Part 07
-
Activating WiFi

Version: 2023-02-16

Getting wireless LAN network

Starting from Raspberry Pi 3, WiFi hardware is on-board and a WiFi dongle is no longer needed. Should you however prefer a Wi-Fi dongle, see later in this document

Important:
Your WiFi router or Access Point should be setup for **WPA2 only** and not WPA/WPA2. The last setting might not garantuee a probleem-free connectivity**.**

This tutorial works best if your router is broadcasting the SSID. Make sure you have "Broadcast SSID" set up on your router.

Basic setup
On the Raspberry Pi 3B+ and Raspberry Pi 4B, you will also need to set the country code, so that the 5GHz networking can choose the correct frequency bands.
But also, with the latest Buster Raspberry Pi OS release, you must ensure that the `wpa_supplicant.conf` file contains the following information at the top

```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
```

You can edit the `wpa_supplicant.conf` file and add the following. (Note: you need to replace 'BE' with the 2 letter ISO code of your country. See Wikipedia for a list of 2 letter ISO 3166-1 country codes.)

```
sudo nano /etc/wpa_supplicant/wpa_supplicant.conf
```

Make sure it looks like this

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

Save and reboot

```
sudo reboot
```

Checking bands ands channels

```
iw phy phy0 channels
```

You should see a list of bands and their channels

```
Band 1:
        * 2412 MHz [1]
          Maximum TX power: 20.0 dBm
          Channel widths: 20MHz HT40+
        * 2417 MHz [2]
          Maximum TX power: 20.0 dBm
          Channel widths: 20MHz HT40+
        ...

Band 2:
        * 5170 MHz [34] (disabled)
        * 5180 MHz [36]
          Maximum TX power: 20.0 dBm
          Channel widths: 20MHz HT40+ VHT80
        * 5190 MHz [38] (disabled)
        ...
```

Scan the air
To scan for wireless networks, use the command

```
sudo iwlist wlan0 scan
```

This will list all available wireless networks, along with other useful information. Look out for:

- 'ESSID:"testing"' is the name of the wireless network. Ours is called ‘testing’
- 'IE: IEEE 802.11i/WPA2 Version 1' is the authentication used. In this case it's WPA2, the newer and more secure wireless standard which replaces WPA. This guide should work for WPA or WPA2, but may not work for WPA2 enterprise. For WEP hex keys, see the last example here. You'll also need the password for the wireless network. For most home routers, this is found on a sticker on the back of the router. The ESSID (ssid) for the examples below is testing and the password (psk) is testingPassword.

To list only what you need to know, use this command

```
sudo iwlist wlan0 scan | grep “ESSID\|Frequency\|IEEE 802.11i”
```

Adding the network details to the Raspberry Pi
Open the `wpa-supplicant` configuration file in nano

```
sudo nano /etc/wpa_supplicant/wpa_supplicant.conf
```

Go to the bottom of the file and add the following:

```
network={
    ssid="testing"
    psk="testingPassword"
}
```

The password can be configured either as the ASCII representation, in quotes as per the example above.

The password can also be pre-encrypted 32 byte hexadecimal number. You can use the `wpa_passphrase` utility to generate an encrypted PSK. This takes the SSID and the password, and generates the encrypted PSK. With the example from above, you can generate the PSK with

```
wpa_passphrase "testing"
```

Then you will be asked for the password of the wireless network (in this case `testingPassword`). The output is as follows

```
network={
    ssid="testing"
    #psk="testingPassword"
    psk=131e1e221f6e06e3911a2d11ff2fac9182665c004de85300f9cac208a6a80531
}
```

Note that the plain text version of the code is present, but commented out. You should delete this line from the final `wpa_supplicant` file for extra security.

The `wpa_passphrase` tool requires a password with between 8 and 63 characters. To use a more complex password, you can extract the content of a text file and use it as input for `wpa_passphrase`. Store the password in a text file and input it to `wpa_passphrase` by calling `wpa_passphrase "testing" < file_where_password_is_stored`. For extra security, you should delete the `file_where_password_is_stored` afterwards, so there is no plain text copy of the original password on the system.

