

---

# MI11

## TP1 Linux Embarqué

---

Jeanneau Louis, Schulster Alex

Printemps 2022

## **Table des matières**

<b>1</b>	<b>Travail préalable à la séance de TP</b>	<b>2</b>
1.1	Carte Joy-Pi-Note . . . . .	2
1.2	Stocker le noyau et le système de fichiers . . . . .	2
<b>2</b>	<b>Prise en main de l'environnement Yocto</b>	<b>2</b>
2.1	Contenu du dossier /opt/mi11 . . . . .	2
2.2	Édition des fichiers de configuration . . . . .	3
2.3	Comparaison taille VM . . . . .	3
<b>3</b>	<b>Démarrer le noyau et le système de fichiers</b>	<b>4</b>
3.1	Démarrage de la cible . . . . .	4
3.2	Observation de la cible . . . . .	4
<b>4</b>	<b>Ajout de paquets</b>	<b>5</b>
4.1	Compilation . . . . .	5
4.2	Installation . . . . .	5
<b>5</b>	<b>Compilation manuelle du noyau</b>	<b>6</b>
5.1	Configuration par défaut du noyau . . . . .	6
5.2	Personnalisation du noyau . . . . .	6
5.3	Résultat . . . . .	6
<b>A</b>	<b>Log de démarrage</b>	<b>8</b>

# 1 Travail préalable à la séance de TP

## 1.1 Carte Joy-Pi-Note

Le Joy-Pi-Note est un kit d'apprentissage informatique constitué d'un Raspberry Pi 4, ainsi que d'un boîtier bardé de périphériques d'entrée-sortie (écran, interrupteurs, LEDs, moteurs, etc).

La carte possède un processeur **Broadcom BCM2711**, basé sur l'architecture ARM (64 bits) et est cadencé à 1.5 GHz. Le Joy-Pi-Note embarque la version avec 4 GB de RAM.

## 1.2 Stocker le noyau et le système de fichiers

Il existe deux moyens de stocker le noyau et le système de fichiers nécessaires pour démarrer la carte :

- Sur un support de stockage tel qu'une clé USB ou une carte SD ;
- Sur le réseau, stocké sur un autre ordinateur par exemple ;

Le stockage sur le réseau est le plus simple dans notre cas de développement sur le noyau, car nous pouvons directement modifier les fichiers localement depuis notre machine principale.

# 2 Prise en main de l'environnement Yocto

## 2.1 Contenu du dossier /opt/mi11

On observe sur la VM le contenu des dossiers. Les sous-dossiers **meta-raspberrypi**, **meta-joypinote** et **meta-mi11** contiennent pleins de fichiers qui permettent de paramétrer le build de Yocto avec les paramètres adaptés à notre configuration matérielle.

Le sous-dossier **poky/build/conf** contient quant à lui 3 fichiers textes de configuration de Yocto, comme illustré dans la Figure 1.

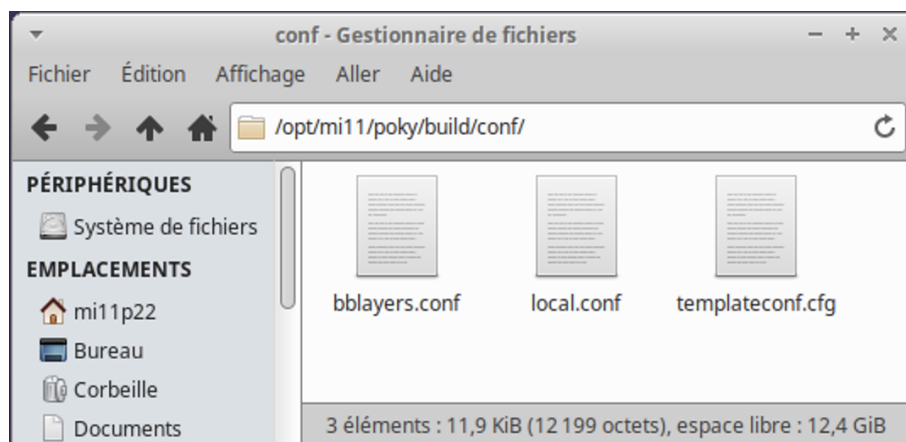
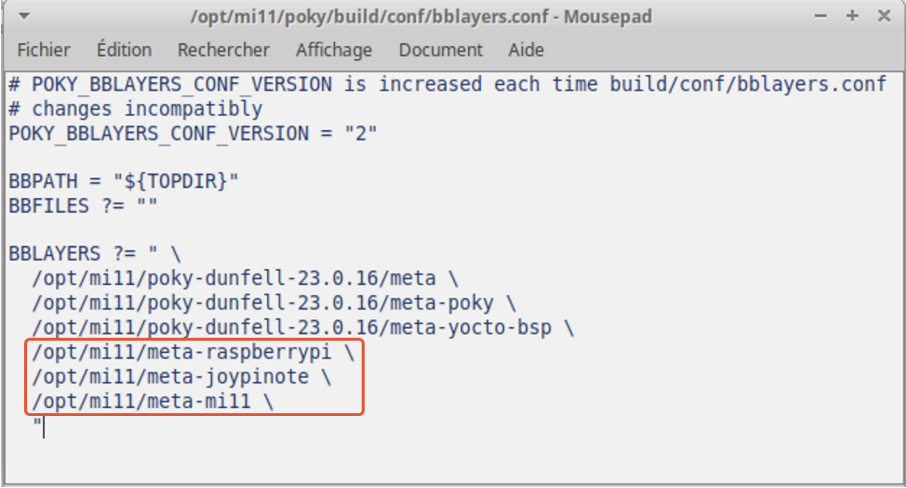


