

## Slackware ARM on the Raspberry Pi 3

The Raspberry Pi 3 has a Broadcom BCM2837 SoC incorporating a Quad-core ARMv8 Cortex-A53 [64 bit] CPU @ 1.2GHz and VideoCore IV GPU @ 400MHz, and comes with 1GB LPDDR2 SDRAM @ 900MHz. This revised and upgraded ARM single board computer succeeds the [Raspberry Pi \(2\)](#), and is considerably quicker and a lot more powerful. Wi-Fi and Bluetooth are now included on-board. Still no RTC though. Think of the RPi3 as a renovation, and not an innovation. Slackware ARM, as you would expect, runs faultlessly on this device, with a very significant increase in speed. Compile times are much shorter compared to the RPi2, for example.

The Raspberry Pi 3 is supported outside of the official Slackware ARM tree by the Slackware community.

### Slackware releases 14.2, -current

Slackware ARM -current or Slackware ARM 14.2 can be installed on the Raspberry Pi 3.

Follow the link(s) in the table below. These are maintained by a separate author as part of the Slackware-on-Raspberry Pi community.

Site	Slackware versions	Using official Slackware packages	Installation methods	Notes
<a href="#">SARPi Project</a>	14.2,-current	Yes	Slackware installer	An end-to-end HOW TO tutorial taking you through the installation and setup process.

### AArch64 ARM64 [Experimental], Slackware ARM -current

Experimental, development, and prototype, Slackware AArch64 ARM64 link(s).

Site	Slackware versions	Using official Slackware packages	Installation methods	Notes
<a href="#">SARPi64 Project</a>	-current	Yes	Slackware installer	A development project for Slackware ARM running AArch64 [ARMv8] kernel and modules. <b>Experimental in nature and purpose.</b>

### Manual install method

This method is for installing Slackware ARM 14.2 on a Raspberry Pi 3 Model B. However, it should work for other Slackware ARM and Raspberry Pi versions.

## 1. Partition and format the SD Card

```
$ sudo fdisk -l /dev/mmcblk0

Disk /dev/mmcblk0: 31.9 GB, 31914983424 bytes
4 heads, 16 sectors/track, 973968 cylinders, total 62333952 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00000000

    Device Boot      Start         End      Blocks   Id  System
/dev/mmcblk0p1          2048       67583        32768    b   W95 FAT32
/dev/mmcblk0p2       67584    62333951   31133184   83   Linux
$ sudo mkfs.vfat /dev/mmcblk0p1
$ sudo mkfs.ext4 /dev/mmcblk0p2
```

Remarks:

- I use a 32 GB SD Card
- I choose 32 MB for the size of the first partition
- I let the empty space left for the second partition

## 2. Put the Raspberry Pi firmware in the SD Card

```
$ git clone https://github.com/raspberrypi/firmware.git
$ sudo mount /dev/mmcblk0p1 ~/mnt
$ sudo cp -r firmware/boot/* ~/mnt
$ sudo umount ~/mnt
$ sudo mount /dev/mmcblk0p2 ~/mnt
$ sudo mkdir -p ~/mnt/lib/modules
$ sudo cp -r firmware/modules/* ~/mnt/lib/modules
$ sudo umount ~/mnt
```

## 3. Put the Slackware ARM mini root file system in the SD Card

```
$ wget -c
ftp://ftp.arm.slackware.com/slackwarearm/slackwarearm-devtools/minirootfs/ro
ots/slack-14.2-miniroot_01Jul16.tar.xz
$ sudo mount /dev/mmcblk0p2 ~/mnt
$ sudo tar -C ~/mnt -xf slack-14.2-miniroot_01Jul16.tar.xz
$ echo "/dev/mmcblk0p1 /boot vfat defaults 0 0" | sudo tee ~/mnt/etc/fstab
$ echo "/dev/mmcblk0p2 /      ext4 defaults 0 0" | sudo tee -a
~/mnt/etc/fstab
$ echo "proc          /proc proc defaults 0 0" | sudo tee -a
~/mnt/etc/fstab
$ PASSWD=$(openssl passwd -1 -salt cetkq/enZx6/c2 password)
$ sudo sed -i "s|\(root:\)\.*\(:16983:0:::\)|\1${PASSWD}\2|"
```

```
~/mnt/etc/shadow
$ sudo sed -i 's|USE_DHCP\[1\]=""|USE_DHCP\[1\]="yes"|'
~/mnt/etc/rc.d/rc.inet1.conf
$ echo "PermitRootLogin yes" | sudo tee -a ~/mnt/etc/ssh/sshd_config
$ sudo umount ~/mnt
```

