

# Working with headless Linux computers

October 31, 2019

## 1 Connecting your Raspberry Pi to Wifi

1

Connecting a Raspberry Pi to OKState University Wifi networks can be a challenge, mostly because Linux wireless networking configuration for the university network is not yet well documented or supported.

### 1.1 Prerequisites

This assumes that you have a headless Linux distribution running, have command line access and sudo privileges. Additionally, since you may often not have a mouse, I have added details for when you have only keyboard access (which makes the copy/paste operation more complicated).

### 1.2 Typical Setup

A typical Raspberry Pi setup goes something like this

1. Assemble your physical kit
2. Install Raspbian
  - (a) Use and SD card image or connect it to ethernet and hold shift during startup
  - (b) Do any software updates needed. If you won't have a mouse, you may want to install xclip while you're on the network to make copy/paste easier later.
3. Install any peripherals you like (mouse, keyboard, etc)
4. Install & Configure your display (if applicable).
  - (a) Install your display using git repository
  - (b) Configure display resolution in `/boot/conf.txt`
5. Configure & connect wifi
  - (a) Configure wifi, either your home network or university wifi
  - (b) Be sure to obfuscate your password using a hash!

### 1.3 More on connecting WIFI

The basic workflow is to edit the `wpa_supplicant.conf` configuration file, and then restart networking to reread this file. The exact edits to the config file depend on what network you are connecting to. I recommend you try a home network first (it's easier), then an enterprise network. If you're working on a headless unit, you can edit this file, and then put it into the `/boot` folder, and Raspbian will read it and use it when it boots up. If you are already connected or working on the computer, you can directly edit it in the

---

<sup>1</sup>Prepared by Imraan A. Faruque [i.faruque@okstate.edu](mailto:i.faruque@okstate.edu).

### 1.3.1 Connecting to home network

Connecting to a home network is fairly straightforward. You'll enter your SSID, and password. Because everyone who can get on your computer can see this `wpa_supplicant` file, you should obfuscate your password in some way rather than writing it in plaintext. There is more detail on that below.

To enter your connection information, add a new "network" section to your `wpa_supplicant` file. Note that you'll need super user access to do this. To start editing this file, your commands will look like

```
cd /etc/wpa_supplicant/  
cp wpa_supplicant.conf wpa_supplicant.conf.bak  
sudo nano wpa_supplicant.conf
```

The first line navigates you to the folder containing this file, the second line creates a backup copy (in case we somehow break the file), and the third opens a small, lightweight editor called nano. Sudo (super-user do) is needed because we're editing an operating system file that has some protections on it.

Add a section with the following:

```
network={  
  ssid="MYSSID"  
  #psk = "passphrase"  
  psk=LONGLISTOFCHARACTERS
```

Replace MYSSID and passphrase with your network SSID and the wifi password you use. I listed both `psk` and `#psk` options because you can either type your password in plaintext or type the preshared key, which combines the SSID and passphrase into a "hash" that is used in authentication. I think you only need one of the lines, but I haven't verified this in a while. On most systems, Raspbian included, there is a built in command line tool that can help. Type

```
wpa_passphrase MYSSID passphrase
```

When you're done editing the file, push CTRL X to indicate you want to exit, push Y for yes when it asks you if you want to save the file, and push return to confirm you want to overwrite the current file.

