Amity Institute of Information Technology

3rd Floor, Amity University, Sec-125, Noida, U.P., India

NTCC Term Paper

Student Name	:	Arpit Agarwal
Enrollment Number	:	A1004815039
Program	:	BCA
Semester	:	2nd
Batch	:	2015-2018
Faculty Guide's Name	:	M/s. Ruchika Bathla
Project Title	:	Raspberry Pi

Installing Raspbian OS in Raspberry Pi

This is the first step everyone has to do in order to use the Raspberry Pi(RPi). This guide explains how to install Raspbian Operating System. RPi uses SD Card as primary storage device so we need SD Card of 4GB or more.

You need to have Raspbian image downloaded on to your computer. You can get the latest version from https://www.raspberrypi.org/downloads/raspbian/ . Download the full version not the Lite version. Extract the IMG file from the zip file. Get a SD card reader to insert SD card into your computer.

For Windows

- Insert the SD Card in your computer. Your SD card will be assigned drive letter. In my case `G:`
- Download SDFormatter from https://www.sdcard.org and format your SD card using this software
- Download Win32DiskImager utility from https://sourceforge.net/projects/win32diskimager/
- Choose the drive letter assigned to your SD card
- Browse the image file of the Raspbian which you downloaded
- Press the Write button

For Linux

- Open terminal and list all the mounted devices

```
$ df -h
```

- Insert the SD card into your computer. Your SD card will be mounted within a few seconds

- Again list all mounted devices

```
$ df -h
```

- You will see that few more partitions are listed this time. Note the device name it can be `/dev/mmcblk0p1` or `/dev/sdd1`. The last part `p1` or `1` is the partition number. In my case it is `/dev/mmcblk0p1` and `/dev/mmcblk0p2`
- We will have to unmount all the partitions of the SD card before writing the image.

```
$ umount /dev/mmcblk0p1
$ umount /dev/mmcblk0p2
```

- We want to write the image to the whole device not to one partition. So, you need to remove partition part from the device name example `/dev/mmcblk0`
- We will use `dd` tool preinstalled in Linux. Note that this tool can format any partition of your system so use it carefully
- Write the image onto SD card

```
 $ dd bs=4M if=2016-05-27-raspbian-jessie.img of=/dev/mmcblk0
```

- Please note that block size set to 4M will work most of the time; if not, please try 1M, although this will take considerably longer
- Also note that if you are not logged in as root you will need to prefix this with `sudo`
- `dd` command does not give any kind of information about the progress so it will appear as frozen
- After the command has successfully ran. Safely remove the SD card from your system

Now, you are ready with your Raspbian installed on your SD card. Same steps will be used to install any Raspberry Pi Linux Image on SD card.

SSH Connection

Plug SD card into RPi. Provide power supply to it. Plug the Ethernet cable into RPi connecting it with your Home network. If you do not have Monitor or LED for RPi then we will use SSH to remotely access RPi. Connect your system and RPi on same Network. Use `Angry IP Scanner` available at http://angryip.org/download/ to scan the network to find the connected devices with their respective allotted IP address. Once you have got the IP address assigned to RPi by your Router, use SSH to connect to it.

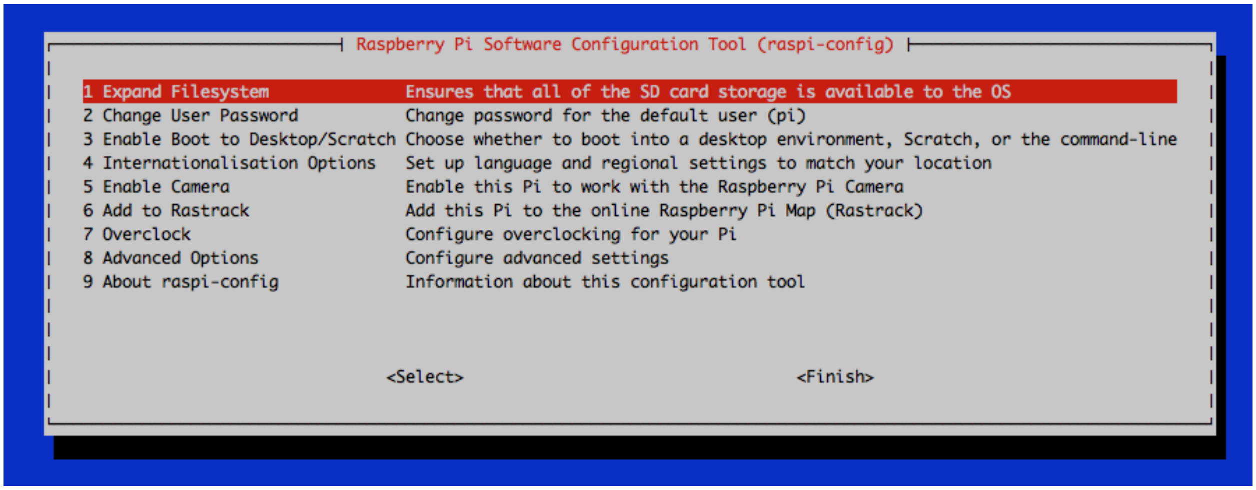
Windows users can use `PuTTY` available here http://www.putty.org . Linux users can use preinstalled tool `ssh` or `PuTTY` as you wish. The default username is “pi” and password is “raspberry”.

Configuration after First Boot

The first thing to do after first boot is to expand File System.

