University of Nottingham
UK | CHINA | MALAYSIA

Department of
Electrical & Electronic Engineering

EEEE1002: Lab Week 4

“Pi3 A+ as a Wireless Station”

Alex Ottway

February 2022

EEEE1002: Lab Week 4

Configuring the Pi3 A+ as a WiFi Station

Contents

Overview 2
Background research 3
Quick hints for working with Raspberry Pi 4
1. Enable WiFi 7
2. Download hostapd and dnsmasq packages. 7
3. Alter the default setup configuration in dnsmasq.conf 7
4. Edit the hostapd configuration file. 8
5. Add our new Access Point network to the network interfaces file: 9
6. Create a file to run when hostapd starts: 9
7. Change permissions on /usr/local/bin/hostapdstart 10
8. Edit rc.local 10
9. Assign parameters to our AP network in the interfaces file: 10
References 12

Overview

SO FAR, OUR PI HAS BEEN SET UP TO COMMUNICATE WITH THE HOST PC USING A USB A-A CABLE.

For the next part of the labs, we're going to use the Pi as a WiFi station (it's a similar setup to your home router)

It will allow multiple devices to connect to the Pi using the Pi's WiFi port.

This way we'll be able to support multiple devices communicating to/from our Pi simultaneously.

Background research

Since we'll be diving head-first into connecting multiple devices together, it might be helpful, if you're not especially familiar with it, to understand some important terms and abbreviations used in networking.

This will help you to get a handle on what's happening and will save a huge amount of time when asking questions and finding faults.

The terms below are a (very non-exhaustive!) list of some important networking concepts. You will benefit from a quick read on each of them.

Google is your friend.

IP Addresses (v4 and v6), **static** versus **DHCP addressing**.

Lease time, **subnet mask and network classes**.

SSID, **LAN**, **WAN**, **MAC address**, the difference between a **switch** and a **router**, and **NAT**.

WEP, **PSK**, **WPA** (and **WPA2**), **WPS** and **Enterprise security.**

Gateway, **Broadcast Address, DNS** and **mDNS**.

You never know…there might be a quiz later!

Quick hints for working with Raspberry Pi

New to Raspberry Pi?

Don’t panic.

Some stuff we’ll do just like in Windows (using a VNC connection, and your PC’s keyboard and mouse to point-and-click.)

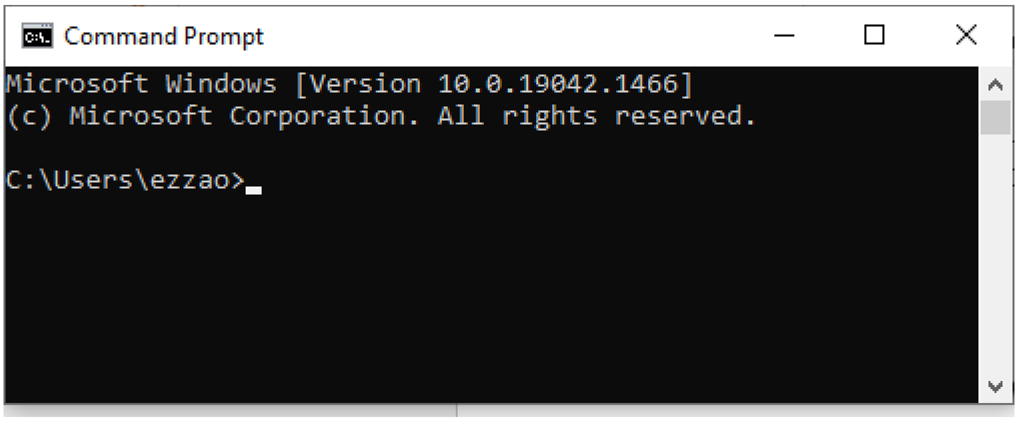
Other commands will be run using a Terminal window (which is just like the **CMD** prompt in Windows.)

Never used the Windows Command CMD prompt?

You should try it some time…it’s really useful!

In the Windows start bar search box, type **CMD** and press enter.

