Setting up a Raspberry Pi 3 as an Access Point

CONTRIBUTORS: *SHAWN HYMEL*

FAVORITE 6

Introduction

Note: This tutorial was based on instructions found on this blog.

The Raspberry Pi 3 comes with a built-in wireless adapter, which makes it easy to configure it as a WiFi hotspot to share Internet or host your own internal web site. The first part of this guide will show you how to set up the Pi to broadcast its SSID, accept WiFi connections, and hand out IP addresses (using DHCP). The next section shows you how to enable a pass-through Ethernet connection should you want to share your Internet connection.

Raspberry Pi 3 B+
DEV-14643
$39.95
★★★★⯪ 19

Before You Get Started!

You will want a Raspberry PI 3 or Raspberry Pi Zero W along with any hookup accessories you might need (for example, a power adapter and micro SD card).

You will want to load an operating system (OS) onto the SD card and be able to log into the Pi and open a terminal.

Note: This tutorial was created with Raspbian Stretch (version: March 2018). Using a different version may require performing different steps than what's shown in this tutorial. If you would like to download the March 2018 version of Raspbian, it can be found below.

RASPBIAN STRETCH (VERSION: MARCH 2018) DOWNLOAD (ZIP)

If you need help installing an OS onto the Raspberry Pi, these tutorials can be helpful:

- The getting an OS part of the Raspberry Pi 3 Starter Kit guide walks you through a few options to installing NOOBS on the Pi.
- Getting Started with the Raspberry Pi Zero Wireless tutorial is a good place to start if you have a Pi Zero W.
- The Headless Raspberry Pi Setup guide is useful if you are looking to configure your Pi without a monitor, keyboard, or mouse (e.g. log in through serial or SSH).

Suggested Reading

If you aren't familiar with the following concepts, we recommend checking out these tutorials before continuing.

Raspberry Pi 3 Starter Kit Hookup Guide
Guide for getting going with the Raspberry Pi 3 Model B and Raspberry Pi 3 Model B+ starter kit.

Getting Started with the Raspberry Pi Zero Wireless
Learn how to setup, configure and use the smallest Raspberry Pi yet, the Raspberry Pi Zero - Wireless.

Headless Raspberry Pi Setup
Configure a Raspberry Pi without a keyboard, mouse, or monitor.

Set Up WiFi Access Point

Make sure your Raspberry Pi is connected to the Internet. Open a terminal window (either in a console window or over a serial/SSH connection).

Install Packages

To install the required packages, enter the following into the console:

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

Hostapd is a program that allows you to use the WiFi radio as an access point, and *Dnsmasq* is a lightweight combination of DHCP and DNS services (handing out IP addresses and translating domain names into IP addresses).

Set Static IP Address

In recent versions of Raspbian, network configuration is managed by the *dhcpcd* program. As a result, we'll need to tell it to ignore the wireless interface, *wlan0*, and set a static IP address elsewhere.

Note: If you are connected to your Raspberry Pi using SSH over wireless, you will want to connect with a keyboard/mouse/monitor, Ethernet, or serial instead until we get the access point configured.

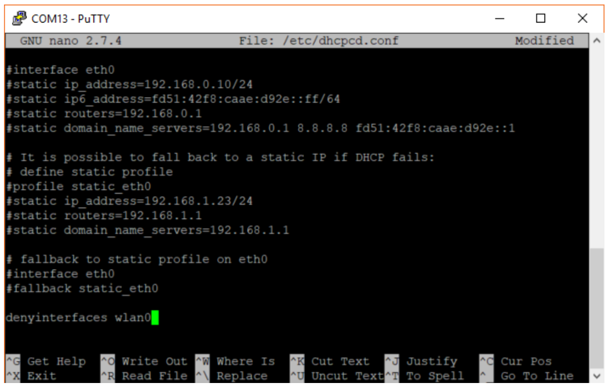
Edit the *dhcpcd* file:

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

Scroll down, and at the bottom of the file, add:

```
denyinterfaces wlan0
```

Your terminal window should look similar to the image below.


COM13 - PuTTY

```
  GNU nano 2.7.4              File: /etc/dhcpcd.conf                Modified

#interface eth0
#static ip_address=192.168.0.10/24
#static ip6_address=fd51:42f8:caae:d92e::ff/64
#static routers=192.168.0.1
#static domain_name_servers=192.168.0.1 8.8.8.8 fd51:42f8:caae:d92e::1

# It is possible to fall back to a static IP if DHCP fails:
# define static profile
#profile static_eth0
#static ip_address=192.168.1.23/24
#static routers=192.168.1.1
#static domain_name_servers=192.168.1.1

# fallback to static profile on eth0
#interface eth0
#fallback static_eth0

denyinterfaces wlan0


^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
```

Save and exit by pressing `ctrl` + `x` and `y` when asked.