Remarks:

- I set password as password for the root user
- I set DHCP on the eth1 network interface
- I allow the root user to connect through SSH

## 4. Insert the SD Card in the Raspberry Pi

Your SD Card is ready so you can insert it in the Raspberry Pi and boot.

You can connect remotely to your Raspberry Pi as root through SSH.

```
$ ssh root@raspberrypi
```

As soon as you are logged, you might want to install additional Slackware ARM packages:

```
$ wget --mirror ftp://ftp.arm.slackware.com/slackwarearm/slackwarearm-14.2
$ upgradepkg --install-new
ftp.arm.slackware.com/slackwarearm/slackwarearm-14.2/slackware/*/*.txz
$ removepkg
ftp.arm.slackware.com/slackwarearm/slackwarearm-14.2/slackware/*/kernel_*.txz
```

Remarks:

- I consider that the Raspberry Pi hostname is raspberrypi
- I recommend to add a normal user by using the adduser command and use this user instead of root
- I recommend to change the root user password by using the passwd command
- I recommend to disallow the root user to connect through SSH by editing the /etc/ssh/sshd\_config file
- I recommend to [build your own Linux kernel](#) packages because the kernel you are running does not match with the installed Slackware ARM packages
- I recommend to change the slackware.localdomain hostname by editing the /etc/HOSTNAME file

## 5. Tips and tricks

### 5.1. Bluetooth

The Raspberry Pi has a Broadcom chip providing Bluetooth. However, the required firmware is not installed on Slackware ARM. It means that you need to download and install it:

```
$ git clone https://github.com/OpenELEC/misc-firmware.git
$ mkdir -pv /where/you/want/to/install/etc/firmware
$ cp -v misc-firmware/firmware/brcm/BCM43430A1.hcd
/where/you/want/to/install/etc/firmware
```

Then build your own Slackware ARM bluez - firmware - brcm package and install it.

In order to enable Bluetooth you need to add the Bluetooth module, run the Bluetooth daemon, attach the device then open and initialize the device. Add the following lines to the end of the `/etc/rc.d/rc.local` file:

```
if ps axc | grep -q bluetoothd; then
    killall bluetoothd
fi
if ps axc | grep -q hciattach; then
    killall hciattach
fi
if lsmod | grep -q hci_uart; then
    rmmod hci_uart
fi
if lsmod | grep -q btbcm; then
    rmmod btbcm
fi
modprobe btbcm
bluetoothd &
while ! hciattach /dev/ttyAMA0 bcm43xx; do
    echo "Unable to attach the device! We try again..."
done
hciconfig hci0 up
```

Remark: Sometimes there is a failure with the `hciattach` command so that is why you need a `while`.

You can check that Bluetooth is working by typing:

```
$ hcitool dev
$ hcitool scan
$ sudo bluetoothctl
```

Now, the Bluetooth is correctly set.

## 5.2. Keyboard

The default keyboard map on Slackware ARM is the one of United Kingdom. If you want to load an other keyboard map, edit the `/etc/rc.d/rc.keymap` file.

If you want to change the keyboard layout for X11, you need to copy the X11 configuration file then edit it:

```
sudo cp /usr/share/X11/xorg.conf.d/90-keyboard-layout.conf
/etc/X11/xorg.conf.d
```

Now, the keyboard is correctly set.

### 5.3. Memory

Unfortunately, the Raspberry Pi has only 1 GB of memory. It means that you will eventually not be able to run some applications requiring more memory. However, you can extend your memory by creating a temporary 4 GB swap partition:

```
$ dd if=/dev/zero of=/tmp/swap bs=1M count=4000
$ mkswap /tmp/swap
$ sudo swapon /tmp/swap
```

You can check the current memory by typing:

```
$ free
```

Now, the memory is correctly set.