FIGURE 1 – Contenu du sous dossier poky/build/conf

## 2.2 Édition des fichiers de configuration

Dans le fichier `bblayers.conf`, nous avons ajouté l'import des 3 couches pour Yocto (Cf. Figure 2).



```

/opt/mi11/poky/build/conf/bblayers.conf - Mousepad
Fichier  Édition  Rechercher  Affichage  Document  Aide

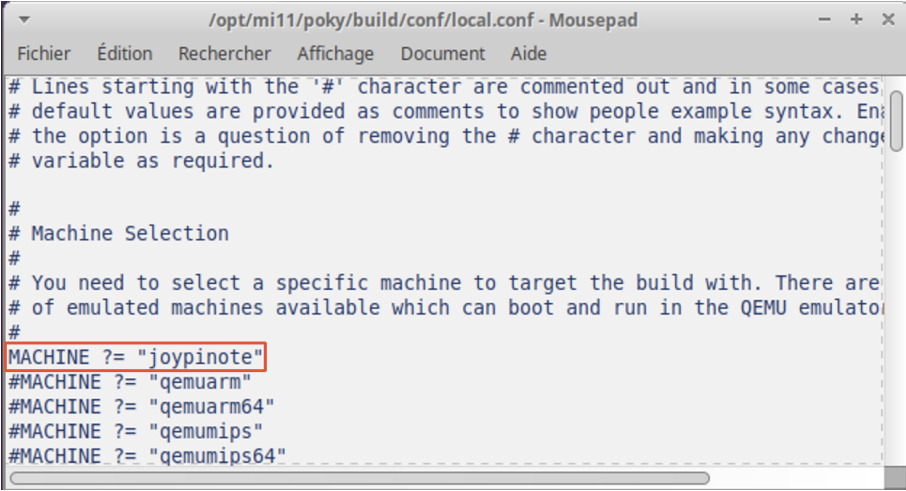
# POKY_BBLAYERS_CONF_VERSION is increased each time build/conf/bblayers.conf
# changes incompatibly
POKY_BBLAYERS_CONF_VERSION = "2"

BBPATH = "${TOPDIR}"
BBFILES ?= ""

BBLAYERS ?= " \
/opt/mi11/poky-dunfell-23.0.16/meta \
/opt/mi11/poky-dunfell-23.0.16/meta-poky \
/opt/mi11/poky-dunfell-23.0.16/meta-yocto-bsp \
/opt/mi11/meta-raspberrypi \
/opt/mi11/meta-joypinote \
/opt/mi11/meta-mill \
"
```

FIGURE 2 – Modification du fichier `bblayers.conf`

Dans `local.conf`, nous avons défini la variable `MACHINE='joypinote'` afin que la compilation de l'image soit adaptée à notre cible (Cf. Figure 3).



```

/opt/mi11/poky/build/conf/local.conf - Mousepad
Fichier  Édition  Rechercher  Affichage  Document  Aide

# Lines starting with the '#' character are commented out and in some cases,
# default values are provided as comments to show people example syntax. En
# the option is a question of removing the # character and making any change
# variable as required.

#
# Machine Selection
#
# You need to select a specific machine to target the build with. There are
# of emulated machines available which can boot and run in the QEMU emulator
#
MACHINE ?= "joypinote"
#MACHINE ?= "qemuarm"
#MACHINE ?= "qemuarm64"
#MACHINE ?= "qemumips"
#MACHINE ?= "qemumips64"
```

FIGURE 3 – Choix de la machine dans `local.conf`

## 2.3 Comparaison taille VM

La taille du build résultant est d'environ 26 Mo. C'est bien plus petit que la taille de la VM Ubuntu sur laquelle nous travaillons (~10 Go). Cette différence peut s'expliquer par le fait le noyau que nous venons de compiler comporte beaucoup moins de fonctionnalités qu'Ubuntu (qui seraient inutiles pour une application embarquée) tels que les éléments graphiques où encore tous les packages et driver préinstallés.

## 3 Démarrer le noyau et le système de fichiers

### 3.1 Démarrage de la cible

En essayant de démarrer la cible, nous remarquons qu'il se passe déjà un certain nombre de choses à l'écran : Beaucoup de services parviennent à être chargé. La machine semble cependant bloquer pour le chargement des fichiers que nous avons compilés précédemment dans ce TP. Pour que le Joy-Pi-Note arrive à les récupérer par le réseau, il faut les placer dans le répertoire `/tftpboot/serial_num/`, où `serial_num` est le numéro de série de notre carte (ici `fab61dcd`).

Une fois ces fichiers bien placés, le Joy-Pi-Note parvient à les charger correctement, mais bute maintenant sur le chargement de l'OS. Il faut pour cela placer l'image de l'OS compilée à l'emplacement `/tftpboot/rootfs`.