Start the Raspberry Configuration Tool

```
$ sudo raspi-config
```



```
Raspberry Pi Software Configuration Tool (raspi-config)

1 Expand Filesystem              Ensures that all of the SD card storage is available to the OS
2 Change User Password           Change password for the default user (pi)
3 Enable Boot to Desktop/Scratch Choose whether to boot into a desktop environment, Scratch, or the command-line
4 Internationalisation Options   Set up language and regional settings to match your location
5 Enable Camera                  Enable this Pi to work with the Raspberry Pi Camera
6 Add to Rastrack                Add this Pi to the online Raspberry Pi Map (Rastrack)
7 Overclock                      Configure overclocking for your Pi
8 Advanced Options               Configure advanced settings
9 About raspi-config             Information about this configuration tool

                   <Select>                                  <Finish>
```

Select Expand File System and Press Enter. Your File System will be expanded after a Reboot. You are done with installing Raspbian and Configuring it after First Boot. As you can see there are more options, you can try them if you wish.

Next thing is to make your RPi up-to-date,

```
$ sudo apt-get update
```

This will match the version of every installed software with the latest version available on their respective official websites. Then will start downloading and installing them one by one. Press ‘Y’ if prompted for confirmation.

```
$ sudo apt-get upgrade
```

Then check whether a distribution upgrade is available for your currently running Operating System on RPi.

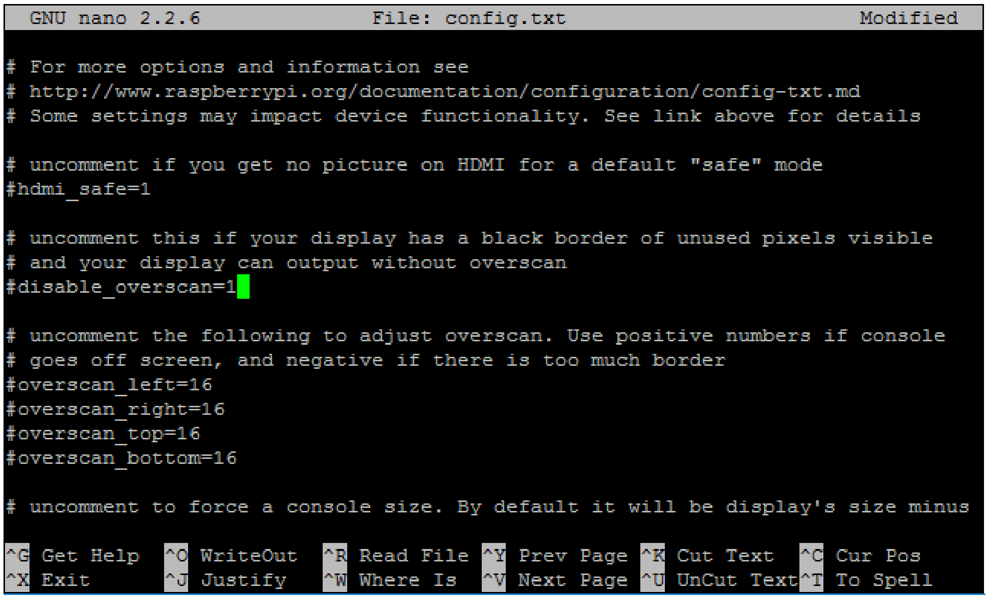
```
$ sudo apt-get dist-upgrade
```

Finally remove the programs which are no longer needed by Operating System.

```
$ sudo apt-get auto-remove
```

One thing I forgot, if you are connecting your RPi to monitor and your monitor is displaying it in wrong resolution or there are unused pixels left on both side and black border is coming, you can edit the `/boot/config.txt` file. You can use SSH with root privileges or simply plug the SD card in your system and you will find a partition named `boot` of around 63MB size will get mounted. Open it up and edit `/boot/config.txt` file.

```
$ sudo nano config.txt
```



```
# For more options and information see
# http://www.raspberrypi.org/documentation/configuration/config-txt.md
# Some settings may impact device functionality. See link above for details

# uncomment if you get no picture on HDMI for a default "safe" mode
#hdmi_safe=1

# uncomment this if your display has a black border of unused pixels visible
# and your display can output without overscan
#disable_overscan=1

# uncomment the following to adjust overscan. Use positive numbers if console
# goes off screen, and negative if there is too much border
#overscan_left=16
#overscan_right=16
#overscan_top=16
#overscan_bottom=16

# uncomment to force a console size. By default it will be display's size minus
```

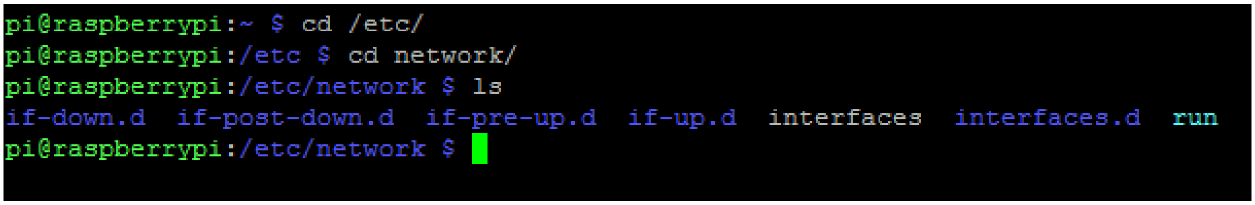
Just uncomment `#disable_overscan=1` line. It should look like `disable_overscan=1`.

Save the file. Press Ctrl+X then Y and then Enter. Now Reboot and your monitor will display it in full screen.

