

Inexpensive All Mode Remote Operating with a Raspberry Pi, Fldigi/Flrig, WSJT-X, and NoMachine

Harry Bloomberg W3YJ
14 August 2020

How can I remotely operate a station located at a vacation home?

Like many amateur radio operators, my primary residence is on a small suburban lot where putting up a good HF antenna is a challenge. A few years ago I inherited 5 acres of woods and an old Amish farmhouse 85 miles away. I hung some antennas from tall walnut and pine trees on the property and spent weekends and holidays operating from there.

I wanted to get on the air more often and started researching how to access the station at my wooded location from my primary residence. Some of the solutions were quite expensive and complex. But I discovered three key pieces of technology that allowed me to operate CW, digital modes, and SSB remotely and inexpensively.

The first piece was the Raspberry Pi. The Pi is a small single-board computer designed in the UK that was initially intended for educational purposes but is now used in many embedded applications. It runs a distribution of the Open Source Linux operating system named Raspberry Pi OS. Raspberry Pi's are very inexpensive, starting at \$35. I splurged and purchased a Raspberry Pi 4 with 4 GB of RAM as part of a kit containing important accessories for \$99.

The next piece was Fldigi and Flrig, part of the NBEMS suite developed by Dave Freese W1HKJ. Most hams associate NBEMS with emergency and public service operations. It is the standard communications package for many ARES groups. NBEMS also provides many features for recreational operation. Fldigi can function as a contest logger and CW keyer, for example. Flrig allows you to control a transceiver through a USB interface. You can change frequency, adjust power, and control other major parameters on a variety of different transceivers. NBEMS runs on Windows, MacOS and Linux, including the Raspberry Pi. You can also use Flrig to act as rig control for WSJT-X.

The final piece was NoMachine remote operating software. I looked into various ways of logging onto the Pi but the software was either expensive or would not send or receive audio. Then a good friend told me about how NoMachine was used to log securely onto computers at a major government lab. NoMachine is free for personal use and will stream audio both ways to and from the Raspberry Pi. I joined a NoMachine support forum and soon learned how to interface NoMachine to the audio system on the Raspberry Pi. NoMachine makes clients for all major platforms

including Windows, MacOS, iOS and Android that can connect to the NoMachine server on the Pi. NoMachine for all platforms along with installation instructions, documentation, and support is available from <http://www.nomachine.com>.

I knew my remote system was fully operational when I connected to my station from a Starbucks and worked FT8 DX from my iPad with WSJT-X and then made a CW contact with a special events station using Fldigi. I also made several SSB contacts in the PA QSO Party from a hotel room in Michigan.

The advantage of this method of remote operating is that it is very inexpensive. The only additional hardware you will need to purchase is a Raspberry Pi. The drawback is you've got to learn a little Linux systems administration. But you're a ham. That means you embrace technical challenges to save a buck. And you love learning new technology. Most ham clubs have some members who are familiar with Linux who might be able to help you.

Detailed technical notes may be found at <http://www.w1hkj.com/W3YJ/>.

Integrating NoMachine with the Raspberry Pi

Please see Figure 1 for an overall view of how the system works. At the center of everything is a Raspberry Pi Model 4. I initially made a prototype system with a Pi Model 3B+. It worked OK but on occasion the IC-7200 would lock up and lose audio connection to the Pi. This has never happened with the Pi 4. I suspect the Pi 4 works better because it is much more powerful than the Pi 3.

The IC-7200 is particularly well suited for connecting to a computer because both audio and rig control commands are carried over the USB cable. I have also operated an Elecraft KX3 remotely with the Pi but an external USB soundcard and cables going to the mic and headphone jacks are required.

Audio is processed on the Pi by a package named PulseAudio which acts as an audio server, sending and receiving streams of audio much as a webserver sends and receives Internet data. NoMachine interfaces to PulseAudio. A common commercial use for NoMachine is logging onto remote systems and streaming audio remotely to remote mics and speakers for VoIP communications. Hams of course want to hear audio coming from the transceiver and send audio to a mic input on the radio. Some Linux shell commands are run once after logging into the Pi to redirect these audio streams.

