My guide for setting up a raspberry pi zero w as a wifi rerouter and access point

references:
1: http://www.0xf8.org/2016/02/using-your-raspberry-pi-zeros-usb-wifi-adapter-as-both-wifi-client-and-access-point/
2: https://www.karlrupp.net/en/computer/nat_tutorial
3: https://www.raspberrypi.org/forums/viewtopic.php?f=36&t=138730&start=25
4: https://learn.adafruit.com/setting-up-a-raspberry-pi-as-a-wifi-access-point/overview

The goal of this guide is to turn a raspberry pi zero w into a device that attempts to connect to a surrounding wifi network and act as an access point. For the robot I'm working with, I'd like to reliably connect to the onboard raspberry pi 2. This raspberry pi zero w will act as a consist network through which I can connect. The added benefit of this is I can also connect to the internet if the raspberry pi zero w finds a network.

Check what channel you're on
If you haven't done so, connect your raspberry pi zero w to a wifi network. Type "iwlist wlan0 channel" and note which channel you're on. This number will be used later.

Install stuff
We'll be using the hostapd and dnsmasq libraries. Type the following to install them:
```
sudo apt-get install hostapd dnsmasq
```

Double checking configuration
Make sure when you type "iw list" this text appears at the bottom of the long output:

```
valid interface combinations:
           * #{ managed } <= 1, #{ P2P-device } <= 1, #{ P2P-client, P2P-GO } <= 1,
             total <= 3, #channels <= 2
           * #{ managed } <= 1, #{ AP } <= 1, #{ P2P-client } <= 1, #{ P2P-device } <=
1,
             total <= 4, #channels <= 1
     Device supports scan flush.
```

This means access points are supported (see reference 1 for details).

Creating the virtual interfaces
The plan is to create a fake interface called ap0 to handle our access point stuff (connecting and handling devices that want to use our raspberry pi network) and route its traffic to wlan0, the actual internet connection. wlan0 is what you usually use for wifi based connections.

udev is a device manager (for linux? raspberry pi only? I dunno). We need to define some rules about these new interfaces. Insert the following into "/etc/udev/rules.d/70-persistent-net.rules" (edit this file with `sudo nano /etc/udev/rules.d/70-persistent-net.rules`):

```
SUBSYSTEM=="ieee80211", ACTION=="add|change", ATTR{macaddress}=="xx:xx:xx:xx:xx:xx", 
KERNEL=="phy0", \
    RUN+="/sbin/iw phy phy0 interface add wlan1 type __ap", \
```

This deletes wlan0 as an interface (it comes back anyway though for some reason) to make way for the access point interface. Make sure to replace "`xx:xx:xx:xx:xx:xx`" with the mac address of your raspberry pi zero w's wlan0 device. Find it by typing "ifconfig." It's the entry titled "HWaddr".

After you reboot (which you **shouldn't** do yet), your file should look like this (udev inserts its own rules into this file):

———————————————————/etc/udev/rules.d/70-persistent-net.rules———————————————————

```
SUBSYSTEM=="ieee80211", ACTION=="add|change", ATTR{macaddress}=="yy:yy:yy:yy:yy:yy", 
KERNEL=="phy0", \
    RUN+="/sbin/iw phy phy0 interface add wlan1 type __ap", \

# Unknown net device (/devices/platform/soc/20300000.mmc/mmc_host/mmc1/mmc1:0001/
mmc1:0001:1/net/wlan0) (brcmfmac_sdio)
SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?*", ATTR{address}=="xx:xx:xx:xx:xx:xx", 
ATTR{dev_id}=="0x0", ATTR{type}=="1", KERNEL=="wlan*", NAME="wlan0"

# USB device 0x:0x (rtl8192cu)
SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?*", ATTR{address}=="yy:yy:yy:yy:yy:yy", 
ATTR{dev_id}=="0x0", ATTR{type}=="1", KERNEL=="wlan*", NAME="wlan1"
```

———

Configuring the virtual interfaces

Now we need to define the behavior of these interfaces. The raspberry pi zero w will assign an ip address of 192.168.200.1 on its own access point so I can log in through ssh if necessary and search for any wifi networks using wpa_supplicant. We'll be editing the file "/etc/network/interfaces".

