx.

## Dépôt GitHub principal

Les images utilisées proviennent du dépôt suivant :

<https://github.com/ophub/amlogic-s9xxx-armbian>

Ce dépôt fournit :

- des images Armbian préconfigurées pour les SoC Amlogic (S905W, S905X, S912, etc.),
- des kernels adaptés (legacy et mainline selon les versions),
- une documentation détaillant les méthodes de boot, de configuration DTB et de dépannage.

Il s'agit aujourd'hui de **la référence communautaire** pour l'exécution d'Armbian sur les TV box Amlogic.

---

## Version utilisée dans ce projet

La build sélectionnée correspond à :

- **Armbian basé sur Debian Bookworm**
- Architecture : **arm64**
- Kernel : **vendor / legacy (5.15 ou 6.1 selon la release)**

Les releases sont accessibles directement depuis la section *Releases* du dépôt :

<https://github.com/ophub/amlogic-s9xxx-armbian/releases>

Le choix d'une version Bookworm a été motivé par :

- la stabilité à long terme de Debian,
  - la compatibilité avec les usages serveur (VPN, SSH, services réseau),
  - la cohérence avec le reste du homelab basé sur Debian / Proxmox.
- 

## Documentation associée

La documentation officielle du projet est disponible ici :

<https://github.com/ophub/amlogic-s9xxx-armbian/blob/main/README.md>

Elle couvre notamment :

- les méthodes de flash sur carte SD,
- la sélection et l'utilisation des Device Tree (`.dtb`),
- les procédures de dépannage courantes,
- les limitations connues selon les modèles de boîtiers.

## Choix de l'OS et justification

### OS retenu

- Armbian
- Base : Debian Bookworm
- Kernel : legacy / vendor (5.15 ou 6.1 selon l'image)

### Armbian\_bookworm\_arm64\_server\_2026.01

**Armbian Image information**

- Default username: `root`
- Default password: `1234`
- Install command: `armbian-install`
- Update command: `armbian-update`

**Applicable platform**

- `arm64`
- Docker image: <https://hub.docker.com/u/ophub>

▶ Assets 138

1 person reacted

## Pourquoi Debian Bookworm

- Stabilité à long terme.
  - Écosystème serveur mature.
  - Compatibilité native avec WireGuard et OpenVPN.
  - Cohérence avec le reste du homelab (serveur Proxmox basé sur Debian).

Dans un contexte headless orienté réseau, Debian a été préféré à Ubuntu.

## Device Tree (DTB) : point central du projet

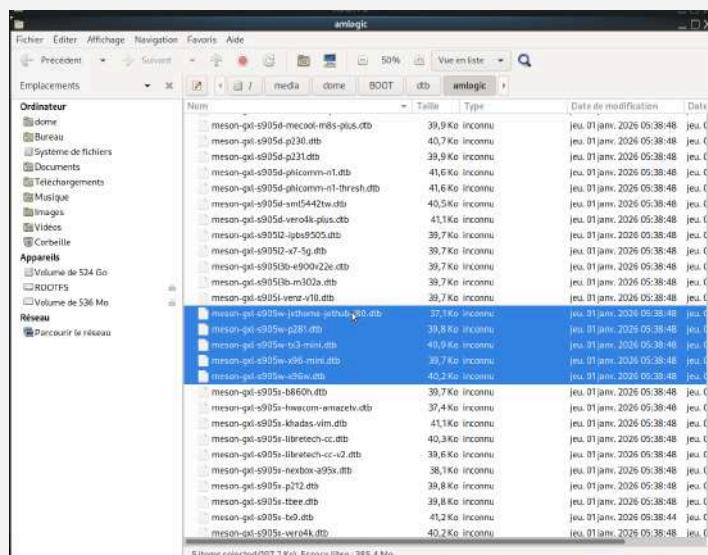
Le **Device Tree** définit la manière dont le matériel est exposé au noyau Linux. Sur S905W, un DTB incorrect entraîne :

- absence de réseau,
  - échec du boot,
  - écran figé sur le logo

## DTB testés :

- meson-gxl-s905w-p281.dtb
  - meson-gxl-s905w-x96-mini.dtb
  - meson-gxl-s905w-tx3-mini.dtb

Voir photo : dossier `/dtb/amlogic` contenant plusieurs fichiers `.dtb`.