A detailed explanation of how to interface NoMachine with PulseAudio is beyond the scope of an article for QST. Please see <http://www.w1hkj.com/W3YJ/> for detailed tech notes. But, only a handful of Linux shell commands are required. I have written a Perl program that probes your Raspberry Pi for soundcards and writes the shell commands for you. This script is named `write_script.pl` and is also available for download from <http://www.w1hkj.com/W3YJ/>. PulseAudio refers to

audio inputs like mics as “sources” and output devices like speakers as “sinks”. Figure 2 shows sinks and sources when my IC-7200 is connected to the Pi and how the Linux shell commands cause sinks and sources to be reconfigured to interface the IC-7200 to NoMachine. The tech notes explain how to obtain the names of sources and sinks for your USB soundcard.

Adding Fldigi and Flrig

Please see Figure 3.

There are two ways to install Fldigi and Flrig. The easiest way is to use the Pi’s Add/Remove software utility. Unfortunately, this will install a version of the software from a Raspberry Pi repository that is quite old. You will not have the latest features or bug fixes. An alternative is to build the software from source code. Excellent instructions may be found at http://www.w1hkj.com/doku/doku.php?id=howto:building_on_debian-ubuntu-mint. Yes, you will need to open a terminal and enter some Linux shell commands, but we are amateur radio operators and when you successfully build from source you will feel a sense of accomplishment as you perform the software equivalent of soldering.

Now that Fldigi and Flrig have been installed, you must first configure Flrig. Go into the Configure menu and select your transceiver and USB communications parameters. You’ll know Flrig is configured correctly when you see your transceiver’s frequency correctly displayed in Flrig.

Next, configure Fldigi to use Flrig for Rig Control. Go to the Configure->Rig Control menu and click on the checkbox to use Flrig. Again, you’ll know Fldigi is properly configured when you see the correct frequency displayed in Fldigi.

You are now ready to tell Fldigi to use PulseAudio for audio and to connect Fldigi to your USB soundcard devices. In Fldigi, go to the Configure->Soundcard menu. Select the Devices tab and click the checkbox for PulseAudio. Leave the box for Server String empty because you’re accessing PulseAudio on this computer.

To connect Fldigi to the correct USB soundcard devices, open the pavucontrol utility. You can install pavucontrol using the Raspberry Pi’s Add/Remove Software utility. Go to the Recording tab. You’ll see one audio stream with the Fldigi icon next to it. This is the volume control for audio being directed into Fldigi. Select your USB soundcard device from the drop down. Then adjust your audio level and make sure you see signals on the waterfall. A good rule of thumb for adjusting input level is the waterfall is mostly blue with signals yellow. If you see signals that are red, your level is too high and you won’t get the full benefit of your soundcard’s dynamic range.

The process is similar for Fldigi’s transmit audio. Go to the Playback tab and find the stream that has a small Fldigi icon. Change the dropdown to your USB audio

device. Adjust the output level so that your rig's ALC just moves a little bit. Note that if you click on the Flrig SWR meter while transmitting, you'll see ALC displayed.

My favorite way to have Fldigi generate CW and key your transceiver is by installing a K1EL Winkeyer. I have had great success with both the WKUSB-SMT and WKmini. The K1EL website has full instructions at <https://www.hamcrafters2.com/> as does the W1HKJ NBEMS website at <http://www.w1hkj.com>. Fldigi can also control a nanoKeyer at <https://nanokeyer.wordpress.com/>.

I made a few CW contacts in the Worked All Europe (WAE) contest while staying in a hotel room on a vacation. To configure Fldigi as a contest logger, go to Configure->Contest/Logging->Contest. The Contest drop-down allows you to set up Fldigi for a variety of popular contests including ARRL Field Day. Fldigi can also check for dupes and automatically generate serial numbers. Macro keys can be edited to automate contest exchanges and logging. Logs can be exported in a variety of formats including ADIF, CSV and Cabrillo. Fldigi has an excellent CW modem to help you copy CW.

Remotely Logging Onto Your Raspberry Pi

Perform the following steps to log onto your Raspberry Pi.

1. Right click on the NoMachine icon on the toolbar at the top of your Pi's screen. Go to the Show the service status menu. Note the nx: address that has a global IP address, that is an IP address that is not on you local network. Also write down the port number. This is the number that is separated from the IP address by a colon.
2. Enter this nx: address into NoMachine on your computer or tablet device.

You can now connect to your Raspberry Pi from your computer or tablet. To ease this in the future, you may want to subscribe to a service like NoIP so that you can connect to your Pi by the machine's name instead of IP address which could be changed by your Internet provider.

Configuring NoMachine Audio

When you log onto your Raspberry Pi using NoMachine, run the shell script described in the tech notes to connect PulseAudio to NoMachine. You must do this while logged in using NoMachine. If you try to run the script while logged directly onto Pi, the PulseAudio sources and sinks for NoMachine won't be available and the script will fail.