To use the wpa_passphrase–encrypted PSK, you can either copy and paste the encrypted PSK into the `wpa_supplicant.conf` file, or redirect the tool's output to the configuration file in one of two ways:

- Either change to root by executing

  ```
  sudo su
  ```

  then call

  ```
  wpa_passphrase "testing" >> /etc/wpa_supplicant/wpa_supplicant.conf
  ```

  and enter the testing password when asked

- Or use

  ```
  wpa_passphrase "testing" | sudo tee -a /etc/wpa_supplicant/wpa_supplicant.conf >/dev/null
  ```

  and enter the testing password when asked. The redirection to `/dev/null` prevents `tee` from **also** outputting to the screen (standard output).

If you want to use one of these two options, make sure you use `>>`, or use `-a` with `tee`, either will append text to an existing file. Using a single chevron `>`, or omitting `-a` when using tee, will erase all contents and then append the output to the specified file.

Reconfigure the interface with

```
wpa_cli -i wlan0 reconfigure.
```

You can verify whether it has successfully connected using

```
iwconfig
```

and

```
ifconfig wlan0
```

If the `inet addr` field has an address beside it, the Raspberry Pi has connected to the network. If not, check that your password and ESSID are correct.

On the Raspberry Pi 3B+ and Raspberry Pi 4B, you will also need to set the country code, so that the 5GHz networking can choose the correct frequency bands.
You can edit the `wpa_supplicant.conf` file and add the following. (Note: you need to replace 'BE' with the 2 letter ISO code of your country. See Wikipedia for a list of 2 letter ISO 3166-1 country codes.)

```
country=BE
```

Note that with the latest Buster Raspberry Pi OS release, you must ensure that the `wpa_supplicant.conf` file contains the following information at the top

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

So, a correct `wpa_supplicant.conf` file should look like this

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

network={
    ssid="testing"
    psk="testingPassword"
}
```

or

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

network={
    ssid="testing"
    psk=131e1e221f6e06e3911a2d11ff2fac9182665c004de85300f9cac208a6a80531
}
```

Unsecured networks
If the network you are connecting to does not use a password, the `wpa_supplicant` entry for the network will need to include the correct `key_mgmt` entry. e.g.

```
network={
    ssid="testing"
    key_mgmt=NONE
}
```

Hidden networks
If you are using a hidden network, an extra option in the `wpa_supplicant` file, `scan_ssid`, may help connection.

```
network={
    ssid="yourHiddenSSID"
    scan_ssid=1
    psk="Your_wireless_network_password"
}
```

You can verify whether it has successfully connected using `iwconfig` and `ifconfig wlan0`. If the `inet addr` field has an address beside it, the Raspberry Pi has connected to the network. If not, check your password and ESSID are correct.

Adding multiple wireless network configurations
On recent versions of Raspberry Pi OS, it is possible to set up multiple configurations for wireless networking. For example, you could set up one for home and one for school.
For example

```
network={
    ssid="SchoolNetworkSSID"
    psk="passwordSchool"
    id_str="school"
}

network={
    ssid="HomeNetworkSSID"
    psk="passwordHome"
    id_str="home"
}
```

If you have two networks in range, you can add the priority option to choose between them. The network in range, with the highest priority, will be the one that is connected.

```
network={
    ssid="HomeOneSSID"
    psk="passwordOne"
    priority=1
    id_str="homeOne"
}

network={
    ssid="HomeTwoSSID"
    psk="passwordTwo"
    priority=2
    id_str="homeTwo"
}
```

Connecting to specific AP
If you want to connect to a specific Acces Point, find out the MAC address of that AP and add it to the config. Eg: AP with MAC-address `AC:84:C6:27:EE:2A`

```
network={
    ...
    bssid=AC:84:C6:27:EE:2A
    ...
}
```

If you want to prevent it connects to a specific AP, find out the MAC address of that AP and add it to the config. Eg: AP with MAC-address `8E:FC:A6:10:02:58`

```
network={
    ...
    bssid_blacklist=8E:FC:A6:10:02:58
    ...
}
```