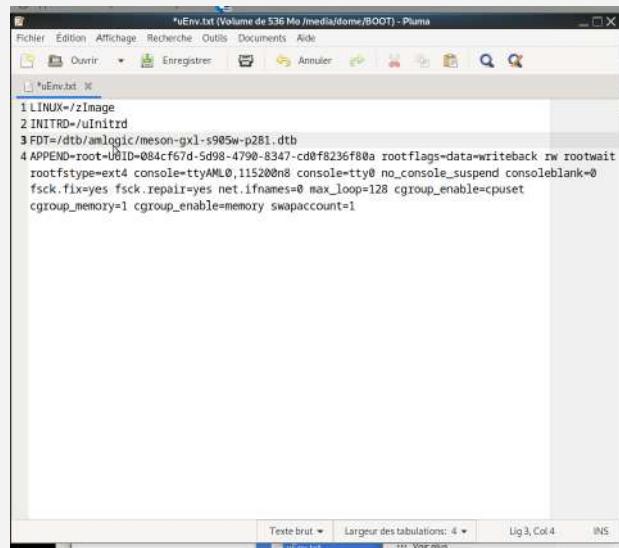


## Configuration finale retenue

Dans le fichier `uEnv.txt` :

- `FDT=/dtb/amlogic/meson-gxl-s905w-p281.dtb`

Voir photo : édition de `uEnv.txt`.



The screenshot shows a window titled "uEnv.txt (Volume de 536 Mo /media/dome/BOOT) - Pluma". The file content is as follows:

```
1 LINUX=/vImage
2 INITRD=/uInitrd
3 FDT=/dtb/amlogic/meson-gxl-s905w-p281.dtb
4 APPEND=root=UUID=084cf67d-5d98-4790-8347-cd0f8236f80a rootflags=data=writeback rw rootwait
rootfstype=ext4 console=ttyAML0,115200n8 console=tty0 no_console_suspend consoleblank=0
fsck.fix=yes fsck.repair=yes net.ifnames=0 max_loop=128 cgroup_enable=cpuset
cgroup_memory=1 cgroup_enable=memory swapaccount=1
```

## Problème critique : alimentation

Un point souvent sous-estimé.

### Symptômes observés

- Boot aléatoire.
- Écran bloqué sur le logo.
- Reboots en boucle.
- Services instables, notamment SSH.



### Solution

- Remplacement par une alimentation 5V 2A stable et de meilleure qualité.
- Boot immédiatement fiable après remplacement.

## ancienne alimentation



nouvelle alimentation



Conclusion : le problème n'était pas logiciel.

**Armbian OS**

v25.11.0 for Aml.S905w running Armbian Linux 6.1.158-ophub

Packages: Debian stable (bookworm)

IPv4: (LAN) 172.16.0.106 (WAN) 77.196.\*ok.\*ok

Performance:

Load: 32% Uptime: 1 min

Memory usage: 18% of 794M

CPU temp: 61°C Usage of /: 3% of 58G

RX today: 4 GiB

Commands:

Configuration : armbian-config

Monitoring : htop

Last login: Thu Jan 8 11:45:58 CET 2026 on tty1

root@armbian:~#

À l'invite :

`armbian login:`

Identifiants par défaut :

`login: root`

`password: 1234`

Note importante: Clavier Qwerty!!

---

## Configuration clavier (US vers FR)

### Problème

- Console configurée par défaut en QWERTY.
- Inconfort important lors des phases de debug.

### Solution temporaire

- `loadkeys fr`

Suffisant pour la phase d'installation.

Une configuration persistante est prévue ultérieurement dans un contexte entièrement headless.

**Important : dans une démarche visant à éviter les problèmes ultérieurs, il est recommandé d'exécuter un**

`apt update && apt upgrade -y`

**afin de partir sur une base système saine et à jour.**

---

## Solution définitive et fiable pour la configuration du clavier (US vers FR)

### Contexte

Sur les images Armbian destinées aux boîtiers TV basés sur des SoC Amlogic, la configuration standard du clavier via les mécanismes Debian classiques (`keyboard-configuration`, `console-setup`) n'est pas systématiquement appliquée au démarrage.