To adjust the volume of the audio from your radio that is being sent to your device, open pavucontrol, go to the Recording tab, and adjust the level of the NoMachine slider. You may need to change the value of the drop down for this slider to "Monitor of Null Output".

After you run the script, you will have to go into NoMachine's settings, then go to the audio icon, and click on the left speaker icon to mute and then unmute. This is because you have rerouted the audio streams from PulseAudio to NoMachine. Muting and then unmuting will cause NoMachine to "hear" these rerouted PulseAudio streams. There is also another audio level control for adjusting the volume.

Before logging out when you have finished operating, enter the "pulseaudio -k" command to reset PulseAudio. You will need to completely log out of your Raspberry Pi and then log back on before you can run the shell script to redirect audio again.

Configuring for SSB Operation

Using NoMachine for remote operating works most simply if you want to operate CW or digital modes. But SSB operation is possible with a USB mic connected to your computer or using the mic built into your computer. You must change the audio settings on your NoMachine client on your computer to unmute your mic. You can also adjust levels in your NoMachine microphone settings. In addition you'll need to adjust levels on your computer. Finally, you must go the Playback tab on the pavucontrol program and change the output for the NoMachine audio stream to your radio's USB soundcard device. Note that you cannot do this unless your mic is enabled in NoMachine. Please see Figure 4.

Operating WSJT-X with NoMachine

WSJT-X has a built-in radio interface to FLrig that will simplify radio configuration. In WSJT-X go to File->Settings->Radio and select "FLrig FLrig" as the radio and CAT as the PTT control. This allows FLrig to control WSJT-X and saves you having to configure your radio directly through WSJT-X.

For WSJT-X audio settings, go to the Audio tab and select your USB soundcard.

Setting audio levels with WSJT-X and pavucontrol is a little tricky. A quirk of WSJT-X is that the audio input from PulseAudio is by default always set to 100%. You will need to change this every time you start WSJT-X. To reduce the level, go to the Recording tab in pavucontrol and find the stream with the WSJT-X icon labeled QtPulseAudio. Use this control to reduce the input audio level. I find a setting around 50% works for me.

You can only adjust the audio level for transmitting while actually transmitting. To set the audio level, first reduce the power level on your transmitter in FLrig to a very low value to minimize QRM. Press the WSJT-X Tune button. While transmitting, go to pavucontrol on the Playback tab. Find the stream with the WSJT-X icon labeled QtPulseAudio. Adjust this level until you see a small amount of ALC on your

transmitter. Fortunately, this needs to be adjusted once and will stay the same every time you execute WSJT-X.

Remote Station Considerations

Should something go wrong with your remote station, you cannot be there to pull the kill switch on your radio or cycle power to your Raspberry Pi. I have installed wi-fi controlled AC power sockets for both my IC-7200 and Pi. This allows me to cycle power to both devices using an app on my cellphone. To increase the reliability of my home network, the AC power for both my wireless router and the Raspberry Pi is connected to a UPS. I also have the Raspberry Pi directly plugged into the wireless router using an Ethernet cable.

Putting It All Together

Are there a lot of moving parts in this method for remote operation? Absolutely! You are almost guaranteed to have total failure if you install NoMachine, Fldigi, Fldigi, and run your PulseAudio scripts all at the same time. It is very important to add one feature at a time, make sure each new feature works, and then move on to the next step. You should add functions in the following order:

1. Build Fldigi and Flrig on your Pi from source, or install from the repository.
2. Connect to your rig using Flrig and make sure all the major controls in Flrig work with your rig.
3. Set up Fldigi. First, make sure it works with Flrig. Next, make sure you can send and transmit audio using PulseAudio. Adjust audio levels with pavucontrol.
4. Connect your K1EL Winkeyer and make sure you can key your radio with Fldigi in CW mode.
5. Install WSJT-X and use Flrig as rig control. Adjust audio levels with pavucontrol.
6. At this point you can make digital and CW contacts using your Raspberry Pi. Spend some time the air to become familiar with operating using your Pi.
7. Run the write_script.pl Perl program to identify your USB audio device and to write the shell commands to reroute audio streams.
8. Install NoMachine on your Pi and your computer or tablet.
9. Log onto your Pi using NoMachine from your local network.
10. Run the scripts written by write_script.pl to redirect audio. Enable your speaker and mic in NoMachine.

You are now ready to connect to your remote system! Try it out in your shack before you head to Starbucks.

Emcomm applications?

