UNIVERSITÉ DE FIANARANTSOA

ÉCOLE NATIONALE D'INFORMATIQUE

RAPPORT PROJET DE SYSTÈME EMBARQUES

Mention : Informatique

Parcours : Informatique générale

Intitulé :

Mise en œuvre sur Raspberry Pi 3 B+ avec Buildroot

Présenté le : 21 Décembre 2025

Par : Monsieur RAMBELOSON Valisoa Ny Aina Charlin

Membres du Jury :

Rapporteurs : Monsieur RAZAFINDRANDRIANTSIMANIRY Dieu Do

Année Universitaire : 2024-2025

1. Introduction

Ce projet vise à construire un système Linux embarqué avec Buildroot sur une Raspberry Pi 3 B+. Il couvre la préparation de l'environnement, la configuration et la compilation du système, le démarrage de la carte et le déploiement d'applications embarquées.

2. Plateforme matérielle : Raspberry Pi 3 B+

La Raspberry Pi 3 B+ est équipée d'un processeur Broadcom BCM2837 (ARM Cortex-A53 quadricœur, 1,4 GHz), 1 Go de RAM, un port HDMI, 4 ports USB, Ethernet, WiFi et Bluetooth. Elle est adaptée aux projets IoT et systèmes embarqués.

Figure 1 – Vue de la Raspberry Pi 3 B+.

3. Préparation de l'environnement de développement

Le matériel inclut la Raspberry Pi 3 B+, une carte microSD, une alimentation, un câble série/USB et un PC hôte. Les logiciels nécessaires incluent Ubuntu, gcc, git et make. Buildroot est

téléchargé depuis son site officiel.

Dans notre projet, nous utilisons **VMware Player (version 16 ou supérieure)** afin d'exécuter une machine virtuelle **Ubuntu 22.04** qui contient les outils nécessaires. Cette machine virtuelle permet de travailler dans un environnement isolé et reproductible.

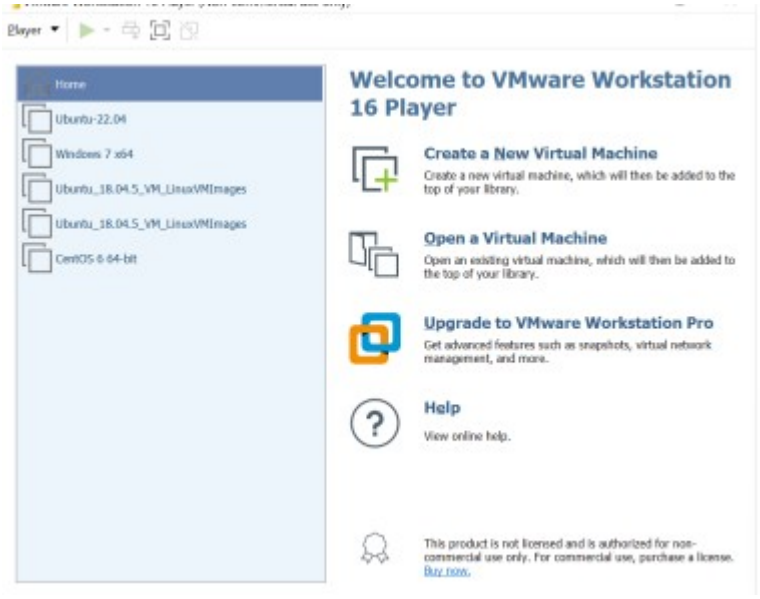

Figure 2 – Écran principal de VMware Player avec une VM Ubuntu disponible.

