Wireless Access Point on Raspberry Pi 3B+ with Blinkt! Internet Monitor

A guide to set up and configure a new WiFi Access Point

Thomas Templeton

ABSTRACT

This document shows users how to set up a Wireless Access Point on Raspberry Pi 3B+ using Raspbian Stretch. There is also an optional project to display the internet connection through the Blinkt! LED display.

DATE:

17 April 2019

Table of Contents

1.0 Overview 3
1.1 Instruction layout 3
1.1.1 Wireless Access Point 3
1.1.2 Blinkt! Internet Monitor 3
1.1.3 Testing 3
1.2 Network Diagram 3
1.3 What you'll need 4

2.0 Preparation 5
2.1 WiFi location 5
2.2 Check MAC addresses of WLANs 6

3.0 Changing the configurations for the networks to connect 7
3.1 WPA Supplicant files 7
3.2 DHCP SERVER INFORMATION 7
3.3 Install programs to use 8
3.4 DNSMASQ 9
3.5 Hostapd 10
3.6 IPTables and IP forwarding 11
3.7 rc.local 11
3.8 DAEMON 12
3.90 Reloading and Restarting 12

4.0 Setting up the Blinkt! WiFi monitor 13
4.1 Blinkt! Set up 13
4.2 Blinkt! Coding 13
4.3 Getting Blinkt! code to work automatically 15

5.0 Testing 16
5.1 To check the connection 16
5.2 To check the wifi devices 16
5.3 Test hostapd 16
5.4 Test dnsmasq 16
5.5 To view iptables 16
5.6 To check dhcpcd 16
5.6 To check rng-tools 16
5.7 To check connectivity to an address 16
5.8 To check connectivity to an address 16

6.0 Conclusion 17

7.0 Reference List 18

1.0 Overview

1.1 Instruction layout

1.1.1 Wireless Access Point

The purpose of these instructions is to set up your Raspberry Pi 3B + (on stretch) as a Wireless Access Point (WiFi to WiFi) with optional LED connection display. This access point will send out an internet signal using a WiFi name, allowing devices to connect to allocated IP addresses. These IP addresses will be sent through one WLAN, processed by the Raspberry Pi and then send back out via another WLAN to the original modem.

1.1.2 Blinkt! Internet Monitor

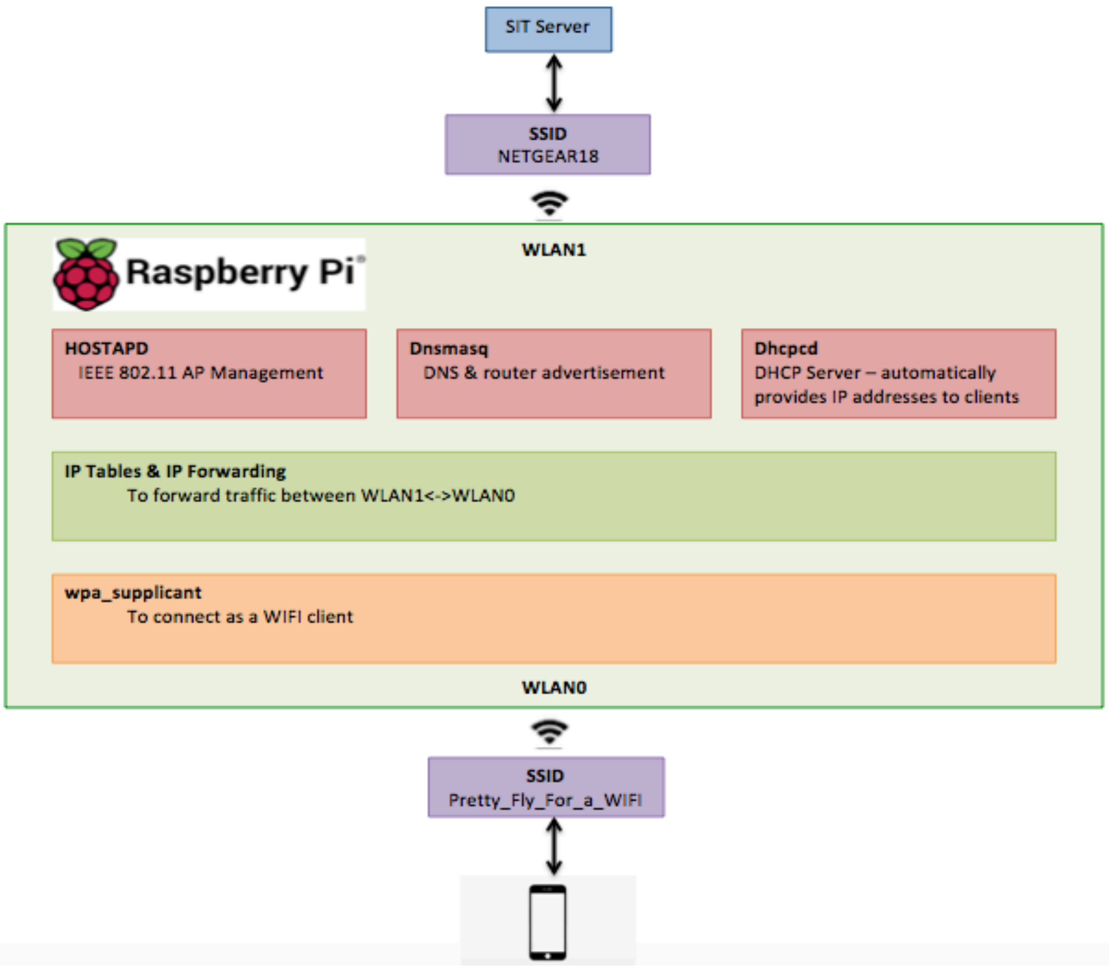
These instructions are to set up an LED device (we used Blinkt!) on your Raspberry Pi to display whether or not the internet is connected.