A black window with some text in it will appear.



```
Microsoft Windows [Version 10.0.19042.1466]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ezzao>_
```

The bit that says **C:\Users\[your userid]>** is the “prompt” which also tells you the current directory path you’re working in (here, ‘C’ drive, ‘Users’ subfolder, ‘ezzao’ subfolder.)

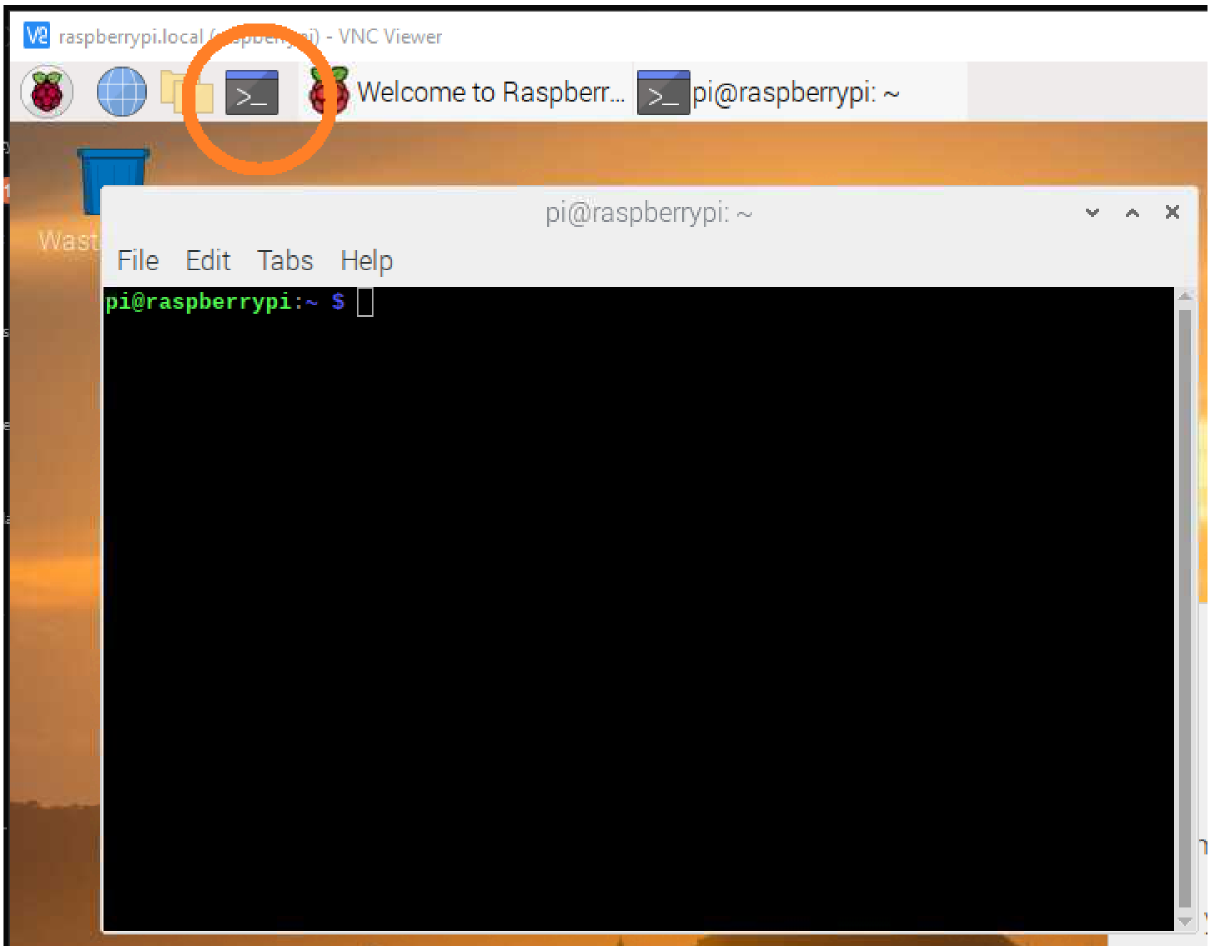
Type **ipconfig/all** at the prompt and press enter.

