RPi: Accélération matérielle et GPU.

Table of Contents

Raspberry est un ordinateur monocarte, compact, puissant et peu coûteux , disposant d'un processeur Cortex-A7 quadricœur fonctionnant à 900 MHz et de 1 Go de RAM, d'une sortie HDMI, etc.

Principalement utilisé avec Java, C et les applications de traitement, qui fonctionnent à bas niveau et sont capables de contrôler les ressources matérielles, mais pour exécuter une application Web, on dépend d'un système d'exploitation qui limite ces ressources. Pour obtenir quelque chose de plus performant et tirer parti de certaines autres ressources, telles que le GPU, il faut créer une distribution personnalisée.

Construction distribution Linux

Habituellement, on installe Raspbian et on utilise simplement le navigateur disponible fourni avec le système d'exploitation, qu'il s'agisse d'Epiphany ou de Midori. Avec ces navigateurs fonctionnant sur ce système d'exploitation, on peut naviguer sans trop de tracas, mais sans les animations et les performances:

- La mémoire est partagée entre le système et le GPU. On a 1 Go, mais il est partagé entre ces deux composants.
- Les navigateurs ne peuvent pas utiliser nativement l'accélération matérielle du GPU pour traiter les animations.

Un hack connu consiste à utiliser QT pour forcer l'utilisation du GPU. La compilation de WebKit dans QT donne accès à cette ressource, mais pour y parvenir, il faut créer une distribution Linux propre et abandonner tout ce qui n'est pas essentiel pour garder la RAM disponible pour l'application.

Construction avec Buildroot

La compilation croisée et la configuration du noyau sont courants que dans le monde Linux ; des

choses vraiment de bas niveau par rapport à l'environnement Web. Mais il existe des outils pour aider, Buildroot en fait partie : un outil simple, efficace et facile à utiliser qui génère des systèmes Linux embarqués par compilation croisée.

Même avec Buildroot, il est important de comprendre et de connaître les dépendances requises par chaque bibliothèque que l'on souhaite compiler.

Le référentiel de Metrological permet les meilleures configurations de performances pour Buildroot afin d'exécuter la bibliothèque QT et le moteur WebKit dans Raspberry.

Il suffit de cloner le dépôt :

```
git clone https://github.com/Metrological/buildroot
cd buildroot
```

Ensuite, appliquer la configuration de base pour RPI Model 2 :

```
make rpi2_qt5webkit_defconfig
```

Ou pour accéder au menu Buildroot et voir les autres bibliothèques disponibles, utiliser :

```
make menuconfig
```

Dans ce référentiel, on trouve un fichier `.config` de base avec toutes les bibliothèques considérées comme essentielles pour exécuter une application Web, telles que git, fbv, websocket et python.

Maintenant, copier le fichier mentionné dans le répertoire Buildroot et exécuter :

```
make
```

Le processus prend un certain temps.

Installation sur la carte SD

On a maintenant une image prête à être clonée dans la SDCard. Utiliser l'une des méthodes décrites ici pour graver l'image sur la carte SD. Lorsque la carte est prête, on peut démarrer le RPI et se connecter au système via SSH (Login : root, Mot de passe : root).

```
ssh root@192.168.1.100 # Remplacer par l'adresse IP de RPI !
```

Ensuite, lancer l'application :

```
qtbrowser --url=http://url
```

La distribution qui en résulte est vraiment propre. On n'y trouvera pas d'éléments Debian classiques, comme apt-get. Certaines fonctionnalités dont on peut avoir besoin doivent être installées manuellement.

