Linux installation (incl. CAN-drivers and WiFi-setup)

There are two ways to set up an SD-card for Linux-based RPi:

1. All data is copied block by block from an already working version
2. A standard RPi-suitable Linux distribution is installed and then tweaked to our needs.

Both methods are described below (both of them are exemplified on a Linux system). In both cases, WiFi-mac address of the RPi-hardware must be registered with CRIT.

Method 1 – straightforward data copying

Check what partition names are assigned to the SD-cards. This is done either by

sudo fdisk -l

or

dmesg

and noticing the difference between the printouts when SD-cards are inserted and when they are not.

Suppose that a working Linux SD-card has been mounted on /dev/sdd, while the new card is mounted on /dev/mmcblk0. Unmount the SD-partitions and use the ”dd”-command to copy data. Be very careful about setting input (if) and output (of) correctly.

sudo -i

umount /dev/sd1

umount /dev/sd2

umount /dev/mmcblk0p1

dd if=/dev/sdd of=/dev/mmcblk0

If using an intermediary storage file (instead of directly copying between two SD-cards):

dd if=/dev/mmcblk0 | gzip > linux-image.gz

gzip -dc linux-image.gz | dd of=/dev/mmcblk0

Method 2 – Standard kernel installation and tweaking

**-1. Background material (not really necessary, but good to know)**

Background materials about how to use CAN board for starting purpose:

[**http://skpang.co.uk/catalog/pican-canbus-board-for-raspberry-pi-p-1196.html**](http://skpang.co.uk/catalog/pican-canbus-board-for-raspberry-pi-p-1196.html)

**0. Install Raspian**

**http://www.raspberrypi.org/help/noobs-setup/**

**1.Prepare the kernel (only needed to do once)**

Instructions about how to compile a KERNEL image (/linux\_kernel) which is containing a CAN driver module:

<http://elinux.org/RPi_CANBus>

(Note: Because the link shows the steps about how to compile the kernel version 3.6, recommend to download the 3.6 kernel source file. Otherwise you cannot follow the instructions.)

**2. TEST WIFI**

Register the MAC address of Wi-Fi to SICS system

Login Pi, Input command:

sudo nano /etc/wpa\_supplicant/wpa\_supplicant.conf

Then add the following configuration to the file:

network={

ssid="phoneName"

psk="password"

proto=RSN

key\_mgmt=WPA-PSK

pairwise=CCMP

auth\_alg=OPEN

priority=60

}

network={

ssid="SICS"

key\_mgmt=NONE

auth\_alg=OPEN

priority=40

}

sudo nano /etc/network/interfaces

Add the following configuration into the file

#allow-hotplug wlan0

#iface wlan0 inet manual

#wpa-roam /etc/wpa\_supplicant/wpa\_supplicant.conf

#iface default inet dhcp

allow-hotplug wlan0

auto wlan0

iface wlan0 inet dhcp

wpa-conf /etc/wpa\_supplicant/wpa\_supplicant.conf

Reboot pi, if everything is working fine, then the Wi-Fi blue led will blink.

**3. Change the Kernel**

Based on the instructions from step 1, you can get /boot/ and /lib/ under opt/raspberrypi/build, also you need find can.ko, can-bcm.ko, can-dev.ko, can-raw.ko, mcp251x.ko,mcp2515.ko and spi-bcm2708.ko under the raspberrypi/linux/

Now you can create a folder called, for example, Kernel, which should include the following things:

/boot, /lib, can.ko, can-bcm.ko, can-dev.ko, can-raw.ko, mcp251x.ko, mcp2515.ko and spi-cm2708.ko

Transfer this “Kernel” folder to the Pi,

Replace the original kernel.img with our new kernel.img by using command:

pi@raspberrypi ~ $ cd kernel/

pi@raspberrypi ~/kernel$ cp boot/kernel.img /boot/

Rename the original /modules/ folder which is under root/lib to /lib/modules\_orig, and copy the new compiling /kernel/lib/modules to the /root/lib:

pi@raspberrypi /lib $ sudo mv modules / modules\_orig

pi@raspberrypi /lib $ sudo cp /home/pi/kernel/lib/modules/ .

Merge our new firmware into the official one.

pi@raspberrypi /lib/firmware $ cp -r /home/pi/kernel/lib/firmware/\* .

Reboot Pi, if everything works, you should see that the Wi-Pi still blinking. Otherwise something wrong, for example, the size of files in the modules is 0.

**4. Install the can-utils**

You can follow this link: <http://elinux.org/Can-utils>

pi@raspberrypi ~ $ git clone https://git.gitorious.org/linux-can/can-utils.git

pi@raspberrypi ~ $ cd can-utils

pi@raspberrypi ~/can-utils $ ./autogen.sh

(if this not support, sudo apt-get install autoreconf)

pi@raspberrypi ~/can-utils $ ./configure

pi@raspberrypi ~/can-utils $ sudo make

pi@raspberrypi ~/can-utils $ sudo make install (with root privileges)

Copy can.ko, can-bcm.ko, can-dev.ko, can-raw.ko, mcp251x.ko, mcp2515.ko from /kernel to / can-utils, and install them under root privilege by using the command:

insmod spi-bcm2708.ko

insmod can.ko

insmod can-dev.ko

insmod can-raw.ko

insmod can-bcm.ko

insmod mcp251x.ko

ip link set can0 up type can bitrate 500000

After that you can send can message by using

root@raspberrypi:/home/pi/kernel/can-utils# ./cansend can0 123#1234

If you have 2 raspberry pis, on the other pi, by using the command

root@raspberrypi:/home/pi/kernel/can-utils# ./candump can0

you will see the result like

can0 123 [2] 12 34