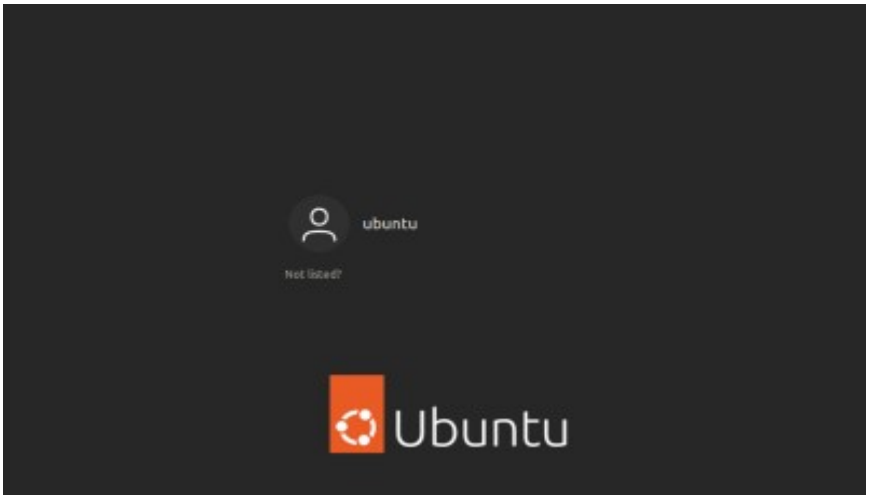

Figure 3 – Capture relative à la préparation de connexion au ubuntu.

Figure 4 – Page principale du site officiel de Buildroot.

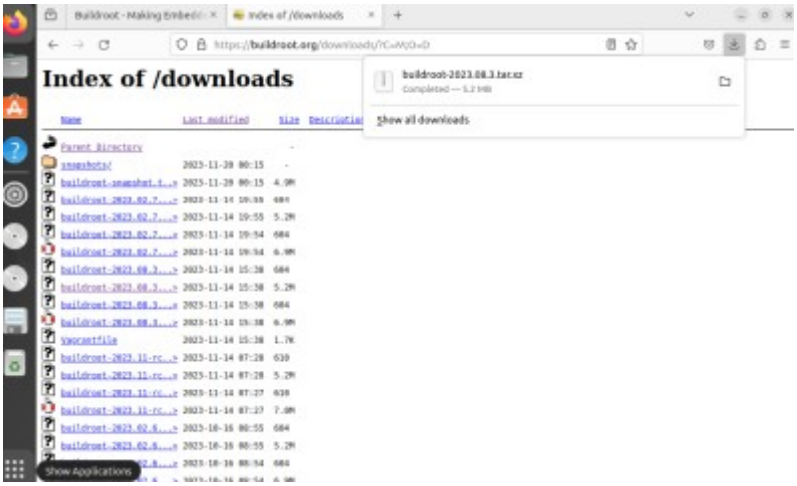

Figure 5 – Terminal Ubuntu (préparation de l'environnement de développement).

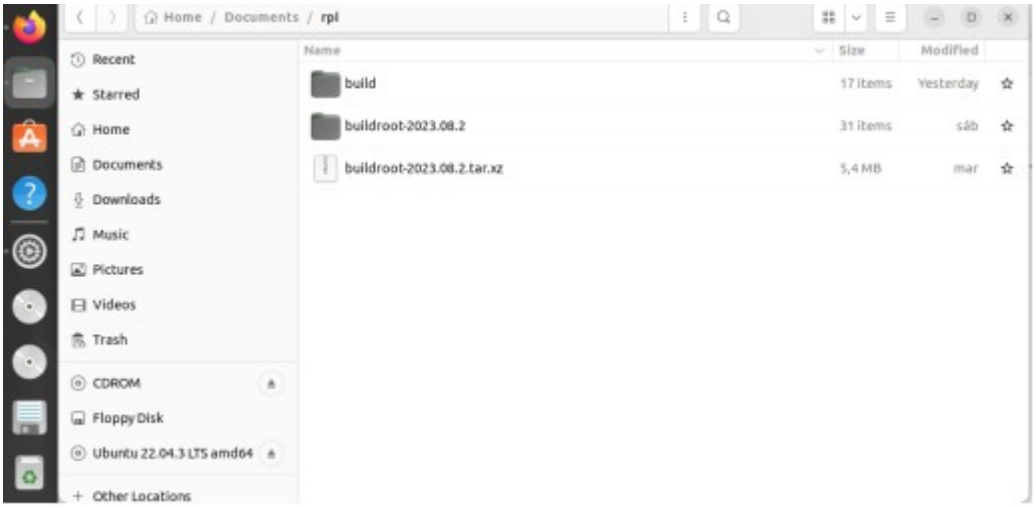

Figure 6 – Dossier Buildroot après extraction (le nom dépend de la version téléchargée).

4. Construction du système Linux avec Buildroot

Buildroot est configuré avec make menuconfig pour cibler ARM Cortex-A53. On choisit la toolchain, active Busybox et configure le noyau Linux. La commande make compile et génère bootloader, noyau, rootfs et toolchain.

