Raspberry Pi Packet Radio Station

by KM4MLS

What you will need:

Items you'll need to run this tutorial/guide:

- Raspberry Pi Model B (or newer, ethernet is required)
- Ethernet Cable
- WiFi Adapter (if not built-in; Raspberry Pi 3 Model B has WiFi on-board)
- SD Card (4GB or larger)
- Power Supply (5v@1.5A+)
- USB-Micro Cable (for power supply)
- USB Sound Card
- Sound Card Radio Interface Circuit (see the appendix for details and B.O.M.)
- Desktop PC (or laptop) for initial set-up of the Raspberry Pi

Required Software for Set-Up:

- Direwolf Software "Sound Card" Modem/TNC
  (See instructions for download Information)
  https://github.com/wb2osz/direwolf
- LINBPQ (BPQ32 for Linux) Packet Node Switch
  (See instructions for download Information)
  http://www.cantab.net/users/john.wiseman/Downloads/
- KM4MLS Raspberry Pi Packet Station Set-Up Scripts
  (See instructions for download information)
  https://github.com/km4mls/KM4MLS-Raspberry-Pi-Packet-Node

Recommended Software for Set-Up:

- Win32DiskImager - Download Here
  https://sourceforge.net/projects/win32diskimager/files/latest/download
- SDFormatter - Download Here
  https://www.sdcard.org/downloads/formatter_4/eula_windows/index.html
- PuTTY – Download Here
  https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html

Getting Started:

First of all we need a freshly formatted SD card (4GB or larger) to burn the Raspbian OS to. Insert the SD card into your SD card reader and open SDFormatter. Click “Option” and make sure “Format Type” is set to “Quick” and “Format Size Adjustment” is set to “On”. Select the drive letter of the SD card and press the “Format” button.

Download the latest version of Raspbian. You may find the latest version on the Raspberry Pi Foundation's web page.
https://www.raspberrypi.org/downloads/raspbian/

After downloading the Raspbian image. Burn the image to an SD card (4GB or larger) using Win32DiskImager (Windows Users) or your favorite image writer application for SD cards for your preferred OS.

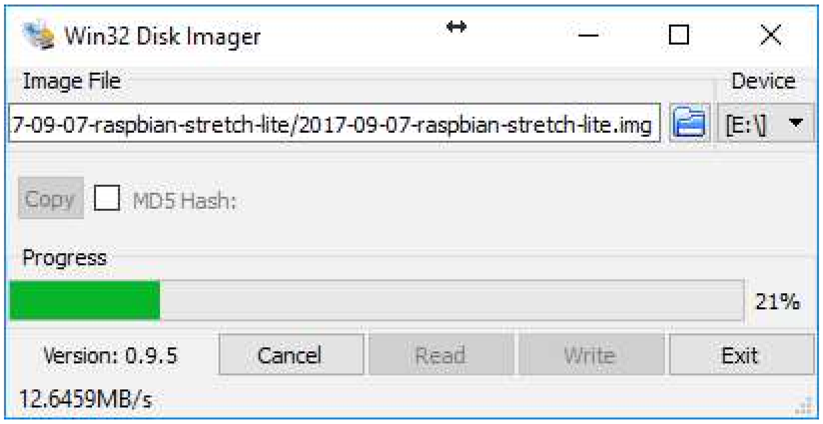

Win32 Disk Imager

Image File: 7-09-07-raspbian-stretch-lite/2017-09-07-raspbian-stretch-lite.img

Device: [E:\]

MD5 Hash:

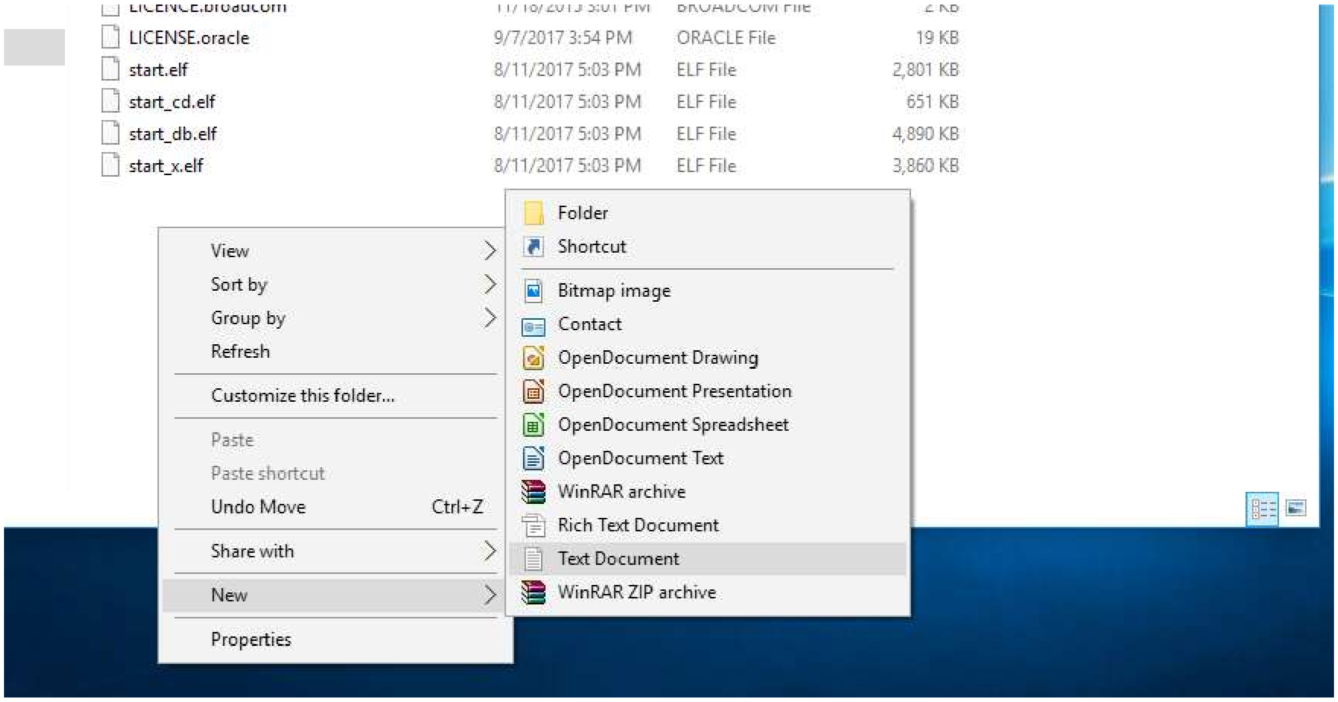
Progress: 21%

Version: 0.9.5

12.6459MB/s

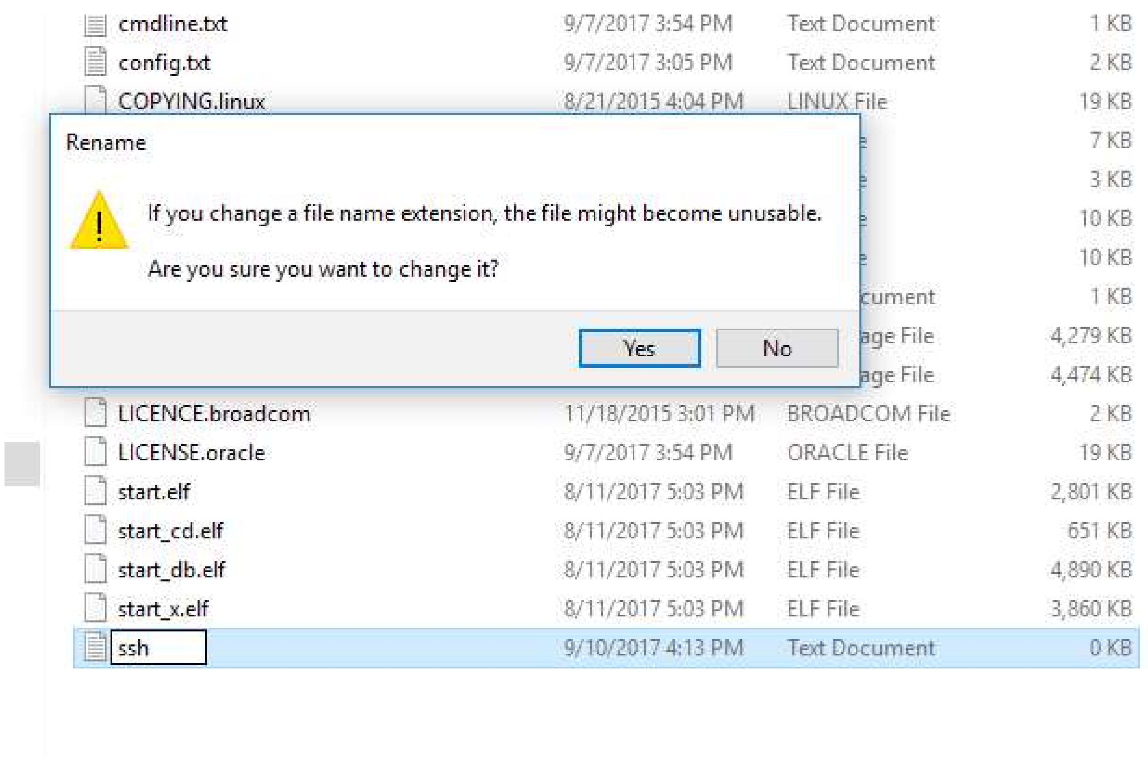
Open the unzipped image and make sure the correct device (drive letter) is selected before pressing “Write”

Once burned add a file named “ssh” (empty file with no file extension) to the BOOT partition. This will enable ssh without having to connect a monitor and keyboard (headless.)



LICENSE.oracle	9/7/2017 3:54 PM	ORACLE File	19 KB
start.elf	8/11/2017 5:03 PM	ELF File	2,801 KB
start_cd.elf	8/11/2017 5:03 PM	ELF File	651 KB
start_db.elf	8/11/2017 5:03 PM	ELF File	4,890 KB
start_x.elf	8/11/2017 5:03 PM	ELF File	3,860 KB

Create a new text document.



cmdline.txt	9/7/2017 3:54 PM	Text Document	1 KB
config.txt	9/7/2017 3:05 PM	Text Document	2 KB
COPYING.linux	8/21/2015 4:04 PM	LINUX File	19 KB
		e	7 KB
		e	3 KB
		e	10 KB
		e	10 KB
		cument	1 KB
		age File	4,279 KB
		age File	4,474 KB
LICENCE.broadcom	11/18/2015 3:01 PM	BROADCOM File	2 KB
LICENSE.oracle	9/7/2017 3:54 PM	ORACLE File	19 KB
start.elf	8/11/2017 5:03 PM	ELF File	2,801 KB
start_cd.elf	8/11/2017 5:03 PM	ELF File	651 KB
start_db.elf	8/11/2017 5:03 PM	ELF File	4,890 KB
start_x.elf	8/11/2017 5:03 PM	ELF File	3,860 KB
ssh	9/10/2017 4:13 PM	Text Document	0 KB

Rename

If you change a file name extension, the file might become unusable.