La carte est maintenant capable de démarrer correctement avec un OS complet et fonctionnel. (Cf. Annexe A pour l'ensemble des logs de démarrage)

### 3.2 Observation de la cible

Afin de communiquer plus facilement avec la cible, nous décidons de communiquer par SSH plutôt que par le port série. Il nous faut pour cela récupérer son adresse IP. Il nous suffit de taper la commande `ip -a` dans le terminal série pour obtenir cette dernière : Ici, notre carte a pour adresse IP `192.168.0.122`. Nous pouvons maintenant nous connecter en SSH avec la commande `ssh 192.168.0.122@root`.

On décide d'observer et d'analyser le contenu de certains répertoires spécifiques. On commence avec `/proc/devices` qui est séparé en deux sections :

- *character* qui liste tous les éléments communiquant caractère par caractère (tel que le terminal, les ports séries...);
- *block* qui contient les éléments de stockage par bloc comme un périphérique USB;

Ci-dessous le résultat obtenu dans le terminal :

```

1 root@joypinote:/proc# cat devices
2 Character devices:
3   1 mem
4   4 /dev/vc/0
5   4 tty
6   4 ttyS
7   5 /dev/tty
8   5 /dev/console
9   5 /dev/ptmx
10  5 ttyprintk
11  7 vcs
12 10 misc
13 13 input
14 29 fb
15 128 ptm
16 136 pts
17 153 spi
18 162 raw
19 180 usb
20 189 usb_device
21 204 ttyAMA
22 226 drm
23 245 vchiq
24 246 vcs
25 247 bcm2835-gpiomem
26 248 vcio
27 249 vc-mem
28 250 bsg
29 251 watchdog
30 252 iio
31 253 rtc
32 254 gpiochip
33
34 Block devices:
35 1 ramdisk
36 7 loop
37 8 sd
38 65 sd
39 66 sd
40 67 sd
41 68 sd
42 69 sd
43 70 sd
44 71 sd
45 128 sd
46 129 sd
47 130 sd
48 131 sd
49 132 sd
50 133 sd
51 134 sd
52 135 sd
53 259 blkext
```

On se penche maintenant sur les répertoires contenant les fichiers binaires compilés : `/bin`, `/sbin`, `/usr/bin` et `/usr/sbin`. On remarque que beaucoup d'éléments sont des liens vers *busybox*, qui permet de rassembler un grand nombre d'utilitaires Linux en un unique fichier exécutable.

## 4 Ajout de paquets

### 4.1 Compilation

On utilise la commande `bitbake nano` pour compiler le paquet `nano`. Les objets compilés sont regroupés dans le répertoire `/opt/mi11/poky/build/tmp/deploy/ipk`. Celui-ci est structuré en sous-répertoire (un par cible). Ainsi, le paquet `nano` se trouve dans le sous-répertoire `cortexa7t2hf-neon-vfpv4/` (qui correspond au processeur présent sur le Raspbessy Pie). Le fichier compilé est `nano_2.2.5-r3.0_cortexa7t2hf-neon-vfpv4.ipk`.

Le noyau quant à lui se trouve dans le sous-répertoire `/joypinote` car il est spécifique à l'ensemble du système (et pas seulement au processeur).

On remarque que le paquet `nano` contient différents composants :

- `dev` ;
- `dbg` (debug) ;
- `doc` ;
- différentes versions pour proposer plusieurs langues ;

### 4.2 Installation

On déplace le fichier `nano` compilé sur le système de fichier de la cible. Ensuite, directement sur la cible (via SSH par exemple) on tente d'installer `nano` avec le gestionnaire de paquet de la cible avec la commande `opkg install paquet_compilé.ipk`. On remarque cependant que l'installation échoue, car le gestionnaire de paquets ne parvient pas à utiliser les dépendances de `nano`. Plusieurs solutions s'offrent alors à nous :

- Ne pas installer `nano` (solution proposée par `opkg`), radicale, mais efficace ;
- Compiler puis installer une à une les dépendances à la main, mais ce processus serait long et fastidieux (sans compter le fait que les dépendances peuvent elles-mêmes avoir des dépendances) ;
- Se connecter à un serveur contenant tous les paquets, nécessitant l'accès à internet ;

Nous choisissons la troisième option pour des raisons pratiques évidentes. Nous modifions donc le fichier de configuration de `opkg` situé à l'adresse `/etc/opkg/opkg.conf`. Nous ajoutons les deux lignes suivantes :

```
src/gz cortexa7t2hf-neon-vfpv4 http://192.168.0.1/cortexa7t2hf-neon-vfpv4
src/gz joypinote http://192.168.0.1/joypinote
```

Nous pouvons maintenant mettre à jour `opkg` et installer `nano` depuis le serveur avec les commandes :

```
opkg update
opkg install nano
```

## 5 Compilation manuelle du noyau

### 5.1 Configuration par défaut du noyau

On souhaite faire clignoter une LED sur la carte. On tente pour cela de la contrôler en modifiant le fichier `/sys/class/leds/<nom_led>/brightness` à l'aide de la commande suivante :

```
echo 1 > /sys/class/leds/nom_led/brightness
```

Cette commande n'aboutit pas, car les LEDs n'ont pas été activées dans le noyau (le dossier `/sys/class/leds` ne contient rien). Nous allons donc le recompiler manuellement en changeant quelques paramètres. Il nous faut tout d'abord activer la chaîne de compilation croisée avec la commande fournie dans le sujet du TP.