Après le démarrage de VMware Player et la connexion à Ubuntu (utilisateur *ubuntu*, mot de passe *ubuntu*), nous téléchargeons **Buildroot** depuis son site officiel (https://buildroot.org). Le fichier est sauvegardé dans le dossier *Documents*, puis déplacé dans un répertoire spécifique `rpi` créé à cet effet. L'archive téléchargée est ensuite décompressée afin de rendre les sources accessibles.

Depuis Ubuntu, un terminal est ouvert afin de préparer la configuration. Les commandes suivantes permettent de créer un dossier de compilation dédié et de lancer l'outil de configuration de Buildroot :

```
rpi@ubuntu:~/Documents/rpi$ mkdir build
rpi@ubuntu:~/Documents/rpi$ cd build
rpi@ubuntu:~/Documents/rpi/build$ make O=$PWD -C
/home/ubuntu/Documents/rpi/buildroot-2023.08.2/ menuconfig
```

Cette étape permet d'accéder au menu de configuration (Fig. 7), où l'on définit les options du noyau, de la toolchain et des paquets embarqués.

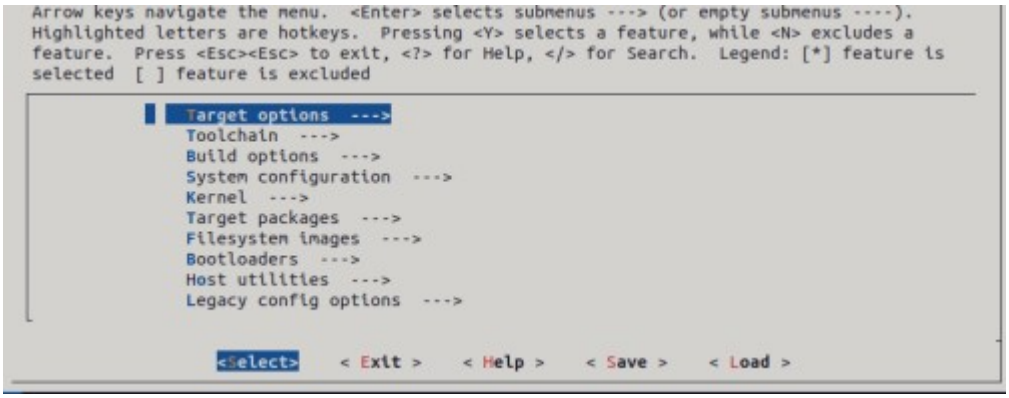

Fig. 7 – Menu principal de la configuration Buildroot (make menuconfig).

```
e/myDocuments/buildroot-2020.02.8/buildroot-2020.02.8/output/images/rpi-firmware
/fixup.dat' '::'" (stderr):
INFO: vfat(boot.vfat): adding file 'rpi-firmware/start.elf' as 'rpi-firmware/sta
rt.elf' ...
INFO: vfat(boot.vfat): cmd: "MTOOLS_SKIP_CHECK=1 mcopy -bsp -i '/home/dte/myDocu
ments/buildroot-2020.02.8/buildroot-2020.02.8/output/images/boot.vfat' '/home/dt
e/myDocuments/buildroot-2020.02.8/buildroot-2020.02.8/output/images/rpi-firmware
/start.elf' '::'" (stderr):
INFO: vfat(boot.vfat): adding file 'rpi-firmware/overlays' as 'rpi-firmware/over
lays' ...
INFO: vfat(boot.vfat): cmd: "MTOOLS_SKIP_CHECK=1 mcopy -bsp -i '/home/dte/myDocu
ments/buildroot-2020.02.8/buildroot-2020.02.8/output/images/boot.vfat' '/home/dt
e/myDocuments/buildroot-2020.02.8/buildroot-2020.02.8/output/images/rpi-firmware
/overlays' '::'" (stderr):
INFO: vfat(boot.vfat): adding file 'zImage' as 'zImage' ...
INFO: vfat(boot.vfat): cmd: "MTOOLS_SKIP_CHECK=1 mcopy -bsp -i '/home/dte/myDocu
ments/buildroot-2020.02.8/buildroot-2020.02.8/output/images/boot.vfat' '/home/dt
e/myDocuments/buildroot-2020.02.8/buildroot-2020.02.8/output/images/zImage' '::'
" (stderr):
INFO: hdimage(sdcard.img): adding partition 'boot' (in MBR) from 'boot.vfat' ...
INFO: hdimage(sdcard.img): adding partition 'rootfs' (in MBR) from 'rootfs.ext4'
 ...
INFO: hdimage(sdcard.img): writing MBR
dte@ubuntu:~/myDocuments/buildroot-2020.02.8/buildroot-2020.02.8$
```

Fig. 8 – Compilation et installation réussies de Buildroot.