You can combine both as well

```
network={
    ...
    bssid=AC:84:C6:27:EE:2A
    bssid_blacklist=8E:FC:A6:10:02:58
    ...
}
```

Connecting to specific band/frequency
If you want to specilically connect to a band/frequency, you can run following command

```
iw wlan0 scan | grep -A5 'freq: 5'
```

and you get a list like this

```
freq: 5180
beacon interval: 100 TUs
capability: ESS Privacy SpectrumMgmt ShortSlotTime (0x0511)
signal: -59.00 dBm
last seen: 0 ms ago
SSID: Engrie
```

Now you can specify the `wpa_supplicant` entry freq_list to only go for that specific one

```
freq_list=5180
```

So your `wpa_supplicant.conf` file should look like this

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

freq_list=5180

network={
    ssid="testing"
    psk="testingPassword"
}
```

Note:

- you can add more channels by listing them leaving a space in between. Eg.

```
freq_list=2412 2417 2422 2427 2432
```

- you can connect to only 2.4 Ghz band by specifying (for Belgium)

```
freq_list=2412 2417 2422 2427 2432 2437 2442 2447 2452 2457 2462 2467 2472
```

- you can connect to only 5 Ghz band by specifying (for Belgium)

```
freq_list=5180 5200 5220 5240 5260 5280 5300 5320 5500 5520 5540 5560 5580 5600 5620 5640 5660 5680 5700
```

Connecting to a channel
If you want connect to a channel, you can first list all channels seen by your Pi

```
iwlist wlan0 channel
```

and get a list like this

```
wlan0     32 channels in total; available frequencies :
          Channel 01 : 2.412 GHz
          Channel 02 : 2.417 GHz
          Channel 03 : 2.422 GHz
          Channel 04 : 2.427 GHz
          Channel 05 : 2.432 GHz
          Channel 06 : 2.437 GHz
          Channel 07 : 2.442 GHz
          Channel 08 : 2.447 GHz
          Channel 09 : 2.452 GHz
          Channel 10 : 2.457 GHz
          Channel 11 : 2.462 GHz
          Channel 12 : 2.467 GHz
          Channel 13 : 2.472 GHz
          Channel 36 : 5.18 GHz
          Channel 40 : 5.2 GHz
          Channel 44 : 5.22 GHz
          Channel 48 : 5.24 GHz
          Channel 52 : 5.26 GHz
          Channel 56 : 5.28 GHz
          Channel 60 : 5.3 GHz
          Channel 64 : 5.32 GHz
          Channel 100 : 5.5 GHz
          Channel 104 : 5.52 GHz
          Channel 108 : 5.54 GHz
          Channel 112 : 5.56 GHz
          Channel 116 : 5.58 GHz
          Channel 120 : 5.6 GHz
          Channel 124 : 5.62 GHz
          Channel 128 : 5.64 GHz
          Channel 132 : 5.66 GHz
          Channel 136 : 5.68 GHz
          Channel 140 : 5.7 GHz
          Current Frequency:2.462 GHz (Channel 11)
```

Next, you can use the band/frequency method to connect to a specific channel

Set up a static ip for your Raspberry Pi

For specific reasons, your Raspberry Pi might need a fixed/static IP address. For this you need to modify the file `dhcpcd.conf` located in the `/etc` folder and add the lines below.
This requires access to the main partition of the SD card, which mean you need a Linux or Mac OS X computer. If you only have a Windows computer, then first connect the Raspberry Pi via DHCP (as above), find the ip (eg. from your router or via Angry IP Scan tool), and do the following steps directly on the Raspberry via SSH connection.

