How to setup the Raspberry Pi to use the LPRF Driver

1. Download and unzip Raspbian image

$ wget [Link to download]

$ unzip *file* -d *destination*

1. Connect the SD-Card to the PC
2. Get the image to the SD-Card

$ df -h // determine the partition number

$ umount /dev/sdb1

$ umount /dev/sdb2

$ sudo dd bs=4M if=*file.img* of=/dev/sdb

$ pkill -USR1 -n -x dd // to get status

$ sync

1. Insert the SD-Card into the RPI and start the RPI
2. RPI configuration (first start config)

* Expand Filesystem
* Boot Options -> B1
* Change Timezone and Keyboard layout

1. Reboot
2. Change network settings

$ sudo nano /etc/network/interfaces

auto lo

iface lo inet loopback

auto eth0

iface eth0 inet dhsp

hwaddress 02:E4:73:03:BE:83

$ sudo reboot

1. Changing the Host

$ sudo nano /etc/hosts

Change “raspberrypi” into the name you want (RPI2)

$ sudo nano /etc/hostname

Replace “raspberrypi” with the name you choose above

$ sudo /etc/init.d/hostname.sh

$ sudo reboot

1. Now you can use ssh for remote to the RPI

$ ssh [pi@137.226.200.211](mailto:pi@137.226.200.211)

1. Update the RPI

$ sudo apt-get update

$ sudo apt-get upgrade

1. Download the linux source tree

$ mkdir kernel

$ cd kernel

$ git clone –depth=1 <https://github.com/raspberrypi/linux.git>

$ cd linux

1. Prepare kernel compiling

$ sudo apt-get install bc libncurses5-dev libncursesw5-dev

$ KERNEL=kernel

$ make bcmrpi\_defconfig

$ make menuconfig

* Device Drivers -> Network device support -> Wireless LAN -> Realtek 8192C USB WiFi
* Device Drivers -> Network device support -> USB Network Adapters -> Multi-purpose USB Networking Framework -> SMSC LAN95XX based USB 2.0 10/100 ethernet devices
* Device Drivers -> SPI support -> BCM2835 SPI controller <\*>
* Device Drivers -> USB support -> USB Mass Storage support <\*>
* Device Drivers -> USB support -> DesignWare USB2 DRD Core Support <\*>
* Networking support -> RF switch subsystem support <\*>
* Networking support -> Wireless -> cfg80211 – wireless configuration API <\*>
* Networking support -> Wireless -> Generic IEEE 802.11 Networking Stack (mac80211) <\*>
* Networking support -> Networking Options -> TCP/IP networking -> The IPv6 protocol <\*>
* Networking support -> Networking Options -> 6LoWPAN Support <\*>
* Networking support -> Networking Options -> IEEE Std 802.15.4 Low-Rate Wireless Personal Area Networks support <\*>
* Networking support -> Networking Options -> IEEE Std 802.15.4 Low-Rate Wireless Personal Area Networks support -> IEEE 802.15.4 socket interface <\*>
* Networking support -> Networking Options -> IEEE Std 802.15.4 Low-Rate Wireless Personal Area Networks support -> 6lowpan support over IEEE 802.15.4 <\*>
* Networking support -> Networking Options -> IEEE Std 802.15.4 Low-Rate Wireless Personal Area Networks support -> Generic IEEE 802.15.4 Soft Networking Stack (mac802154) <\*>
* Networking support -> Networking Options -> NETLINK: mmaped IO <\*>
* Networking support -> Networking Options -> NETLINK: socket monitoring interface <\*>
* Device Drivers -> Network Device Support -> IEEE 802.15.4 drivers -> AT86RF230/231/233/212 transceiver driver <M>

1. Compiling the kernel

$ make zImage modules dtbs

Takes almost 12 hours

1. Install Modules and copy files in the boot folder

$ sudo make modules\_install

$ sudo cp arch/arm/boot/dts/\*.dtb /boot/

$ sudo cp arch/arm/boot/dts/overlays/\*.dtb\* /boot/overlays

$ sudo cp arch/arm/boot/dts/overlays/README /boot/obverlays

$ sudo scripts/mkknlimg arch/arm/boot/zImage /boot/kernel.img

1. Modify /boot/config.txt

Insert at the end of the file

dtoverlay=mmc

1. Reboot

# Variant A: didn’t work

1. Modify arch/arm/boot/dts/bcm2708-rpi-b-plus.dts

Delete the old spi0-node and insert

&spi0 {

status = "okay";

at86rf231@0 {

compatible = "atmel,at86rf231";

reg = <0>;

interrupts = <23 1>;

interrupt-parent = <&gpio>;

reset-gpio = <&gpio 24 1>;

sleep-tpio = <&gpio 25 1>;

spi-max-frequency = <500000>;