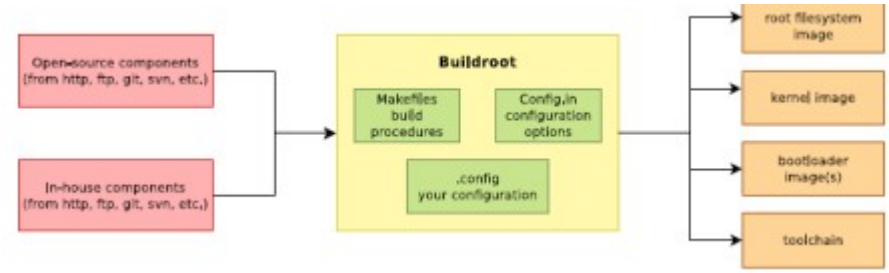

Figure 9 – Représentation schématique de l'outil Buildroot : génération du rootfs, du noyau, du bootloader et de la toolchain (source : Bootlin).

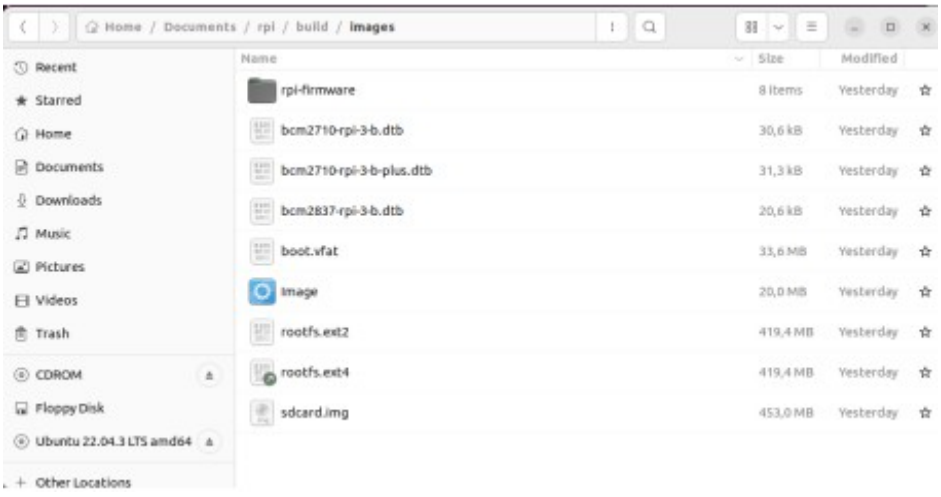

Figure 10 – *Contenu du dossier images/ avec les fichiers binaires du système embarqué.*

5. Démarrage de la Raspberry Pi

L'image générée est copiée sur carte SD et la Raspberry Pi démarre via le processus BootROM → bootcode.bin → start.elf → kernel → rootfs. Une console série (Putty) permet de suivre le démarrage.