Ce fichier sert à définir à certain nombre de variables qui permettront au `Makefile` de compiler un fichier image adapté à l'architecture de la cible. On remarque par ailleurs que le préfixe du fichier à charger est `cortexa7t2hf`, soit le modèle du processeur du Raspberry Pi 3.

Afin d'obtenir la liste des configurations par défaut de compilation du noyau, on effectue la commande `make help` dans le répertoire `/opt/mi11/linux-raspberrypi`. On observe alors qu'il existe une configuration par défaut pour le Joy-Pi-Note, appelée `joypinote_def_config`. On lance la compilation du noyau avec cette configuration avec la commande `make joypinote_def_config`.

### 5.2 Personnalisation du noyau

Ensuite, on lance la personnalisation du noyau (`make menuconfig`) afin d'activer les options nécessaires pour pouvoir contrôler les LEDs. Le menu illustré en Figure 4 apparaît alors à l'écran.

Nous activons donc l'option `Device Drivers`  $\Rightarrow$  `LED Support`  $\Rightarrow$  `LED support for GPIO connected`. On peut soit l'activer de façon permanente, soit en tant que module si l'on souhaite pouvoir l'activer / désactiver plus facilement plus tard pour des raisons d'optimisation. Ici, nous l'activerons de façon permanente avec l'option `Y`.

Ensuite, on ajoute dans le `device tree` la définition des LED 5 et 6 de la carte selon la méthode indiquée dans le sujet, puis on lance la compilation.

### 5.3 Résultat

Le résultat de notre compilation se trouve dans le sous-dossier `arch/arm/boot/`. On observe en particulier le *fichier image* `zImage` et le *device tree* `dts` (Cf. Figure 5).

On place alors l'image et le device tree dans le dossier `serial_num/fab61dcd/`. Pour s'assurer que le Raspberry Pi utilise bien cette nouvelle image, on regarde lors du démarrage de celui-ci la date de compilation du noyau utilisé. Dans notre cas, le premier noyau était compilé aux alentours de 9 h, contre 11 h pour celui personnalisé. On a ainsi pu vérifier le bon chargement de notre nouveau noyau.

