LINUX DEVICE DRIVERS for nRF24L01+

Purpose:

The main objective of this project is to establish RF communication between two nrf24l01+ 2.4ghz wireless transceivers, one connected to RaspberryPi and other to connected to arduino.

Pre- Requirement:

1. **Building Raspberry**

You need a RaspberryPi hardware kit which is to be installed with latest version of OS i.e. raspbian jessie . Once you boot your RaspberryPi with OS installed you need to open the terminal window and type the following command:

```
$ sudo apt-get update

$ sudo apt-get upgrade
```

These two command will help to update and upgrade your RaspberryPi with latest updates and all necessary packages. Then for testing purpose that your RPi is working correctly you need to run a simple kernel module (Hello module) on it. When you make your kernel module, it will generate .ko and .o file of your respective module, if your RPi is having all required RPi header files. Otherwise you will get the following error:

```
make:*** /lib/modules/3.10.25+/build: No such file or directory. Stop.
make: *** [all] Error 2
```

These make errors indicate that your RPi doesn’t have required RPi-kernel-headers. To overcome this error and get things working you need to do the following:

```
$ apt install build-essential bc git wget

$ cd /usr/src

$ git clone --depth 1 https://github.com/raspberrypi/linux.git

$ ln -s linux $(uname -r)

$ ln -s /usr/src/linux /lib/modules/$(uname -r)/build
```

These commands will help you to install all necessary kernel headers required for RPi and using last command you can link your install kernel headers to your existing kernel.

2. **Installing BCM2835 drivers**

Firstly, we need to configure the GPIOs and SPI pins to enable the /dev/spidev0.0 and /dev/spidev1.0 device Node for SPI, spi_bcm2708 Driver. The latest version, of the library, bcm2835-1.46.tar.gz works well with both RaspberryPi and RaspberryPi2. The following version can be downloaded from the link:
http://www.airspayce.com/mikem/bcm2835/bcm2835-1.46.tar.gz

After, that follow the below instruction to install the library to your RaspberryPi.

```
 #download the latest version of the library, say bcm2835-1.xx.tar.gz, then:
$tar zxvf bcm2835-1.xx.tar.gz cd bcm2835-1.xx
$ ./configure
$ make
$ sudo make check
$ sudo make install
```

NOTE: Raspberry Pi 2 (RPI2)
For this library to work correctly on RPI2, you MUST have the device tree support enabled in the kernel. You should also ensure you are using the latest version of Linux. The library has been tested on RPI2 with 2015-02-16-raspbian-wheezy and ArchLinuxARM-rpi-2 as of 2015-03-29.

When device tree suport is enabled, the file /proc/device-tree/soc/ranges will appear in the file system,
and the bcm2835 module relies on its presence to correctly run on RPI2 (it is optional for RPI1). Without
device tree support enabled and the presence of this file, it will not work on RPI2.

To enable device tree support:

sudo raspi-config under Advanced Options - enable Device Tree and Reboot.

The Library allows users to override the BCM2835 Driver for SPIDEV, and use some other customized driver or some other platform, like BeagleBone Black by using the using the command during 'make all' and 'make install',

```
#sudo make all -B RF24_SPIDEV=1
#sudo make install -B RF24_SPIDEV=1
```

About the project:

This project depicts use of nRF24L01+ for communication using two different modes. In first method, a character driver is written to register the nRF24L01+ as a character device. You can build API's to communicate with this module. With API, one can make networking layer glued into the kernel module.With ioctl functions, networking can also be moved out of kernelspace to userspace.

In second method, the project uses the /dev/spidev0.0 node in the /dev directory which allows the user to access BCM2835 driver for SPI function.

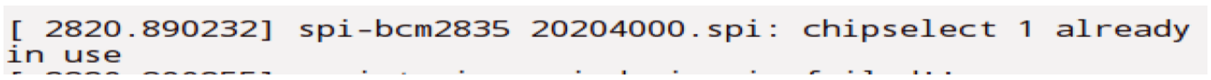
<u>nRF24L01+ Character Device Driver: nrf_character_driver</u>

A character device driver is made to implement the functionalities of the nRF24L01+ device to communicate to other devices using SPI protocol. In this, inbuilt <linux/spi/spi.h> library is used. Following commands has to be used to buils:

$ make

$ sudo insmod ./nrf_1.ko

But since we have enabled the BCM2835 driver initially, the Chipselect will show busy on the busy:



```
[ 2820.890232] spi-bcm2835 20204000.spi: chipselect 1 already
in use
```

So, to make our module usable we need to disable the chip select using dts overlay structure or brute force way:

Forceful disabling the chipselect:

$ sudo git clone https://github.com/msperl/spi-config.git

$ cd spi-config

$sudo depmode –a (just for the first time)

$ modprobe spi-config devices=bus=0:cs=1:force_release (whenever you want to run your module)