What’s returned is the IP configuration of all interfaces/cards in your PC – IP addresses, gateways, the lot.

Try finding that information by the other method…. that’s why command prompt or terminal window is so powerful!

The Pi terminal window works the same way.

The Terminal window on a Pi is accessed either by PuTTY from your PC, or if you have access to the Pi Desktop via VNC, you can click the terminal icon (circled in orange in the picture below) and a predominantly black window will open, with a “$” prompt…That’s also the Pi terminal window.



```
pi@raspberrypi:~ $
```

When working with the Pi terminal, some commands need to be prefixed with a “are you sure” sort of instruction – mostly commands that can modify system settings and important files. This command keyword is **sudo** – “**S**uper**U**ser: **DO**”. The difference it makes is this:

Opening a regular, user-created file using the text editor “**nano**” is achieved with the command:

nano [filepath][filename]

Try **nano mytestfile.txt** to create and open a new text file “mytestfile.txt” in the nano text editor program.

You can read the file, you can type stuff into it. Try saving it, (**CTRL X**, **Y**) - no problem.

Now do the same with a file in a system folder, e.g. the Wi-Fi configuration file **wpa_supplicant.conf**

The command:

nano /etc/wpa_supplicant/wpa_supplicant.conf

attempts to open the configuration file, but comes with a stern warning “**PERMISSION DENIED**”

Now try it again with **sudo** as the prefix to the command:

sudo nano /etc/wpa_supplicant/wpa_supplicant.conf

…and you find the file is there for you to edit and save. Don't change anything in there at this time – press **CTRL+X** to exit. If you had changed anything, you'll be asked "Save to (location)?" and you hit "**y**" to confirm the suggested file location, then press **Enter** to confirm.)

You may find the page **Pi Command Reference Sheet**, available at

https://raspberrytips.com/raspberry-pi-commands/

very useful as an eye-opener on what you can do with the terminal prompt, and how much havoc can be wrought with incautious use of **sudo** and **bash** …

1. Enable WiFi

Let's get started.

By default on this Pi Operating System, WiFi is initially disabled by the **RFKILL** process. (A **PuTTY** connection will advise this just above the command prompt).

WiFi functionality is enabled with the command

```
sudo rfkill unblock 0
```
[i]

With that done, we can now set the Pi up as a wireless station access point. [ii]

2. Download hostapd and dnsmasq packages.

hostapd is a package that enables a HOST computer (the Pi in this case) to become an Access Point Device.

dnsmasq is a small but efficient Domain Name Server – network devices that allow other network devices to connect to them usually require a system to allocate IP addresses/domain names to the connecting devices.

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

During the install process, you may have to hit "y" to confirm that yes, you want to use some additional disk space for this download.

3. Alter the default setup configuration in dnsmasq.conf

```
sudo nano /etc/dnsmasq.conf
```

…opens the "**nano**" text file editor. Use your arrow keys and scroll down until you find a line which initially reads

```
#interface=
```

The **#** symbol at the start of the line indicates that the line is a comment. Delete the # to activate the line.

Then edit the line to read:

```
interface=lo,uap0
```

[for the avoidance of doubt, that's a lower case "L"]
This enables the universal access point in system.

Scroll down further and edit the line that originally read

```
#no-dhcp-interface=
```

And edit it to:

no-dhcp-interface=lo,wlan0

This stops the regular "inbound" Wi-Fi, (e.g., if you were connecting to your domestic Wi-Fi, or eduroam) from grabbing the connection and assigning an address to the device. We want our UAP interface to do that.

Then further down still, change the appropriate line into

dhcp-range=192.168.2.100,192.168.2.200,12h

This defines the range of addresses that will be automatically assigned to connecting devices by our WiFi Station and defines the lease time as 12 hours.

Save and close the file.

4. Edit the hostapd configuration file.

Rather like we did in the initial Raspberry Pi setup (step 10, "Change your Pi hostname"), you **WILL** need to pick an SSID that's unique within the labs. Your surname and bench number would be ideal. (smithb31 or westc29 for example). You will also need a passphrase (password) more than 8 characters long.

sudo nano /etc/hostapd/hostapd.conf

This opens a new, empty file.