### 1.3.2 Connecting to an un-secured Wifi network

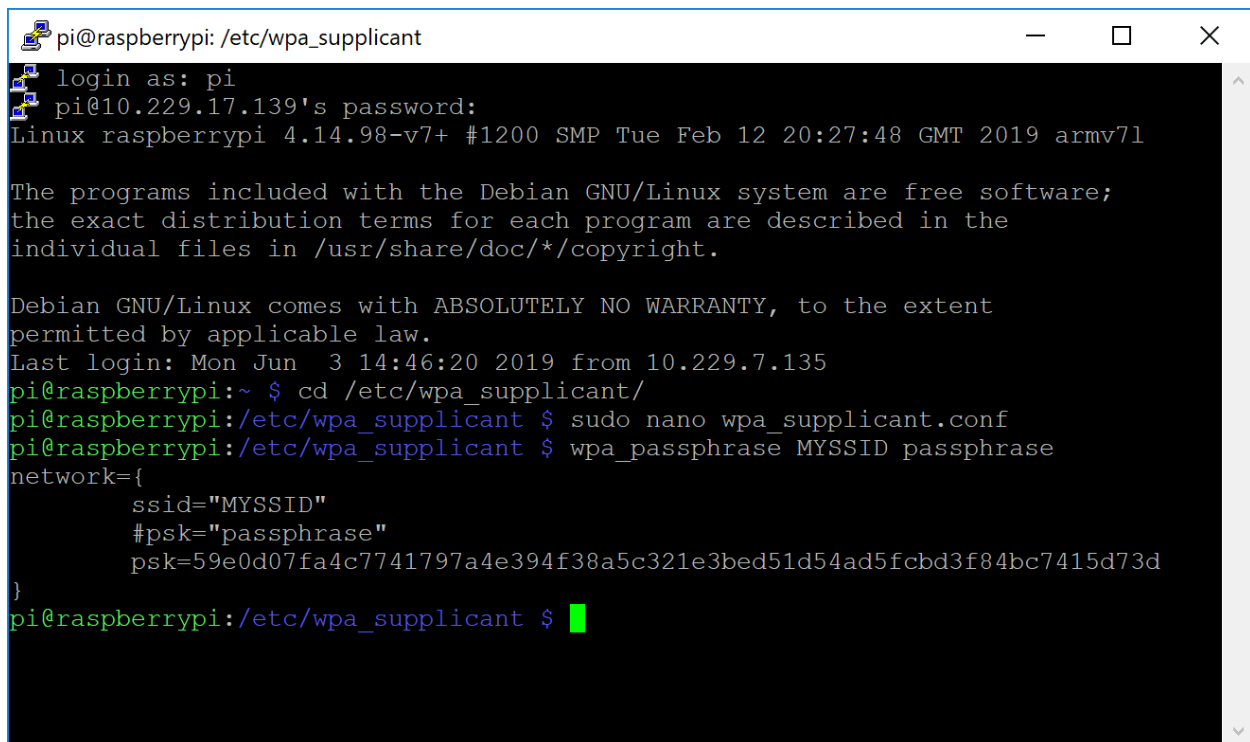
Same process, but add

```
network={  
  ssid="Stillwater RC Flyers"  
  key_mgmt=NONE  
  priority=2  
}
```

### 1.3.3 Connecting to OKState wifi

OKState has transitioned to the eduroam system that logs in by tunneling through to a RADIUS server on the home campus with your credentials. The radius server either approves or denies the login request, which is forwarded back to the local eduroam network. To connect to this kind of enterprise system with a login name, you'll need to add a few more lines to your file. For reference, here is a file to connect to Eduroam wifi:

```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev  
update_config=1  
country=US  
  
network={
```

A terminal window titled 'pi@raspberrypi: /etc/wpa\_supplicant' with standard window controls. The terminal shows a login sequence for user 'pi' at IP '10.229.17.139'. It displays the Linux version '4.14.98-v7+' and the date 'Tue Feb 12 20:27:48 GMT 2019'. After displaying the Debian GNU/Linux copyright notice, the user navigates to '/etc/wpa\_supplicant' and runs 'sudo nano wpa\_supplicant.conf'. The configuration file content is shown, including a network entry for 'MYSSID' with a 'passphrase'.

```
pi@raspberrypi: /etc/wpa_supplicant
login as: pi
pi@10.229.17.139's password:
Linux raspberrypi 4.14.98-v7+ #1200 SMP Tue Feb 12 20:27:48 GMT 2019 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Mon Jun  3 14:46:20 2019 from 10.229.7.135
pi@raspberrypi:~ $ cd /etc/wpa_supplicant/
pi@raspberrypi:/etc/wpa_supplicant $ sudo nano wpa_supplicant.conf
pi@raspberrypi:/etc/wpa_supplicant $ wpa_passphrase MYSSID passphrase
network={
    ssid="MYSSID"
    #psk="passphrase"
    psk=59e0d07fa4c7741797a4e394f38a5c321e3bed51d54ad5fcbd3f84bc7415d73d
}
pi@raspberrypi:/etc/wpa_supplicant $
```

Figure 1: Using WPA passphrase to generate the network entry needed.

```
ssid="eduroam"
priority=1
proto=RSN
key_mgmt=WPA-EAP
pairwise=CCMP TKIP %not needed
auth_alg=OPEN
eap=PEAP
identity="i.faruque@okstate.edu"
password=hash:PASSWORDHASHGOESHERE
phase1="peaplabel=0"
phase2="auth=MSCHAPV2"
}
```

You'll need to replace the text after `password=` with either your plaintext password (this will work, but don't leave it this way because it is a big security hole) or a hashed version of your password.