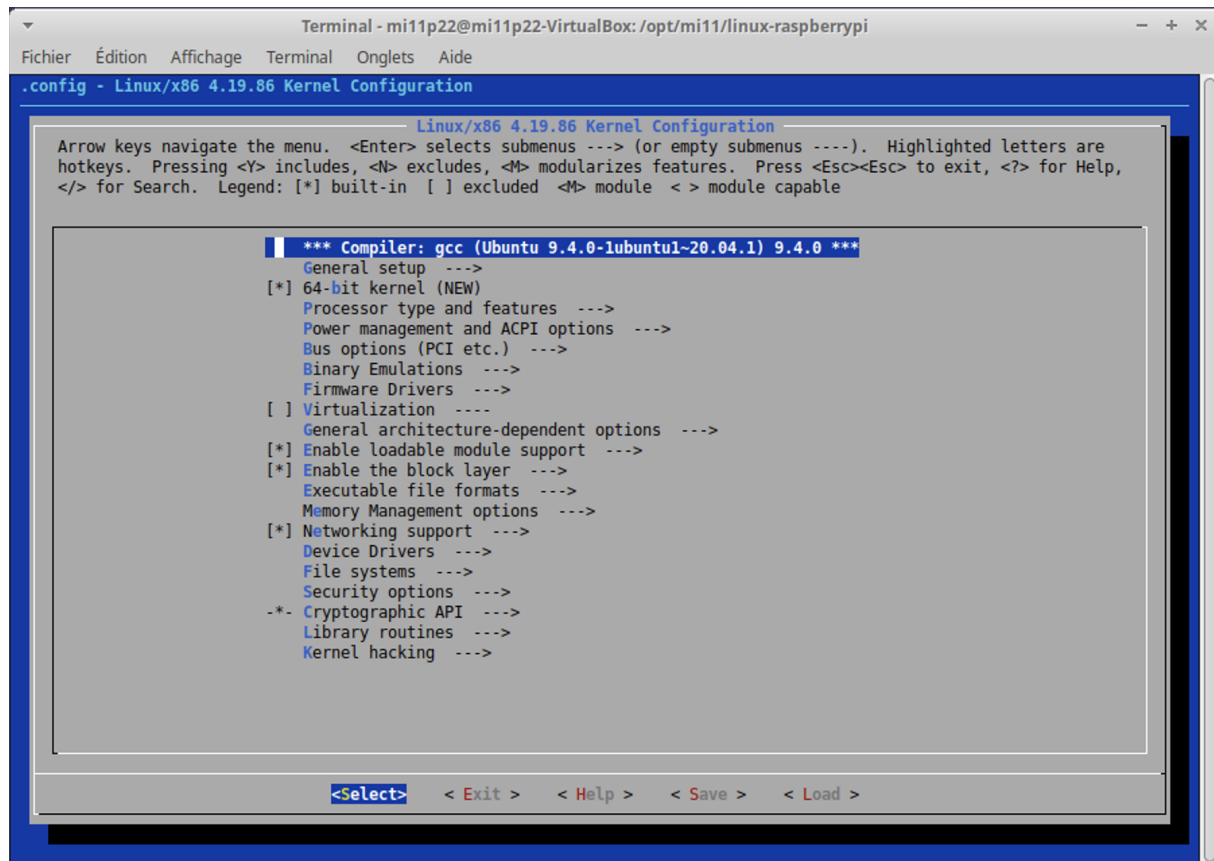


FIGURE 4 – Menu de configuration du noyau

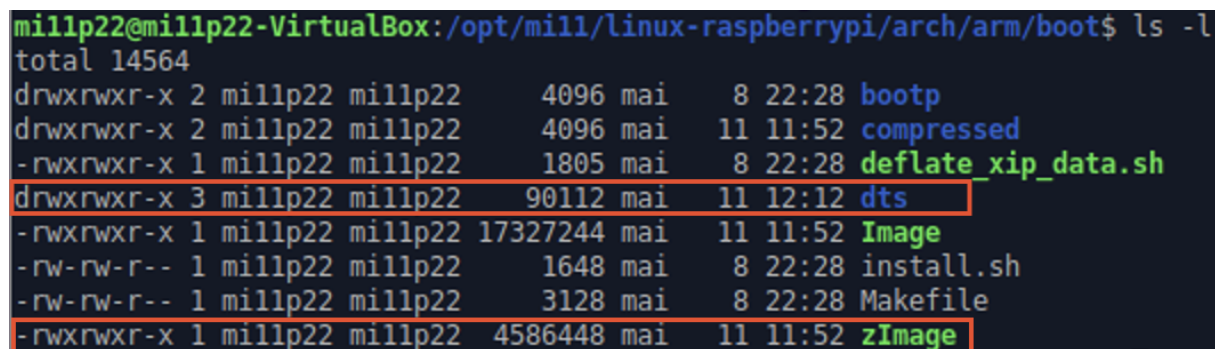


FIGURE 5 – Résultat de la compilation du noyau

On peut maintenant effectuer de nouveau la commande d'activation des LEDs :

```
echo 1 > /sys/class/leds/led_5/brightness
```

Et on constate que la LED sur la carte s'allume en réaction à cette commande.

Afin d'ajouter un *trigger* sur une LED, il suffit d'ajouter le nom du trigger dans le fichier trigger de cette LED avec la commande :

```
echo <nom_du_trigger> > /sys/class/leds/<nom_led>/trigger
```



