Installation archlinux sur BananaPi zero

Table of Contents

- Installation archlinux sur BananaPi zero
  - Prérequis
  - Préparation côté pc (terminal/console)
    - Préparation du système hôte
    - Installer le système de base sur une carte SD
    - Compiler et copier le chargeur de démarrage U-Boot
    - Pilote X.org
    - Dépannage
    - Scripte d'installation
  - Finalisation de l'installation
    - Configuration de l'accès SSH
    - Configuration des locales
    - Installation de python
    - Installation des bibliothèques GPIO

Cet article présente l'installation et la configuration d'Archlinux sur BananaPi M2 Zero.

Quelques sources:

- http://www.banana-pi.org/downloadall.html
- https://wiki.archlinux.org/index.php/Banana_Pi#Compile_and_copy_U-Boot_bootloader

Prérequis

- pc hôte avec linux
- lecteur/graveur de carte sd
- carte micro-sd avec 4 Go ou plus
- accès Internet
- moniteur avec entrée HDMI
- connexion réseau filaire avec dhcp, accès entre pc et bananapi et accès internet
- clavier usb pour saisie directe

Préparation côté pc (terminal/console)

Préparation du système hôte

Les paquets suivants doivent être installés dans l'hôte afin de pouvoir compiler le fichier **dtb**[1)]:

- **u-boot-tools**: outils compagnons pour le chargeur de démarrage **Das U-Boot**

- **swig**: outil logiciel d'interfaçage pour lier des programmes en C/C++ avec des programmes de plus haut-niveau comme Python
- **dtc**: Device Tree Compiler, prend en entrée une arborescence de périphériques dans un format donné et génère un device-tree dans un autre format pour démarrer les noyaux sur les systèmes embarqués. Typiquement, le format d'entrée est “dts”, un format source lisible par l'homme, et crée un format **dtb**, ou binaire en sortie.

```
sudo pacman --needed -S uboot-tools wget swig arm-none-eabi-gcc dtc git
```

```
sudo apt-get install gcc-arm-none-eabi u-boot-tools swig dtc git wget
```

Installer le système de base sur une carte SD

Mettre à zéro le début de la carte SD en remplaçant sdX par le périphérique de bloc du lecteur :

```
dd if=/dev/zero of=/dev/sdX bs=1M count=8
```

Utiliser **fdisk** pour partitionner la carte SD avec une table de partition MBR et formater la partition racine, en remplaçant sdX1 par la partition racine :

```
mkfs.ext4 -O ^metadata_csum,^64bit /dev/sdX1
```

Monter le système de fichiers Ext4 :

```
mkdir mnt
mount /dev/sdX1 mnt
```

Télécharger ArchLinuxARM avec wget :

```
wget https://archlinuxarm.org/os/ArchLinuxARM-armv7-latest.tar.gz
```

Extraire le système de fichiers racine :

```
bsdtar -xpf ArchLinuxARM-armv7-latest.tar.gz -C mnt/
```

Créer un fichier `boot.cmd` avec le script de démarrage suivant:

```
part uuid ${devtype} ${devnum}:${bootpart} uuid
setenv bootargs console=${console} root=PARTUUID=${uuid} rw rootwait

if load ${devtype} ${devnum}:${bootpart} ${kernel_addr_r} /boot/zImage; then
  if load ${devtype} ${devnum}:${bootpart} ${fdt_addr_r}
/boot/dtbs/${fdtfile}; then
```

```
    if load ${devtype} ${devnum}:${bootpart} ${ramdisk_addr_r}
/boot/initramfs-linux.img; then
      bootz ${kernel_addr_r} ${ramdisk_addr_r}:${filesize} ${fdt_addr_r};
    else
      bootz ${kernel_addr_r} - ${fdt_addr_r};
    fi;
  fi;
fi

if load ${devtype} ${devnum}:${bootpart} 0x48000000 /boot/uImage; then
  if load ${devtype} ${devnum}:${bootpart} 0x43000000 /boot/script.bin; then
    setenv bootm_boot_mode sec;
    bootm 0x48000000;
  fi;
fi
```

Le compiler et l'écrire sur la carte SD à l'aide du package **uboot-tools**.

```
mkimage -A arm -O linux -T script -C none -a 0 -e 0 -n "Script de démarrage
BananPI" -d boot.cmd mnt/boot/boot.scr
umount mnt
```

Compiler et copier le chargeur de démarrage U-Boot

L'étape suivante consiste à créer une image u-boot. Les outils arm-none-eabi-gcc, dtc, git, swig et uboot-tools doivent être installés sur le système. Cloner ensuite le code source de u-boot et compiler une image, en veillant à utiliser la configuration defconfig adéquate.