Dans ce projet, plusieurs méthodes ont été testées afin de rendre la disposition clavier AZERTY persistante :

- reconfiguration via `dpkg-reconfigure keyboard-configuration`,
- redémarrage des services `keyboard-setup`,
- modification des fichiers de configuration standards Debian.

Malgré une configuration correcte sur le papier, ces approches se sont révélées **inefficaces après redémarrage**, le clavier revenant systématiquement en disposition US.

Ce comportement est lié au processus de boot spécifique des images Armbian pour TV box, qui surcharge ou ignore certaines étapes classiques d'initialisation.

---

## Choix de la solution

Face à l'échec des méthodes conventionnelles, il a été décidé d'opter pour une **solution plus radicale mais totalement fiable** : forcer explicitement le chargement du keymap français à chaque démarrage, indépendamment des mécanismes internes d'Armbian.

Cette approche repose sur un principe simple :

- appliquer la configuration clavier **après l'initialisation de la console**,
- garantir son exécution **à chaque boot**,
- ne dépendre d'aucun comportement implicite du système.

### Étape 1 – Vérifier que le keymap FR existe

```
ls /usr/share/keymaps/i386/azerty/fr.kmap.gz
```

Si le fichier existe → OK

Sinon :

```
apt install console-data
```

## Étape 2 – Créer un service systemd dédié

```
nano /etc/systemd/system/keyboard-fr.service
```

Taper exactement ceci :

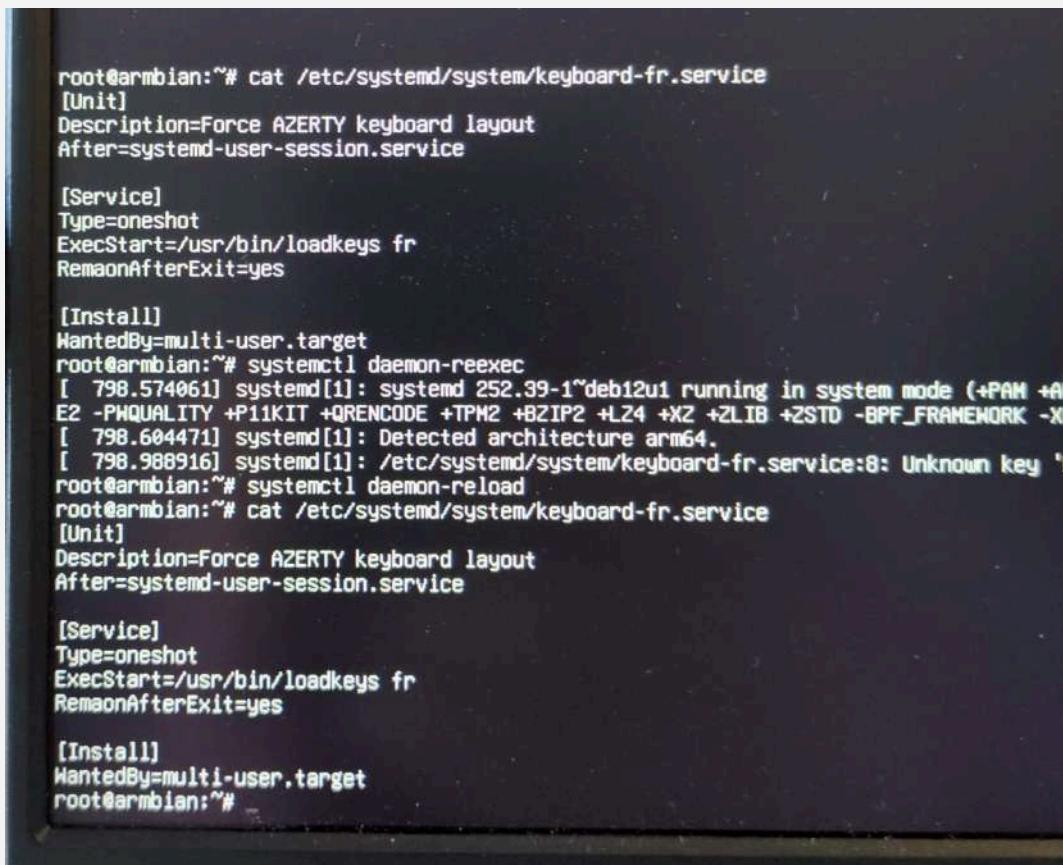
```
[Unit]
Description=Force AZERTY keyboard layout
After=systemd-user-sessions.service

[Service]
Type=oneshot
ExecStart=/usr/bin/loadkeys fr
RemainAfterExit=yes

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

## Étape 3 – Activer le service