1.1.3 Testing

There are some suggestions later in this part for testing the different components which make up the Wireless Access Point.

1.2 Network Diagram

The diagram below (Almeida, 2012) shows how the network will operate. It displays the Raspberry Pi (in light green) with its internal programs and their functions. It also explains how the different components will interact with each other.


SIT Server
SSID
NETGEAR18
WLAN1
Raspberry Pi
HOSTAPD
IEEE 802.11 AP Management
Dnsmasq
DNS & router advertisement
Dhcpcd
DHCP Server – automatically provides IP addresses to clients
IP Tables & IP Forwarding
To forward traffic between WLAN1<->WLAN0
wpa_supplicant
To connect as a WIFI client
WLAN0
SSID
Pretty_Fly_For_a_WIFI

1.3 What you'll need

Mouse

Monitor

Keyboard

Ethernet Cable

Raspberry Pi 3B+

Blinkt! LED display

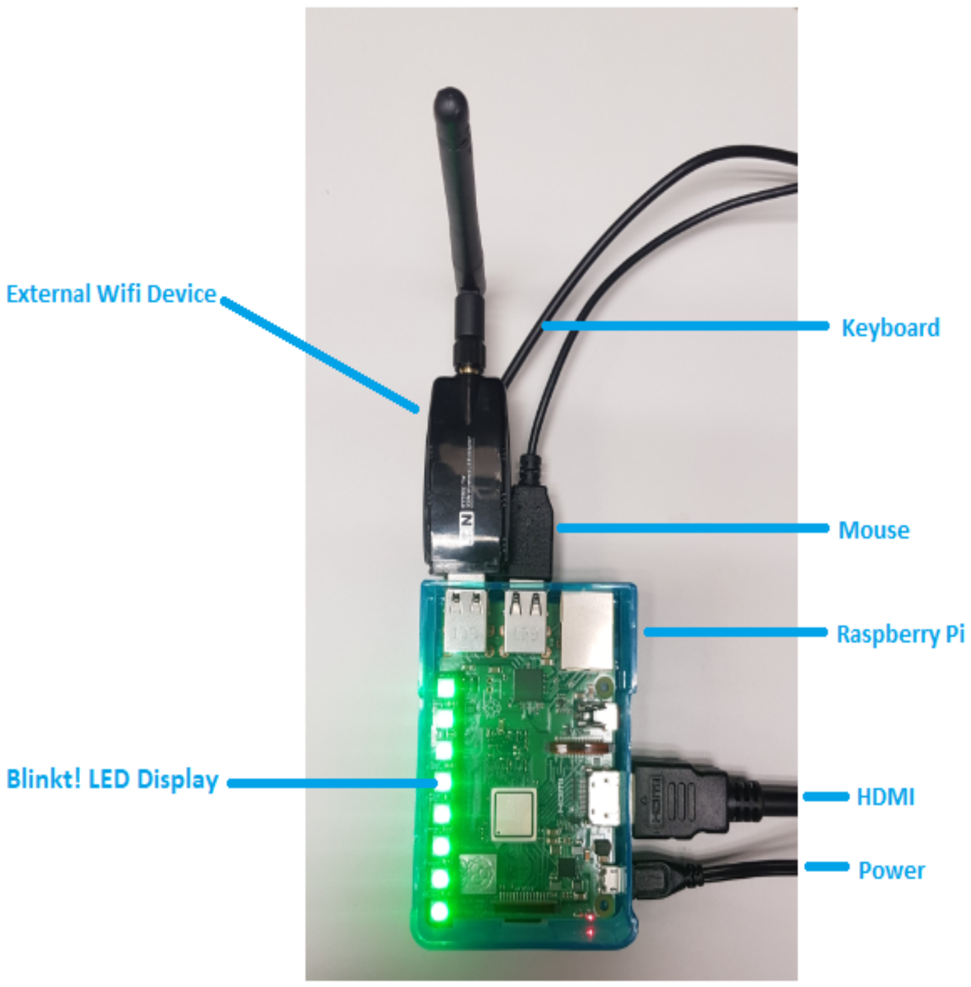
External WiFi device

HDMI (or similar) cable

Micro USB cable (for power)

Internet connection (via modem)

Flash drive with Raspbian Stretch pre-installed