Configuration

Fichiers de boot

Placer le contenu suivant dans `/boot/cmdline.txt`. Cela empêchera le journal de démarrage d'apparaître:

```
dwc_otg.fiq_fix_enable=1 sdhci-bcm2708.sync_after_dma=0 dwc_otg.lpm_enable=0
vt.global_cursor_default=0 console=tty3 root=/dev/mmcblk0p2 rootwait
loglevel=3 quiet
```

Dans le fichier `/boot/config.txt`, il faut au minimum définir les propriétés utilisées par ce projet :

- **gpu_mem_1024=512**[1)]: mémoire vidéo
- **hdmi_group=1**[2)]: force le mode tv
- **hdmi_mode=16**: pour utiliser le Full HD 60p
- **hdmi_force_hotplug=1**: force la sortie HDMI

```
disable_splash=1
disable_overscan=1
boot_delay=0
arm_freq=1000
gpu_freq=500
over_voltage=6
avoid_warnings=1
force_turbo=0
gpu_mem_256=128
gpu_mem_512=256
gpu_mem_1024=512
kernel=zImage
hdmi_group=1
hdmi_mode=16
hdmi_force_hotplug=1
```

Liste complète des résolutions :

HDMI mode	**VGA**[3)]			**DVI**		
hdmi_mode=1	VGA			640×350	85Hz	
hdmi_mode=2	480p	60Hz		640×400	85Hz	
hdmi_mode=3	480p	60Hz	H	720×400	85Hz	

HDMI mode	VGA[3)]			DVI		
hdmi_mode=4	720p	60Hz		640×480	60Hz	
hdmi_mode=5	1080i	60Hz		640×480	72Hz	
hdmi_mode=6	480i	60Hz		640×480	75Hz	
hdmi_mode=7	480i	60Hz	H	640×480	85Hz	
hdmi_mode=8	240p	60Hz		800×600	56Hz	
hdmi_mode=9	240p	60Hz	H	800×600	60Hz	
hdmi_mode=10	480i	60Hz	4x	800×600	72Hz	
hdmi_mode=11	480i	60Hz	4x H	800×600	75Hz	
hdmi_mode=12	240p	60Hz	4x	800×600	85Hz	
hdmi_mode=13	240p	60Hz	4x H	800×600	120Hz	
hdmi_mode=14	480p	60Hz	2x	848×480	60Hz	
hdmi_mode=15	480p	60Hz	2x H	1024×768	43Hz	DO NOT USE
hdmi_mode=16	1080p	60Hz		1024×768	60Hz	
hdmi_mode=17	576p	50Hz		1024×768	70Hz	
hdmi_mode=18	576p	50Hz	H	1024×768	75Hz	
hdmi_mode=19	720p	50Hz		1024×768	85Hz	
hdmi_mode=20	1080i	50Hz		1024×768	120Hz	
hdmi_mode=21	576i	50Hz		1152×864	75Hz	
hdmi_mode=22	576i	50Hz	H	1280×768		reduced blanking
hdmi_mode=23	288p	50Hz		1280×768	60Hz	
hdmi_mode=24	288p	50Hz	H	1280×768	75Hz	
hdmi_mode=25	576i	50Hz	4x	1280×768	85Hz	
hdmi_mode=26	576i	50Hz	4x H	1280×768	120Hz	reduced blanking
hdmi_mode=27	288p	50Hz	4x	1280×800		reduced blanking
hdmi_mode=28	288p	50Hz	4x H	1280×800	60Hz	
hdmi_mode=29	576p	50Hz	2x	1280×800	75Hz	
hdmi_mode=30	576p	50Hz	2x H	1280×800	85Hz	
hdmi_mode=31	1080p	50Hz		1280×800	120Hz	reduced blanking
hdmi_mode=32	1080p	24Hz		1280×960	60Hz	
hdmi_mode=33	1080p	25Hz		1280×960	85Hz	
hdmi_mode=34	1080p	30Hz		1280×960	120Hz	reduced blanking
hdmi_mode=35	480p	60Hz	4x	1280×1024	60Hz	
hdmi_mode=36	480p	60Hz	4xH	1280×1024	75Hz	
hdmi_mode=37	576p	50Hz	4x	1280×1024	85Hz	
hdmi_mode=38	576p	50Hz	4x H	1280×1024	120Hz	reduced blanking
hdmi_mode=39	1080i	50Hz		1360×768	60Hz	
hdmi_mode=40	1080i	100H		1360×768	120Hz	reduced blanking
hdmi_mode=41	720p	100H		1400×1050		reduced blanking
hdmi_mode=42	576p	100H		1400×1050	60Hz	
hdmi_mode=43	576p	100H	H	1400×1050	75Hz	
hdmi_mode=44	576i	100H		1400×1050	85Hz	
hdmi_mode=45	576i	100H	H	1400×1050	120Hz	reduced blanking
hdmi_mode=46	1080i	120H		1440×900		reduced blanking
hdmi_mode=47	720p	120H		1440×900	60Hz	
hdmi_mode=48	480p	120H		1440×900	75Hz	