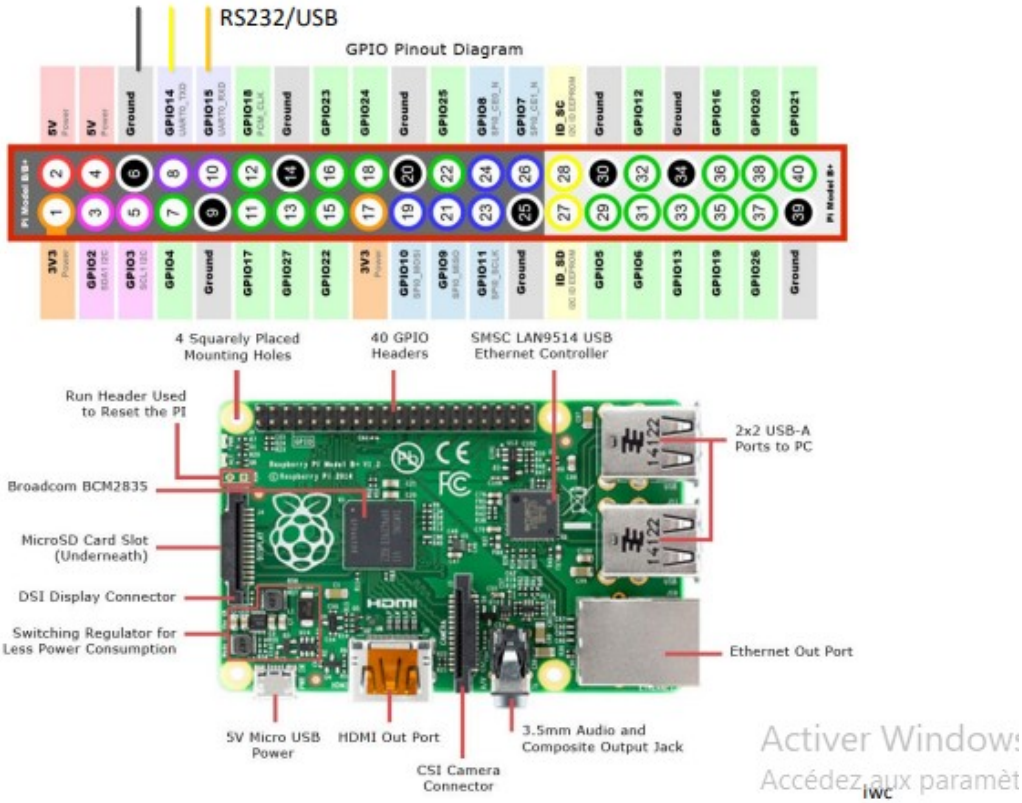

Figure 11 – Raspberry Pi 3 B+ avec identification des principaux éléments matériels.

Figure 12 – Identification des broches du connecteur GPIO de la Raspberry Pi 3 B+.

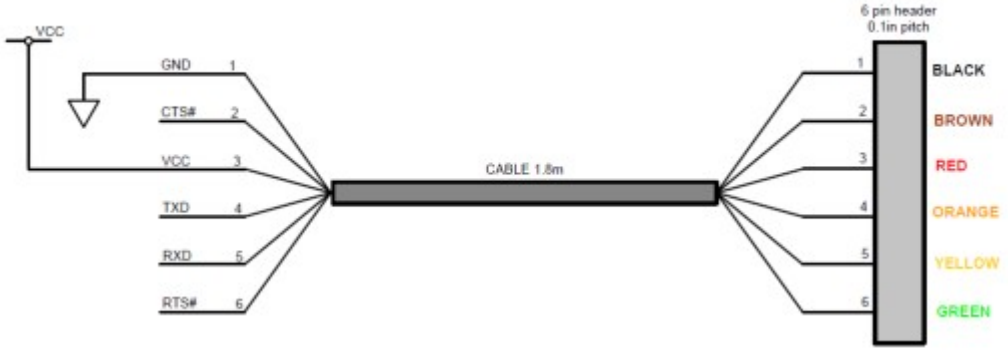

Figure 13 – Capture relative au démarrage de la Raspberry Pi.