Next, we need to tell the Raspberry Pi to set a static IP address for the WiFi interface. Open the *interfaces* file with the following command:

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

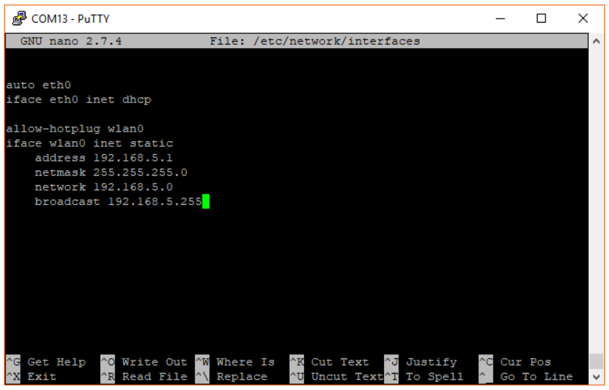
At the bottom of that file, add the following:

```
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet dhcp

allow-hotplug wlan0
iface wlan0 inet static
    address 192.168.5.1
    netmask 255.255.255.0
    network 192.168.5.0
    broadcast 192.168.5.255
```

Your terminal window should look similar to the image below.



```
auto eth0
iface eth0 inet dhcp

allow-hotplug wlan0
iface wlan0 inet static
    address 192.168.5.1
    netmask 255.255.255.0
    network 192.168.5.0
    broadcast 192.168.5.255
```

Save and exit by pressing `ctrl + x` and `y` when asked.

Configure Hostapd

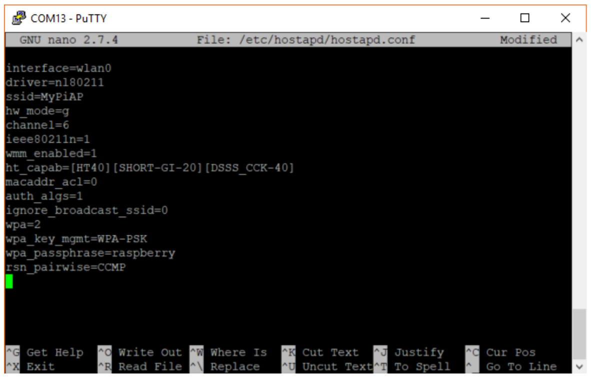
We need to set up *hostapd* to tell it to broadcast a particular SSID and allow WiFi connections on a certain channel. Edit the *hostapd.conf* file (this will create a new file, as one likely does not exist yet) with this command:

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

Enter the following into that file. Feel fee to change the `ssid` (WiFi network name) and the `wpa_passphrase` (password to join the network) to whatever you'd like. You can also change the `channel` to something in the 1-11 range (if channel 6 is too crowded in your area).

```
interface=wlan0
driver=nl80211
ssid=MyPiAP
hw_mode=g
channel=6
ieee80211n=1
wmm_enabled=1
ht_capab=[HT40][SHORT-GI-20][DSSS_CCK-40]
macaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=0
wpa=2
wpa_key_mgmt=WPA-PSK
wpa_passphrase=raspberry
rsn_pairwise=CCMP
```

Your terminal window should look similar to the image below.



```
COM13 - PuTTY
  GNU nano 2.7.4              File: /etc/hostapd/hostapd.conf              Modified

interface=wlan0
driver=nl80211
ssid=MyPiAP
hw_mode=g
channel=6
ieee80211n=1
wmm_enabled=1
ht_capab=[HT40][SHORT-GI-20][DSSS_CCK-40]
macaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=0
wpa=2
wpa_key_mgmt=WPA-PSK
wpa_passphrase=raspberry
rsn_pairwise=CCMP

^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
```

Save and exit by pressing `ctrl` + `x` and `y` when asked.