HDMI mode	VGA[3]			DVI		
hdmi_mode=49	480p	120H	H	1440×900	85Hz	
hdmi_mode=50	480i	120H		1440×900	120Hz	reduced blanking
hdmi_mode=51	480i	120H	H	1600×1200	60Hz	
hdmi_mode=52	576p	200H		1600×1200	65Hz	
hdmi_mode=53	576p	200H	H	1600×1200	70Hz	
hdmi_mode=54	576i	200H		1600×1200	75Hz	
hdmi_mode=55	576i	200H	H	1600×1200	85Hz	
hdmi_mode=56	480p	240H		1600×1200	120Hz	reduced blanking
hdmi_mode=57	480p	240H	H	1680×1050		reduced blanking
hdmi_mode=58	480i	240H		1680×1050	60Hz	
hdmi_mode=59	480i	240H	H	1680×1050	75Hz	
hdmi_mode=60				1680×1050	85Hz	
hdmi_mode=61				1680×1050	120Hz	reduced blanking
hdmi_mode=62				1792×1344	60Hz	
hdmi_mode=63				1792×1344	75Hz	
hdmi_mode=64				1792×1344	120Hz	reduced blanking
hdmi_mode=65				1856×1392	60Hz	
hdmi_mode=66				1856×1392	75Hz	
hdmi_mode=67				1856×1392	120Hz	reduced blanking
hdmi_mode=68				1920×1200		reduced blanking
hdmi_mode=69				1920×1200	60Hz	
hdmi_mode=70				1920×1200	75Hz	
hdmi_mode=71				1920×1200	85Hz	
hdmi_mode=72				1920×1200	120Hz	reduced blanking
hdmi_mode=73				1920×1440	60Hz	
hdmi_mode=74				1920×1440	75Hz	
hdmi_mode=75				1920×1440	120Hz	reduced blanking
hdmi_mode=76				2560×1600		reduced blanking
hdmi_mode=77				2560×1600	60Hz	
hdmi_mode=78				2560×1600	75Hz	
hdmi_mode=79				2560×1600	85Hz	
hdmi_mode=80				2560×1600	120Hz	reduced blanking
hdmi_mode=81				1366×768	60Hz	
hdmi_mode=82				1080p	60z	
hdmi_mode=83				1600×900		reduced blanking
hdmi_mode=84				2048×1152		reduced blanking
hdmi_mode=85				720p	60Hz	
hdmi_mode=86				1366×768		reduced blanking

Démarrage automatique des applications

Un point intéressant sur l'utilisation du RPI est de construire un système dans lequel l'interaction humaine est rarement nécessaire. Il peut démarrer et se mettre à jour sans aucune interaction. Les scripts peuvent aider dans ce processus

L'exemple suivant utilise git pour mettre à jour un dépôt local:

```
cat >> /etc/init.d/S80init <<EOF
#!/bin/sh
#
# Start processing
#

wget -q --spider http://google.com
if [ $? -eq 0 ]; then
    cd /home/default/yourdirectory
    git pull --rebase
fi
EOF
```

L'exemple suivant permet de lancer l'application **qtbrowser**[4]:

```
cat >> /etc/init.d/S90app <<EOF
#!/bin/sh

nohup /root/setup.sh & /root/run_app.sh & /root/run_appconnect.sh &
/root/run_qtbrowser.sh
EOF
```