Add the following, taking care to use your chosen SSID and passphrase where indicated. These configuration lines define the system parameters for our new network.

```
interface=uap0

ssid=*********THIS IS WHERE YOUR CHOSEN SSID GOES*********

hw_mode=g

channel=6

macaddr_acl=0

auth_algs=1

ignore_broadcast_ssid=0

wpa=2

wpa_passphrase=*********THIS IS WHERE YOUR CHOSEN PASSPHRASE GOES**********

wpa_key_mgmt=WPA-PSK

wpa_pairwise=TKIP

rsn_pairwise=CCMP
```

Save and close.

5. Add our new Access Point network to the network interfaces file:

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

Add the following configuration lines:

```
auto uap0

iface uap0 inet static

address 192.168.2.1

netmask 255.255.255.0
```

Save and close the file.

6. Create a file to run when hostapd starts:

Edit a new file:

```
sudo nano /usr/local/bin/hostapdstart
```

add these lines, which define our AP network parameters and tells hostapd to read the config file:

```
iw dev wlan0 interface add uap0 type __ap

service dnsmasq restart

sysctl net.ipv4.ip_forward=1

iptables -t nat -A POSTROUTING -s 192.168.2.0/24 ! -d 192.168.2.0/24 -j MASQUERADE

ifup uap0

hostapd /etc/hostapd/hostapd.conf
```

Save and close.

7. Change permissions on /usr/local/bin/hostapdstart

Using **chmod** we'll alter the owner/group/world read/write/execute/delete permissions[iii]

```
chmod 667 /usr/local/bin/hostapdstart
```

8. Edit rc.local

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

Arrow down and add this line BEFORE "**exit 0**"

```
hostapdstart >1&
```

This auto-initialises the hostapd file on start up.
Save and close.

9. Assign parameters to our AP network in the interfaces file:

Step 1: Generate a **PSK** from your SSID and passphrase:

At the command prompt, type:

```
wpa_passphrase yourSSID yourpassphrase
```

Press enter.

Copy the long alphanumeric code after "**psk=**"

Step 2:
Edit the Network interfaces file:

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

Remove any block of text that refers to wlan0 and add these lines:

```
auto wlan0

iface wlan0 inet dhcp

wpa-ssid *****your SSID HERE****

wpa-psk ***the passphrase code you just copied, here*****
```