```
sudo nano /etc/dhcpcd.conf

interface eth0
static ip_address = 192.168.1.100 / 24
static routers = 192.168.1.1
static domain_name_servers = 192.168.1.1

interface wlan0
static ip_address = 192.168.1.101 / 24
static routers = 192.168.1.1
static domain_name_servers = 192.168.1.1
```

Let's take a look at all this, here `interface eth0` corresponds to a connection of wired type and `interface wlan0` to a Wi-fi connection. So you have to choose the one that corresponds to your setup.
`static ip_address` is used to assign the ip that your Raspberry Pi will have. Generally the ip is of type `192.168.1.x` , replace the x with the value of your choice, be careful not to conflict with other devices .
For `static router` (also known as `gateway`) and `static domain_name_servers` (also known as DNS servers), this is the ip of your router, which is `192.168.1.1` .

Save the file. The SD card is now ready to be inserted into the Raspberry Pi or reboot your Pi.

Controlling WiFi power management
By default, power manangement on WiFi is on. You can see that by typing the command

```
iwconfig wlan0

wlan0     IEEE 802.11  ESSID:"Engrie-IoT"
          Mode:Managed  Frequency:2.462 GHz  Access Point: CC:40:D0:5D:13:A2
          Bit Rate=72.2 Mb/s   Tx-Power=31 dBm
          Retry short limit:7   RTS thr:off   Fragment thr:off
          Encryption key:off
          Power Management:on
          Link Quality=57/70  Signal level=-53 dBm
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:1  Invalid misc:0   Missed beacon:0
```

Using the command

```
iwconfig wlan0 power off
```

you turn off power management

```
wlan0     IEEE 802.11  ESSID:"Engrie-IoT"
          Mode:Managed  Frequency:2.462 GHz  Access Point: CC:40:D0:5D:13:A2
          Bit Rate=72.2 Mb/s   Tx-Power=31 dBm
          Retry short limit:7   RTS thr:off   Fragment thr:off
          Encryption key:off
          Power Management:off
          Link Quality=57/70  Signal level=-53 dBm
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:1  Invalid misc:0   Missed beacon:0
```

but once you reboot, power management will be on again. To make this off at every boot, add the command in `rc.local` like this

```
nano /etc/rc.local


#!/bin/sh -e
#
# rc.local
#
# This script is executed at the end of each multiuser runlevel.
# Make sure that the script will "exit 0" on success or any other
# value on error.
#
# In order to enable or disable this script just change the execution
# bits.
#
# By default this script does nothing.

# Print the IP address
_IP=$(hostname -I) || true
if [ "$_IP" ]; then
  printf "My IP address is %s\n" "$_IP"
fi
iwconfig wlan0 power off
exit 0
```

Ad-hoc WiFi Network

If you want to create an ad-hoc wifi network between your pi and your computer or phone to be able to ssh directly into it without the proximity of a router, execute the following command lines. We Assume your WiFi dongle is recognised as `wlan0`

```
ifconfig wlan0 up
iwconfig wlan0 mode ad-hoc
iwconfig wlan0 essid "Pi"
ifconfig wlan0 192.168.1.1 netmask 255.255.255.0
```

Add all of the above to /etc/rc.local to make it permanent.

```
sudo nano /etc/rc.local
```

Add the lines:

```
#!/bin/sh -e
#
# rc.local
#
# This script is executed at the end of each multiuser runlevel.
# Make sure that the script will "exit 0" on success or any other
# value on error.
#
# In order to enable or disable this script just change the execution
# bits.
#
# By default this script does nothing.

# Print the IP address
_IP=$(hostname -I) || true
if [ "$_IP" ]; then
  printf "My IP address is %s\n" "$_IP"
fi

ifconfig wlan0 up
iwconfig wlan0 mode ad-hoc
iwconfig wlan0 essid "Pi"
ifconfig wlan0 192.168.1.1 netmask 255.255.255.0

exit 0
```

When you have finished press [Ctrl] + x. This will ask if you want to save the modified files. Press 'Y' and then hit [Return] to save the file with the same name.

Reboot your Pi

```
sudo reboot
```

Note: all commands in rc.local are being executed as root.

How to set up a wireless ad-hoc network, but only if it is not already connected

This will show you how to set up a wireless ad-hoc network, but only if it not already connected. The wireless hotspot will also Dynamically assign IP Addresses to clients that connect.

1. Install the packages hostapd (WiFi hotspot manager) and udhcpd (dhcp server that will hand out the IP Addresses to clients that connect)

```
sudo apt-get –y install hostapd
sudo apt-get –y install udhcpd
```

2. Configure udhcpd

   Open /etc/udhcp.conf in your favourite text editor.