Setting up Wi-Fi Connection

You can purchase Wi-Fi adapter if you want to connect your RPi to Wireless Network. RPi 3 comes with inbuilt Wi-Fi chip from Broadcom. This resource will explain you how to connect to Wireless Network.

Connect to your RPi with SSH or with Monitor, Mouse, Keyboard. In terminal, go to `/etc/network` directory and list that directory. Here you will find a file named `interfaces`.

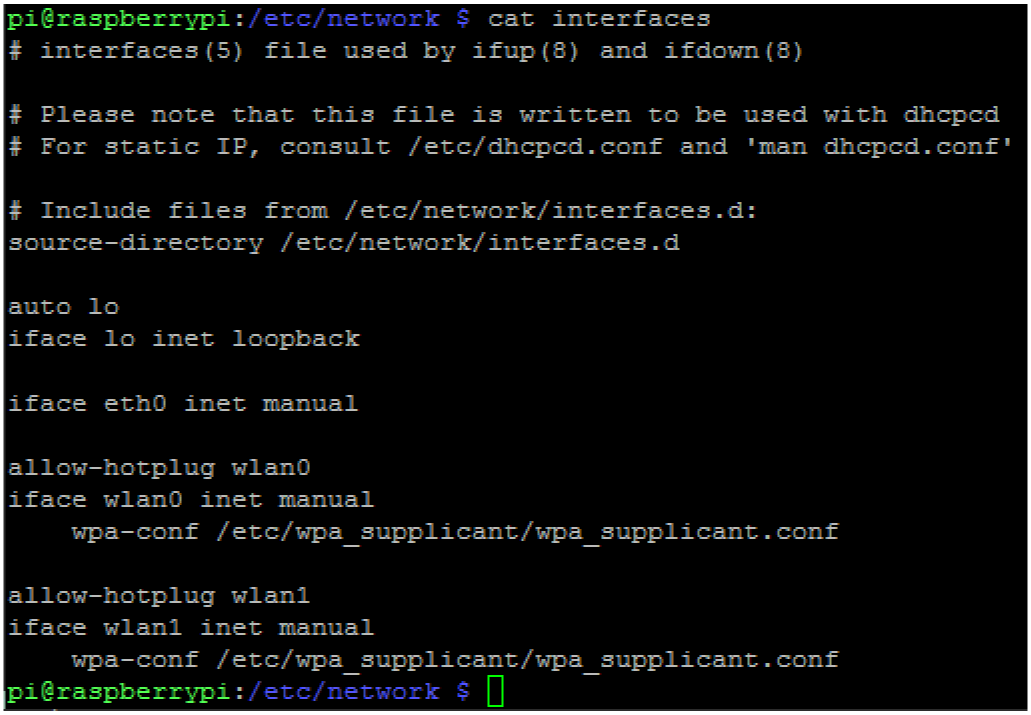


```
pi@raspberrypi:~ $ cd /etc/
pi@raspberrypi:/etc $ cd network/
pi@raspberrypi:/etc/network $ ls
if-down.d  if-post-down.d  if-pre-up.d  if-up.d  interfaces  interfaces.d  run
pi@raspberrypi:/etc/network $
```

List the contents of `interfaces` file

```
$ cat interfaces
```

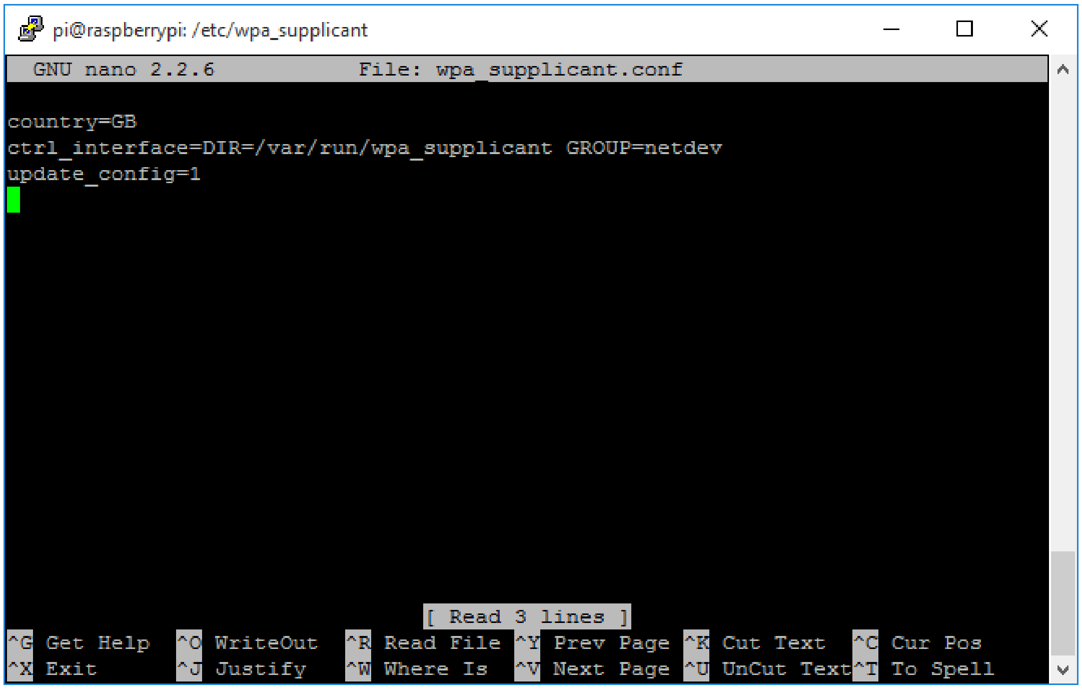
You will see something like this:



```
pi@raspberrypi:/etc/network $ cat interfaces
# interfaces(5) file used by ifup(8) and ifdown(8)

# Please note that this file is written to be used with dhcpcd
# For static IP, consult /etc/dhcpcd.conf and 'man dhcpcd.conf'

# Include files from /etc/network/interfaces.d:
source-directory /etc/network/interfaces.d

auto lo
iface lo inet loopback

iface eth0 inet manual

allow-hotplug wlan0
iface wlan0 inet manual
    wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf

allow-hotplug wlan1
iface wlan1 inet manual
    wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf
pi@raspberrypi:/etc/network $
```