```
systemctl daemon-reexec
systemctl daemon-reload
systemctl enable keyboard-fr.service
```



The screenshot shows a terminal session on an Armbian system. The user runs 'cat' on the service file to verify its contents. Then, they run 'systemctl daemon-reexec' to reload the daemon. After that, they run 'systemctl daemon-reload' to update the daemon's configuration. Finally, they run 'systemctl enable keyboard-fr.service' to ensure the service starts automatically at boot.

```
root@armbian:~# cat /etc/systemd/system/keyboard-fr.service
[Unit]
Description=Force AZERTY keyboard layout
After=systemd-user-session.service

[Service]
Type=oneshot
ExecStart=/usr/bin/loadkeys fr
RemainAfterExit=yes

[Install]
WantedBy=multi-user.target
root@armbian:~# systemctl daemon-reexec
[ 798.574061] systemd[1]: systemd 252.39-1~deb12u1 running in system mode (+PAM +AL
E2 -PWQUALITY +F11KIT +QRENCODE +TPM2 +BZIP2 +LZ4 +XZ +ZLIB +ZSTD -BPF_FRAMEWORK -XK
[ 798.604471] systemd[1]: Detected architecture arm64.
[ 798.988916] systemd[1]: /etc/systemd/system/keyboard-fr.service:8: Unknown key 'R
root@armbian:~# systemctl daemon-reload
root@armbian:~# cat /etc/systemd/system/keyboard-fr.service
[Unit]
Description=Force AZERTY keyboard layout
After=systemd-user-session.service