### 1.3.4 Hashing your password

To hash your password,

```
echo -n password_here | iconv -t utf16le | openssl md4
```

Here, we are using the “|” symbol, which is known as the “pipe” symbol, to send the output of a command to a different place.<sup>2</sup>

---

<sup>2</sup>If the key labeled does not produce what you like, it is possible you have the wrong keyboard layout, use `sudo raspi-config` or `sudo dpkg-reconfigure keyboard-configuration` to select the correct keyboard layout, and then `sudo setupcon` to apply it.

```
pi@oknav: /etc/wpa_supplicant
pi@oknav:/etc/wpa_supplicant $ cd /etc/wpa_supplicant/
pi@oknav:/etc/wpa_supplicant $ cat wpa_supplicant.conf
country=US
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1

network={
    ssid="EML33T2"
    #psk="emrooftop"
    psk=c150059ac2d9df589127a8ee0a1cc475099c2d6e60ea48644c21684cb7ee6b23
}

network={
    ssid="eduroam"
    proto=RSN
    key_mgmt=WPA-EAP
    auth_alg=OPEN
    eap=PEAP
    identity="i.faruque@okstate.edu"
    password=hash:
    phase1="peaplabel=0"
    phase2="auth=MSCHAPV2"
    priority=1
}

network={
    ssid="Rachraan"
    psk=
}

network={
    ssid="Stillwater RC Flyers"
    priority=2
    key_mgmt=NONE
}

network={
    priority=-999
    key_mgmt=NONE
}
```

Figure 2: Example to connect to OKState WIFI (password hash redacted).

You'll get something out that looks like

```
ssl md4
(stdin)= LONGLISTOFCHARACTERS
```

The long list of characters is a md4 hash of the password you entered. It's enough to connect to the network, but the hashed password is computationally-difficult to un-hash (relatively). You'll likely now want to copy paste this into your file, but Linux and its derivatives are actually older than the CTRL-C and CTRL-V shortcuts you may be used to. In fact, by the time Windows introduced these, they already had a purpose in the operating system. I'll presume you're not using a mouse, in which case you need to also select these without a mouse to drag. There are a few ways to do this,<sup>3</sup>.

**Save to file** A neat one is to pipe the output to a file. Unfortunately, you're in a write protected directory and you might need to give yourself permissions.

**XClip** in an Xterminal, I use a tiny program called xclip. To install it, use `sudo apt-get install xclip`. By default, xclip is set to copy piped input to the clipboard. `xclip-o` will let you see what this looks like. In my case, it is the hashed password because that is the last piped input. If this wasn't the case, you could use pipe it into the keyboard using `COMMAND |xclip -sel clip`. Now, open the

<sup>3</sup>[https://wiki.archlinux.org/index.php/Copying\\_text\\_from\\_a\\_terminal](https://wiki.archlinux.org/index.php/Copying_text_from_a_terminal)