Are you sure you want to change it?

Then, rename to ssh, removing the .txt extension. If you get a warning like the one above, click "Yes"

Insert the SD card into the Pi and connect the Pi to a wired ethernet connection on your LAN and turn it on. The Raspberry Pi requires a 5v power supply capable of delivering at least 1.5 Amps. To power the Pi, you may use a smart-phone charger with a USB-micro charging cable.

(Note: When powering down the Pi, always use the shutdown command,

```
sudo shutdown -h now
```

to properly shut down the Pi. Just pulling the power cord out without properly shutting down the Pi will likely corrupt data on the SD card rendering it useless.)

Wait a moment for the Pi to boot and connect to the network then open your router's web configuration page or whatever app you use to configure your network.

Look for the DHCP table in your routers configuration app/ web configuration and find the listing for raspberry-pi. Take note the IP address of the PI and write it down. Keep it handy; we'll need it later.

Connecting to the Pi:

Now that you have the Pi's IP address you can open your favorite SSH client and connect to the Pi from your PC. My favorite SSH client is a program called PuTTY. It's free to use and also has other communication protocols that you may need later (ie. RS-232 Serial and Telnet clients) to control other devices of your station.

To connect to the Pi over SSH from PuTTY, first open PuTTY. The default Connection Type should already be set to SSH, if not, make sure the radio box for SSH is ticked under “Connection Type”. Now in the text box under “Host Name (or IP address)” enter the IP address of the Pi; leave the port set to 22. Now click “Open” at the bottom of the window to open the connection.

The first time you open a new connection to a new device, PuTTY will display a warning about a potential security breach due to the SSH encryption keys not matching what PuTTY has cached. This is because it's the first time PuTTY has seen this device and therefore doesn't have the keys cached. Click “Yes” to continue, you shouldn't see this message again unless you change the Pi's IP address or MAC address, or if you change the network adapter on the Pi.

Logging in:

Once connected you will be presented with a log-on prompt asking for your user-name (“login as”.) Enter the default user-name “pi”. Then you will be asked for the password. Enter the default password “raspberry”. Congratulations! You are now logged-in to your new Raspberry Pi!

Setting Up your User Account:

Now that you are connected it's time to change the default user-name and password to something unique. You may leave everything default if you like, but every Raspberry Pi in the world contains the same default log-on credentials, so it is a good idea to change them.

The default user “pi” can not be changed while you're logged-in to it, so before we change it, we must create a temporary user account to log-in to to make the changes.
We'll start by creating that temporary user. The lines following in `blue` are commands that should be typed into the command line of the Pi from the SSH client while logged-in. `Black` text depicts text returned by the SSH console after entering a command. Any text appearing in `RED` should be edited to suit your application before entering the command.

For Example:

```
This depicts a command to be entered and this should be changed
```

```
And this is text returned or displayed by the console
```

To create the new temp user enter in to the command line:

```
sudo useradd -m tempuser
```

Now to set the password for tempuser:

```
sudo passwd tempuser
```

Now you will be prompted twice to “Enter new UNIX password:”, the tempuser's new password, we'll use the password “temp” for now:

```
temp
```

```
temp
```

Now in order to give the tempuser account the permissions it needs to modify an administrator account we need to had it to the “sudo” group.

```
sudo usermod -a -G sudo tempuser
```

To check and make sure tempuser was indeed added to the “sudo” group, we can use the command:

```
grep sudo /etc/group
```

The command should return something like this:

```
sudo:x:27:pi, tempuser
```

The exact string isn't important, so long as you see tempuser listed.

Now that we have created the tempuser account, reboot the Pi:

```
sudo reboot
```

Once rebooted log back in, this time using the “tempuser” account, instead of “pi”

```
login as:
tempuser
```

```
pi@192.168.0.07's password:
temp
```

Now, before we change the name of the “pi” account, we want to back up all the files we're about to change. Lucky for us, Linux has a nifty command just for such things. See, command line interfaces aren't so bad, right? Heck, they can be down right useful! Ok, first let's move to the /etc directory:

```
cd /etc
```

Now, to back up the files we want we must enter the command properly, the command will do only what you tell it, so we must be specific. It may be easier for you to just copy and paste the command, but be very careful to get is absolutely as written (the following command is all one line, the text wraps the screen).