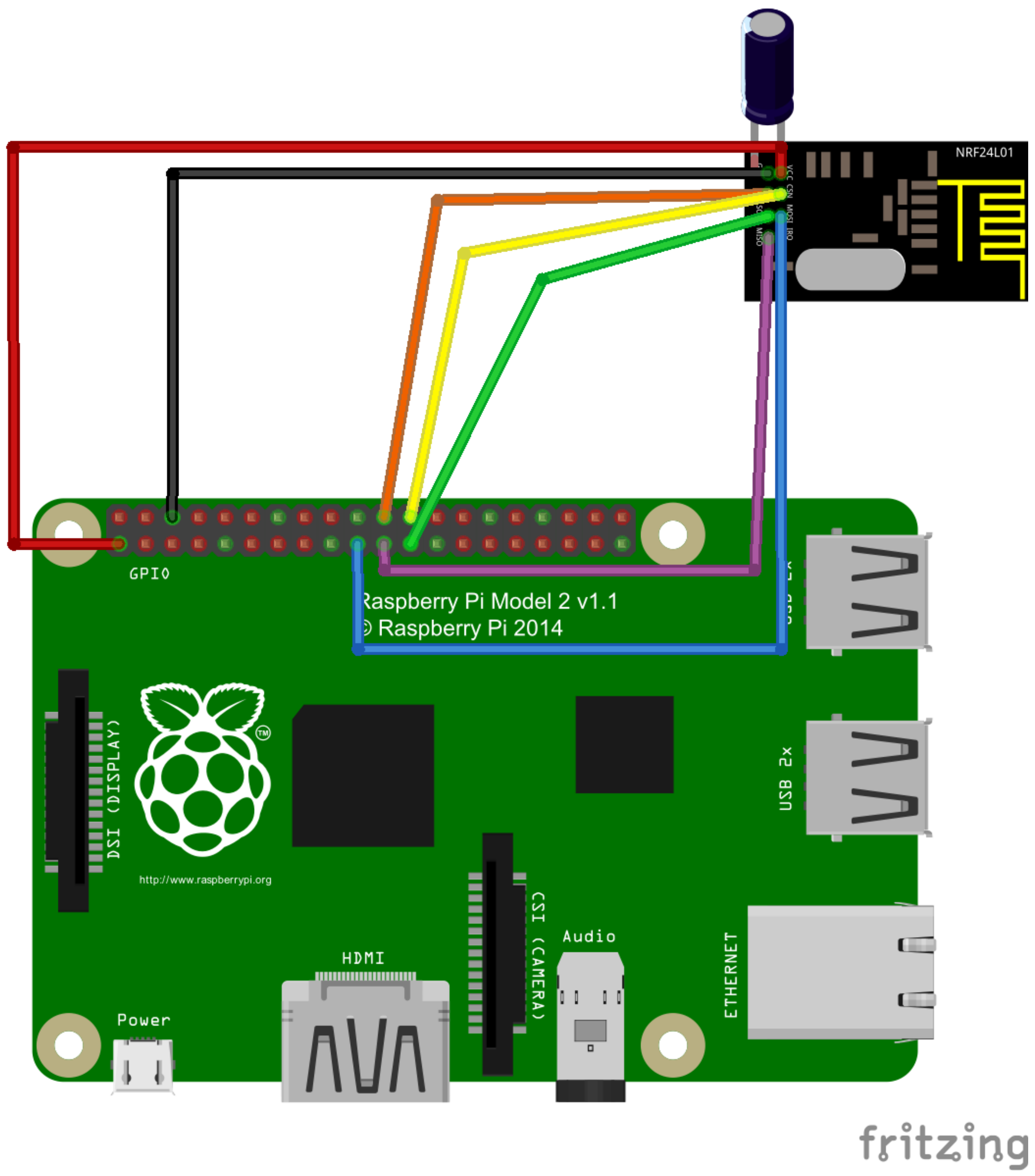
$ cd

$ sudo insmod ./nrf_1.ko

Necessary Connections:

1. RaspberrPi2

RaspberryPi2(GPIO)	Physical pin	nRF24L01+
GND	6	GND PIN1
3.3POWER	1	3.3V PIN2
GPIO 25	22	CE PIN3
GPIO 8	24	CS PIN4
GPIO 11	23	SCK PIN5
GPIO 10	19	MOSI PIN6
GPIO 9	21	MISO PIN7
-	-	IRQ PIN8



2. Arduino Connections

Arduino Pins	nRF24L01+
GND	GND PIN1
3.3V	3.3V PIN2
PIN 8	CE PIN3
PIN 9	CS PIN4
PIN 13	SCK PIN5
PIN 11	MOSI PIN6
PIN 12	MISO PIN7
-	IRQ PIN8

<u>nRF24L01+ User space module: nrf_user_program</u>

this module uses the SPIDEV library. This library utilizes the functions of <linux/spi/spi.h>. in this module, we have used the one of two nodes created by /dev/spidev0.0(chip select 0) and /dev/spidev0.1(chip select 1). The example_linux folder contains the user space program to the implemented on RPi and the example folder contains the Arduino code. Steps to be followed:

```
$ cd nrf_final

$ ./configure --driver=SPIDEV

$ sudo make install –B

$ cd examples_linux

$make

$ sudo ./nrf_testing
```

Necessary Connections:

RaspberryPi2(GPIO)	Physical pin	nRF24L01+
GND	25	GND PIN1
3.3POWER	1	3.3V PIN2
GPIO 25	22	CE PIN3
GPIO 8	24	CS PIN4
GPIO 11	23	SCK PIN5
GPIO 10	19	MOSI PIN6
GPIO 9	21	MISO PIN7
-	-	IRQ PIN8

Arduino connections will be same as previous ones. In code check the (CE, CS) pins.

General Problems:

Most of the problems with their solutions are mentioned above. Other common problems are:

1. To powerup nRF24L01+, we need to connect a capacitor (10 -100µF) between ground and 3.3V.
2. Header file #include<unistd.h> should be included if sleep() function is used.
3. Update and Upgrade your RPi to install latest packages.
4. Enable SPI configuration in the RaspberryPi to enable SPI protocol.