```
sudo nano /etc/udhcpd.conf
```

   Then enter the following example configuration:

```
# This is the range of IPs that the hotspot will give to client devices.
start 192.168.0.2
end 192.168.0.20
# The device udhcp listens on.
interface wlan0
remaining yes
# The DNS servers client devices will use.
opt dns 8.8.8.8 4.2.2.2
opt subnet 255.255.255.0
# The Pi's IP address on wlan0 which we will set up shortly.
opt router 192.168.0.1
# 1 hour DHCP lease time in seconds
opt lease 3600
```

   When you have finished press [Ctrl] + x. This will ask if you want to save the modified files. Press 'Y' and then hit [Return] to save the file with the same name.

   The next step is to 'enable' the udhcpd server. To do this, edit the `/etc/default/udhcpd` file and comment out the following line by prepending a hash symbol:

```
sudo nano /etc/default/udhcpd

DHCPD_ENABLED="no"
```

   Like this:

```
#DHCPD_ENABLED="no"
```

   When you have finished press [Ctrl] + x. This will ask if you want to save the modified files. Press 'Y' and then hit [Return] to save the file with the same name.

   This will enable the udhcpd server. Later, we will prevent it from starting automatically, just in case we do actually manage to connect to another wireless network.

3. Configuring hostapd

   The next step is to configure hostapd. Edit the following file (creating it if it doesn't exist) `/etc/hostapd/hostapd.conf`. It should contain the following text:

```
interface=wlan0
driver=nl80211
ssid=Raspi_wifi
hw_mode=g
channel=11
macaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=0
wpa=2
wpa_passphrase=somepassword
wpa_key_mgmt=WPA-PSK
wpa_pairwise=TKIP
rsn_pairwise=CCMP
```

   When you have finished press [Ctrl] + x. This will ask if you want to save the modified files. Press 'Y' and then hit [Return] to save the file with the same name.

   The above configuration will create an ad-hoc wireless g network on channel 11. You should change the channel to one that is not in use in the area where you will be using your Pi. It will be secured with WPA-2, and password will be "somepassword". You should change this to a more secure password.

4. To check the connectivity on boot and performing the appropriate action, run the following commands:

```
sudo update-rc.d hostapd remove
sudo update-rc.d udhcpd remove
```

   This will prevent udhcpd and hostapd from running on boot and preventing your Pi from connecting to other wireless networks automatically.

   Next, edit the file: `/etc/rc.local`. It should look something like this:

```
#[...a lot of stuff commented here...]
# By default this script does nothing.

# Print the IP address
_IP=$(hostname -I) || true
if [ "$_IP" ]; then
  printf "My IP address is %s\n" "$_IP"
fi

exit 0
```

   Change it to look something like this:

```
#[...a lot of stuff commented here...]
# By default this script does nothing.

# Print the IP address
sleep 10
_IP=$(hostname -I) || true
if [ "$_IP" ]; then
  printf "My IP address is %s\n" "$_IP"
fi

if [ "$_IP" ]; then
```

```
  echo "[AP] - The wifi is already connected, no access point needed"
else
  echo "[AP] - The wifi is not connected, firing up an access point..."
  sudo ifconfig wlan0 192.168.0.1
  sudo service hostapd start
  sudo service udhcpd start
fi
```

The `sleep 10` makes sure that your Pi has time to connect to any wireless networks it can. The code starting after the "My IP Address is…" will output a message telling the console whether or not we are setting up an ad-hoc wireless access point.

When you have finished press [Ctrl] + x. This will ask if you want to save the modified files. Press 'Y' and then hit [Return] to save the file with the same name.

Reboot your Pi

```
sudo reboot
```

Disable WiFi at boot time

Use

```
crontab -e
```

and add:

```
@reboot sudo iwconfig wlan0 txpower off
```

So that command runs at each boot.

Configuration of WiFi with Dongle

Pre-checks:

1. Make sure your Raspberry Pi is powerd off