If you used the “Desktop” version of Raspbian use this command:

```
sudo tar -cvf authfiles.tar passwd group shadow gshadow sudoers
lightdm/lightdm.conf systemd/system/autologin@.service sudoers.d/* polkit-
1/localauthority.conf.d/60-desktop-policy.conf
```

If instead you used the “Lite” version of Raspbian, like I generally do with systems that I run without a monitor. Use this command:

```
sudo tar -cvf authfiles.tar passwd group shadow gshadow sudoers
systemd/system/autologin@.service sudoers.d/*
```

You will likely be asked for your password again when running the above command, enter it if asked (“temp”) and continue. If not, disregard.

```
temp
```

Now that the files are backed up, we need to make our changes. First, ensure we're still in the /etc directory:

```
cd /etc
```

Then we will make the changes needed. Before entering the following command, you need to enter the name that you wish to change the “pi” account to into the command. In this example, I will use my callsign. When you set up your Raspberry Pi packet station, you should change this to the user-name you wish to use. You do not have to use your callsign here, but you may if you wish (I did.) Before running the following command, change the **RED** text to your desired user-name (also remember this is all one line):

Again if you're running the “Desktop” version of Raspbian, use this command:

```
sudo sed -i.$(date +'%y%m%d_%H%M%S') 's/\bpi\b/km4mls/g' passwd group
shadow gshadow sudoers lightdm/lightdm.conf
systemd/system/autologin@.service sudoers.d/* polkit-
1/localauthority.conf.d/60-desktop-policy.conf
```

Otherwise, use this one:

```
sudo sed -i.$(date +'%y%m%d_%H%M%S') 's/\bpi\b/km4mls/g' passwd group
shadow gshadow sudoers systemd/system/autologin@.service sudoers.d/*
```

The “sed” command above changes every occurrence of the word “pi” in each of the files to “km4mls”. Before the change is made, however, another backup is made of each file, just in case something went wrong. This is probably not needed, but better safe than sorry, as they say.

To check that the changes were made properly you may enter the following command, again replacing the **RED** text with your chosen user-name:

```
grep km4mls /etc/group
```

You should see something like this returned:

```
adm:x:4:km4mls
dialout:x:20:km4mls
cdrom:x:24:km4mls
sudo:x:27:km4mls,tempuser
audio:x:29:km4mls
video:x:44:km4mls
plugdev:x:46:km4mls
games:x:60:km4mls
users:x:100:km4mls
input:x:101:km4mls
netdev:x:108:km4mls
km4mls:x:1000:
spi:x:999:km4mls
i2c:x:998:km4mls
gpio:x:997:km4mls
```

The last thing to do is to take care of the “pi” user's home directory, renaming it to your new user-name, and again replacing the **RED** text with your new user-name:

```
sudo mv /home/pi /home/km4mls
```

Then we create a “symbolic link” to re-direct any applications that may try to write to the “pi” user's home directory which is no longer there to the home directory of your new user account. Do I have to mention the **RED** text anymore, I think you have the hang of it now, right?

```
sudo ln -s /home/km4mls /home/pi
```

Ok, that's it for renaming the “pi” user to your choice of user-name. Before we get rid of the tempuser account, let's reboot and test our new account. We did not change the password for “pi” so when logging in to your new user-name, use the same password as was previously set for “pi” (“raspberry”.)

```
sudo reboot
```

Log back in under your new user-name, if all is well you should have no issues logging-in and your new account should have all the same privileges as the old “pi” account. To check this enter:

```
ls -al
```

You should see a list similar to this returned (if you're running the “Lite” version of Raspbian you won't see most of these directories listed):

```
total 96
drwxr-xr-x 16 km4mls km4mls 4096 Sep  4 03:16 .
drwxr-xr-x  4 root   root   4096 Sep  4 03:51 ..
-rw-------  1 km4mls km4mls  107 Sep  4 03:16 .bash_history
-rw-r--r--  1 km4mls km4mls  220 Aug 16 00:23 .bash_logout
-rw-r--r--  1 km4mls km4mls 3523 Aug 16 00:23 .bashrc
drwxr-xr-x  5 km4mls km4mls 4096 Sep  4 03:10 .cache
drwx------  7 km4mls km4mls 4096 Sep  4 03:10 .config
drwxr-xr-x  2 km4mls km4mls 4096 Aug 16 01:41 Desktop
drwxr-xr-x  5 km4mls km4mls 4096 Aug 16 01:11 Documents
drwxr-xr-x  2 km4mls km4mls 4096 Aug 16 01:41 Downloads
drwx------  3 km4mls km4mls 4096 Aug 16 01:41 .gnupg
drwxr-xr-x  3 km4mls km4mls 4096 Aug 16 01:11 .local
drwxr-xr-x  2 km4mls km4mls 4096 Aug 16 01:41 Music
drwxr-xr-x  2 km4mls km4mls 4096 Aug 16 01:41 Pictures
-rw-r--r--  1 km4mls km4mls  675 Aug 16 00:23 .profile
drwxr-xr-x  2 km4mls km4mls 4096 Aug 16 01:41 Public
drwxr-xr-x  2 km4mls km4mls 4096 Aug 16 01:11 python_games
drwxr-xr-x  2 km4mls km4mls 4096 Aug 16 01:41 Templates
drwxr-xr-x  3 km4mls km4mls 4096 Sep  4 03:10 .themes
drwxr-xr-x  2 km4mls km4mls 4096 Aug 16 01:41 Videos
-rw-------  1 km4mls km4mls   56 Sep  4 03:12 .Xauthority
-rw-------  1 km4mls km4mls 4234 Sep  4 03:14 .xsession-errors
-rw-------  1 km4mls km4mls 4055 Sep  4 03:10 .xsession-errors.old
```