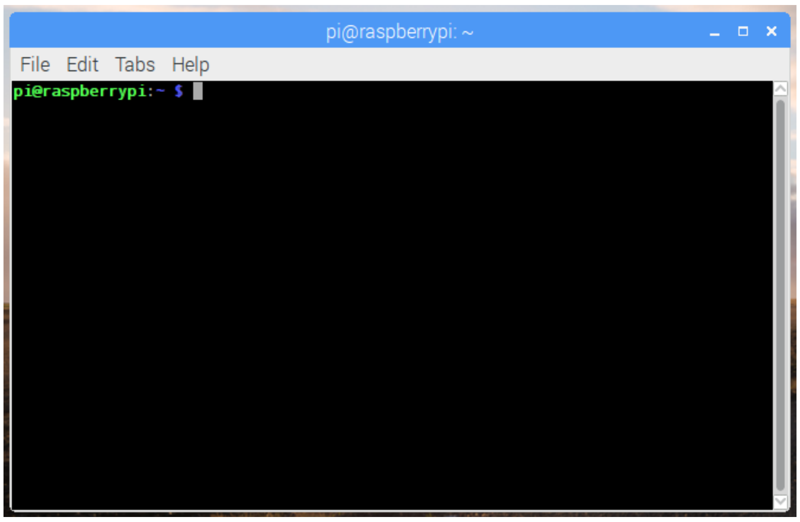
The image above shows what your Raspberry Pi should look like once it's all set up.

2.0 Preparation

For the reader to use these instructions, they will need to have all equipment in section "1.3 What you'll need", stretch should also be installed and working on the Raspberry Pi. If not, another instruction will need to be sourced to get to that point. These next steps will set your Raspberry Pi up to ensure the rest of the process will flow nicely. All commands are run in the terminal window. These instructions can be done either via ssh connection (with Ethernet cable connected) or locally with keyboard and mouse.

2.1 WiFi location

2.1.1 Open the Terminal Window. All commands in these instructions should be entered into the terminal window. Your input will appear after the $ sign at each stage.



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

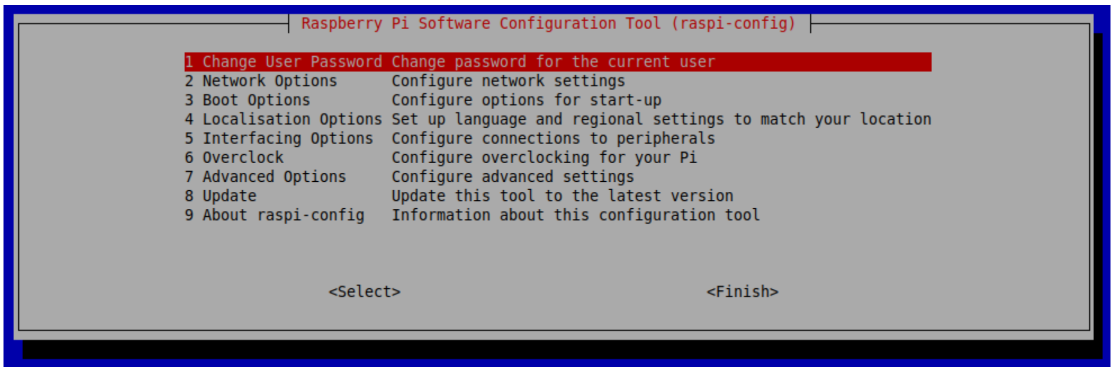
2.1.2 Before we start we should ensure the Raspberry Pi is up to date. Run:

```
sudo apt-get update
```