```
sudo poweroff
```

2. Insert your WiFi dongle and power on the Pi. Wait another 30 seconds after it completed powered on.

3. Check which wlan it uses (normaly wlan0)

```
iwconfig

...
wlan0  IEEE 802.11bgn  ESSID:"YourSSID"
       Mode:Managed  Frequency:2.422 GHz  Access Point: 00:18:E7:F2:E3:36
       Bit Rate=65 Mb/s   Tx-Power=20 dBm
...
```

4. To scan for WiFi networks, use the command. It might be usefull to note down some information about the WiFi network you will use like ESSID, IE, Group, Pairwise, Authentication, …

```
sudo iwlist wlan0 scan

...
wlan0     Scan completed :
          Cell 01 - Address: 00:18:E7:F2:E3:36
                    ESSID:"YourSSID"
                    Protocol:IEEE 802.11bgn
                    Mode:Master
                    Frequency:2.427 GHz (Channel 4)
                    Encryption key:on
                    Bit Rates:144 Mb/s
                    Extra:wpa_ie=dd1a0050f20101000050f...
                    IE: WPA Version 1
                        Group Cipher : TKIP
                        Pairwise Ciphers (2) : TKIP CCMP
                        Authentication Suites (1) : PSK
                    IE: IEEE 802.11i/WPA2 Version 1
                        Group Cipher : TKIP
                        Pairwise Ciphers (2) : TKIP CCMP
                        Authentication Suites (1) : PSK
                    IE: Unknown: DD0E0050F204104A0001101044000102
                    Quality=100/100  Signal level=75/100
...
```

This scan shows I have excellent signal (**Quality=100/100 Signal level=75/100)** for my network with SSID "**YourSSID**". My AP uses the standard 802.11b/g/n and I am at high bitrate (**144 Mb/s**) to transfer data. Encryption is active (on) and 2 versions are available **WPA (Version 1)** with related info and **WPA2 (Version 1)** with related info.

5. Find your WiFi device if you do use a WiFi dongle. If you use the build-in WiFi controller of the Raspberry Pi 3, you willn not see it.

```
lsusb
```

- For Ralink Technology RT5370

```
...
Bus 001 Device 005: ID 148f:5370 Ralink Technology Corp. RT5370 Wireless
Adaptor
...
```

- For EdiMax 8192cu

```
...
Bus 001 Device 006: ID 0bda:8178 Realtek Semiconductor Corp. RTL8192CU
802.11n WLAN Adapter
...
```

6. Find your WiFi driver

```
lsmod
```

- For Ralink Technology RT5370

```
Module                  Size  Used by
...
rt2800usb              14940  0
rt2800lib              55351  1 rt2800usb
rt2x00usb              11215  1 rt2800usb
rt2x00lib              42335  3 rt2x00usb,rt2800lib,rt2800usb
...
```

- For EdiMax 8192cu

```
Module                  Size  Used by
...
8192cu                528485  0
...
```

WiFi Dongle Powermanagement considerations

Even after buying a powerful and reliable power supply, USB WiFi adaptors often fail to perform. They'll work fine on boot, but as soon as the network traffic dies down, they become unreliable.

The problem here is that many of these adaptors use a power-saving mode designed for smartphones and tablets where the network traffic is typically predominantly outbound. If the Raspberry Pi doesn't send any network traffic for a while, the wireless adaptors go into low power mode and become difficult to contact from an external source. For example, you may not be able to ping or establish an SSH connection from your desktop PC to your Raspberry Pi, or an SSH session may become unreliable after being left idle for a few minutes.

These power-saving modes can be turned off. The actual power saving is pretty minimal, especially if you actually want to have the Pi serving data pretty much constantly.

For the WiFi dongles driven by the module 8192cu, add the following two lines to the configuration file and save the file:

```
sudo nano /etc/modprobe.d/8192cu.conf

# Disable power management
options 8192cu rtw_power_mgnt=0 rtw_enusbss=0
options 8192cu rtw_power_mgnt=0 rtw_enusbss=1 rtw_ips_mode=1
```

When you have finished press [Ctrl] + X. This will ask if you want to save the modified files. Press 'Y' and then hit [Return] to save the file with the same name.

For most other modules, you can add