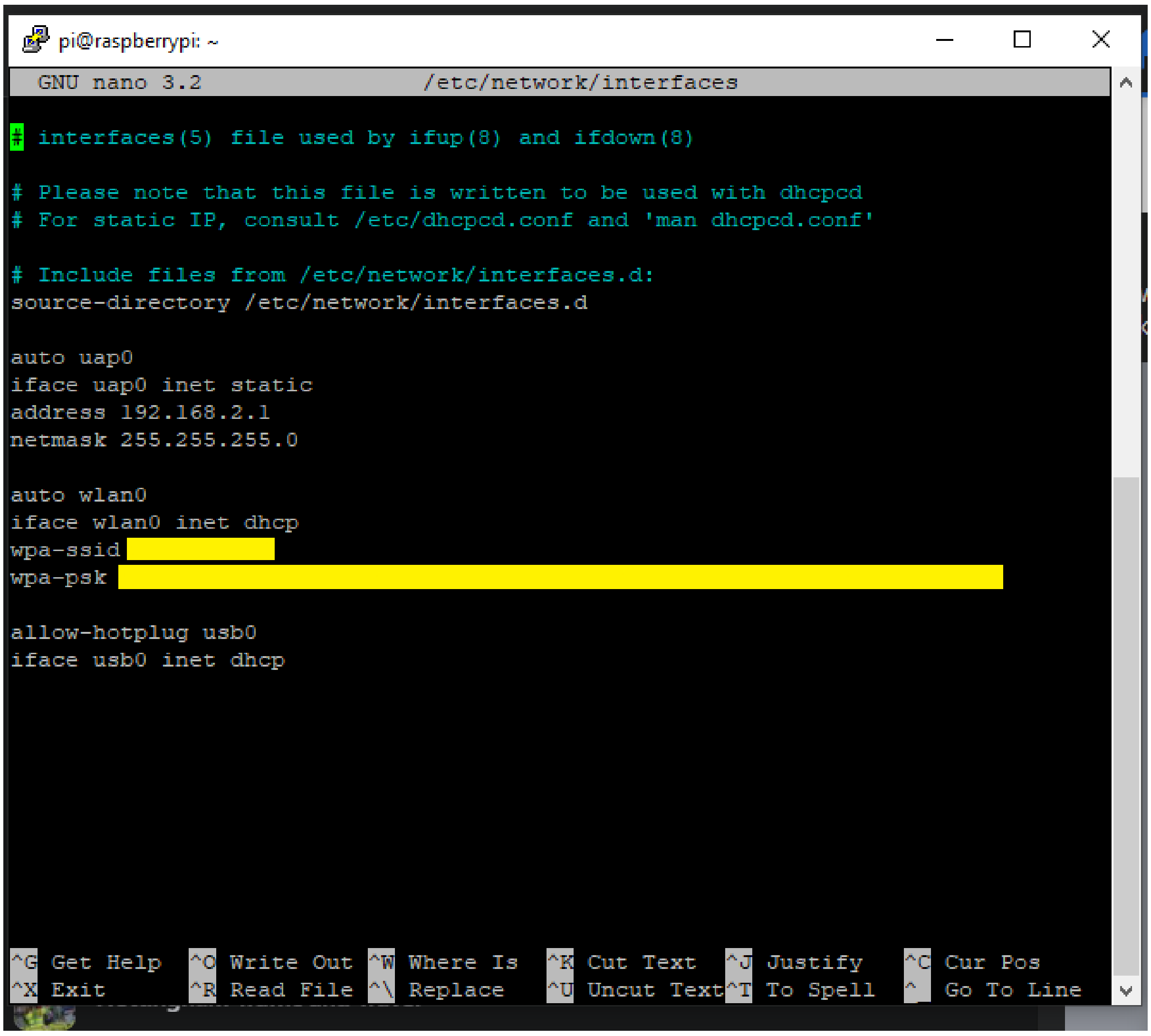
Now we need to add in our original USB connection parameters, otherwise they won't be initialised automatically.

Add these lines:

```
allow-hotplug usb0

iface usb0 inet dhcp
```

As a check, your /etc/network/interfaces file should look like this…



```
pi@raspberrypi: ~
  GNU nano 3.2                /etc/network/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 uap0
iface uap0 inet static
address 192.168.2.1
netmask 255.255.255.0

auto wlan0
iface wlan0 inet dhcp
wpa-ssid [redacted]
wpa-psk [redacted]

allow-hotplug usb0
iface usb0 inet dhcp

^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos
^X Exit      ^R Read File ^\ Replace   ^U Uncut Text^T To Spell   ^_ Go To Line
```

…with your SSID and alphanumeric PSK where the yellow blocks are shown here.

Save and close the file.

Reboot your Pi using

sudo reboot now

Remember to go back into Windows > networking > change adapter options and firstly STOP sharing your primary internet connection, and then when the Pi has rebooted, SHARE that connection again.

Congratulations – your Raspberry Pi is now set up as a WiFi Station/access point.

If you look on your phone's Wi-Fi available networks list, you will see your SSID, amongst many others, when your Pi has rebooted, which will take a little while longer than previously.

There's no need to connect your phone to your new access point network, but if you want to try, to prove that it's working, then go ahead (just remember to go back onto eduroam after!)

Once this section is successfully completed, you can go on the document describing the implementation of MQTT on the Pi.

References

[i] https://forums.raspberrypi.com/viewtopic.php?t=206223#p1278333
[ii] https://forums.raspberrypi.com/viewtopic.php?p=938306&sid=b950f5108774f76e0f3876c87c195961#p938306
[iii] https://chmodcommand.com/chmod-667/