Board	**Defconfig**
M1	Bananapi_defconfig
M1+	bananapi_m1_plus_defconfig
M2	Sinovoip_BPI_M2_defconfig
M2 Berry	bananapi_m2_berry_defconfig
M2PLUS-H3	bananapi_m2_plus_h3_defconfig
M2PLUS-H5	bananapi_m2_plus_h5_defconfig
M2 Ultra	Bananapi_M2_Ultra_defconfig
M2 Magic	Bananapi_m2m_defconfig
M2 Zero	bananapi_m2_zero_defconfig
M3	Sinovoip_BPI_M3_defconfig
R1	Lamobo_R1_defconfig

Les cartes M2 Ultra et Magic ont des fichiers d'arborescence de périphériques disponibles. Ils, ainsi que d'autres defconfigs, peuvent être trouvés sur le U-Boot GitLab.

```
git clone git://git.denx.de/u-boot.git
```

```
cd u-boot
make -j4 ARCH=arm CROSS_COMPILE=arm-none-eabi- defconfig
make -j4 ARCH=arm CROSS_COMPILE=arm-none-eabi-
```

Dans le cas où l'erreur suivante apparaît lors de la compilation :

```
 Traceback (most recent call last):
 File "./tools/binman/binman", line 31, in <module>
   import control
 File "/u00/thomas/Downloads/bananapi/u-boot/tools/binman/control.py", line
15, in <module>
   import fdt
 File "/u00/thomas/Downloads/bananapi/u-boot/tools/binman/../dtoc/fdt.py",
line 13, in <module>
   import libfdt
 File "tools/libfdt.py", line 17, in <module>
   _libfdt = swig_import_helper()
 File "tools/libfdt.py", line 16, in swig_import_helper
   return importlib.import_module('_libfdt')
 File "/usr/lib/python2.7/importlib/__init__.py", line 37, in import_module
   __import__(name)
 ImportError: No module named _libfdt
 make: *** [Makefile:1149: u-boot-sunxi-with-spl.bin] Fehler 1
```

Il faut s'assurer que l'interpréteur python2 est utilisé. Pour forcer cela, on peut utiliser par exemple :

```
virtualenv -p /usr/bin/python2.7 my_uboot
source my_uboot/bin/activate
```

Puis compiler à nouveau comme ci-dessus.

Si tout s'est bien passé, on devrait avoir une image U-Boot : `u-boot-sunxi-with-spl.bin`. Maintenant, ajouter l'image sur la carte SD, où `/dev/sdX` est la carte SD.

```
dd if=u-boot-sunxi-with-spl.bin of=/dev/sdX bs=1024 seek=8
```

Pilote X.org

Le pilote X.org pour Banana Pi peut être installé avec le package xf86-video-fbdev.

Dépannage

Ethernet ne fonctionne pas

Dans certains cas, la connexion Ethernet Gbit est instable ou ne fonctionne pas correctement. Il peut

être utile de limiter la vitesse de la liaison à 100 Mbps en utilisant ethtool :

```
ethtool -s eth0 speed 100 duplex half autoneg off
```

L'écran s'éteint après l'inactivité et ne se rallume plus

Si vous rencontrez également un problème avec l'affichage qui s'éteint après un certain temps d'inactivité et ne se rallume pas, vous pouvez désactiver DPMS. Ajoutez donc ces arguments X11 à la bonne configuration de votre gestionnaire d'affichage.

```
-s 0 -dpms
```

Par exemple, pour utiliser SLiM, modifier `/etc/slim.conf`:

```
xserver_arguments -nolisten tcp vt07 -s 0 -dpms
```

Si, pour une raison quelconque, l'écran continue de s'éteindre, par ex. au redémarrage de votre appareil de réception, vous pouvez le rallumer, en modifiant temporairement la résolution :

```
# echo "D:1280x720p-60" > /sys/class/graphics/fb0/mode
# echo "D:1920x1080p-60" > /sys/class/graphics/fb0/mode
```