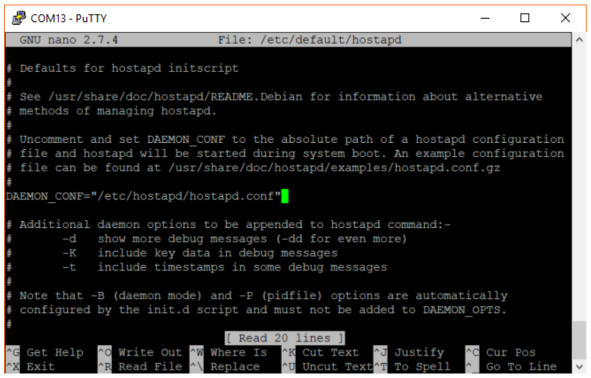
Unfortunately, *hostapd* does not know where to find this configuration file, so we need to provide its location to the *hostapd* startup script. Open */etc/default/hostapd*:

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

Find the line `#DAEMON_CONF=""` and replace it with:

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

Your terminal window should look similar to the image below.



```
COM13 - PuTTY
  GNU nano 2.7.4              File: /etc/default/hostapd

# Defaults for hostapd initscript
#
# See /usr/share/doc/hostapd/README.Debian for information about alternative
# methods of managing hostapd.
#
# Uncomment and set DAEMON_CONF to the absolute path of a hostapd configuration
# file and hostapd will be started during system boot. An example configuration
# file can be found at /usr/share/doc/hostapd/examples/hostapd.conf.gz
#
DAEMON_CONF="/etc/hostapd/hostapd.conf"

# Additional daemon options to be appended to hostapd command:-
#       -d   show more debug messages (-dd for even more)
#       -K   include key data in debug messages
#       -t   include timestamps in some debug messages
#
# Note that -B (daemon mode) and -P (pidfile) options are automatically
# configured by the init.d script and must not be added to DAEMON_OPTS.
#
                                [ Read 20 lines ]
^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
```

Save and exit by pressing `ctrl` + `x` and `y` when asked.

Configure Dnsmasq

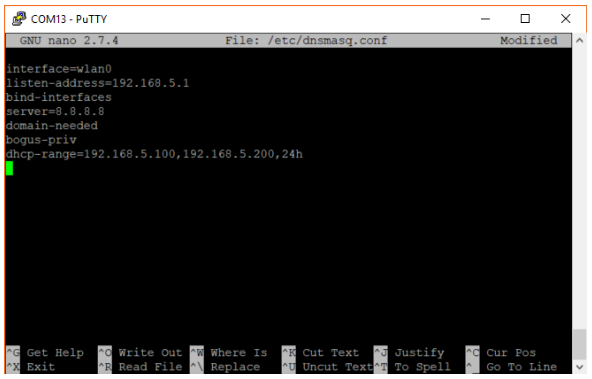
Dnsmasq will help us automatically assign IP addresses as new devices connect to our network as well as work as a translation between network names and IP addresses. The **.conf** file that comes with *Dnsmasq* has a lot of good information in it, so it might be worthwhile to save it (as a backup) rather than delete it. After saving it, open a new one for editing:

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

In the blank file, paste in the text below. Note that we set up DHCP to assign addresses to devices between `192.168.5.100` and `192.168.5.200`. Remember that `192.168.5.1` is reserved for the Pi. So, anything between `192.168.5.2 - 192.168.5.9` and between `192.168.5.201 - 192.168.5.254` can be used for devices with static IP addresses.

```
interface=wlan0
listen-address=192.168.5.1
bind-interfaces
server=8.8.8.8
domain-needed
bogus-priv
dhcp-range=192.168.5.100,192.168.5.200,24h
```

Your terminal window should look similar to the image below.



```
  GNU nano 2.7.4              File: /etc/dnsmasq.conf              Modified

interface=wlan0
listen-address=192.168.5.1
bind-interfaces
server=8.8.8.8
domain-needed
bogus-priv
dhcp-range=192.168.5.100,192.168.5.200,24h

^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
```

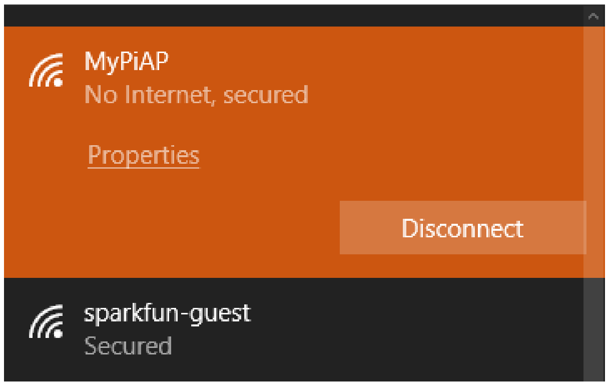
Save and exit by pressing `ctrl + x` and `y` when asked.