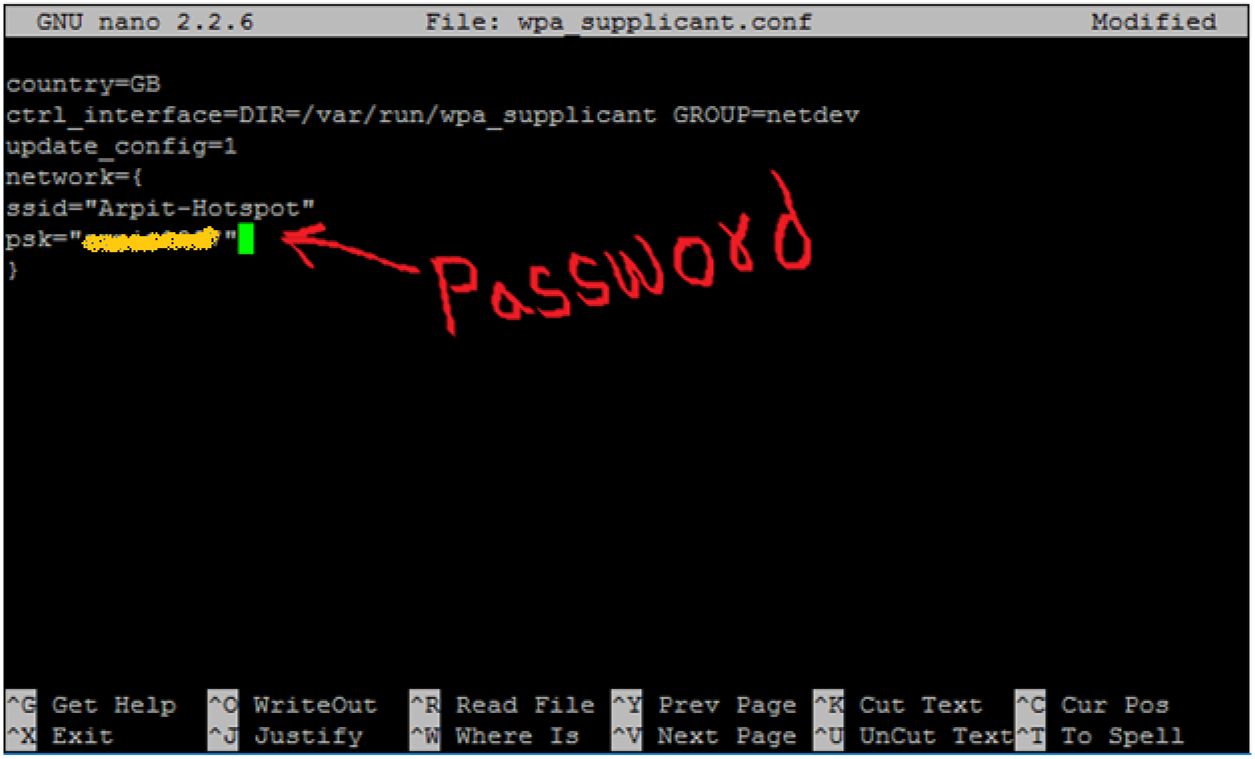
This is the network interface file. As you can see wlan0 interface has configuration file stored at `/etc/wpa_supplicant/wpa_supplicant.conf`. Forget wlan1. Open up the `wpa_supplicant.conf` in your favorite editor with root privileges.

File looks like this with `nano`:



```
  GNU nano 2.2.6              File: wpa_supplicant.conf

country=GB
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1

                              [ Read 3 lines ]
^G Get Help   ^O WriteOut   ^R Read File  ^Y Prev Page  ^K Cut Text   ^C Cur Pos
^X Exit       ^J Justify    ^W Where Is   ^V Next Page  ^U UnCut Text^T To Spell
```

Write the details of your Wi-Fi network so that `wlan0` can connect with it on boot. Make the file look like this:



```
country=GB
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
network={
ssid="Arpit-Hotspot"
psk="[redacted]"
}
```

You can add more networks also. Save the file. Now reboot your RPi and you will be able to connect to the specified Wireless Network.

```
$ sudo reboot
```

VNC Connection

In order to get VNC connection, you need to install a VNC server in RPi. You can install TightVNC but it creates a separate desktop for every connection. You can go with x11vnc, if you want single desktop to be shared among every VNC client and the RPi itself.