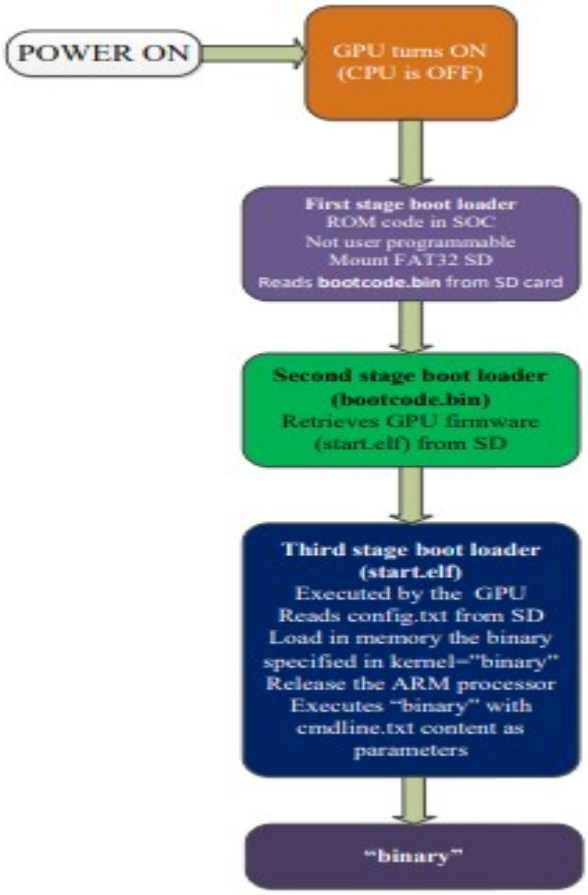

Figure 14 – Processus de démarrage du processeur BCM2837 de la Raspberry Pi 3 B+.

6. Mise en réseau et support WiFi

Le réseau filaire (eth0) est configuré en DHCP par défaut. Le support WiFi nécessite le firmware Broadcom BCM43438 et wpa_supplicant. La configuration se fait dans les fichiers /etc/network/interfaces et wpa_supplicant.conf.

7. Développement et déploiement d'applications

Eclipse CDT permet de développer en C/C++ en utilisant la toolchain générée par Buildroot. Les binaires sont transférés sur la Raspberry Pi via scp/ssh et exécutés. Le débogage est possible avec gdb/gdbserver.

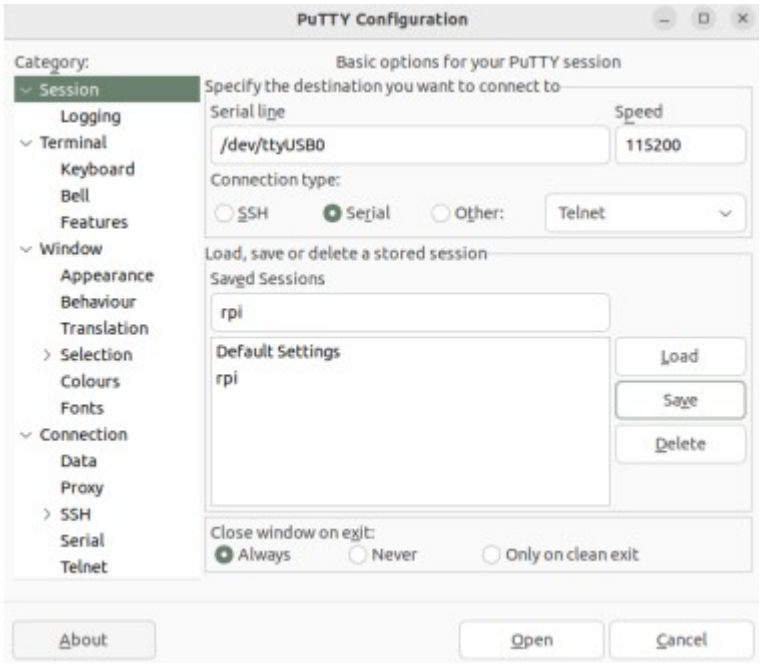

Figure 15 – Fenêtre principale du programme Putty (configuration de la console série).

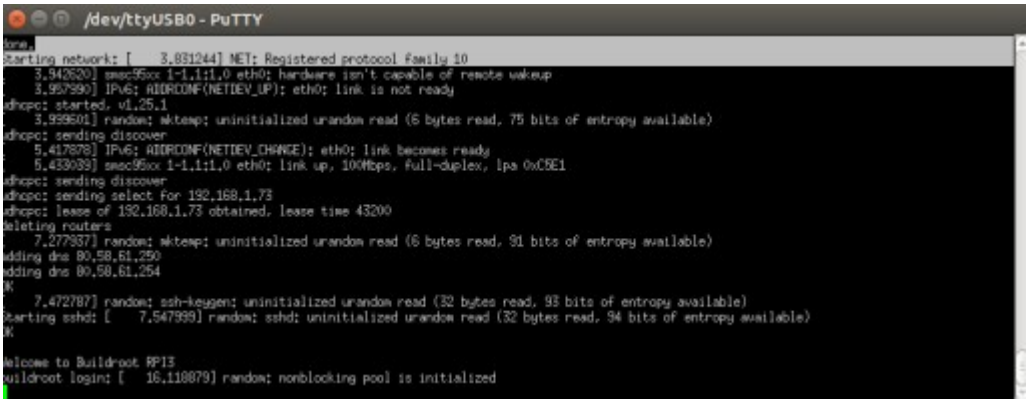

Figure 16 – Exemple de session série ouverte avec Putty (messages de démarrage Linux).