# A Log de démarrage

```
1 RPi: BOOTLOADER release VERSION:507b2360 DATE: 2022/04/26 TIME: 11:24:28 BOOTMODE:
   0x00000006 part: 0 BUILD_TIMESTAMP=1650968668 0xfab61dcd 0x00b03114 0x0005c63a
2 PM_RSTS: 0x00001000
3 part 00000000 reset_info 00000000
4 uSD voltage 3.3V
5 Initialising SDRAM 'Samsung' 16Gb x1 total--size: 16 Gbit 3200
6 DDR 3200 0 0 16 152
7
8 XHCI--STOP
9 xHC ver: 256 HCS: 05000420 fc000031 00e70004 HCC: 002841eb
10 USBSTS 11
11 xHC ver: 256 HCS: 05000420 fc000031 00e70004 HCC: 002841eb
12 xHC ports 5 slots 32 intrs 4
13 Boot mode: SD (01) order f2
14 USB2[1] 400202e1 connected
15 USB2 root HUB port 1 init
16 DEV [01:00] 2.16 000000:01 class 9 VID 2109 PID 3431
17 HUB init [01:00] 2.16 000000:01
18 SD HOST: 250000000 CTL0: 0x00000000 BUS: 200000 Hz actual: 200000 HZ div: 1250 (625) status:
   0x1fff0000 delay: 540
19 SD HOST: 250000000 CTL0: 0x00000f00 BUS: 200000 Hz actual: 200000 HZ div: 1250 (625) status:
   0x1fff0000 delay: 540
20 EMMC
21 SD retry 1 oc 0
22 SD HOST: 250000000 CTL0: 0x00000000 BUS: 200000 Hz actual: 200000 HZ div: 1250 (625) status:
   0x1fff0000 delay: 540
23 SD retry 2 oc 0
24 SD HOST: 250000000 CTL0: 0x00000000 BUS: 200000 Hz actual: 200000 HZ div: 1250 (625) status:
   0x1fff0000 delay: 540
25 SDV1
26 SD retry 3 oc 0
27 SD HOST: 250000000 CTL0: 0x00000000 BUS: 200000 Hz actual: 200000 HZ div: 1250 (625) status:
   0x1fff0000 delay: 540
28 SD CMD: 0x371a0010 (55) 0x0 0x1fff0001
29 Failed to open device: 'sdcard' (cmd 371a0010 status 1fff0001)
30 Retry SD 1
31 SD HOST: 250000000 CTL0: 0x00000000 BUS: 200000 Hz actual: 200000 HZ div: 1250 (625) status:
   0x1fff0000 delay: 540
32 SD HOST: 250000000 CTL0: 0x00000f00 BUS: 200000 Hz actual: 200000 HZ div: 1250 (625) status:
   0x1fff0000 delay: 540
33 EMMC
34 SD retry 1 oc 0
35 SD HOST: 250000000 CTL0: 0x00000000 BUS: 200000 Hz actual: 200000 HZ div: 1250 (625) status:
   0x1fff0000 delay: 540
36 SD retry 2 oc 0
37 SD HOST: 250000000 CTL0: 0x00000000 BUS: 200000 Hz actual: 200000 HZ div: 1250 (625) status:
   0x1fff0000 delay: 540
38 SDV1
39 SD retry 3 oc 0
40 SD HOST: 250000000 CTL0: 0x00000000 BUS: 200000 Hz actual: 200000 HZ div: 1250 (625) status:
   0x1fff0000 delay: 540
41 SD CMD: 0x371a0010 (55) 0x0 0x1fff0001
42 Failed to open device: 'sdcard' (cmd 371a0010 status 1fff0001)
43 Boot mode: NETWORK (02) order f
44 GENET: RESET_PHY 0
45 CTL 1140 PHY ID 600d 84a2
46 MII_CONTROL 1140
47 MII APD 0001 SCTLS 0001
48
49 NET_BOOT: e4:5f:01:07:de:02 wait for link TFTP: 0.0.0.0
50 LINK STATUS: speed: 100 full duplex
51 Link ready
52 GENET START: 64 16 32
53 GENET: UMAC_START 0xe45f0107 0xde020000
54 RX: 0 IP: 0 IPV4: 0 MAC: 0 UDP: 0 UDP REC: 0 IP_CSUM_ERR: 0 UDP_CSUM_ERR: 0
55 DHCP src: 08:00:27:2a:be:09 192.168.0.1
56 YI_ADDR 192.168.0.122
57 OPTIONS:--