Installing an HF antenna at a mobile or portable command post can be problematic due to the nature of HF antennas, especially for the 80M band where a full-size antenna is approximately 135 feet long. Yes, you can use a portable or mobile antenna, but even the best of these will not work as well as a full-sized antenna. You

might consider connecting to a remote HF station during a deployment. Think back to your most recent Field Day operation and remember how long it took to set up your HF station and how much planning was required. Think of all the little things that can go wrong like a bad piece of coax or a poorly soldered rf connector. Now imagine setting up an HF station under the duress of a drill or actual emergency deployment. You may be better off trying to connect to a well-maintained permanent station if a network is available rather than hastily setting up an HF station with a poor-performing temporary antenna at the deployment site.

HF stations at EOCs can be pose difficulties also. EOCs are often located in urban areas with lots of noise. It may be difficult to properly install an HF antenna on the roof. One of the most effective HF antennas is the classic 135 foot dipole fed with twin-lead. These might difficult to install in your EOC because the twin-lead would be compromised by running it through a metal building. Because of these reasons, a remotely-operated HF station could give your EOC better HF performance.

Some of you may say that using a remote HF station adds unnecessary complexity to a deployment or to the EOC. The same was said years ago about NBEMS, Winlink, and mesh networking. They're now standard parts of our emcomm toolbox. "But it requires use of the Internet!" you might say. It's been my experience that most EMAs cannot manage an emergency without Internet connectivity, so there's a very high probability the Internet will be made available to the EMA and you'll be able to connect to your remote HF station.

Future Considerations

With a price of \$35-55, the Raspberry Pi is easily affordable by almost any amateur radio operator. It also has many input/output ports that can be adopted for controlling equipment in a ham shack. The Linux operating system is an ideal platform for amateur radio operators because it is Open Source and therefore suitable for tinkering and experimenting. Vendors of commercial amateur radio equipment should consider porting their rig control and radio programming software to Linux so their software can run on a Raspberry Pi.

Why write an article for QEX? One reason is fame and fortune. Another is to ask for help from other hams. I worked on this alone and your collective wisdom is welcomed. Please review my work! Is there another way to do this? I love NoMachine but I'd rather be using software that is Open Source. Is there something out there that works as well as NoMachine and supports so many different platforms? And, what can be done to simplify all this so that a ham with minimal Linux admin skills can make this work?

Acknowledgements

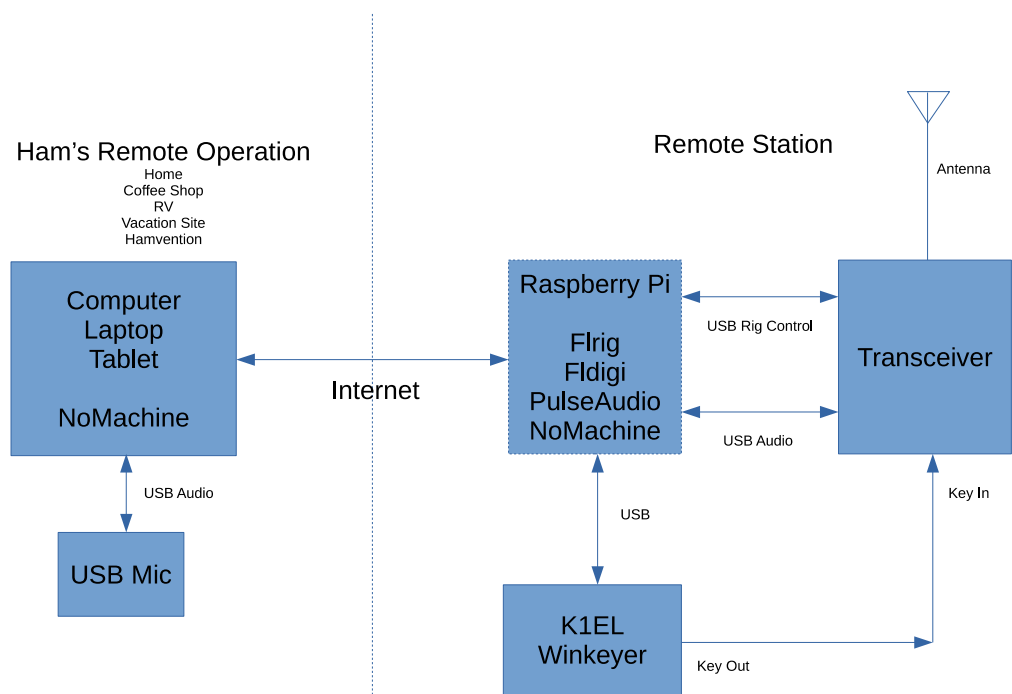
I would like to thank the following:

- Juan Manfredi NA0B of Panther Amateur Radio Club of the University of Pittsburgh.