Afin d'automatiser un maximum sans devoir saisir un mot de passe sudo crée un fichier `/etc/sudoers.d/90-rpi` contenant:

```
pi ALL = NOPASSWD:/usr/bin/fbi
```

Écran de démarrage

On peut implémenter un écran de démarrage en quelques étapes simples :

- Créer une séquence PNG nommée frame*.png et la placer dans un répertoire à l'intérieur de RPI.
- Modifier /etc/init.d/S01logging et inclure le code suivant dans la première ligne :

```
# Splash initiale !
cat /dev/zero 1> /dev/fb0 2>/dev/null
fbv -i -c /home/default/bootanimations/frame*.png --delay 1
```

mpv accéléré matériellement

Pour utiliser la lecture vidéo accélérée par le matériel de mpv sur le Raspberry Pi il faut construire deux packages basés sur les packages officiels, modifiés

- **ffmpeg** avec –enable-mmal pour la prise en charge de l'accélération matérielle MMAL
- **mpv** avec –enable-rpi pour la prise en charge de Raspberry Pi

Construction de mpv

Pour compiler pour une architecture différente, il faut installer les prérequis nécessaires :

```
apt install git make gcc device-tree-compiler bison flex libssl-dev libncurses-dev
```

Puis installer un compilateur croisé à jour et l'ensemble des outils associés. Il faut choisir la version adaptée à l'architecture de la machine hôte (celle sur laquelle on compile) et de la machine cible.

Par exemple, pour une compilation sur une architectuer aarch64 vers une architecture armhf, utiliser la version gcc-linaro-12.0.1-2022.02-aarch64_arm-linux-gnueabihf.tar.xz.

```
wget https://snapshots.linaro.org/gnu-toolchain/12.0-2022.02-1/arm-linux-gnueabihf/gcc-linaro-12.0.1-2022.02-aarch64_arm-linux-gnueabihf.tar.xz
sudo tar xf gcc-linaro-7.3.1-2018.05-x86_64_arm-linux-gnueabihf.tar.xz -C /opt
```

Configurer le compilateur croisé :

```
export ARCH=arm
export CROSS_COMPILE=/opt/gcc-linaro-12.0.1-2022.02-aarch64_arm-linux-gnueabihf-
```

Bien que ffmpeg prenne une heure à construire, la prise en charge de MMAL est requise pour l'accélération matérielle dans mpv.

```
sudo apt-get install -y  gperf bison flex autoconf automake make libharfbuzz-dev libfreetype6-dev libx11-dev libxrandr-dev libvdpau-dev libva-dev mesa-common-dev libegl1-mesa-dev yasm libasound2-dev libpulse-dev libuchardet-dev zlib1g-dev libfribidi-dev git libgnutls28-dev libgl1-mesa-dev libsdl2-dev wget texinfo help2man libtool libtool-bin ncurses-dev git
```

```
yasm mercurial cmake cmake-curses-gui libfribidi-dev checkinstall
libfontconfig1-dev libgl1-mesa-dev libgles2-mesa-dev gnutls-dev
libsmbclient-dev libpulse-dev libbluray-dev libdvdread-dev libluajit-5.1-dev
libjpeg-dev libv4l-dev libcdio-cdda-dev libcdio-paranoia-dev python g++
git clone https://github.com/mpv-player/mpv-build.git
cd mpv-build
echo --enable-libmpv-shared > mpv_options
echo --disable-cplayer >> mpv_options
echo --enable-rpi >> mpv_options
echo --enable-mmal >> ffmpeg_options
./use-mpv-release
./use-ffmpeg-release
./rebuild -j4
./update
sudo ./install
sudo ldconfig
```