TightVNC Server

Install TightVNC Server

```
$ sudo apt-get install tightvncserver
```

Next run TightVNC Server which will prompt you for the password and an optional View Only password.

```
$ tightvncserver
```

Once that is done you can start a VNC server from shell prompt using below command:

```
$ vncserver :0 -geometry 1366x768 -depth 24
```

Run at Boot

Create a file in `/etc/init.d` with any name such as `vncserver` with following content:

```
#!/bin/sh
### BEGIN INIT INFO
# Provides: vncboot
# Required-Start: $remote_fs $syslog
# Required-Stop: $remote_fs $syslog
# Default-Start: 2 3 4 5
# Default-Stop: 0 1 6
# Short-Description: Start VNC Server at boot time
# Description: Start VNC Server at boot time.
### END INIT INFO

USER=root
HOME=/root

export USER HOME

case "$1" in
 start)
   echo "Starting VNC Server"
   #Insert your favoured settings for a VNC session
   /usr/bin/vncserver :0 -geometry 1280x800 -depth 16 -pixelformat rgb565
   ;;

 stop)
   echo "Stopping VNC Server"
   /usr/bin/vncserver -kill :0
   ;;

 *)
   echo "Usage: /etc/init.d/vncboot {start|stop}"
   exit 1
   ;;
esac

exit 0
```

Make the file executable using this command

```
$ chmod 755 /etc/init.d/vncboot
```

Enable dependency based boot sequencing

```
$ update-rc.d /etc/init.d/vncboot defaults
```

If enabling dependency based boot sequencing was successful, it says

```
$ update-rc.d: using dependency based boot sequencing
```

But if it says

```
$ update-rc.d: error: unable to read
/etc/init.d//etc/init.d/vncboot
```

then try the following command

```
$ update-rc.d vncboot defaults
```

X11VNC Server

Install x11vnc Server

```
$ sudo apt-get install x11vnc
```

Store Password

```
$ x11vnc -storepasswd
```

Run the server from shell prompt with password

```
$ x11vnc -usrpw -display :0
```

Run at Boot

Make a directory if not already made

```
$ mkdir ~/.config/autostart
```

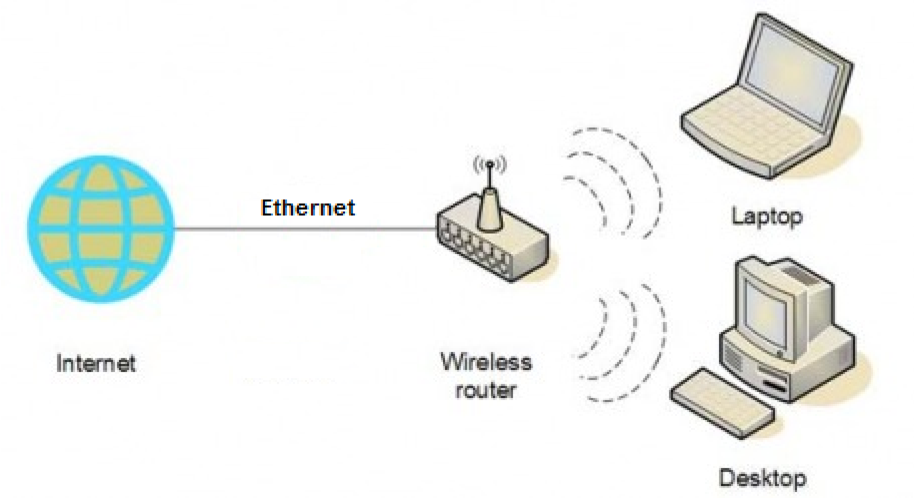
Create a file named `x11vnc.desktop` or any with the following content

```
[Desktop Entry]
Encoding=UTF-8
Type=Application
Name=X11VNC
Exec=x11vnc -forever -usepw -display :0
StartupNotify=false
Terminal=false
Hidden=false
```

Now reboot your RPi, and you will have a VNC Server ready.

Wired-to-Wi-Fi Router

Yes, you are thinking right. You can make RPi a Wireless Router. Below image depicts the concept of Wi-Fi Router.



Let’s get started, I am using Raspberry Pi 3 which comes with inbuilt BCM4334 Chip. This chip is supported by open-source brcmfmac driver.

- We will use `hostapd` and `dnsmasq` for this purpose

  hostapd – This package allows you to use RPi’s Wi-Fi as an access point
  dnsmasq – This is combined DHCP and DNS server which is easy to configure

  If you want something a little more 'heavyweight', you can use `isc-dhcp-server` and `bind9` packages for DHCP and DNS respectively, but for our purposes, `dnsmasq` works just fine.

```
$ sudo apt-get install hostapd dnsmasq
```

- We need to configure interfaces. We will assign a static IP address to `wlan0` which will be used as gateway. Open the `interfaces` file

```
$ sudo nano /etc/network/interfaces
```

Edit the `wlan0` section like this:

```
allow-hotplug wlan0
iface wlan0 inet static
    address 192.168.2.1
    netmask 255.255.255.0
    network 192.168.2.0
    broadcast 192.168.2.255
#wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf
```

- Next, we need to configure `hostapd` to create the access point. Create a configuration file and open it with `nano` editor at the same time

```
$ sudo nano /etc/hostapd/hostapd.conf
```

Add the following content in it:

```
#This is the name of the interface we just configured
interface=wlan0
# Use the nl80211 driver with the brcmfmac driver
driver=nl80211
# This is the name of the network
ssid=Arpit-Raspberry
# Use the 2.4GHz band
hw_mode=g
# Use channel 6
channel=6
# Enable 802.11n
ieee80211n=1
# Enable WMM
wmm_enabled=1
# Enable 40MHz channels with 20ns guard interval
ht_capab=[HT40][SHORT-GI-20][DSSS_CCK-40]
# Accept all MAC addresses
macaddr_acl=0
# Use WPA authentication
auth_algs=1
# Require clients to know the network name
ignore_broadcast_ssid=0
# Use WPA2
wpa=2
```

```
# Use a pre-shared key
wpa_key_mgmt=WPA-PSK
# The network passphrase
wpa_passphrase=arpit1997
# Use AES, instead of TKIP
rsn_pairwise=CCMP
```

We can check if it's working at this stage by running `sudo /usr/sbin/hostapd /etc/hostapd/hostapd.conf`. If it's all gone well thus far, you should be able to see to the network **Arpit-Raspberry**! If you try connecting to it, you will see some output from the Pi, but you won't receive and IP address until we set up `dnsmasq` in the next step. Use **Ctrl+C** to stop it.