Before we remove the tempuser account, lets change our password on our new account from the default “raspberry” so something a bit more suitable and secure:

```
passwd km4mls
```

You will be asked to enter your current password (“raspberry”) then your desired new password twice, once you've done so, your password is now changed.

```
raspberry
```

```
yournewpass
```

```
yournewpass
```

Ok, now we're finally ready to remove the tempuser account, after which we can get to the good stuff. Thanks for hangin' with me so far. See this whole Linux thing isn't too bad, it just takes a bit of patience! :) To delete the tempuser account, enter:

```
sudo userdel tempuser
```

Now the tempuser account is deleted. It's good practice, when making changes to the system, to reboot; let's go ahead and reboot again then log back in to your new account.

```
sudo reboot
```

That's it! Your Raspberry Pi is now customized with your own user-name and password, and is well on it's way to becoming an AX.25 Packet Radio Station!

Direwolf Sound Card Modem/TNC:

In order for your station to be able to send packets over the air, you have to have a way of interfacing your radio with the Raspberry Pi. Back in the day, when PC's weren't as powerful, this meant purchasing a Terminal Node Controller (TNC) that interfaced the radio with your PC over an RS-232 serial connection (COM port). Now days, we have PC's the size of a pack of chewing gum that can handle the task of decoding the packet signals as well as running all sorts of other services that may come in handy (ie. RMS, BBS, Chat, APRS, etc...) all in one machine.

Our Raspberry Pi packet station is a little larger than a pack of gum, but can handle all of the above mention tasks and more. So, instead of using a traditional TNC, we'll be using a “sound card TNC”. Therefore, we need to install some software made for just this task. There are a few different software options to choose from, but in this example we'll be using the Direwolf TNC emulator software package written by WB2OSZ. Direwolf was designed with the Raspberry Pi in mind, so it runs well on the platform. As WB2OSZ states on the github page where Direwolf is hosted:

"Dire Wolf is a software "soundcard" modem/TNC and APRS encoder/decoder. It can be used stand-alone to observe APRS traffic, as a digipeater, APRStt gateway, or Internet Gateway (IGate). It can also be used as a virtual TNC for other applications such as APRSIS32, UI-View32, Xastir, APRS-TW, YAAC, UISS, Linux AX.25, SARTrack, RMS Express, BPQ32, Outpost PM and many others. "

In our application we will be using Direwolf as our TNC for interfacing to LINBPQ (the Linux version of BPQ32). LINBPQ is the packet node controller/switch that runs all of the services we wish to have on our node and makes sure the packets go where they are supposed to. But more on BPQ later.

Installing Direwolf is fairly straight-forward, the author has instructions posted on the github page where he hosts the downloads. The github page can be found Here: https://github.com/wb2osz/direwolf .

Installing Direwolf:

The easiest way to install the most up-to-date version of Direwolf is to get is directly from github. Fortunately, Linux has a handy program just for this called git. Git may already be installed if you're running the "Desktop" version, but just in case it's not we will run the install command to check.

```
sudo apt-get install -y git libasound2-dev screen
```

If git was not installed (or if there's a newer version) this will authorize the system to install/upgrade it along with the other two packages Direwolf needs in order to be able to run automatically on the Pi without a monitor. Now just wait until it completes the install.

Now that we have git installed and Direwolf's other dependencies, we can download and install Direwolf.

Execute the command for git to download the Direwolf package:

```
git clone https://www.github.com/wb2osz/direwolf
```

Wait for the download to complete, then move to the newly created direwolf directory.

```
cd direwolf
```

Then run the make command to build/compile the software and wait for it to finish. This may take a while, have patience.

```
make
```

Now we have it built, we'll install it.

```
sudo make install
```

And finally we'll copy the config file to our home directory.

```
make install-conf
```

And then copy an auto-start script to our home directory.

```
cp dw-start.sh ~
```

Now go ahead and set up Direwolf to autostart on boot and restart if it shuts down during operations for any reason. In this case we'll be using the crontab task scheduler to launch the dw-start.sh script whenever the Pi boots up and if for some reason Direwolf crashes or gets closed, crontab will check once per minute to see whether or not Direwolf is still running and if not will relaunch it.

```
crontab -e
```

If this is the first time you've ran crontab, it will ask you which text editor you prefer, select Nano (/bin/nano); it was selection number 2 in my case.
Once crontab is opened, scroll to the bottom and enter the following lines, each on a separate line:

```
@reboot /home/km4mls/dw-start.sh >/dev/null 2>&1
```

```
* * * * * /home/km4mls/dw-start.sh >/dev/null 2>&1
```

Be sure that there are no blank lines in the crontab file.

To save and exit press [CTRL]+[X], and answer “y” followed by [RETURN].

Now lets change the auto-start script run-mode so it starts in CLI (command line interface) mode since we're not using the desktop environment:

```
cd ~
```

```
nano dw-start.sh
```

Find the line that reads (around line #32):

```
RUNMODE=AUTO
```

And change it to:

```
RUNMODE=CLI
```

Then save and exit by entering:

```
[CTRL] + [X]
```

and answering,

```
y
```

followed by,

```
[RETURN]
```

Now let's reboot to make sure our changes are implemented.

```
sudo reboot
```

Then log back in.