### 5.4. Processor

The Raspberry Pi processor can reach 1.2 GHz. However, by default, it is stuck to 600 MHz even if it is used at 100%. You can check the current frequency of the processor by typing:

```
$ cpufreq-info
```

In order to reach 1.2 GHz when the processor is used at 100% (i.e., use the frequency scaling), you need to change the default governors. Add the following line to the end of the `/etc/rc.d/rc.local` file:

```
echo ondemand | sudo tee
/sys/devices/system/cpu/cpu*/cpufreq/scaling_governor
```

Now, the processor is correctly set.

### 5.5. Time

Unfortunately, the Raspberry Pi does not provide a Real-Time Clock (RTC). That is why there is no battery included with the board. It means that each time you shutdown the Raspberry Pi, the time is reset! However, if you have internet access, you can update the time during the Slackware ARM boot. Add the following line to the end of the `/etc/rc.d/rc.local` file:

```
ntpdate pool.ntp.org
```

Now, the time is correctly set.

## 5.6. Video

Unfortunately, the Raspberry Pi is not compatible with OpenGL (it is compatible with OpenGL ES that is a subset of OpenGL). It means that, by default, each application requiring OpenGL will be slow. However, you can reach 60 FPS with OpenGL applications on the Raspberry Pi by using the correct driver.

Firstly, you need to build Mesa ( $\geq 17.1.10$ ) with the VC4 DRI driver:

```
$ CFLAGS="-O2 -march=armv8-a -mtune=cortex-a53 -mfloat-abi=softfp -
mfpu=neon-vfpv4" \
  CXXFLAGS="-O2 -march=armv8-a -mtune=cortex-a53 -mfloat-abi=softfp -
mfpu=neon-vfpv4" \
  ./configure \
  --prefix=/usr \
  --sysconfdir=/etc \
  --with-dri-driverdir=/usr/lib/xorg/modules/dri \
  --with-egl-platforms=x11,drm \
  --with-gallium-drivers=vc4
$ make -j4
$ make install DESTDIR=/where/you/want/to/install
```

Then build your own Slackware ARM mesa package and install it (you can safely upgrade the one provided by Slackware ARM).

Secondly, add the following line to the end of the `/boot/config.txt` file:

```
dtoverlay=vc4-fkms-v3d
```

Then reboot the Raspberry Pi.

You can check that you are able to get 60 FPS with OpenGL applications on the Raspberry Pi by typing the following command in an X11 terminal:

```
$ glxgears
```

An other problem with the video is that the default resolution is 1824×984. It means that you can see black borders around your 1920×1080 screen. In order to fix that, add the following line to the end of the `/boot/config.txt` file:

```
disable_overscan=1
```

Then reboot the Raspberry Pi.

You can check that you are using a 1920×1080 resolution on the Raspberry Pi by typing the following command in an X11 terminal:

```
$ xrandr
```

Now, the video is correctly set.

## 5.7. UART

In the `/etc/inittab` file, `s0` refers to the mini [UART](#) that is disabled by default on the Raspberry Pi. Consequently, every 5 minutes, this line will be written to `/dev/tty1` by the `init` process:

```
INIT: Id "s0" respawning too fast: disabled for 5 minutes
```

An easy fix is to edit the `/etc/inittab` file and replace the following line:

```
s0:12345:respawn:/sbin/agetty 115200 ttyS0 vt100
```

with this one:

```
#s0:12345:respawn:/sbin/agetty 115200 ttyS0 vt100
```

Then reboot the Raspberry Pi. Now, the UART is correctly set.

## Sources

- Originally written by [Exaga](#)
- Contributions by [yugiohjcj](#)

[howtos](#), [hardware](#), [arm](#), [author exaga](#)

From:  
<https://docs.slackware.com/> - **SlackDocs**

Permanent link:  
<https://docs.slackware.com/howtos:hardware:arm:raspberrypi3>

Last update: **2019/08/23 05:02 (UTC)**