58 op: 53 len: 1 DHCP recv OFFER (2) expect OFFER
59 op: 54 len: 4 192.168.0.1
60 op: 51 len: 4
61 op: 1 len: 4 255.255.255.0
62 op: 66 len: 11 192.168.0.1[66]: 192.168.0.1
63
64 NET 192.168.0.122 255.255.255.0 gw 0.0.0.0 tftp 192.168.0.1
65 ARP 192.168.0.1 08:00:27:2a:be:09
66 NET 192.168.0.122 255.255.255.0 gw 0.0.0.0 tftp 192.168.0.1
67 RX: 2 IP: 0 IPV4: 1 MAC: 1 UDP: 1 UDP REC: 1 IP_CSUM_ERR: 0 UDP_CSUM_ERR: 0
68 DHCP src: 08:00:27:2a:be:09 192.168.0.1
69 YI_ADDR 192.168.0.122
70 OPTIONS:--
71 op: 53 len: 1 DHCP recv ACK (5) expect ACK
72 op: 54 len: 4 192.168.0.1
73 op: 51 len: 4
74 op: 1 len: 4 255.255.255.0
75 op: 66 len: 11 192.168.0.1[66]: 192.168.0.1
76
77 TFTP_GET: 08:00:27:2a:be:09 192.168.0.1 fab61dcd/start4.elf
78
79 RX: 3 IP: 0 IPV4: 2 MAC: 2 UDP: 2 UDP REC: 2 IP_CSUM_ERR: 0 UDP_CSUM_ERR: 0
80 RX: 4 IP: 0 IPV4: 3 MAC: 3 UDP: 3 UDP REC: 3 IP_CSUM_ERR: 0 UDP_CSUM_ERR: 0
81 TFTP_GET: 08:00:27:2a:be:09 192.168.0.1 fab61dcd/config.txt
82
83 RX: 4 IP: 0 IPV4: 3 MAC: 3 UDP: 3 UDP REC: 3 IP_CSUM_ERR: 0 UDP_CSUM_ERR: 0
84 TFTP: complete 87
85 RX: 6 IP: 0 IPV4: 5 MAC: 5 UDP: 5 UDP REC: 5 IP_CSUM_ERR: 0 UDP_CSUM_ERR: 0
86 Read fab61dcd/config.txt bytes 87 hnd 0x0
87 TFTP_GET: 08:00:27:2a:be:09 192.168.0.1 fab61dcd/pieeprom.sig
88
89 RX: 6 IP: 0 IPV4: 5 MAC: 5 UDP: 5 UDP REC: 5 IP_CSUM_ERR: 0 UDP_CSUM_ERR: 0
90 TFTP 1: File not found
91 RX: 7 IP: 0 IPV4: 6 MAC: 6 UDP: 6 UDP REC: 6 IP_CSUM_ERR: 0 UDP_CSUM_ERR: 0
92 Read fab61dcd/pieeprom.upd failed
93 TFTP_GET: 08:00:27:2a:be:09 192.168.0.1 fab61dcd/recover4.elf
94
95 RX: 7 IP: 0 IPV4: 6 MAC: 6 UDP: 6 UDP REC: 6 IP_CSUM_ERR: 0 UDP_CSUM_ERR: 0
96 TFTP 1: File not found
97 RX: 8 IP: 0 IPV4: 7 MAC: 7 UDP: 7 UDP REC: 7 IP_CSUM_ERR: 0 UDP_CSUM_ERR: 0
98 Read fab61dcd/recover4.elf failed
99 TFTP_GET: 08:00:27:2a:be:09 192.168.0.1 fab61dcd/recovery.elf
100
101 RX: 8 IP: 0 IPV4: 7 MAC: 7 UDP: 7 UDP REC: 7 IP_CSUM_ERR: 0 UDP_CSUM_ERR: 0
102 TFTP 1: File not found
103 RX: 9 IP: 0 IPV4: 8 MAC: 8 UDP: 8 UDP REC: 8 IP_CSUM_ERR: 0 UDP_CSUM_ERR: 0
104 Read fab61dcd/recovery.elf failed
105 TFTP_GET: 08:00:27:2a:be:09 192.168.0.1 fab61dcd/start4.elf
106
107 RX: 9 IP: 0 IPV4: 8 MAC: 8 UDP: 8 UDP REC: 8 IP_CSUM_ERR: 0 UDP_CSUM_ERR: 0
108 HDMIO edid block 0 offset 0
109 00 ffffffff004a8b3b2a01010101
110 17150103803c2278ealec5ae4f34b126
111 0e5054a54b008180a940d1c0714f0101
112 0101010101011a36801871382d40582c
113 450055502100001e000000ff004a3235
114 374d3936423030464c0a000000fc0052
115 544b204648440a2020202020000000fd
116 00384c1e5111000a20202020202001a7
117 HDMIO edid block 1 offset 128
118 020329f15090050403020716011f1213
119 14201511062309070767030c00100038
120 2d83010000e30503011a36801871382d
121 40582c450006442100001e0111d801871
122 1c1620582c250006442100009e011d00
123 7251d01e206e28550006442100001e8c
124 0ad08a2Ge02d10103e96000644210000
```