- NoMachine Tech Support for helping me to configure the interface with PulseAudio.
- Dave Freese W1HKJ, the leader of the NBEMS team and everybody else working on the project.



The author's iPad in a Starbucks working FT8 DX remotely using NoMachine. Yes, this photo was taken before the coronavirus pandemic. During the Spring of 2020 Pittsburgh was under a stay-at-home order for almost two months. By logging onto my Raspberry Pi at my remote station, I was able to remain on the air throughout. It was a blessing. I operated so much that I now hold DXCC and WAS.



Remote Amateur Radio Operation System Architecture

W3YJ 24 July 2019

Figure 1

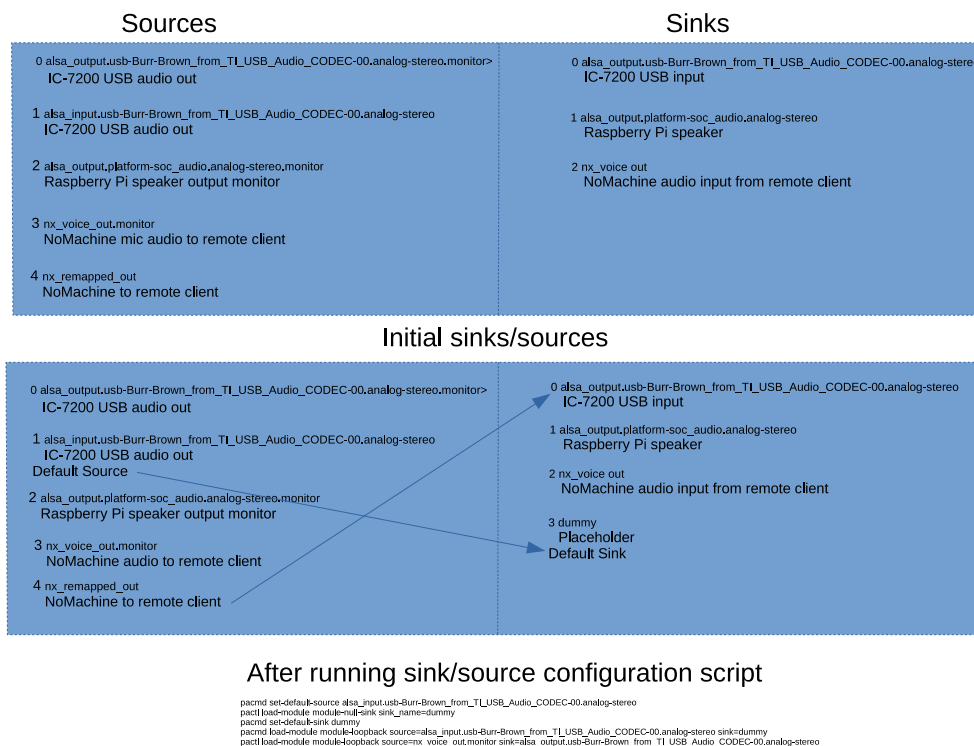
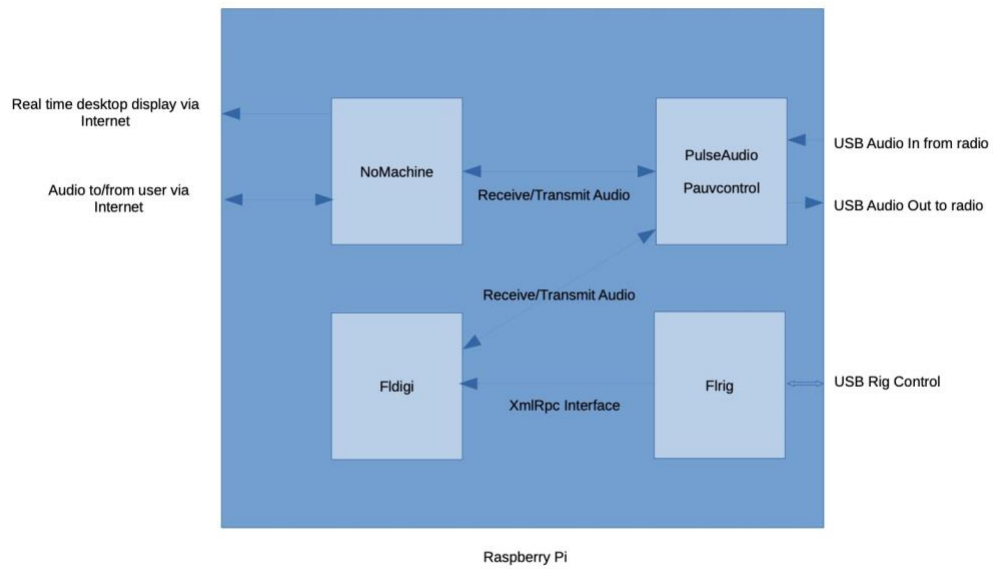