Construction de jellyfin_media_player

jellyfin media player est un client de bureau utilisant jellyfin-web avec lecteur MPV intégré. Il est basé sur Plex Media Player et prend en charge Windows, Mac OS et Linux. Les médias sont lus dans la même fenêtre en utilisant l'interface jellyfin-web contrairement à Jellyfin Desktop. Il prend en charge le flux audio.

```
sudo apt install autoconf automake libtool libharfbuzz-dev libfreetype6-dev
libfontconfig1-dev libx11-dev libxrandr-dev libvdpau-dev libva-dev mesa-
common-dev libegl1-mesa-dev yasm libasound2-dev libpulse-dev libuchardet-dev
zlib1g-dev libfribidi-dev git libgnutls28-dev libgl1-mesa-dev libsdl2-dev
cmake wget python g++ qtwebengine5-dev qtquickcontrols2-5-dev
libqt5x11extras5-dev libcec-dev qml-module-qtquick-controls qml-module-
qtwebengine qml-module-qtwebchannel qtbase5-private-dev
mkdir jmp; cd jmp
git clone git://github.com/iwalton3/jellyfin-media-player
cd jellyfin-media-player
mkdir build
cd build
wget
https://github.com/iwalton3/jellyfin-web-jmp/releases/download/jwc-1.7.2-2/d
ist.zip
unzip dist.zip
cmake -DCMAKE_BUILD_TYPE=Debug -DCMAKE_INSTALL_PREFIX=/usr/local/ ..
make -j4
sudo make install
rm -rf ~/jmp/
```

jellyfin-mpv-shim permet de diffuser des médias depuis Jellyfin Mobile et les applications Web vers MPV. On peut l'installer à partir via pip, mais cela nécessite

d'installer les prérequis suivants:

```
sudo apt installer python3-tk python3-jinja2 python3-webview
gir1.2-webkit2-4.0
```

avant d'installer python-mpv, pystray et jellyfin-mpv-shim:

```
sudo pip3 install --upgrade python-mpv pystray jellyfin-mpv-shim
```

1)

gpu_mem définit la mémoire GPU si le Pi dispose de 256M, 512M ou 1024. **gpu_mem_256**, **gpu_mem_512** et **gpu_mem_1024** remplacent cela s'ils s'exécutent respectivement sur un pi disposant de 256M , 512M et 1024M.

2)

hdmi_group définit le type HDMI, Les modes CEA (**hdmi_group=1**) sont destinés à la télévision, ils incluent de nombreux modes entrelacés et progressifs, généralement avec des fréquences d'images de 25/50/100 Hz (PAL) ou 30/60/120 Hz (NTSC) et des résolutions TV de 288/480/576/720/1080 lignes de balayage. Les modes DMT (**hdmi_group=2**) sont destinés aux écrans d'ordinateur, il n'y a donc aucun des modes entrelacés, les résolutions sont 640/720/800/1024/1280 et les fréquences d'images sont compatibles avec les écrans d'ordinateur, quelque chose comme 60/70/75/80/ 85/120Hz. Ne pas spécifier le groupe, ou définir sur 0 utilisera le groupe préféré signalé par l'edid.

3)

Mode VGA: **H** signifie variante 16:9 d'un mode normal 4:3, **2x** signifie pixel doublé c'est-à-dire une fréquence d'horloge plus élevée, chaque pixel étant répété deux fois, **4x** signifie pixel quadruplé c'est-à-dire une fréquence d'horloge plus élevée, chaque pixel étant répété quatre fois

4)

QTBrowser n'est pas Chrome ou Safari. Il contient quelques détails et différentes manières de traiter le HTML, et il est important de développer un code tout en testant directement sur le RPI: Pendant le chargement de la page, les classes avec des animations CSS ne fonctionneront pas. Il faut ajouter la classe par programmation après le chargement de la page, le images d'une taille supérieure à 1000 pixels affectent négativement la fréquence d'images, il faut abandonner jQuery, il faut éviter les dégradés et les images avec opacité, plusieurs images présentées à l'écran fonctionnent mieux que plusieurs toiles, le clonage des éléments DOM est préférable à leur création par programmation via JavaScript