```
post-up iw dev wlan0 set power_save off
```

to the file `/etc/network/interfaces`

```
auto lo
iface lo inet loopback

auto eth0
allow-hotplug eth0
iface eth0 inet manual

auto wlan0
allow-hotplug wlan0
iface wlan0 inet manual
    wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf
    post-up iw dev wlan0 set power_save off

auto wlan1
allow-hotplug wlan1
iface wlan1 inet manual
    wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf
```

When you have finished press [Ctrl] + X. This will ask if you want to save the modified files. Press 'Y' and then hit [Return] to save the file with the same name.

Reboot your Pi

```
sudo reboot
```

Check that the wifi dongle's power management has been disabled by typing this command:

```
sudo iwconfig
```

And look for `Power Management:off`

```
...
wlan0  IEEE 802.11bgn  ESSID:"YourSSID"
       Mode:Managed  Frequency:2.422 GHz  Access Point: 00:18:E7:F2:E3:36
       Bit Rate=65 Mb/s   Tx-Power=20 dBm
       Retry short limit:7   RTS thr:off   Fragment thr:off
       Power Management:off
       Link Quality=70/70  Signal level=-37 dBm
       Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
       Tx excessive retries:0  Invalid misc:14   Missed beacon:0
 ...
```

Configuration without WPA Supplicant (Raspbian OS before May 2015)

1. Set/Change/Add its settings

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

- For Ralink Technology RT5370

```
auto wlan0
allow-hotplug wlan0
iface wlan0 inet dhcp
    wpa-ssid "YourSSID"
    wpa-psk "YourKEY"
```

- For EdiMax 8192cu

```
auto wlan0
allow-hotplug wlan0
iface wlan0 inet manual
    wpa-ssid "YourSSID"
    wpa-psk "YourKEY"
```

Note: In the case of WEP, add the following instead

```
wireless-essid "YourSSID"
wireless-key "YourKEY"
```

Note: If you need to connect to a private network (i.e. no broadcast SSID)

- Generate a PSK version of your WLAN password with wpa_passphrase utility

```
wpa_passphrase "<Your Wifi SSID>" "<Your Wifi PASSWORD>"
```

and note down the PSK value it give you

Note: Quotes are needed for whitespace

- Edit /etc/network/interfaces and add the

```
sudo nano /etc/network/interfaces

...
auto wlan0
allow-hotplug wlan0
iface wlan0 inet dhcp
   wpa-scan-ssid 1
   wpa-ap-scan 1
   wpa-key-mgmt WPA-PSK
   wpa-proto RSN WPA
   wpa-pairwise CCMP TKIP
   wpa-group CCMP TKIP
   wpa-ssid "<Your Wifi SSID>"
   wpa-psk <Your PSK Value>
...
```

When you have finished press [Ctrl] + x. This will ask if you want to save the modified files. Press 'Y' and then hit [Return] to save the file with the same name.

2. Reboot your Pi

```
sudo reboot
```

or just restart the network services

```
sudo /etc/init.d/networking restart
```

Wait a minute or 2 after reboot completed before checking your wireless connection

```
sudo iwconfig
...
wlan0  IEEE 802.11bgn  ESSID:"YourSSID"
       Mode:Managed  Frequency:2.422 GHz  Access Point: 00:18:E7:F2:E3:36
       Bit Rate=65 Mb/s   Tx-Power=20 dBm
...
```

3. Check Internet connectivity by pinging google.com or 8.8.8.8

```
sudo ping -c 3 8.8.8.8

PING google.com (173.194.69.100) 56(84) bytes of data.
64 bytes from google.com (173.194.69.100): icmp_req=1 ttl=45 time=26.7 ms
64 bytes from google.com (173.194.69.100): icmp_req=2 ttl=45 time=32.3 ms
64 bytes from google.com (173.194.69.100): icmp_req=3 ttl=45 time=34.8 ms
--- google.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 26.752/31.338/34.863/3.395 ms
```