wpa\_supplicant.conf file again with nano and use SHIFT-INS.<sup>4</sup>

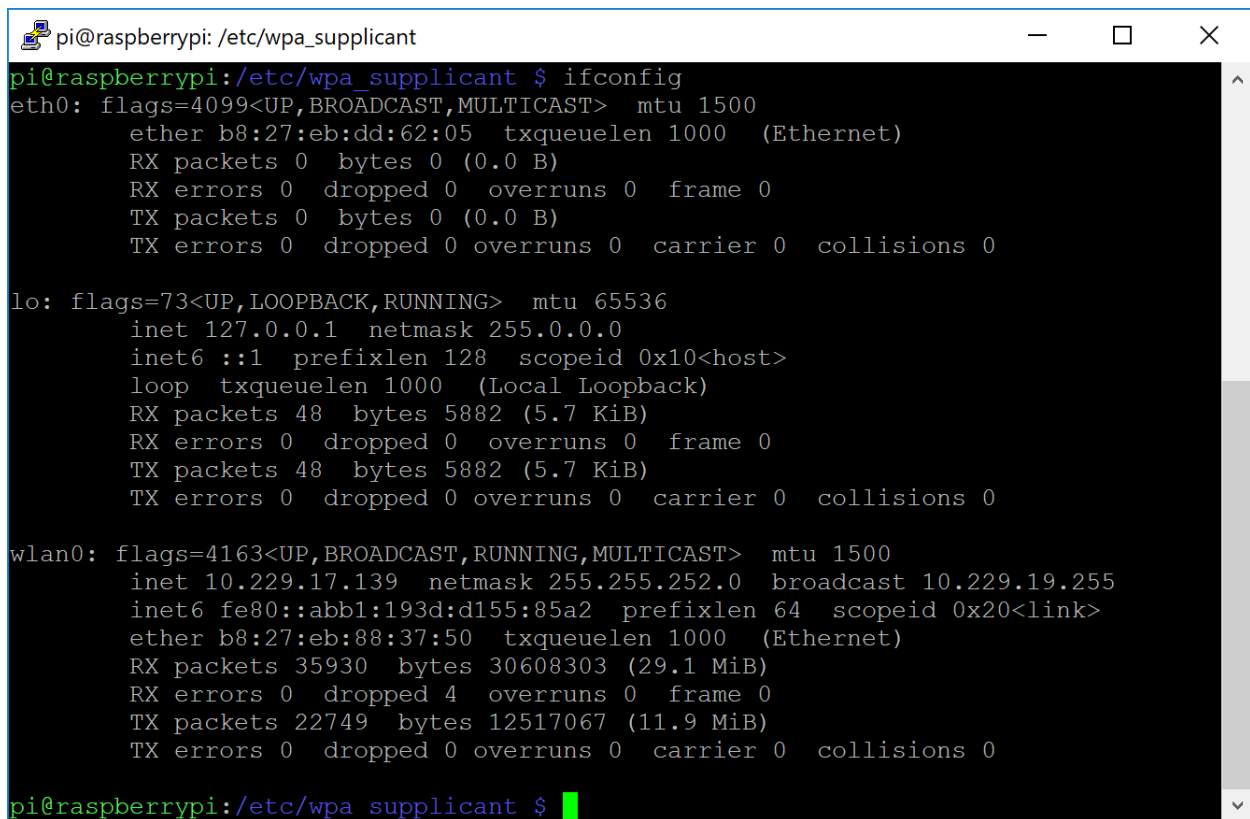
**clip.exe** If this is Windows subsystem for Linux, you have access to all Windows executables, and you can use the Windows program **clip.exe**. To do this, pipe the input into it. For example, `echo funny | clip.exe` copys the word “funny” onto the clipboard, while `cat report.txt | clip.exe` copies the content of report.txt to the clipboard.

Your system is now configured for wifi. To connect, you need to restart networking.

First check to see what its name is and if it has gotten an IP address using **ifconfig**. You should see a list of interfaces, usually eth0, lo, and wlan0. eth0 is the wired LAN port, lo is a loopback (software) interface, and wlan0 is the wirelass LAN adapter we care about. It will probably not list an ip address yet (it would look like inet num.num.num.num. Let’s restart networking. One way to do this is to restart thet computer. But there are plenty of times when you might not want to do that. Instead, use the following commands:

```
ifconfig
sudo ifconfig wlan0 down
sudo ifconfig wlan0 up
```

The first one lets us see the names of the interfaces (and if they are connected), the second and third turn off the interface and bring it back up. Now try ifconfig again and you should see an ip address listed under wlan0.



```
pi@raspberrypi: /etc/wpa_supplicant
pi@raspberrypi:/etc/wpa_supplicant $ ifconfig
eth0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
    ether b8:27:eb:dd:62:05 txqueuelen 1000  (Ethernet)
    RX packets 0  bytes 0 (0.0 B)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 0  bytes 0 (0.0 B)
    TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000  (Local Loopback)
    RX packets 48  bytes 5882 (5.7 KiB)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 48  bytes 5882 (5.7 KiB)
    TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.229.17.139 netmask 255.255.252.0 broadcast 10.229.19.255
    inet6 fe80::abb1:193d:d155:85a2 prefixlen 64 scopeid 0x20<link>
    ether b8:27:eb:88:37:50 txqueuelen 1000  (Ethernet)
    RX packets 35930  bytes 30608303 (29.1 MiB)
    RX errors 0  dropped 4  overruns 0  frame 0
    TX packets 22749  bytes 12517067 (11.9 MiB)
    TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

pi@raspberrypi:/etc/wpa_supplicant $
```

Figure 3: Here we can see that an IP address has been issued.

<sup>4</sup>If you’re like me and have a mac keyboard without an insert key, use “SHIFT-FN-RETURN” as “FN-RETURN” works like insert.

## 2 Finding your computer (Hostnames and IP addresses)

The previous instructions should get you on the network. If you know the computer's IP address, you can connect to it. Sometimes this is tough to know because they change, but if we know its hostname, the network will translate that to it. How you can do this depends on the network.