In the end my file looked like this:

——————————————————————————/etc/network/interfaces——————————————————————————

```
iface default inet dhcp

auto lo
iface lo inet loopback

allow-hotplug wlan1
iface wlan1 inet static
    address 192.168.200.1
    netmask 255.255.255.0
    hostapd /etc/hostapd/hostapd.conf

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

———

Configure the access point

Now we need to define the behavior the of access point. This configuration file is in "/etc/hostapd/hostapd.conf".

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

#ctrl_interface=/var/run/hostapd
#ctrl_interface_group=0
interface=wlan1

# put the network name you want here
ssid=

# Change this (see the crda(8) and regulatory.bin(5) man pages)
country_code=US

ieee80211d=1
ieee80211h=1
ieee80211n=1
hw_mode=g

# Put the channel you found in the beginning here.
channel=1

macaddr_acl=0
wmm_enabled=1
wpa=2

# put a passphrase >=8 characters here
wpa_passphrase=

wpa_key_mgmt=WPA-PSK
wpa_pairwise=CCMP


—————————————————————————————————————————————————————————————————————————————————————
```

The reason the channel number must match the beginning is because the access point is the using the same hardware as wlan0. It can only be operating at one frequency at a time. So if you put a different channel, nothing happens and you don't see your network.

Start the service with: "sudo service hostapd start". This should be a one time command.

Add networks to connect to

Add any networks you'd like to connect to in "/etc/wpa_supplicant/wpa_supplicant.conf". This is where you should enter any credentials for any new networks you'd like to connect to. This file is pretty standard, but here's what mine looks like:

```
————————————————————————/etc/wpa_supplicant/wpa_supplicant.conf———————————————————————
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
country=US

network={
     ssid="somenetwork"
     psk="somepassword"
     key_mgmt=""
}
```

It’s recommended that you generate “psk” by typing: “sudo wpa_passphrase yourSSID yourPassword” and pasting the output into this file.

Configure DNS server
The access point isn’t quite done yet. This is where dnsmasq comes in. This library will assign IP addresses to devices that connect. We just need to tell it how we want it to be done. Open this file: “/etc/dnsmasq.conf”. It’s a big file, so I won’t paste the whole thing in. The only entries I changed were the following:

```
——————————————————————————————————/etc/dnsmasq.conf——————————————————————————————————

interface=lo,wlan1
dhcp-range=wlan1,192.168.200.5,192.168.200.200,255.255.255.0,48h
```

This tells dnsmasq which interface to assign IPs through and what values it should set. The dhcp-range parameters parses as “interface, lower IP value, upper IP value, mask, connection lease time”

Start the service with: “sudo service dnsmasq start”. This should be a one time command.

Route traffic between wlan0 and wlan1
All that’s left is to connect the two interfaces. Open “/etc/sysctl.conf” and uncomment the line that says “net.ipv4.ip_forward=1” (or add it if it’s not there).

Open “/etc/rc.local” and the following line ***before*** the “exit 0” command (if it’s there):

```
iptables -t nat -A POSTROUTING --out-interface wlan0 -j MASQUERADE
iptables -A FORWARD --in-interface wlan1 -j ACCEPT
```

The second link at the top explains how postrouting works. Essentially, packets are now being routed to the access point IP addresses.

Restart the pi and you should see your network! I made a lot of mistakes in getting this process to work because I wasn’t reading carefully. Make sure you did all the steps.

Troubleshooting
If you can connect to the network fine, but there’s no internet, make sure the pi is actually connected to a network. If that isn’t it, check if you can run the iptables command.

If you don’t see your network, check hostapd.conf to see if all the entries are fine. The channel thing is really important.

If you see your network, but are just waiting to connect, dnsmasq probably isn’t handing out an IP address. Check dnsmasq.conf.

If it says it failed to load rc.local when the RPi boots up, make sure “#!/bin/sh -e” is the very first line (no whitespace)

If “rfkill list all” says phy0 is soft blocked, run “sudo rfkill unblock all”

I recommend installing wicd-curses: “sudo apt-get install wicd-curses". It’s a command line interface for connecting to wifi. If you can’t connect, try disabling dhcpcd: “sudo systemctl disable dhcpcd” (after testing this probably doesn’t work…)

If the RPi can’t see your interface for some reason, try “sudo ip addr flush dev wlan1”. “rfkill list all” will show it’s soft blocked, so you’ll have to run “sudo rfkill unblock all”

If you find the internet interface (wlan0) is disconnecting after some long stretch of time, try disabling power save mode by adding “wireless-power off” to your /etc/network/interfaces file:

```
auto wlan0
allow-hotplug wlan0
iface wlan0 inet manual
    wpa-roam /etc/wpa_supplicant/wpa_supplicant.conf
    wireless-power off
```

Summary of files

/etc/udev/rules.d/70-persistent-net.rules
/etc/hostapd/hostapd.conf
/etc/dnsmasq.conf
/etc/network/interfaces
/etc/wpa_supplicant/wpa_supplicant.conf
/etc/rc.local