};

};

1. Change directory and remake dtbs files

$ cd ~/kernel/linux

$ make dtbs

$ sudo cp arch/arm/boot/dts/\*.dtb /boot

1. Reboot and go to step 17.

# Variant B: use the at86rf233 overlay file

1. Edit arch/arm/boot/dts/overlays/at86rf233-overlay.dts

spi-max-frequency = <2000000>;

1. Change directory and remake dtbs files

$ cd ~/kernel/linux

$ make dtbs

$ sudo cp arch/arm/boot/dts/overlays/\*.dtb\* /boot/overlays

1. Insert at the end of /boot/config.txt

dtoverly=at86rf233

1. Reboot and go to step 17.

# Variant C: use a lprf overlay file

1. Change directory to the kernel overlays and create a new overlay for the LPRF Driver

$ cd arch/arm/boot/dts/overlays

$ nano lprf-overlay.dts

1. Insert

/dts-v1/;

/plugin/;

/\* Overlay for LPRF Transceiver Chip spi0.0 \*/

/ {

compatible = "brcm,bcm2835", "brcm,bcm2836", "brcm,bcm2708", "brcm,bcm2709";

fragment@0 {

target = <&spi0>;

\_\_overlay\_\_ {

#address-cells = <1>;

#size-cells = <0>;

status = "okay";

lowpan0: lprf@0 {

compatible = "ias,lprf";

reg = <0>;

interrupt-parent = <&gpio>;

interrupts = <23 4>; /\* active high \*/

reset-gpio = <&gpio 24 1>;

sleep-gpio = <&gpio 25 1>;

spi-max-frequency = <2000000>;

xtal-trim = /bits/ 8 <0xf>;

};

};

};

fragment@1 {

target = <&spidev0>;

\_\_overlay\_\_ {

status = "disabled";

};

};

fragment@2 {

target = <&gpio>;

\_\_overlay\_\_ {

lowpan0\_pins: lowpan0\_pins {

brcm,pins = <23 24 25>;

brcm,function = <0 1 1>; /\* in out out \*/

};

};

};

\_\_overrides\_\_ {

interrupt = <&lowpan0>, "interrupts:0",

<&lowpan0\_pins>, "brcm,pins:0";

reset = <&lowpan0>, "reset-gpio:4",

<&lowpan0\_pins>, "brcm,pins:4";

sleep = <&lowpan0>, "sleep-gpio:4",

<&lowpan0\_pins>, "brcm,pins:8";

speed = <&lowpan0>, "spi-max-frequency:0";

trim = <&lowpan0>, "xtal-trim.0";

};

};

1. Create the \*.dtbo file and copy it into the /boot-folder

$ dtc -0 dtb -o lprf.dtbo -b 0 -@ lprf-overlay.dts

$ sudo cp arch/arm/boot/dts/overlays/lprf.dtbo /boot/overlays

1. Modify the /boot/config.txt and insert at the end

dtoverlay=lprf

1. Reboot and go to step 17.
2. Build and install the WPAN tools

$ cd

$ sudo apt-get install libnl-3-dev libnl-genl-3-dev

$ sudo apt-get install dh-autoreconf

$ git clone https://github.com/linux-wpan/wpan-tools

$ cd wpan-tools

$ ./autogen.sh

$ ./configure CFLAGS='-g -O0' --prefix=/usr --sysconfdir=/etc --libdir=/usr/lib

$ make

$ sudo make install

1. Copy lprf-files to RPI
2. Create file linux/include/linux/spi/lprf.h and insert

#ifndef LPRF\_H

#define LPRF\_H

struct lprf\_platform\_data {

int rstn;

int slp\_tr;

int dig2;

u8 xtal\_trim;

};

#endif

1. Everything is ready and the driver can be loaded.

$ sudo ./load\_driver.sh

If you downloaded the linux source tree in a different folder, you have to change *load\_driver.sh* (line 15) and the *Makefile* (line 6).

1. If you want to unload the driver manually, you have to set down the wpan device first.

$ sudo ip link set wpan0 down

$ sudo rmmod lprf\_tx