Alright, Direwolf is now installed and will now start on boot-up. For now we'll wait on modifying the configuration for our station.

LINBPQ Packet Node Switch:

If you have made it this far give yourself a pat on the back, you're doing great!

Next we will install LINBPQ, the Linux version of the BPQ32 Packet Node Switch software package, developed by John Wiseman, G8BPQ.

Mr. Wiseman, has been developing BPQ for over twenty years and it has evolved into a very powerful and extensive software package for managing packet radio nodes. Our station will only implement the basics needed to run a packet station which implements a Mailbox (BBS) node, a Chat node, and an RMS (Winlink)

node. Note that for the RMS node to function properly, you must have an internet connection established on the ethernet port of the Raspberry Pi or over WiFi. LINBPQ does have the capability to create a V/UHF packet gateway to HF RMS and CMS stations, but that is beyond the scope of this how-to. If you are interested in such a gateway or anything else concerning BPQ32, you may find further documentation on the BPQ32 website Here
http://www.cantab.net/users/john.wiseman/Documents/index.html . You can also get support for BPQ32 (all variants) at the BPQ32 Yahoo Group .
https://groups.yahoo.com/neo/groups/BPQ32/info

Installing LINBPQ:

Installing LINBPQ, is just as easy as installing Direwolf (actually easier in my opinion); it's just a matter of downloading the package, configuring, and then running it. The configuration is the hard part, but we'll save that for later. We'll start by creating a directory for LINBPQ and then downloading the latest version. Then we'll rename the file downloaded from pilinbpq to linbpq and make it executable.

```
mkdir linbpq
```

```
cd linbpq
```

```
wget http://www.cantab.net/users/john.wiseman/Downloads/pilinbpq
```

```
mv pilinbpq linbpq
```

```
sudo chmod +x linbpq
```

After the initial configuration of LINBPQ is done in the bpq32.cfg file in the linbpq directory (we'll do that later), the rest of the configuration is done via LINBPQ's web interface, so we need to get the HTML files associated with the web interface before we can do so. Make sure you're still in the linbpq directory then execute the command to download the files.

```
wget http://www.cantab.net/users/john.wiseman/Downloads/HTMLPages.zip
```

Now create the HTML (case-sensitive) directory to hold the files and unzip them

into it, then delete the zip file.

```
mkdir HTML
```

```
unzip HTMLPages.zip -d HTML
```

```
rm HTMLPages.zip
```

```
cd ~
```

Configuring Direwolf:

Ok, now we have Direwolf and LINBPQ installed, let's configure Direwolf with our callsign and make sure all of the beacons are disabled. We don't need Direwolf to send beacons, LINBPQ will handle that task.

At this point if you haven't already, plug in the USB sound card you plan to use as your sound card radio interface. The Pi's on-board audio is not suitable for this task and does not have a microphone input, so a USB sound card must be used. For more information on the Sound Card to Radio Interface, refer to the circuit diagram (see fig. 1 in the appendix.)

We need to find out what name the Linux OS has given to our sound card, to do this enter the following command:

```
arecord -l
```

The console should return a list similar to this:

```
**** List of CAPTURE Hardware Devices ****
card 1: Headset [Logitech USB Headset], device 0: USB Audio [USB Audio]
  Subdevices: 0/1
  Subdevice #0: subdevice #0
```

If you have multiple devices attached, look for the name of the one you plan to use. In the example above, I'm using “Logitech USB Headset”. Now once you've found your device, note the “card” number (1 in this case) and the “device” number (0 in this case). Write these down somewhere handy and save this for later.

Now let's open the direwolf.conf file for editing.

```
nano direwolf.conf
```

This will open the direwolf.conf file in Nano, the default CLI (command line interface) text editor for Raspbian. Use the arrow keys on your keyboard to navigate the document.

First, we will change the ADEVICE setting, this sets the audio device that Direwolf will use to receive and transmit on. This is where the “card” and “device” numbers we just found are used.

Scroll down, using the down arrow key on your keyboard to approximately line number 69. Alternatively, you may enter [CTRL] + [_] (control-key and underscore at the same time) followed by 69, then [RETURN]. [CTRL] + [_] is Nano's keyboard shortcut to “go to line”. It saves a lot of time if you know what line to go to. If any of the following settings mentioned are on a different line than indicated in this how-to, move your cursor to the line that setting is located on. The settings in direwolf.conf are not on a specific line, so the line numbers are subject to change. I'm providing line numbers here merely to make it easier to locate them, but it is not critical so long as the setting is not duplicated elsewhere in the file.

```
[CTRL] + [_]
```

```
69
```

```
[RETURN]
```

The line should look something like this:

```
#ADEVICE plughw:1,0
```

The “#” in front of everything, tells Direwolf that this line should be ignored, because it's a “comment”. Sometimes, programmers leave “comments” in their code to remind themselves about what's going on there, or to leave general notes. Other times, it's just a useful tool to keep the program from “seeing” that bit of code without having to delete it. In our case, we need the program to “see” this line, so we need to remove the “#”. Also, in this case, the plughw:1,0 is already set to what we want it. The 1 and 0 are the sound card's “card” number, and “device” number, respectively. If your card and device numbers are different, you will need to change these to match your sound card.

```
ADEVICE plughw:1,0
```

Next, we will change the callsign for the MYCALL setting, which at the time of this writing is located on line 132.

```
[CTRL] + [_]
```

```
[1] [3] [2]
```

```
[RETURN]
```

So, now we have found the MYCALL setting, we need to add our callsign, in my case, KM4MLS. In your case, use your own callsign. The line should be changed from:

```
MYCALL NOCALL
```

to

```
MYCALL KM4MLS
```

Be sure to use all-caps when adding your callsign here.