[illegible]

```

259 [ 0.134591] brcm-pcie fd500000.pcie: could not get clock
260 [ 0.134654] brcm-pcie fd500000.pcie: host bridge /scb@7d500000 ranges:
261 [ 0.134699] brcm-pcie fd500000.pcie: MEM 0x600000000-0x603fffff -> 0xf8000000
262 [ 0.191063] brcm-pcie fd500000.pcie: link up, 5.0 Gbps x1 (!SSC)
263 [ 0.191271] brcm-pcie fd500000.pcie: PCI host bridge to bus 0000:00
264 [ 0.191299] pci_bus 0000:00: root bus resource [bus 00-01]
265 [ 0.191326] pci_bus 0000:00: root bus resource [mem 0x600000000-0x603fffff] (bus address
    [0xf8000000-0xfbffff])
266 [ 0.194235] PCI: bus0: Fast back to back transfers disabled
267 [ 0.194263] pci 0000:00:00.0: bridge configuration invalid ([bus 00-00]), reconfiguring
268 [ 0.197566] PCI: bus1: Fast back to back transfers disabled
269 [ 0.197632] pci 0000:00:00.0: BAR 8: assigned [mem 0x600000000-0x6000ffff]
270 [ 0.197663] pci 0000:01:00.0: BAR 0: assigned [mem 0x600000000-0x60000fff 64bit]
271 [ 0.197735] pci 0000:00:00.0: PCI bridge to [bus 01]
272 [ 0.197761] pci 0000:00:00.0: bridge window [mem 0x600000000-0x6000ffff]
273 [ 0.197939] pcieport 0000:00:00.0: enabling device (0140 -> 0142)
274 [ 0.198076] pcieport 0000:00:00.0: Signaling PME with IRQ 54
275 [ 0.198216] pcieport 0000:00:00.0: AER enabled with IRQ 54
276 [ 0.198620] bcm2708_fb soc:fb: FB found 1 display(s)
277 [ 0.233669] Console: switching to colour frame buffer device 240x67
278 [ 0.244039] bcm2708_fb soc:fb: Registered framebuffer for display 0, size 1920x1080
279 [ 0.245802] Serial: 8250/16550 driver, 1 ports, IRQ sharing enabled
280 [ 0.247436] iproc-rng200 fe104000.rng: hwrng registered
281 [ 0.247679] vc-mem: phys_addr:0x00000000 mem_base=0x0ec00000 mem_size:0x10000000(256 MiB)
282 [ 0.248147] vc-sm: Videocore shared memory driver
283 [ 0.248507] gpptomem-bcm2835 fe200000.gpptomem: Initialised: Registers at 0xfe200000
284 [ 0.255440] brd: module loaded
285 [ 0.261355] loop: module loaded
286 [ 0.262319] spi-bcm2835 fe204000.spi: could not get clk: -517
287 [ 0.263579] libphy: Fixed MDIO Bus: probed
288 [ 0.263971] bcmgenet fd580000.genet: failed to getenet clock
289 [ 0.264043] bcmgenet fd580000.genet: GENET 5.0 EPHY: 0x0000
290 [ 0.264111] bcmgenet fd580000.genet: failed to getenet-wol clock
291 [ 0.264182] bcmgenet fd580000.genet: failed to getenet-eee clock
292 [ 0.264261] bcmgenet: Skipping UMAC reset
293 [ 0.264505] unimac-mdio unimac-mdio.-19: DMA mask not set
294 [ 0.281049] libphy: bcmgenet MII bus: probed
295 [ 0.321464] unimac-mdio unimac-mdio.-19: Broadcom UniMAC MDIO bus at 0x(ptrval)
296 [ 0.322442] usbcore: registered new interface driver uas
297 [ 0.322543] usbcore: registered new interface driver usb-storage
298 [ 0.322740] mousedev: PS/2 mouse device common for all mice
299 [ 0.324355] bcm2835-wdt bcm2835-wdt: Broadcom BCM2835 watchdog timer
300 [ 0.325180] ledtrig-cpu: registered to indicate activity on CPUs
301 [ 0.325869] vchiq: vchiq_init_state: slot_zero = (ptrval), is_master = 0
302 [ 0.327053] [vc_sm_connected_init]: start
303 [ 0.333203] [vc_sm_connected_init]: end - returning 0
304 [ 0.334139] Initializing XFRM netlink socket
305 [ 0.334209] NET: Registered protocol family 17
306 [ 0.334315] Key type dns_resolver registered
307 [ 0.335065] registered taskstats version 1
308 [ 0.335123] Loading compiled-in X.509 certificates
309 [ 0.340343] uart-pl011 fe201000.serial: cts_event_workaround enabled
310 [ 0.342980] fe201000.serial: ttyAMA0 at MMIO 0xfe201000 (irq = 34, base_baud = 0) is a PL011 rev2
311 [ 0.348358] console [ttyS0] disabled
312 [ 0.350922] fe215040.serial: ttyS0 at MMIO 0x0 (irq = 37, base_baud = 62500000) is a 16550
313 [ 1.847177] console [ttyS0] enabled
314 [ 1.853645] bcm2835-power bcm2835-power: Broadcom BCM2835 power domains driver
315 [ 1.864627] mcp320x spi0.1: spi0.1 supply vref not found, using dummy regulator
316 [ 1.874736] mcp320x spi0.1: Linked as a consumer to regulator.0
317 [ 1.884327] input: joyypinote_keypad as /devices/platform/joyypinote_keypad/input/input0
318 [ 1.895562] input: joyypinote_joystick as /devices/platform/joyypinote_joystick/input/input1
319 [ 1.908575] brcmstb_thermal fd5d2200.thermal: registered AVS TMON of-sensor driver
320 [ 2.138843] random: fast init done
321 [ 2.233512] of_cfs_init
322 [ 2.238736] of_cfs_init: OK
323 [ 2.244688] bcmgenet: Skipping UMAC reset
324 [ 2.351546] bcmgenet fd580000.genet: configuring instance for external RGMII (no delay)
325 [ 3.441163] bcmgenet fd580000.genet eth0: Link is Down
326 [ 5.521217] bcmgenet fd580000.genet eth0: Link is Up - 100Mbps/Full - flow control rx/tx
327 [ 5.571038] Sending DHCP requests., OK
328 [ 5.637597] IP-Config: Got DHCP answer from 192.168.0.1, my address is 192.168.0.122
329 [ 5.648158] IP-Config: Complete:
330 [ 5.654092] device=eth0, hwaddr=e4:5f:01:07:de:02, ipaddr=192.168.0.122, mask=255.255.255.0,
    gw=255.255.255.255
331 [ 5.667507] host=192.168.0.122, domain=, nis-domain=(none)
332 [ 5.676276] bootserver=0.0.0.0, rootserver=192.168.0.1, rootpath=
333 [ 5.676285] nameserver0=192.168.0.1
334 [ 5.702759] VFS: Mounted root (nfs filesystem) on device 0:15.
335 [ 5.712033] devtmpfs: mounted
336 [ 5.723105] Freeing unused kernel memory: 2048K
337 [ 5.730667] Run /sbin/init as init process
338 [ 6.784746] udevd[90]: starting version 3.2.9
339 [ 6.814932] random: udevd: uninitialized urandom read (16 bytes read)
340 [ 6.823178] random: udevd: uninitialized urandom read (16 bytes read)
341 [ 6.834218] random: udevd: uninitialized urandom read (16 bytes read)
342 [ 6.912344] udevd[91]: starting eudev-3.2.9
343 [ 7.141663] vc_sm_cma: module is from the staging directory, the quality is unknown, you have been
    warned.
344 [ 7.192307] bcm2835_vc_sm_cma_probe: Videocore shared memory driver
345 [ 7.198677] [vc_sm_connected_init]: start
346 [ 7.272262] [vc_sm_connected_init]: installed successfully
347 [ Fri Mar 9 12:34:56 UTC 2018
348 [ 8.564597] urandom_read: 1 callbacks suppressed
349 [ 8.564604] random: dd: uninitialized urandom read (512 bytes read)
350 [ ALSA: Restoring mixer settings...
351 [ /usr/sbin/alsactl: load_state:1735: No soundcards found...
352 [ INIT: Entering runlevel: 5
353 [ Configuring network interfaces... ip: RTNETLINK answers: File exists
354 [ ifup skipped for nfsroot interface eth0
355 [ run-parts: /etc/network/if-pre-up.d/nfsroot: exit status 1
356 [ Starting system message bus: [ 9.213016] random: dbus-daemon: uninitialized urandom read (12 bytes
    read)
357 [ 9.246108] random: dbus-daemon: uninitialized urandom read (12 bytes read)
358 [ dbus.
359 [ Starting Dropbear SSH server: dropbear.
360 [ Starting rpcbind daemon...done.
361 [ Starting advanced power management daemon: No APM support in kernel
362 [ (failed.)
363 [ Starting bluetooth: bluetoothd.
364 [ Starting syslogd/klogd: done
365 [ 9.790049] random: crng init done
366 [ 9.793505] random: 1 urandom warning(s) missed due to ratelimiting
367 [ * Starting Avahi mDNS/DNS-SD Daemon: avahi-daemon
368 [ ... done.
369 [ Starting Telephony daemon
370 [ Starting Linux NFC daemon
371 [
372 [ Poky (Yocto Project Reference Distro) 3.1.16 joyypinote /dev/ttyS0
373 [
374 [ joyypinote login:

```