Figure 2



PulseAudio sources/destinations/levels controlled by Pauvcontrol

Raspberry Pi Software Architecture

W3YJ 24 July 2018

Figure 3

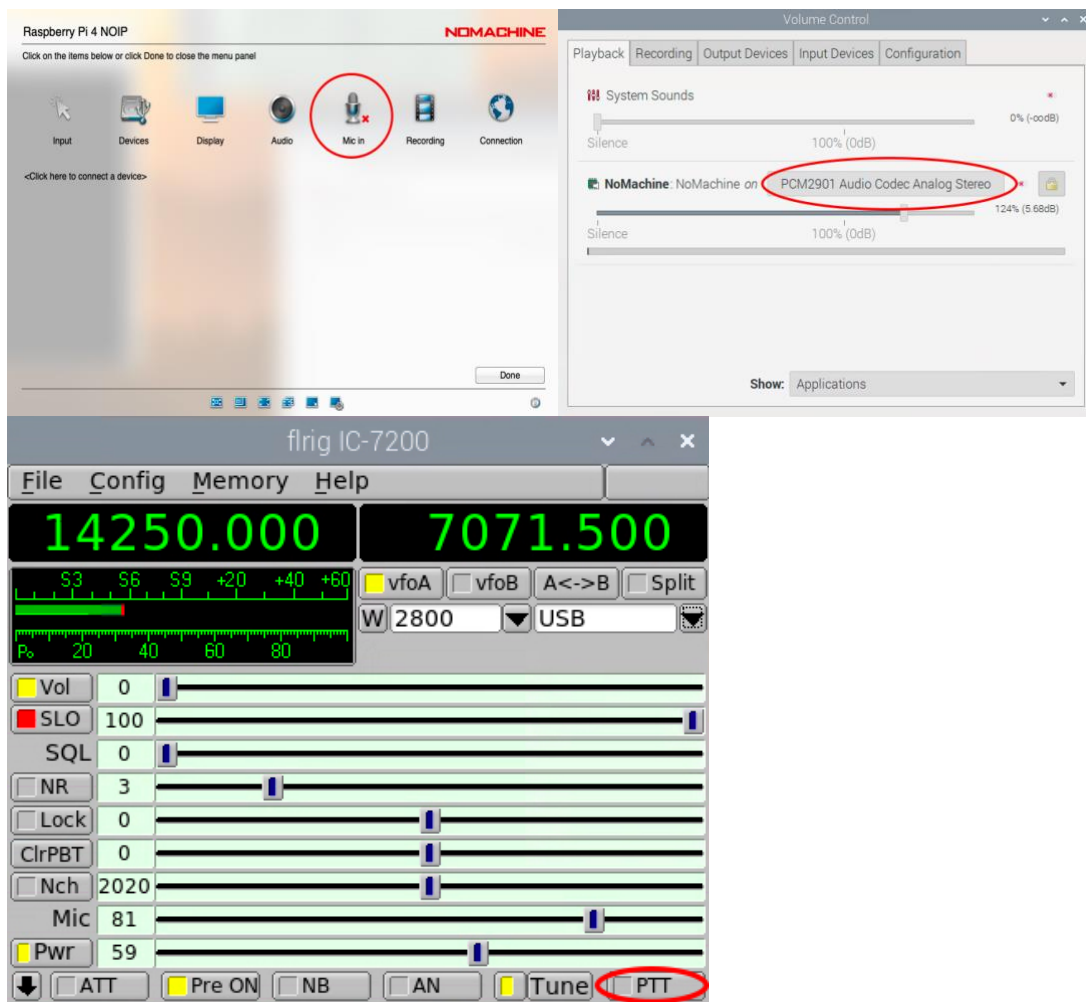


Figure 4 – Configuring for SSB