2.1.3 Now we need to access some setting for the Raspberry Pi, run the following from the command line:

```
sudo raspi-config
```

2.1.4 A blue screen with a grey box will appear as seen below.



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

1 Change User Password Change password for the current user
2 Network Options      Configure network settings
3 Boot Options         Configure options for start-up
4 Localisation Options Set up language and regional settings to match your location
5 Interfacing Options  Configure connections to peripherals
6 Overclock            Configure overclocking for your Pi
7 Advanced Options     Configure advanced settings
8 Update               Update this tool to the latest version
9 About raspi-config   Information about this configuration tool

                <Select>                                  <Finish>
```

2.1.5 Use the up and down keys to move down to highlight and select:

```
4 Localisation Options Set up language Set up language and
regional settings to match your location
```

2.1.6 Highlight and select

```
I4 Change WiFi Country
```

2.1.7 select your country and click on ok then select finish. The system may need to reboot to work correctly. Do this by entering

```
sudo reboot
```

2.2 Check MAC addresses of WLANs

There will be two WLANs (Wireless Local Area Network) in use for this project; WLAN0 and WLAN1.

- WLAN0 should be the on board WiFi device – this is also sending out a WiFi network.
- WLAN1 should be the USB connected device – this will connect all devices and the raspberry pi to the internet.

2.2.1 To note the MAC address of you internal WiFi device first disconnect the USB device and then run:

```
ifconfig
```

The MAC address will be written in hexadecimal and will be displayed next to WLAN0.

2.2.2 Now connect the USB device and repeat the process, Run:

```
ifconfig
```

2.2.3 Now we want to identify the initial server the system is running on, to do this run:

```
sudo cat /etc/resolv.conf
```

Note your server address as you will need this for step 3.2.2.

3.0 Changing the configurations for the networks to connect

3.1 WPA Supplicant files

3.1.1 First we need to create and edit a WPA (WiFi Protected Access) Supplicant file for WLAN1. This will allow the Raspberry Pi to connect WLAN1 to an existing network (something already connected to the internet). To do this run

```
sudo nano /etc/wpa_supplicant/wpa_supplicant-wlan1.conf
```

3.1.2 This will open a new file, now we need to edit this file by adding credentials of our existing network. Add this to the file:

```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
country=NZ
network={
         ssid=”your_existing_network_wifi_ssid”
         psk=”your_existing_wifi_password”
         key_mgmt=WPA-PSK
}
```

Press ctrl + x to close, Y to save and hit enter to accept and close.

3.1.3 Next open another file for WLAN0, To open this file run:

```
sudo nano /etc/wpa_supplicant/wpa_supplicant-wlan0.conf
```

3.1.4 In this file we don’t want such specific information as last time. In this file you should add in:

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

Press ctrl + x to close, Y to save and hit enter to accept and close.

3.1.5 next delete the original file so it does not cause confusion with the new files:

```
sudo rm /etc/wpa_supplicant/wpa_supplicant.conf
```

3.2 DHCP SERVER INFORMATION

This part will need to include information which will create interfaces that creates IP addresses for WLAN0 and WLAN1.

3.2.1 First, open the dhcpcd configuration file to make some changes, run:

```
sudo nano /etc/dhcpcd.conf
```

3.2.2 Scroll to the end of the page and add the following:

```
interface wlan0
static ip_address=192.168.2.254/24
static routers=
static domain_name_servers= 202.37.42.73
static domain_search=

interface wlan1
static ip_address=10.170.180.150/24
static routers=10.170.180.254
static domain_name_servers=202.37.42.73
```

You may need to amend the server address and ip_address to something that works in with your current WiFi settings. Use the server number from step 2.2.3.

Press ctrl + x to close, Y to save and hit enter to accept and close.

3.3 Install programs to use

3.3.1 Firstly, to install HostAPD, run:

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

3.3.2 Now install DNSMASQ. Run:

```
sudo apt-get install -y dnsmasq
```

3.3.3 Stop DNSMASQ from running so changes aren't made while we run the program.