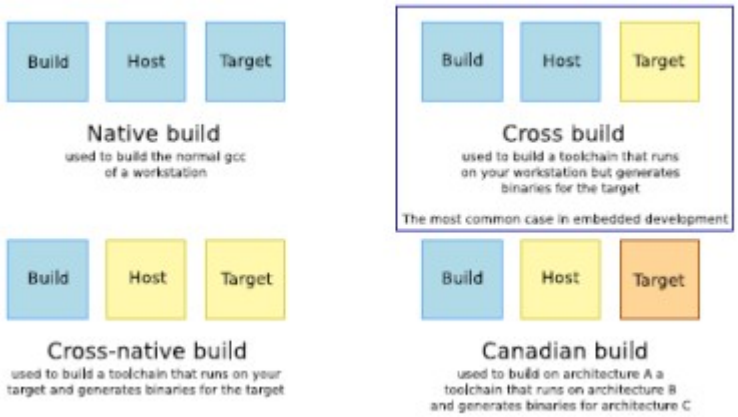

Figure 17 – Résumé des différentes configurations pour le développement d’applications pour systèmes embarqués (source : Free Electrons / Bootlin).

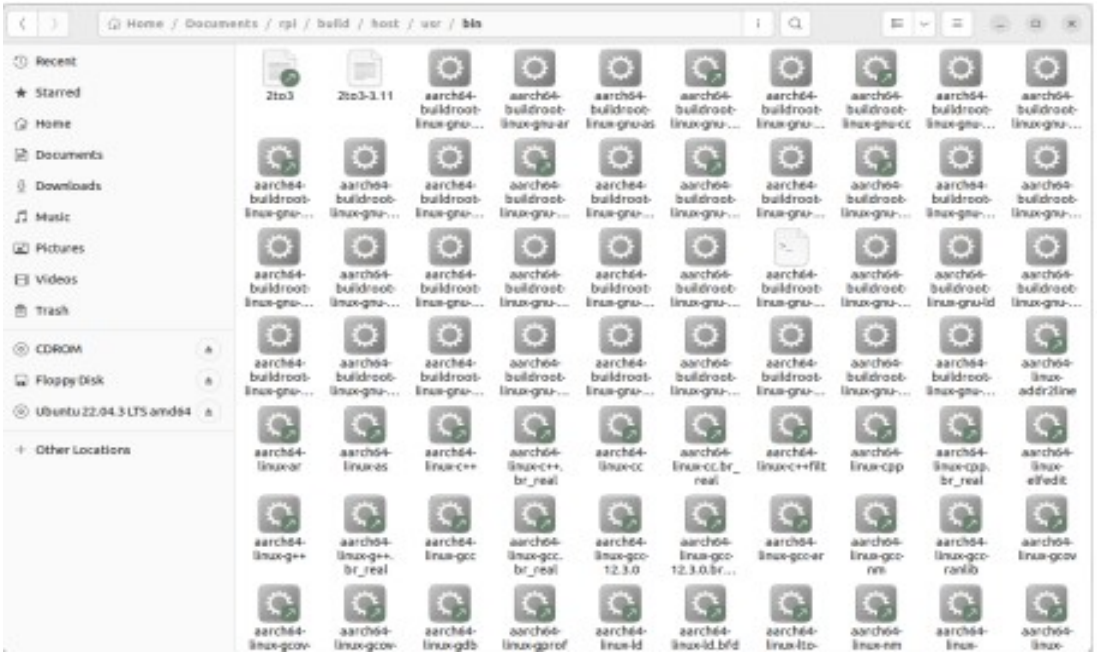

Figure 18 – Outils de compilation croisée installés sur l’ordinateur hôte.

8. Conclusion et perspectives

Le projet a permis de mettre en place un système Linux embarqué minimal sur Raspberry Pi 3 B+. Les perspectives incluent l’optimisation du noyau, l’ajout de services réseaux (web, MQTT), la sécurisation (SSH, firewall) et l’utilisation de conteneurs légers comme Docker.