Tech Notes for Remote Operation Using NoMachine and Raspberry Pi

Harry Bloomberg W3YJ
26 March 2024

March 26, 2024 Update

I have upgraded to a Raspberry Pi 5 and Bookworm. There are so many configuration changes that I wrote a separate document you can download at: http://www.w1hkj.com/W3YJ/Configuring%20Remote%20Operation%20with%20NoMachine%20on%20Bookworm%20on%20Raspberry%20PI.pdf

May 14, 2023 Update

NoMachine has changed the way that UPnP is enabled on a server starting with Version 8. This support page explains how to do this.
https://kb.nomachine.com/AR04S01122

July 5, 2020 Update

The hardest part of using NoMachine for remote operation is writing the Linux shell script that is necessary to redirect PulseAudio streams to NoMachine. I have written a Perl program named write_script.pl to write this script for you. It is available for download as write_script.tar.gz at http://www.w1hkj.com/W3YJ/ . Once this file is downloaded, you can extract the Perl script as follows:

tar -xzvf write_script.tar.gz

The write_script.pl Perl program does the following:

- Interrogates your Raspberry Pi for audio sinks and sources. You will want to use a sink and source associated with the USB soundcard connected to your radio, or your radio's internal USB soundcard. To simplify matters, I suggest you temporarily remove all USB soundcard devices from your Raspberry Pi except the one your radio uses.
- Provides you with a list of sinks and sources.
- Asks you which sink and source you'd like to redirect to NoMachine.
- Writes a shell script for you that will perform the audio redirection to NoMachine. You will need to log into your Raspberry Pi using NoMachine to run this script.

Run the Perl script as follows and follow its directions:

perl write_script.pl

May 10 2020 Update

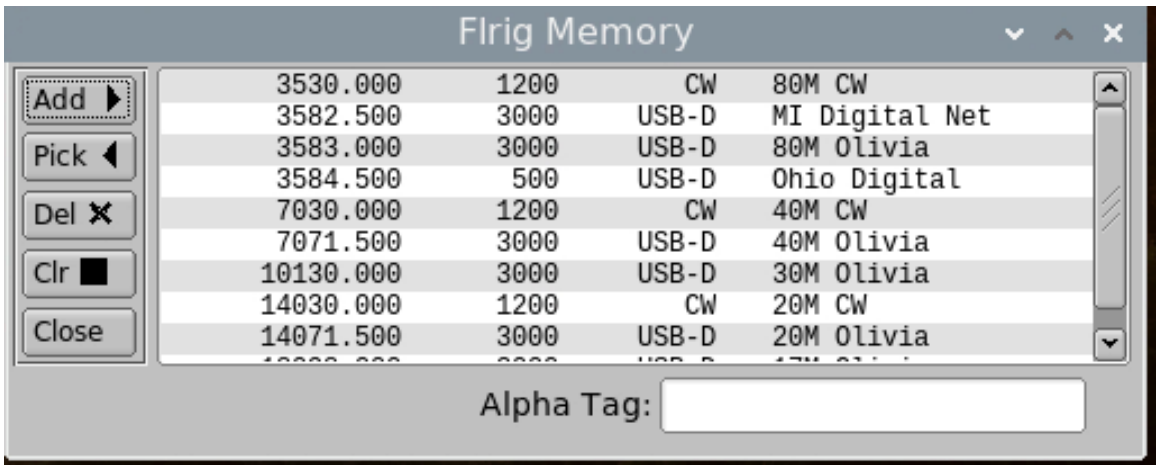
I am now operating an Icom IC-7300 remotely. It is a much better receiver than my old IC-7200 with superior filtering and noise reduction. With the IC-7300, noise reduction can be dialed in from Flrig. I find a setting of 3-4 works best with CW. When the NR is enabled, CW signals just pop out of the passband and are easier to hear. With the IC-7200, receiver noise increases greatly as you narrow the receiver's passband. In the IC-7300 this does not happen nearly as much, especially if the NR is enabled.

I wish it would be possible to view the IC-7300's excellent waterfall display remotely with Open Source software. The ability to make this happen is beyond my programming abilities. I think this would make a very useful project for anybody looking for a project. Considering the popularity of the IC-7300, I think it would be very useful for many hams.

I have solved the mystery of why you must disable and then reenable the audio in NoMachine after running the Linux script to redirect audio from you receiver to NoMachine to be streamed back to you. By default, NoMachine is looking for audio from you remote computer's speaker. Disabling and then reenabling NoMachine audio causes your remote NoMachine client to use your Pi's new audio streaming configuration instead.