The last necessary entry to modify is the GPIO pin that is used for Direwolf to trigger the radio's PTT line. In my case this is GPIO30. You can use any GPIO line you want, just note the number and add it here. I found the entry on line 195.

```
#PTT GPIO 25
```

Should be changed to:

```
PTT GPIO 30
```

This completes the modification of the direwolf.conf file for our purposes. Since we are not dealing with APRS, or the other features Direwolf has built-in, in our station, the rest of the direwolf.conf file can be left as-is. To save and exit enter:

```
[CTRL]+[X]
```

and answer,

```
y
```

followed by,

```
[RETURN]
```

Configuring LINBPQ:

Ok, now that we have Direwolf configured, we need to configure LINBPQ and also set up an auto-run script for it, so that all we have to do once configured is turn on the Pi Packet Station and let it run.

Lucky for us, I have created a nifty little script that will make setting up LINBPQ “easy as pi!” *ahhem*. All we have to do is go get it.

```
cd ~
```

```
https://raw.githubusercontent.com/km4mls/KM4MLS-Raspberry-Pi-Packet-
Node/master/Bash%20Scripts/bpq_cfg_edit.sh
```

Allow that to download then make sure it's executable.

```
sudo chmod +x bpq_cfg_edit.sh
```

Then run it by entering:

```
./bpq_cfg_edit.sh
```

Now follow the on-screen prompts. This is a script I wrote to accompany this how-to that will ask the user for information about the station needed to set-up LINBPQ properly. After entering the info, the script will then download a starter config file and place the relevant information in the proper places within it. It saves a lot of time by entering the information automatically rather than the user having to look through the file and place the info, which is duplicated in many places, manually.

Now that you have the configuration file edited, we need to start up linbpq in order to configure the Mail, and Chat servers. This configuration is absolutely necessary for the Mail and Chat servers to work. This is probably the easiest part of the whole process, though, because LINBPQ has put the ability to edit these configurations into the web server that's built-in. We just need to start LINBPQ then on our PC enter the web address of the Pi. To start linbpq enter:

```
cd linbpq
```

```
./linbpq
```

This will start LINBPQ and it will create the needed Mail and Chat config files.

Now while that's running, browse to [YOUR-PI's-IP-ADDRESS]:8080 in your favorite internet browser. You should see your BPQ32 Node Webpage.

Now click on “Chat Server Pages”. You'll be asked to log-in. Enter the Callsign and password you entered when running the set-up script. Once logged-in click on “Configuration”.

Now in the box next to “Chat APPL No” enter “2” (without quotes). And in the box next to “Streams” enter “10”. Then click on the “Save” button at the bottom of the page. Finally, click on “Node Menu” at the top of the page.

Next, click on “Mail Server Pages” then “Configuration”. And this time enter “1” in the “BBS APPL No” box and “10” in the “Streams” box. Click the “Save” button at the bottom and then click on “Node Menu” at the top of the page. That's it! Your basic packet node is now running!

Anytime you need to manage any of the node's settings, this website is where you will do it. Return here to manage the BBS, user settings, view stats, etc... You can even use this site as your terminal program using the built-in terminal page!

Make sure to restart LINBPQ to put the changes in place. Any time changes are made to LINBPQ's configuration, it must be restarted for them to work.

To restart LINBPQ, go to the terminal it's running in and press [CTRL] + [C] then type ./linbpq or you may reboot the Pi instead. If you are unable to get to the terminal where linbpq is running (ie. it's running in the background) you can use the command “pidof linbpq”, if linbpq is in fact running, the command will return a number (the process id number). To shutdown linbpq using this number, use the command “sudo kill <thenumber>” (ie. If the number was 845, you'd type: sudo kill 845). Afterwards, you can start it back up as usual.

In most cases we'll want to not have to enter any commands in to the Pi to get things started up, which is why we set up Direwolf to auto-start. Now, let's do the same for linbpq.

Start by returning to the home directory.

```
cd
```

Then we'll download a linbpq-start.sh script for auto-starting linbpq.

```
wget https://raw.githubusercontent.com/km4mls/KM4MLS-Raspberry-Pi-Packet-
Node/master/scripts/linbpq-start.sh
```

Now we need to edit one line of that script, to customize it for your system.

```
nano linbpq-start.sh
```

Find the line near the top that looks like this:

```
USERNAME="ENTER YOUR USERNAME HERE"
```

And change it to your login user name, like so (be careful this is case-sensitive):

```
USERNAME=”km4mls”
```

Now save and exit by entering:

```
[CTRL] + [X]
```

followed by,

```
y
```

then,

```
[RETURN]
```

Now make that file executable.

```
sudo chmod +x linbpq-start.sh
```

Now to make LINBPQ auto-start on boot and if it fails, we need to go back to crontab and add entries for this script.

```
crontab -e
```

Now, on two new lines, enter the following lines:

```
@reboot /bin/bash /home/<username>/linbpq-start.sh
```

```
* * * * * /bin/bash /home/<username>/linbpq-start.sh
```

Save and exit:

```
[CTRL] + [X]
```

```
y
```

[RETURN]

That's it! Now when you reboot, Direwolf will start up, followed by LINBPQ. And LINBPQ will start it's Web Server. From there just open a web browser to change any configuration options you need to change in LINBPQ if needed. Don't forget to always reboot the Pi after making any config changes. Congratulations on your new Raspberry Pi AX.25 Packet Radio Node! If you have any questions you can reach me at km4mls@gmail.com Have a great day, hope to see you on the air!