### 2.1 Your own local network

A lot of students like to hotspot their phones and setup both devices to connect to it—this becomes your own home network and the phone (acting as a router) will show you the IP addresses of the connected devices.

### 2.2 Home (small) network

If it's a small network like your home, there are few enough devices on it that you can often list them easily. An app on your phone like Fing will keep track of these IP addresses, however, watch out for port scan programs on a network that's not your own because they try lots of computers and a system administrator might get worried if it looks like you are looking for a hole.

If you don't want to keep track of a changing IP address, give your Raspberry Pi a hostname, like "oknav." To determine your local hostname, type `hostname`. To change this, edit the file `/etc/hostname`. Go ahead and take a minute to set your hostname to something that makes sense, like your team name.

Once you know the hostname then you can find it via something like `ping oknav.local`. This command is likely to work on Linux, Unix, or Mac systems out of the box. Since Windows has historically had some broken networking things behind the scenes, you might need to add a multicast mDNSResponder. Windows 10 now supports mDnS, and many programs (e.g., iTunes) adds it without telling you, so try the command before you make configuration changes. If the command doesn't work, install any program that uses mDNS, like Apple's Bonjour print server, which monitors your network for printers.

### 2.3 On Eduroam

On a bigger enterprise network, listing the devices is impractical because you'll only be on the same router as a few of them. Furthermore, our network has only internal IP addresses for security, and thus even if we set that computer to dial out to try to tell you its IP address, it would just show the outside IP address of the whole university. So we first rename our hostname, and then take advantage of the structure. Our network issues IP addresses like `hostname.eduroam-corew1cs-XX.okstate.edu`, where XX is a number corresponding to the terminal you are connected to.

## 3 Running commands remotely

First, SSH into your computer, using `ssh pi@IpAddressHere`. After you enter your password, you can now run programs like you were sitting at that computer.

### 3.1 Disconnects

However, a program started via an SSH terminal will stop when you disconnect your session, which is likely to happen when your vehicle flies out of WiFi range. To prevent this, create a `screen` session that you can find later. Create a screen by `screen -S friendlyName` where `friendlyName` is something you will remember. If you get disconnected, your programs will keep running, and the next time you login, you can find this screen by `screen -R friendlyName`. You can create multiple screens if needed, just use a different name. If you forget the name, use `screen -l` to see a list of the screens available.

## 3.2 Running programs on boot

For an embedded system like an autopilot, a better solution is to start the program code on startup to avoid the need for us to login and start things. If you later need to edit this program, it is helpful to learn how to stop this program, or kill this process.

A great way to do this is by creating a system service. You'll create a file to define the system, then reload the daemon that indexes it, then start it. Create a file using

```
sudo nano /etc/systemd/system/YOURSERVICENAME.service
```

In this file, define the service:

```
Description=GIVE_YOUR_SERVICE_A_DESCRIPTION
```

```
Wants=network.target
```

```
After=syslog.target network-online.target
```

```
[Service]
```

```
Type=simple
```

```
ExecStart=YOUR_COMMAND_HERE
```

```
Restart=on-failure
```

```
RestartSec=10
```

```
KillMode=process
```

```
[Install]
```

```
WantedBy=multi-user.target
```

Save and exit, then we reload services

```
sudo systemctl daemon-reload
```

Enable this new service:

```
sudo systemctl enable YOUR_SERVICE_NAME
```

Start this new service:

```
sudo systemctl start YOUR_SERVICE_NAME
```

Verify that it did start:

```
sudo systemctl status YOUR_SERVICE_NAME
```

This process will restart if killed (here it waits 10 seconds), when you tell it to stop, it will send a SIGINT signal.

If you need to, you can also **stop** or **disable** the service by replacing one word above.

## 3.3 Transferring files to/from

If you edit your program on your home computer, one way to upload it is via secure copy, or the `scp` command.

## 4 References

5

---

<sup>5</sup>The following references are helpful for using Raspberry Pi in this context.<https://www.instructables.com/id/Connect-Raspberry-Pi-to-College-WIFI/> <https://bbs.archlinux.org/viewtopic.php?id=144471> <https://pimylifeup.com/raspberry-pi-mount-usb-drive/> <https://www.raspberrypi.org/forums/viewtopic.php?t=22957> <https://pimylifeup.com/raspberry-pi-ntfs/> <https://www.raspberrypi.org/documentation/remote-access/ssh/>