- Next, we need to tell `hostapd` where to look for configuration file when it starts up on boot. Open up the default configuration file of `hostapd`

```
$ sudo nano /etc/default/hostapd
```

Replace the line `#DAEMON_CONF=""` with `DAEMON_CONF="/etc/hostapd/hostapd.conf"`

- Till here, the Wi-Fi access point will start on boot but no user can connect to it because they will not get IP address. We will now configure `dnsmasq` for this purpose.
  The shipped `dnsmasq` file contains a lot of information on how to use it. So, I will advise to move it and create a new one.

```
$ sudo mv /etc/dnsmasq.conf /etc/dnsmasq.conf.orig
$ sudo nano /etc/dnsmasq.conf
```

Paste the following into the new file

```
interface=wlan0      # Use interface wlan0
listen-address=192.168.2.1 # listen on
# Bind to the interface to make sure we aren't sending things
# elsewhere
bind-interfaces
server=8.8.8.8       # Forward DNS requests to Google DNS
domain-needed        # Don't forward short names
# Never forward addresses in the non-routed address spaces.
bogus-priv
# Assign IP addresses between 192.168.2.2 and 192.168.2.100 with a
# 12 hours lease time
dhcp-range=192.168.2.2,192.168.2.100,12h
```

- Now, the device can easily connect to Raspberry Wi-Fi access point but it will not be able to access Internet.

  Open the `sysctl.conf` file

```
$ sudo nano /etc/sysctl.conf
```

  Remove the `#` from the beginning of the line containing `net.ipv4.ip_forward=1`
  This will enable packet forwarding on next reboot. But if you want to try it right now without reboot then do this.

```
$ sudo sh -c "echo 1 > /proc/sys/net/ipv4/ip_forward"
```

- We also need to share RPi's internet connection with the devices connected over Wi-Fi. We will configure a NAT between `eth0` and `wlan0`. Before applying new rules, you have to first delete any existing rules using the commands `sudo iptables -F` and `sudo iptables -t nat -F`

```
$ sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
$ sudo iptables -A FORWARD -i eth0 -o wlan0 -m state --state
RELATED,ESTABLISHED -j ACCEPT
$ sudo iptables -A FORWARD -i wlan0 -o eth0 -j ACCEPT
```

  However, we need these rules to be applied every time we reboot the Pi, so run `sudo sh -c "iptables-save > /etc/iptables.ipv4.nat"` to save the rules to the file `/etc/iptables.ipv4.nat`. Now we need to run this after each reboot, so open the `/etc/rc.local` file with `sudo nano /etc/rc.local` and just above the line `exit 0`, add the following line:

```
$ iptables-restore < /etc/iptables.ipv4.nat
```

- And that's all! Now just Reboot your RPi and you will be able to access Internet

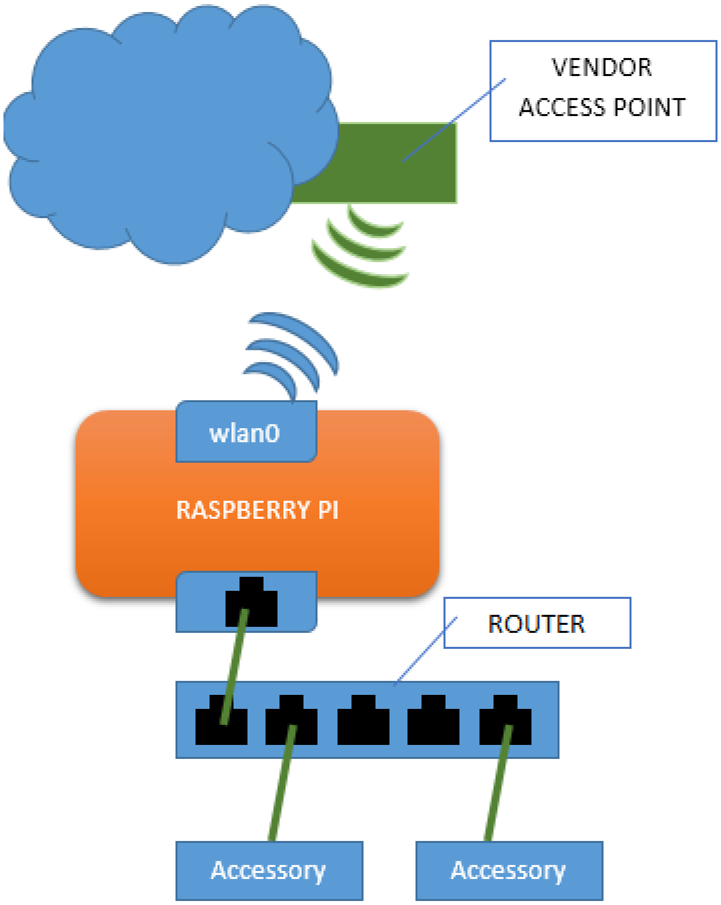
```
$ sudo reboot
```

Wireless-to-Wired Router

This can also be called an Ethernet Router. Suppose, you have a Broadband connection of any company and you have many users connected to a Router. Things are running smoothly and suddenly you got a 3G/4G SIM providing more Bandwidth than your current Broadband with more speed and at less cost. You will obviously think to better use Mobile data than Broadband but how will you shift!!! Here comes the idea of using Raspberry Pi as Wired Router.