-KM4MLS

Appendix:

Sound Card Interface Circuit:

The sound card interface circuit consists of a USB sound card and an audio isolation, circuit using 600Ω audio isolation transformers, as well as level control potentiometers for TX and RX audio and an opto-isolator for the PTT signal. This keeps the digital circuitry galvanicly isolated from the RF/radio circuitry so as to eliminate any ground-loops that may otherwise have been formed. The schematic diagram for the interface circuit is shown in Fig. 1.

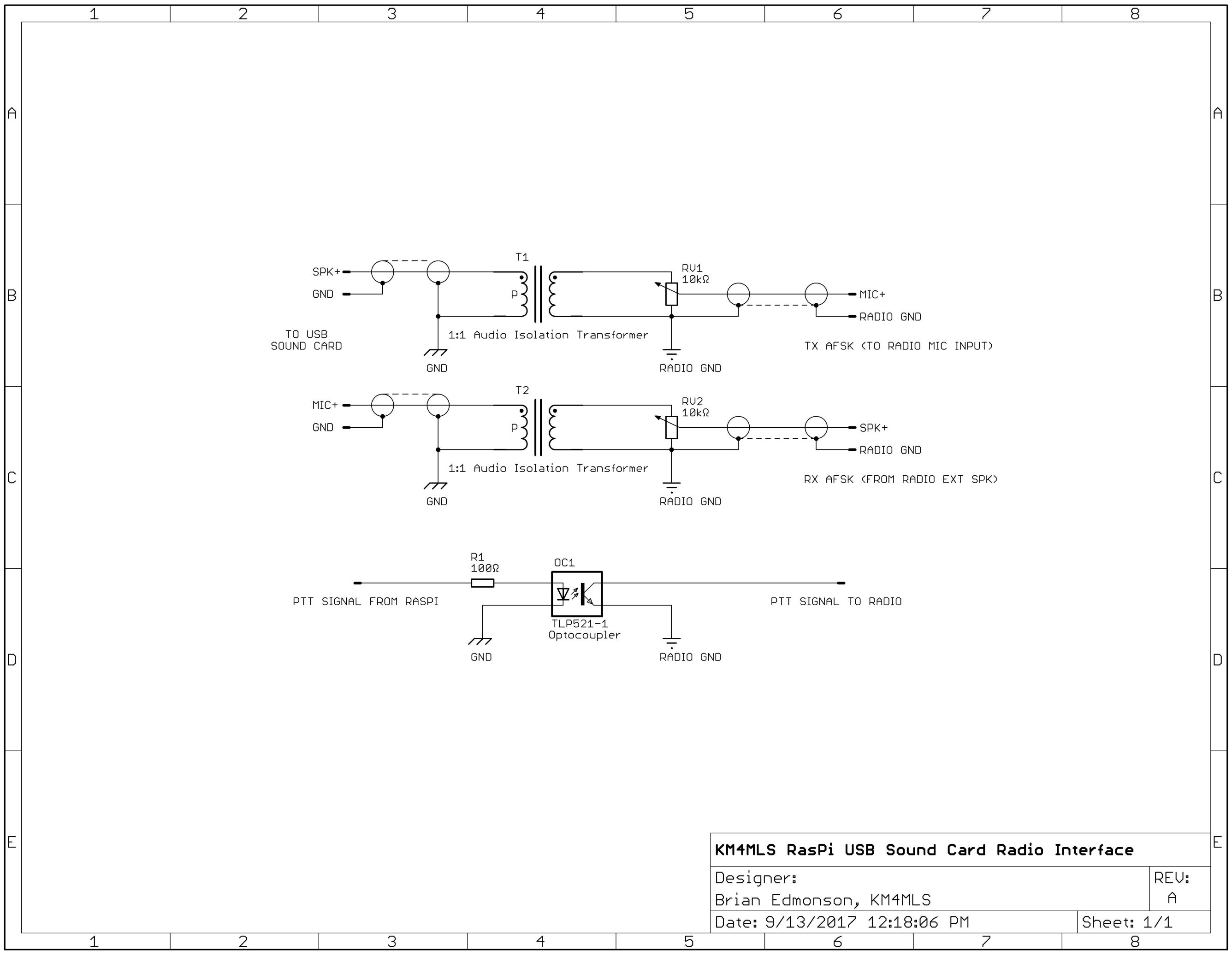


FIG. 1

Sound card radio interface circuit for ground-loop isolation.

For a full-sized drawing see the KM4MLS RasPi USB Sound Card Radio Interface.pdf file in the "docs" directory of the github repository located at https://github.com/km4mls/KM4MLS-Raspberry-Pi-Packet-Node/raw/master/docs/KM4MLS%20RasPi%20USB%20Sound%20Card%20Radio%20Interface.pdf.