Test WiFi connection

Restart the Raspberry Pi using the following command:

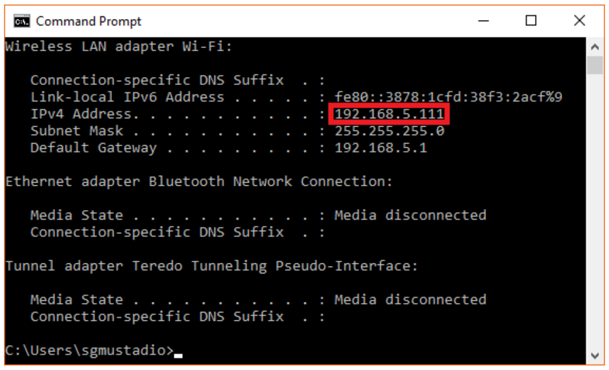
```
sudo reboot
```

After your Pi restarts (no need to log in), you should see **MyPiAP** appear as a potential wireless network from your computer.



Connect to it (the network password is **raspberry**, unless you changed it in the *hostapd.conf* file). Open a terminal on your computer and enter the command `ipconfig` (Windows) or `ifconfig` (Mac, Linux). You should see that you have been assigned an IP address in the `192.168.5.100 - 192.168.5.200` range.

Here's an example of what you may see after connecting wirelessly to the Pi.



```
Wireless LAN adapter Wi-Fi:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::3878:1cfd:38f3:2acf%9
   IPv4 Address. . . . . . . . . . . : 192.168.5.111
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.5.1

Ethernet adapter Bluetooth Network Connection:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Tunnel adapter Teredo Tunneling Pseudo-Interface:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

C:\Users\sgmustadio>
```

If you just want to use the Pi as a standalone WiFi access point, you can stop here. If you want to connect it to another network (over Ethernet) in order to share Internet (much like a WiFi router), continue on.

Enable Packet Forwarding

We can use the Raspberry Pi as a router by being able to connect it to another network over Ethernet and have WiFi-connected devices be able to talk to that network. By doing this, we can share an Internet connection from the Pi.

Configure NAT

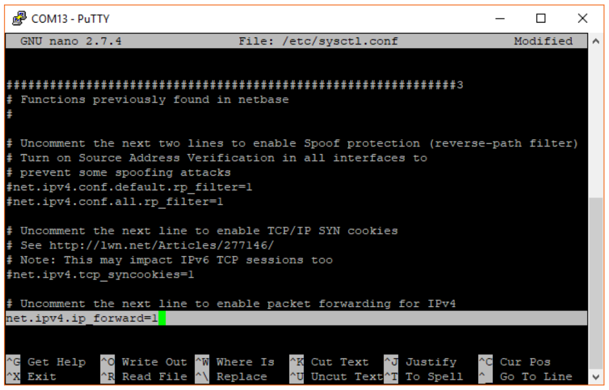
Make sure you are logged into your Pi. Edit the */etc/sysctl.conf* file:

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

Look for the line `#net.ipv4.ip_forward=1` , and uncomment it by deleting the `#` .

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

Your terminal window should look similar to the image below.



```
  GNU nano 2.7.4              File: /etc/sysctl.conf                Modified


###############################################################3
# Functions previously found in netbase
#

# Uncomment the next two lines to enable Spoof protection (reverse-path filter)
# Turn on Source Address Verification in all interfaces to
# prevent some spoofing attacks
#net.ipv4.conf.default.rp_filter=1
#net.ipv4.conf.all.rp_filter=1

# Uncomment the next line to enable TCP/IP SYN cookies
# See http://lwn.net/Articles/277146/
# Note: This may impact IPv6 TCP sessions too
#net.ipv4.tcp_syncookies=1

# Uncomment the next line to enable packet forwarding for IPv4
net.ipv4.ip_forward=1


^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
```

Save and exit by pressing `ctrl + x` and `y` when prompted.

Finally, we need to configure Network Address Translation (NAT) between the Ethernet and WiFi interfaces to allow devices on both networks to communicate with each other. In the terminal, enter the following:

```
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
```

This will work for now, but on reboot, the Pi will revert back to its previous state. To fix this, we need these NAT rules to be applied each time it starts up. Save the current rules to a file with this command:

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

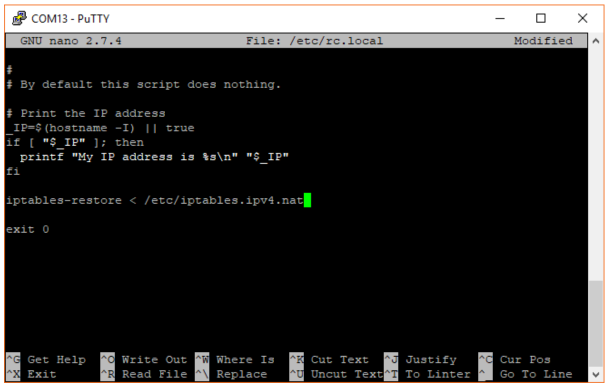
Linux provides us with a number of ways to run commands on boot. Usually, the easiest is to put those commands into the */etc/rc.local* script. To have our NAT rules restored on boot, we edit the *rc.local* file:

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

Just above the `exit 0` line (which ends the script), add the following:

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

Your terminal window should look similar to the image below.



```
  GNU nano 2.7.4              File: /etc/rc.local                   Modified

#
# 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

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

exit 0








^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 Linter  ^_ Go To Line
```

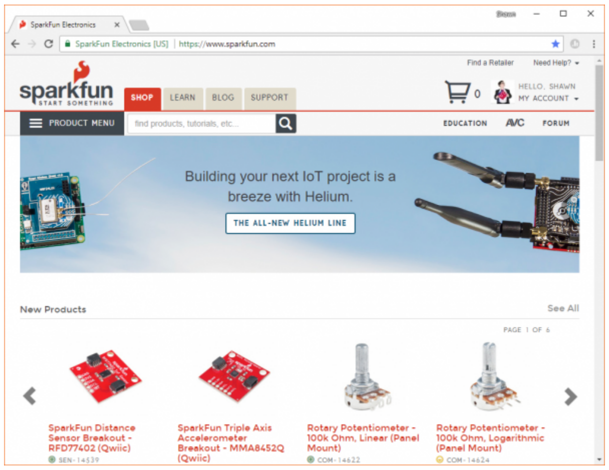
Save and exit by pressing `ctrl + x` and `y` when prompted.

Test It Out

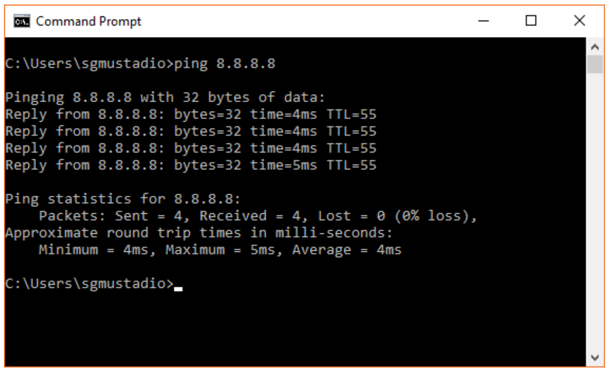
Restart your Pi:

```
sudo reboot
```

Give your Pi a couple minutes to restart (once again, no need to log in). Connect an Ethernet cable from your Internet router (or switch, etc.) to your Pi. Once the Pi has started, connect to the **MyPiAP** network from your computer. Open a web browser, and navigate to the website of your choice.


SparkFun Electronics
https://www.sparkfun.com
sparkfun START SOMETHING
SHOP
LEARN
BLOG
SUPPORT
PRODUCT MENU
Building your next IoT project is a breeze with Helium.
THE ALL-NEW HELIUM LINE
New Products
See All
SparkFun Distance Sensor Breakout - RFD77402 (Qwiic)
SparkFun Triple Axis Accelerometer Breakout - MMA8452Q (Qwiic)
Rotary Potentiometer - 100k Ohm, Linear (Panel Mount)
Rotary Potentiometer - 100k Ohm, Logarithmic (Panel Mount)

You can also open a terminal on your computer and ping a known Internet address (e.g. `8.8.8.8` is one of Google's Public DNS servers).



```
C:\Users\sgmustadio>ping 8.8.8.8

Pinging 8.8.8.8 with 32 bytes of data:
Reply from 8.8.8.8: bytes=32 time=4ms TTL=55
Reply from 8.8.8.8: bytes=32 time=4ms TTL=55
Reply from 8.8.8.8: bytes=32 time=4ms TTL=55
Reply from 8.8.8.8: bytes=32 time=5ms TTL=55

Ping statistics for 8.8.8.8:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 4ms, Maximum = 5ms, Average = 4ms

C:\Users\sgmustadio>_
```

Resources and Going Further

With the Raspberry Pi set up as an access point, you can now configure it to host a web site (e.g. using Apache), share drive space (e.g. Samba), and any other fun router customizations you might want (but on a Pi, instead!).

Looking for some inspiration for your Raspberry Pi? Check out these tutorials:

Raspberry Pi Twitter Monitor
How to use a Raspberry Pi to monitor Twitter for hashtags and blink an LED.

Raspberry gPIo
How to use either Python or C++ to drive the I/O lines on a Raspberry Pi.