You can also use it to provide Wi-Fi capabilities to the system which needs internet connection from Wireless Network and do not have inbuilt Wi-Fi like Laptop.

Below image can explain you the basic concept of this.



You need to have a working Internet Connection on RPi through Wi-Fi. See my Wi-Fi Connection topic if you don’t know how to configure Wi-Fi on RPi.

- We will use `dnsmasq` package for this purpose because it is combined DHCP and DNS server and also easy to configure.
  If you want something a little more 'heavyweight', you can use the `isc-dhcp-server` and `bind9` packages for DHCP and DNS respectively, but for our purposes, `dnsmasq` works just fine.

```
$ sudo apt-get install dnsmasq
```

- We need to configure interfaces. We will assign a static IP address to `eth0` which will be used as gateway. Open the `interfaces` file

```
$ sudo nano /etc/network/interfaces
```

Edit the `eth0` section like this:

```
allow-hotplug eth0
iface eth0 inet static
    address 192.168.2.1
    netmask 255.255.255.0
    network 192.168.2.0
    broadcast 192.168.2.255
```

- Next, we will configure `dnsmasq`. The shipped `dnsmasq` file contains a lot of information on how to use it. So, I will advise to move it and create a new one.

```
$ sudo mv /etc/dnsmasq.conf /etc/dnsmasq.conf.orig
$ sudo nano /etc/dnsmasq.conf
```

Paste the following into the new file

```
interface=eth0      # Use interface eth0
listen-address=192.168.2.1 # listen on
# Bind to the interface to make sure we aren't sending things
# elsewhere
bind-interfaces
server=8.8.8.8       # Forward DNS requests to Google DNS
domain-needed        # Don't forward short names
# Never forward addresses in the non-routed address spaces.
bogus-priv
# Assign IP addresses between 192.168.2.2 and 192.168.2.100 with a
# 12 hours lease time
dhcp-range=192.168.2.2,192.168.2.100,12h
```

- Edit the `/etc/sysctl.conf` file to enable packet forwarding

```
$ sudo nano /etc/sysctl.conf
```

Remove the `#` from the beginning of the line containing `net.ipv4.ip_forward=1`
This will enable packet forwarding on next reboot. But if you want to try it right now without reboot then do this.

```
$ sudo sh -c "echo 1 > /proc/sys/net/ipv4/ip_forward"
```

- We also need to share RPi’s internet connection with the devices connected over Wi-Fi. We will configure a NAT between `eth0` and `wlan0`. Before applying new rules, you have to first delete any existing rules using the commands `sudo iptables -F` and `sudo iptables -t nat -F`

```
$ sudo iptables -t nat -A POSTROUTING -o wlan0 -j MASQUERADE
$ sudo iptables -A FORWARD -i wlan0 -o eth0 -m state --state
RELATED,ESTABLISHED -j ACCEPT
$ sudo iptables -A FORWARD -i eth0 -o wlan0 -j ACCEPT
```

However, we need these rules to be applied every time we reboot the Pi, so run `sudo sh -c "iptables-save > /etc/iptables.ipv4.nat"` to save the rules to the file `/etc/iptables.ipv4.nat`. Now we need to run this after each reboot, so open the `/etc/rc.local` file with `sudo nano /etc/rc.local` and just above the line `exit 0`, add the following line:

```
$ iptables-restore < /etc/iptables.ipv4.nat
```

- And that’s all! Now just Reboot your RPi and you will be able to access Internet

```
$ sudo reboot
```

Tor Router

In this article, I will explain you how to make your RPi work as a Tor Router. It will help you to browse Internet anonymously by encrypting the traffic. To get started, we have to first convert RPi to Wi-Fi Router then we will configure Tor. You need a working Internet connection on your RPi. Connect to it using SSH.

Let’s get started, I am using Raspberry Pi 3 which comes with inbuilt BCM4334 Chip. This chip is supported by open-source brcmfmac driver.

- We will use `hostapd` and `dnsmasq` for this purpose

  hostapd – This package allows you to use RPi’s Wi-Fi as an access point
  dnsmasq – This is combined DHCP and DNS server which is easy to configure

  If you want something a little more 'heavyweight', you can use `isc-dhcp-server` and `bind9` packages for DHCP and DNS respectively, but for our purposes, `dnsmasq` works just fine.

```
$ sudo apt-get install hostapd dnsmasq
```

- We need to configure interfaces. We will assign a static IP address to `wlan0` which will be used as gateway. Open the `interfaces` file

```
$ sudo nano /etc/network/interfaces
```

Edit the `wlan0` section like this:

```
allow-hotplug wlan0
iface wlan0 inet static
    address 192.168.2.1
    netmask 255.255.255.0
    network 192.168.2.0
    broadcast 192.168.2.255
#wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf
```

- Next, we need to configure `hostapd` to create the access point. Create a configuration file and open it with `nano` editor at the same time

```
$ sudo nano /etc/hostapd/hostapd.conf
```