[Service]
Type=oneshot
ExecStart=/usr/bin/loadkeys fr
RemainAfterExit=yes

[Install]
WantedBy=multi-user.target
root@armbian:~#
```

## Étape 4 – Reboot et validation

`reboot`

Après reboot, tester immédiatement :

- `a` → doit écrire `a`
- `q` → doit écrire `a`
- `;` → doit écrire `m`

Si c'est bon → **c'est définitivement réglé.**

---

## Problèmes SSH et résolution

### Symptômes

- Service `ssh.service` en échec.
- Impossible de se connecter à distance.
- `systemctl status ssh` indiquant un état FAILED.

Voir photos : erreurs SSH, codes de sortie et journaux systemd.

### Diagnostic

- Fichier `sshd_config` partiellement corrompu (directive `Subsystem sftp`).
- Absence totale de clés hôte SSH.

Erreur observée :

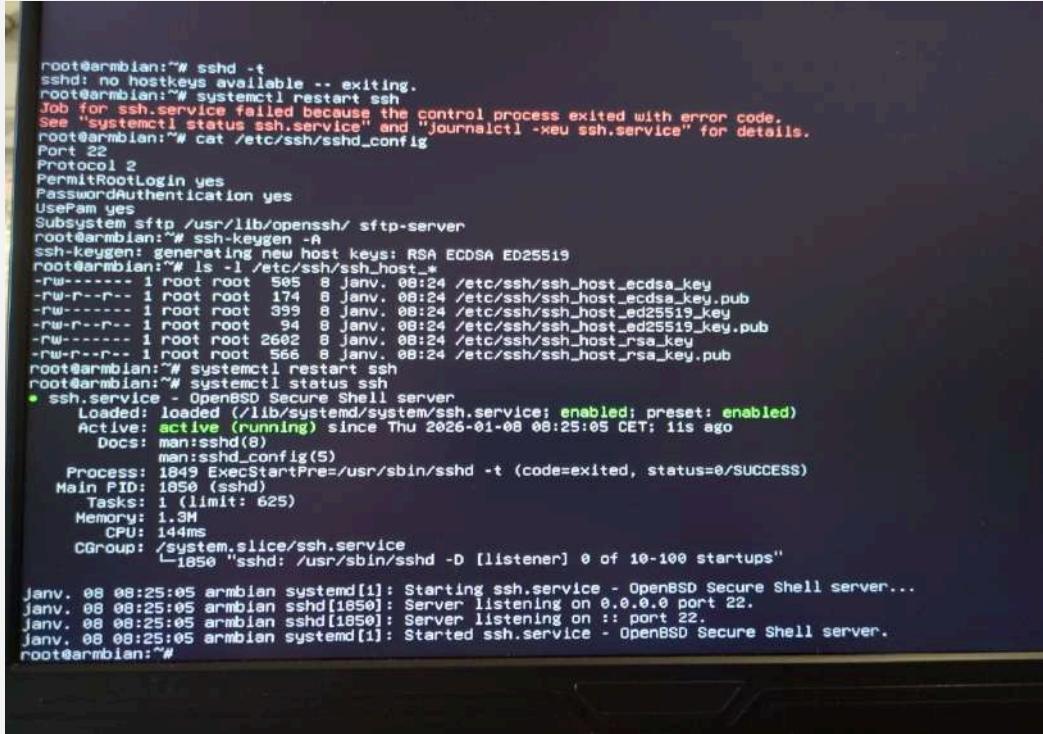
- `sshd: no hostkeys available`

### Résolution

- `ssh-keygen -A`

- `systemctl restart ssh`

Voir photo : génération des clés et service SSH en état actif.



```

root@armbian:~# sshd -t
sshd: no hostkeys available -- exiting.
root@armbian:~# systemctl restart ssh
Job for ssh.service failed because the control process exited with error code.
See "systemctl status ssh.service" and "journalctl -xeu ssh.service" for details.
root@armbian:~# cat /etc/ssh/sshd_config
Port 22
Protocol 2
PermitRootLogin yes
PasswordAuthentication yes
UsePam yes
Subsystem sftp /usr/lib/openssh/ sftp-server
root@armbian:~# ssh-keygen -A
ssh-keygen: generating new host keys: RSA ECDSA ED25519
root@armbian:~# ls -l /etc/ssh/ssh_host_*
-rw----- 1 root root 505 8 Janv. 08:24 /etc/ssh/ssh_host_ecdsa_key
-rw----r-- 1 root root 174 8 Janv. 08:24 /etc/ssh/ssh_host_ecdsa_key.pub
-rw----- 1 root root 399 8 Janv. 08:24 /etc/ssh/ssh_host_ed25519_key
-rw----r-- 1 root root 94 8 Janv. 08:24 /etc/ssh/ssh_host_ed25519_key.pub
-rw----- 1 root root 2602 8 Janv. 08:24 /etc/ssh/ssh_host_rsa_key
-rw----r-- 1 root root 566 8 Janv. 08:24 /etc/ssh/ssh_host_rsa_key.pub
root@armbian:~# systemctl restart ssh
root@armbian:~# systemctl status ssh
● ssh.service - OpenBSD Secure Shell server
   Loaded: loaded (/lib/systemd/system/ssh.service; enabled; preset: enabled)
   Active: active (running) since Thu 2026-01-08 08:25:05 CET; 11s ago
     Docs: man:sshd(8)
           man:sshd_config(5)
  Process: 1849 ExecStartPre=/usr/sbin/sshd -t (code=exited, status=0/SUCCESS)
 Main PID: 1850 (sshd)
    Tasks: 1 (limit: 625)
      Memory: 1.3M
        CPU: 144ms
       CGroup: /system.slice/ssh.service
               └─ 1850 "sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups"