```
sudo systemctl stop dnsmasq
```

3.3.4 Next install iptables, run:

```
sudo apt-get install -y iptables
```

3.3.5 Now install rng-tools. Rng-tools will assist HostAPD to run as it increases entropy.

```
sudo apt-get install rng-tools
```

3.3.6 Finally, to get all these new programs running correctly we need to restart the Raspberry Pi, run:

```
sudo reboot
```

3.4 DNSMASQ

Dnsmasq "provides network infrastructure for small networks: DNS, DHCP, router advertisement and network boot" (Kelley, n.d.).

3.4.1 The default dnsmasq.conf file has too much information and the program can become confused due to this. It is best to move this original file to another name.

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

3.4.2 Open a new dnsmasq configuration file:

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

3.4.3 add the following to the end of the page:

```
interface=wlan0
domain-needed
bogus-priv
dhcp-range=192.168.2.1,192.168.2.20,255.255.255.0,24h
dhcp-authoritative
```

Press ctrl + x to close, Y to save and hit enter to accept and close.

3.4.4 The DNSMASQ program now needs to be started

```
sudo systemctl start dnsmasq
```

3.5 Hostapd

Hostapd is a “user space daemon for access point and authentication servers. It implements IEEE 802.11 access point management, IEEE 802.1X/WPA/WPA2/EAP Authenticators, RADIUS client, EAP server, and RADIUS authentication server” (Malinen, 2013).

3.5.1 To begin with Hostapd, edit the hostapd configuration file, run:

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

3.5.2 This will open a blank file, where the new WiFi network is created. Add the following into this file:

```
interface=wlan0
driver=nl80211
hw_mode=g
channel=7
wmm_enabled=0
macaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=0
ht_capab=[HT40][SHORT-GI-20][DSSS_CCK-40]
ieee80211n=1
wpa=2
ssid=Enter_New_Wifi_Name_Here
wpa_passphrase=NewPassWordGoesHere
wpa_key_mgmt=WPA-PSK
wpa_pairwise=TKIP
rsn_pairwise=CCMP
```

Press ctrl + x to close, Y to save and hit enter to accept and close.

3.5.3 Sometimes when in headless mode Hostapd does not load in the correct order. To fix this issue Hostapd needs to be delayed. Firstly open a new file:

```
sudo nano /home/pi/startHostapd.sh
```

3.5.4 Add the following information into this file:

```
#!/bin/bash
sleep 10s hostapd /etc/hostapd/hostapd.conf > /home/pi/hostapd.log
```

Press ctrl + x to close, Y to save and hit enter to accept and close.

3.5.5 Now change the permissions

```
sudo chmod +x /home/pi/startHostapd.sh
```

3.5.6 Now get this file to load on start up by editing crontab:

`crontab –e` Then enter `2`

3.5.7 Add the following line to the bottom of the page:

```
@reboot sudo /home/pi/startHostapd.sh
```

3.6 IPTables and IP forwarding

IPTables are a tool that will tell WLAN0 and WLAN1 how and when to forward internet communications to each other. The following information needs to be entered in the terminal to make these connections. Press enter after entering each instruction below:

3.6.1

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

3.6.2

```
sudo iptables -t nat -A POSTROUTING -o wlan1 -j MASQUERADE
```

3.6.3

```
sudo iptables -A FORWARD -i wlan1 -o wlan0 -m state –state
RELATED,ESTABLISHED -j ACCEPT
```

3.6.4

```
sudo iptables -A FORWARD -i wlan0 -o wlan1 -j ACCEPT
```

3.6.5

```
sudo sh -c "iptables-save > /etc/iptables.ipv4.nat"
```

3.6.6 Edit the /etc/sysctl.conf file to allow IPv4 forwarding:

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

3.6.7 Now uncomment the line `#net.ipv4.ip_forward=1` it should now look like this:

```
net.ipv4.ip_forward=1
```

Press ctrl + x to close, Y to save and hit enter to accept and close.

3.7 rc.local

3.7.1 So the iptables rules can take effect after rebooting we need to edit the rc.local file. Run:

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

3.7.2 Now scroll to the bottom of the page and add this command before on the line above `exit 0`:

```
sudo sh -c "echo 1 > /proc/sys/net/ipv4/ip_forward"
sudo iptables-restore < /etc/iptables.ipv4.nat
```

Press ctrl + x to close, Y to save and hit enter to accept and close.

3.8 DAEMON

3.8.1 Open and edit the default Hostapd file, run:

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

3.8.2 Remove the `#` from in front of the DAEMON_CONF line and change to:

```
DAEMON_CONF="etc/hostapd/hostapd.conf"
```

3.8.3 Open and edit the init Hospapd file and run:

```
sudo nano /etc/init.d/hostapd
```

3.8.4 Uncomment DAEMON_CONF line and change to

```
DAEMON_CONF=etc/hostapd/hostapd.conf
```

3.90 Reloading and Restarting

The Raspberry Pi is just about ready to make new internet connections and send out new IP addresses, but first we need to reload and restart some programs:

3.9.1 First reload Daemon:

```
sudo systemctl daemon-reload
```

3.9.2 By default hostapd is masked and disabled so it cannot run without network settings being set by the user. Unmask hostapd:

```
sudo systemctl unmask hostapd
```

3.9.3 Then enable hostapd to operate at all times:

```
sudo systemctl enable hostapd
```

3.9.4 Now start hostapd so it will be functioning:

```
sudo service hostapd start
```

3.9.5 Reboot to disconnect Ethernet and check all is working.

```
sudo halt
```

3.9.6 The Raspberry Pi will now be shutdown. It is now safe to disconnect the Ethernet cable.

3.9.7 Turn the power off and on again to reboot the Raspberry pi, wait until it has fully loaded and then reopen the terminal. Once Raspbian is loaded other devices should now be able to see your new WiFi name is operational.

3.9.8 With another device (i.e. laptop, smart phone, etc) click on the WiFi name in WiFi settings and add the password to connect. The new device should now have internet access!

4.0 Setting up the Blinkt! WiFi monitor

These instructions were obtained by Gleeson (2016) and have been amended slightly to suit our own needs here. At this point, your Raspberry Pi should be operating as Wireless Access Point and other devices have been able to connect to via it, to the internet. Now we are going to connect the Blinkt! Phat and tune it to act as a WiFi monitor.

4.1 Blinkt! Set up

4.1.1 With the Raspberry Pi turned off, attach the Blinkt! LED phat to the GPIO breadboard (ensuring curved edge of Blinkt! is face outwards).

4.1.2 Turn on Raspberry Pi, allow to boot.

4.1.3 Download Pimoroni Blinkt! from the internet. Open a new terminal and add the following:

```
curl https://get.pimoroni.com/blinkt | bash
```

Enter Y or Yes where required to continue the installation.

4.1.4 Once installed, reboot to ensure effectiveness. Run:

```
sudo reboot
```

4.2 Blinkt! Coding

4.2.1 Create a new python file in the Pimoroni/blinkt/examples folder:

```
sudo nano /home/pi/Pimoroni/blinkt/examples/inet_monitor.py
```

This is what will control the Blinkt! light display.

4.2.2 Now add the following code into this file:

```
#!/usr/bin/env python

import colorsys
import signal
import random
import socket
import time
import sys
import os

from blinkt import set_brightness, set_pixel, show

def signal_handler(signal, frame):
    “””
     This signal handler allows us to background the process
     and  send it a simple SIGINT to tell it to exit cleanly
    “””
    sys.exit(0)

#Start the signal handler for SIGINT
signal.signal(signal.SIGINT, signal_handler)

spacing = 360.0 / 16.0
hue = 0
set_brightness(0.1)
hostname = ‘google.com’

set_pixel (1, 0, 0, 0,)
show()
timer = 5 #how often to check for connectivity

while True:
    #Test for the ability to open socket and resolve DNS.
    print(“trying to connect to google.com”)
    response = os.system(“ping -c 1 “ + hostname)

    #and then check response…
    if response == 0:
      print (“is up!”)
      connectUp = True
    else:
      print (“is down!”)
      connectUp = False

    if connectUp:
      #We have internet, show green lights.
      for i in range(8):
        set_pixel(i, 0, 255, 0)
    else:
      #DNS/internet are down, play the red lights
      for i in range(8):
        set_pixel(i, 255, 0, 0)
    show()
    time.sleep(timer)
```

Press ctrl + x to close, Y to save and hit enter to accept and close.

4.3 Getting Blinkt! code to work automatically

4.3.1 To launch this script on booting the Raspberry Pi open Crontab by entering:

```
sudo crontab -e
```

4.3.2 Now enter the following line at the bottom of the crontab file:

```
@reboot python /home/pi/Pimoroni/blankt/examples/inet_monitor.py &
```

Press ctrl + x to close, Y to save and hit enter to accept and close.

4.3.3 Reboot to test:

```
sudo reboot
```

The raspberry Pi should now reboot, once it commences the reboot the lights should all be Red. When the Raspberry Pi finally connects to the internet the Blinkt! LED panel should change to Green lights.

5.0 Testing

Use some of the following tools to identify if you have any problems with your Raspberry Pi working.

5.1 To check the connection

```
ifconfig
```

5.2 To check the wifi devices

```
iwconfig
```

5.3 Test hostapd

5.3.1

```
sudo /usr/sbin/hostapd /etc/hostapd/hostapd.conf
```

For more information on errors add `-d` or `-dd` at the end of this command

5.3.2 or try

```
sudo systemctl status hostapd
```

5.4 Test dnsmasq

```
systemctl status dnsmasq –l
```

For more information on errors add `-d` or `-dd` at the end of this command

5.5 To view iptables

```
sudo iptables –L
```

5.6 To check dhcpcd

```
systemctl status dhcpcd.service
```

For more information on errors add `-d` or `-dd` at the end of this command

5.7 To check rng-tools

```
sudo service rng-tools status
```

5.8 To check connectivity to an address

```
ping 8.8.8.8
```

This is ping the Google domain address. Troubleshoot by pinging other IP addresses. Start with IP addresses within the Pi and then move further out (i.e. network router)

5.9 To check connectivity to an address

```
traceroute 8.8.8.8
```

This command is similar to 'ping', but it will show more information by displaying the path the packet follows to reach the final IP address. Again, choose any IP address needed.

6.0 Conclusion

These instructions have shown how to set up a Raspberry Pi 3b+ to work as a Wireless Access Point for internet traffic (WiFi to WiFi connections). Furthermore, it has given instruction on how to set up an LED display internet monitor which shows whether the internet is connected or not.

7.0 Reference List

Almeida, B. A. (2012, Feb). *RogueOne: Creating a Rogue Wi-Fi Access Point using a Raspberry Pi*. Retrieved Apr 13, 2019 from A Medium Corporation: https://medium.com/@brunoamaroalmeida/rogueone-creating-a-rogue-wi-fi-access-point-using-a-raspberry-pi-79e1b7e628c6

Gleeson, C. (2016, Aug 31). *blinkt_internet_monitor*. Retrieved Apr 13, 2019 from GitHub: https://github.com/randomInteger/blinkt_internet_monitor

'gs7495945jglolfkgtjoiu'. (2018, Mar 15). *RPI3B Wifi Extender*. Retrieved Apr 13, 2019 from GitHub: https://gist.github.com/gs7495945jglolfkgtjoiu/fb126712c29b7d2e5b81be5600b0d8b6

Kelley, S. (n.d.). *DNSMASQ*. Retrieved Apr 15, 2019 from The Kelleys: http://www.thekelleys.org.uk/dnsmasq/doc.html

Lady Ada. (2018, Aug 22). *AdaFruit*. Retrieved Apr 13, 2019 from Setting up a Raspberry Pi as a WiFi access point: https://cdn-learn.adafruit.com/downloads/pdf/setting-up-a-raspberry-pi-as-a-wifi-access-point.pdf

Malinen, J. (2013, Jan 12). *hostapd: IEEE 802.11 AP, IEEE 802.1X/WPA/WPA2/EAP/RADIUS Authenticator*. Retrieved Apr 15, 2019 from https://w1.fi/hostapd/

Raspberry Pi. (n.d). *Raspberry Pi*. Retrieved Apr 13, 2019 from raspi-config: https://www.raspberrypi.org/documentation/configuration/raspi-config.md

RDPUser. (2018, Jan 30). *Hostapd not working correctly solution provided* . Retrieved Apr 13, 2019 from Raspberry Pi: https://www.raspberrypi.org/forums/viewtopic.php?f=28&t=206784