Add the following content in it:

```
#This is the name of the interface we just configured
interface=wlan0
# Use the nl80211 driver with the brcmfmac driver
driver=nl80211
# This is the name of the network
ssid=Arpit-Raspberry
# Use the 2.4GHz band
hw_mode=g
# Use channel 6
channel=6
# Enable 802.11n
ieee80211n=1
# Enable WMM
wmm_enabled=1
# Enable 40MHz channels with 20ns guard interval
ht_capab=[HT40][SHORT-GI-20][DSSS_CCK-40]
# Accept all MAC addresses
macaddr_acl=0
# Use WPA authentication
auth_algs=1
# Require clients to know the network name
```

```
ignore_broadcast_ssid=0
# Use WPA2
wpa=2
# Use a pre-shared key
wpa_key_mgmt=WPA-PSK
# The network passphrase
wpa_passphrase=arpit1997
# Use AES, instead of TKIP
rsn_pairwise=CCMP
```

We can check if it's working at this stage by running `sudo /usr/sbin/hostapd /etc/hostapd/hostapd.conf`. If it's all gone well thus far, you should be able to see to the network **Arpit-Raspberry**! If you try connecting to it, you will see some output from the Pi, but you won't receive and IP address until we set up `dnsmasq` in the next step. Use **Ctrl+C** to stop it.

- Next, we need to tell `hostapd` where to look for configuration file when it starts up on boot. Open up the default configuration file of `hostapd`

```
$ sudo nano /etc/default/hostapd
```

Replace the line `#DAEMON_CONF=""` with `DAEMON_CONF="/etc/hostapd/hostapd.conf"`

- Till here, the Wi-Fi access point will start on boot but no user can connect to it because they will not get IP address. We will now configure `dnsmasq` for this purpose.
  The shipped `dnsmasq` file contains a lot of information on how to use it. So, I will advise you to move it and create a new one.

```
$ sudo mv /etc/dnsmasq.conf /etc/dnsmasq.conf.orig
$ sudo nano /etc/dnsmasq.conf
```

Paste the following into the new file

```
interface=wlan0      # Use interface wlan0
listen-address=192.168.2.1 # listen on
# Bind to the interface to make sure we aren't sending things
# elsewhere
bind-interfaces
server=8.8.8.8       # Forward DNS requests to Google DNS
domain-needed        # Don't forward short names
# Never forward addresses in the non-routed address spaces.
bogus-priv
```

```
# Assign IP addresses between 192.168.2.2 and 192.168.2.100 with a
# 12 hours lease time
dhcp-range=192.168.2.2,192.168.2.100,12h
```

- Now, the device can easily connect to Raspberry Wi-Fi access point but it will not be able to access Internet.

  Open the `sysctl.conf` file

```
$ sudo nano /etc/sysctl.conf
```

Remove the `#` from the beginning of the line containing `net.ipv4.ip_forward=1`
This will enable packet forwarding on next reboot. But if you want to try it right now without reboot then do this.

```
$ sudo sh -c "echo 1 > /proc/sys/net/ipv4/ip_forward"
```

- We also need to configure `iptables` to route all `wlan0` traffic to port 9040 which we will use in configuring `tor` later. Before applying new rules, you have to first delete any existing rules using the commands `sudo iptables -F` and `sudo iptables -t nat -F`

```
$ sudo iptables -t nat -A PREROUTING -i wlan0 -p tcp --dport 22 -j
REDIRECT --to-ports 22
$ sudo iptables -t nat -A PREROUTING -i wlan0 -p udp --dport 53 -j
REDIRECT --to-ports 53
$ sudo iptables -t nat -A PREROUTING -i wlan0 -p tcp --syn -j
REDIRECT --to-ports 9040
```

However, we need these rules to be applied every time we reboot the Pi, so run `sudo sh -c "iptables-save > /etc/iptables.ipv4.nat"` to save the rules to the file `/etc/iptables.ipv4.nat`. Now we need to run this after each reboot, so open the `/etc/rc.local` file with `sudo nano /etc/rc.local` and just above the line `exit 0`, add the following line:

```
$ iptables-restore < /etc/iptables.ipv4.nat
```

- Next, we will install `tor`

```
$ sudo apt-get install tor
```

- Tor configuration is stored in `/etc/tor/torrc` file but I will recommend to make a copy of it because it contains lots of information on how to use it.

```
$ sudo mv /etc/tor/torrc /etc/tor/torrc.orig
```

- Open `/etc/tor/torrc` file with `nano`

```
$ sudo nano /etc/tor/torrc
```

Write the following content in it

```
VirtualAddrNetwork 10.192.0.0/10

AutomapHostsSuffixes .onion,.exit

AutomapHostsOnResolve 1

TransPort 9040

TransListenAddress 192.168.2.1

DNSPort 53

DNSListenAddress 192.168.2.1
```

- One last step is to handle `log` file. Tor appends the new logs into the existing log file which increases the size of it every time you use it.

  Default location of `log` file is `/var/log/tor/`. We have to delete this file every time. So, to get this done on every boot. Open `/etc/init.d/tor` file.

```
$ sudo nano /etc/init.d/tor
```

Add `sudo rm -rf $TORLOGDIR` line immediately after `TORLOGDIR=/var/log/tor` line.

This section should look like this

```
.....................

DAEMON=/usr/bin/tor

NAME=tor

DESC="tor daemon"

TORLOGDIR=/var/log/tor

sudo rm -rf $TORLOGDIR
```

```
TORPIDDIR=/var/run/tor

TORPID=$TORPIDDIR/tor.pi

…………………
```

- Everything is done now. Just `reboot` and you will be able to use Internet anonymously on the devices connected to your RPi Wi-Fi access point.

```
$ sudo reboot
```

Go to https://check.torproject.org on your device and you will see “Congratulation” message.