Janv. 08 08:25:05 armbian systemd[1]: Starting ssh.service - OpenBSD Secure Shell server...
Janv. 08 08:25:05 armbian sshd[1850]: Server listening on 0.0.0.0 port 22.
Janv. 08 08:25:05 armbian sshd[1850]: Server listening on :: port 22.
Janv. 08 08:25:05 armbian systemd[1]: Started ssh.service - OpenBSD Secure Shell server.
root@armbian:~#

```

Résultat :

- Connexion SSH fonctionnelle.
- Objectif initial atteint.

## Possibilités de flash sur les cartes Amlogic S905W

Sur ce type de boîtier, trois approches principales existent.

### 1. Boot sur carte SD (non destructif – choix initial)

- Armbian est lancé depuis la carte micro-SD.
- L'eMMC interne reste intacte.
- Méthode la plus sûre pour tester et itérer.

## 2. Installation sur eMMC (destructif)

- Armbian remplace Android.
- Démarrage plus rapide.
- Procédure difficilement réversible sans outils Amlogic.

Voir photo : commande `armbian-install` indiquant l'absence d'eMMC détectée.

```
root@armbian:~# lsblk
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
mmcblk1 179:0   0 59,5G  0 disk
└─mmcblk1p1 179:1 0  511M  0 part /boot
└─mmcblk1p2 179:2 0  59G  0 part /var/log.hdd
/
zram0    253:0   0 397,5M  0 disk [SWAP]
zram1    253:1   0   50M  0 disk /var/log
zram2    253:2   0    0B  0 disk
root@armbian:~# armbian install
armbian : commande introuvable
root@armbian:~# armbian-install
La commande « armbian-install » n'a pas été trouvée, mais il y en a 590 similaires
root@armbian:~# armbian-install
[ STEPS ] Installing Armbian to internal eMMC...
[ STEPS ] Checking dependencies...
[ INFO ] Dependency check completed. Proceeding installation...
[ STEPS ] Initializing the environment...
[ INFO ] Use mainline u-boot: [ no ]
[ INFO ] Use ampart tool: [ yes ]
[ INFO ] Show all devices: [ no ]
[ ERROR ] Internal eMMC storage wasn't found in this device!
root@armbian:~# _
```

Sur certains boîtiers Amlogic S905W, l'installation d'Armbian sur le stockage interne (eMMC) est théoriquement possible via l'outil `armbian-install`.

Dans le cadre de ce projet, cette option a été étudiée mais **n'a pas pu être mise en œuvre**, pour les raisons suivantes :

- l'eMMC interne n'a pas été détectée comme périphérique cible par Armbian,
- l'outil `armbian-install` a explicitement indiqué l'absence de stockage interne exploitable,
- ce comportement est connu sur certains modèles de TV box, en fonction du matériel exact et du support du bootloader.

En conséquence, le système Armbian a été conservé en exécution **depuis la carte micro-SD**, ce qui reste une solution fonctionnelle, stable et suffisante pour les usages prévus (VPN, accès distant, services réseau légers).

Cette limitation n'est **pas liée à une instabilité du système**, mais à des contraintes matérielles et de support propres à ce modèle de boîtier.

### 3. Flash via USB Burning Tool (non retenu)

- Nécessite un environnement Windows.
  - Dépendance à des drivers propriétaires.
  - Peu cohérent avec un workflow Linux / homelab.
- 

## État final du système

- Armbian démarre de manière stable.
- Réseau Ethernet fonctionnel.
- Accès SSH opérationnel.
- Installation sur eMMC volontairement non effectuée.
- Système prêt pour un usage VPN ou gateway réseau.

Voir photo :



---

## Prochaine étape

Un dépôt séparé sera consacré à la mise en place du VPN :

Nom prévu :

- `mxq-pro-vpn-node`

Contenu envisagé :

- WireGuard,
- accès distant sécurisé,
- intégration au homelab Proxmox,
- routage et NAT si nécessaire.

*“Si vous êtes arrivé jusqu’ici, félicitations : vous avez survécu aux Device Tree, aux alimentations douteuses, aux claviers QWERTY rebelles et à un SSH plus têtu qu’un bootloader Amlogic.*

*Merci d’avoir pris le temps de lire ce retour d’expérience, né de nombreuses itérations, de quelques frustrations, et de beaucoup d’apprentissage pour moi.”*