Scripte d'installation

Le scripte suivant permet de préparer une carte SD en exécutant l'ensemble des instructions ci-dessus

```


# détection de la carte SD
# souvent /dev/sd...
# ou
# /dev/mmcblk... !!!que la substitution du 1/2 ajouté à p1/p2!!!
sudo fdisk -l
export target_disc=/dev/...

# effacer la carte SD
sudo dd if=/dev/zero of=${target_disc} bs=1M count=8

# générer des partitions
sudo parted -s ${target_disc} mklabel msdos
sudo parted -a optimal -- ${target_disc} mkpart primary 2048s 150M
sudo parted -a optimal -- ${target_disc} mkpart primary 150M 100%

# formatage
```

```
sudo mkfs.ext4 -O ^metadata_csum,^64bit ${target_disc}1
sudo mkfs.ext4 -O ^metadata_csum,^64bit ${target_disc}2
export target_uuid1=$(sudo blkid ${target_disc}1 | awk '{print $2}')
export target_uuid2=$(sudo blkid ${target_disc}2 | awk '{print $2}')
export target_partuuid=$(sudo blkid ${target_disc}2 | awk '{print $4}')

# écrire les images
mkdir -p BananaM2Zero
cd BananaM2Zero
mkdir -p root
mkdir -p boot
sudo mount ${target_disc}1 boot
sudo mount ${target_disc}2 root
wget http://archlinuxarm.org/os/ArchLinuxARM-armv7-latest.tar.gz
sudo bsdtar -xpf ArchLinuxARM-armv7-latest.tar.gz -C root
sudo mv root/boot/* boot/.
echo "${target_uuid2}       /               ext4            rw,relatime
0 1" | sudo tee -a root/etc/fstab
echo "${target_uuid1}       /boot           ext4            rw,relatime
0 2" | sudo tee -a root/etc/fstab

# installation du système de démarrage
cat <<EOF > boot.cmd.sd-card
part uuid \${devtype} \${devnum}:\${bootpart} uuid
setenv bootargs console=\${console} root=${target_partuuid} rw rootwait

if load \${devtype} \${devnum}:\${bootpart} \${kernel_addr_r} zImage; then
  if load \${devtype} \${devnum}:\${bootpart} \${fdt_addr_r}
dtbs/\${fdtfile}; then
    if load \${devtype} \${devnum}:\${bootpart} \${ramdisk_addr_r}
initramfs-linux.img; then
      bootz \${kernel_addr_r} \${ramdisk_addr_r}:\${filesize}
\${fdt_addr_r};
    else
      bootz \${kernel_addr_r} - \${fdt_addr_r};
    fi;
  fi;
fi

if load \${devtype} \${devnum}:\${bootpart} 0x48000000 uImage; then
  if load \${devtype} \${devnum}:\${bootpart} 0x43000000 script.bin; then
    setenv bootm_boot_mode sec;
    bootm 0x48000000;
  fi;
fi
EOF

sudo mkimage -A arm -O linux -T script -C none -a 0 -e 0 -n "BananaM2Zero
boot script" -d boot.cmd.sd-card boot/boot.scr
sudo umount boot
sudo umount root
```

```
# installer u-boot

git clone git://git.denx.de/u-boot.git
cd u-boot
make -j4 ARCH=arm CROSS_COMPILE=arm-none-eabi- bananapi_m2_zero_defconfig
make -j4 ARCH=arm CROSS_COMPILE=arm-none-eabi-
sudo dd if=u-boot-sunxi-with-spl.bin of=${target_disc} bs=1024 seek=8 &&
sync
```

Finalisation de l'installation

- mettre la carte sd dans bananapi
- connecter hdmi, réseau, clavier
- connecter l'alimentation USB

Configuration de l'accès SSH

Se connecter à la console en tant qu'utilisateur **root** et mot de passe **root**.

Type	Nom d'utilisateur	Mot de passe
Root	root	root
User	alarm	alarm

exécuter maintenant ces commandes :

```
loadkeys fr
passwd root
pacman-key --init
pacman-key --populate archlinuxarm
pacman -ySu sudo uboot-tools dialog wpa_supplicant
useradd -m -g users -G wheel -s /bin/bash $username$
passwd $username$
userdel alarm
systemctl enable sshd.service
# notice ip-adresse for eth0
ip a
```

Configurer l'accès wifi

Des appareils comme le M2 Zero fournissent une puce pour le WiFi et le Bluetooth, qui nécessite le firmware brcm 43430. dmesg affichera cela:

```
brcmfmac mmc1:0001:1: Direct firmware load for brcm/brcmfmac43430-
sdio.sinovoip,bpi-m2-zero.txt failed with error -2
```

Le référentiel Arch Linux ARM fournit le package `firmware-raspberrypi`.

```
sudo wifi-menu
```

Redémarrer

```
reboot
```

La connexion SSH pour root est désactivée par défaut. Se connecter avec le compte utilisateur par défaut et utiliser su.

```
ssh $user_name$@$ipadresse$
sudo su
```

Configuration des locales

```
echo "fr_FR.UTF-8 UTF-8" >> /etc/locale.gen
echo "fr_FR ISO-8859-1" >> /etc/locale.gen
locale-gen
echo "LANG=fr_FR.UTF-8 UTF-8"" >> /etc/locale.conf
echo "LANGUAGE=fr_FR" >> /etc/locale.conf
echo "LC_ALL=C" >> /etc/locale.conf
echo "%wheel ALL=(ALL) ALL" >> /etc/sudoers
exit
reboot
```

Installation de python

```
sudo pacman --needed -S python python-pip
```

Installation des bibliothèques GPIO

Le package https://github.com/LeMaker/RPi.GPIO_BP fournit une classe pour contrôler le GPIO sur un BananaPi.

```
sudo pacman --needed -S gpio-utils python-periphery wiringx-git git base-
devel
git clone https://github.com/LeMaker/RPi.GPIO_BP -b bananapi
python setup.py install
sudo python setup.py install
```

Pour tester on va modifier l'état du GPIO(17) de LOW à HIGH et de HIGH à LOW, chaque seconde

```
cat <<EOF > test.py
import RPi.GPIO as GPIO
from time import sleep
GPIO.setmode(GPIO.BCM)
GPIO.setup(17, GPIO.OUT)
while True:
    GPIO.output(17, GPIO.HIGH)
    sleep(1)
    GPIO.output(17, GPIO.LOW)
    sleep(1)
EOF

sudo python test.py
```

[1)]

Un fichier **dtb** contient un **D**evice **T**ree **B**lob, permettant de transmettre des informations matérielles sur la carte au noyau Linux. Il peut être chargé en mémoire et transmis au noyau par u-Boot