It is important to use remotely controlled wi-fi power sockets on your radio and Raspberry Pi and to have your Raspberry Pi hard-wired to your home network's router with a cable. A few times I've not been able to turn on my radio remotely because the wi-fi power sockets were not showing up in the iPhone app I use to control them. I logged onto the Raspberry Pi, connected to the router through its web app, and forced the router to reboot. Had the Pi been connected with wi-fi like the power sockets, it's likely that I would not have been able to log onto it either. A few times, I've not been able to log onto the Pi at all for unknown reasons. Cycling the power remotely forced a reboot that corrected this situation. Do this only as a last resort because of the possibility of corrupting your Pi's SD card.

Flrig has a built-in utility to memorize frequencies and modes. This simplifies jumping around to your favorite frequency. You can get to it from the Memory->Manage menu.



I am writing this in the middle of the coronavirus pandemic. I've not been able to physically get to my station for two months due to my state's stay-at-home order which I take seriously and respect. The remote setup has allowed me to stay on the air and make many contacts. Otherwise I would have been burned out from watching too many reruns of Star Trek.

Introduction

NoMachine is software that allows you connect between computers running a variety of operating systems. You can run NoMachine as a server and accept incoming connections, or as a client to log onto remote systems. NoMachine is available for free personal use at http://www.nomachine.com . NoMachine connections use the NX protocol, which is based upon SSL, so remote sessions are encrypted and secure. NoMachine can be installed on all major operating systems, including Windows, MacOS, Linux, and Android.

The NoMachine server for Raspbian, the Linux distribution that runs on the Raspberry Pi, has the ability to interface with a Linux PulseAudio streaming audio server. This allows us to use a NoMachine server to stream audio coming into and out of a Raspberry Pi. We can not only use NoMachine to listen to audio from an amateur radio transceiver, we can send audio into the transceiver for voice contacts.

NoMachine's streaming audio capabilities are primarily intended for logging onto a remote system for two-way video or VoIP conferencing. This means that by default, audio on the remote system's output device, most likely a speaker, is streamed back to the speakers or headphones on the user's computer. Audio from the mic of the user's computer is streamed to the mic of the remote system.

In amateur radio use, we instead want audio from the user's NoMachine client sent to the mic input of a radio connected to the NoMachine server on the Raspberry Pi. This might be audio going into a radio's mic input from a USB soundcard, or more commonly in newer radios, to the radio's built-in USB audio interface. Inversely, we want output audio from the radio connected to the Raspberry Pi to be streamed to headphones or a speaker on the user's computer. This audio might be sampled by a USB soundcard connected to the Raspberry Pi or in more modern radios, this might be output by a built-in USB audio interface. We use Linux shell commands to reconfigure NoMachine's interface to PulseAudio on the Raspberry PI to redirect audio streams to and from the NoMachine program running on the user's computer.

Please see Figure 1 for an overview of the entire remote operation system architecture. Figure 2 shows details of software inside the Raspberry Pi.
Flrig is used for control of the remote radio. Actual operating is done with the Fldigi and WSJT-X programs for digital modes and CW. For voice operations, you can use the mic to the client computer for audio to be streamed to the Raspberry Pi.

Install PulseAudio Before Installing NoMachine

According to NoMachine, you should install PulseAudio on your Raspberry Pi and other ARM systems before you install NoMachine, otherwise you won't be able to stream audio. NoMachine does provide a workaround, but it's simpler to install the software in the preferred order.
https://www.nomachine.com/AR07N00896

Identifying PulseAudio Sources and Sinks

You will need to write a short Linux shell script on your Raspberry Pi to connect your radio's audio to NoMachine so that audio can then be streamed to and from your remote device. To do this you will need to find names of your radio's input and output devices.

PulseAudio describes audio streams as either *sources* or *sinks*. A source is an audio input like a mic that generates sound. A sink is an audio output device like a speaker. The first step is to identify the PulseAudio source and sink for the USB soundcard that is connected to your radio.

The following command will list all your system's sources and save them to a file named sources.txt:

```
pacmd list-sources | grep -e 'index:' -e device.string -e 'name:' > sources.txt
```

This is the command to list your system's sinks and save them to a file named sinks.txt.

```
pacmd list-sinks | grep -e 'index:' -e device.string -e 'name:' > sinks.txt
```

On my Raspberry Pi connected to the IC-7200 USB soundcard, my sources are as follows:

```
  index: 0
        name: <alsa_output.usb-Burr-Brown_from_TI_USB_Audio_CODEC-00.analog-stereo.monitor>
              device.string = "1"
 * index: 1
        name: <alsa_input.usb-Burr-Brown_from_TI_USB_Audio_CODEC-00.analog-stereo>
              device.string = "front:1"
  index: 2
        name: <alsa_output.platform-soc_audio.analog-mono.monitor>
              device.string = "0"
```

Based up this, we see that the input USB device to the IC-7200 is named "alsa_input.usb-Burr-Brown_from_TI_USB_Audio_CODEC-00.analog-stereo".

Here is the list of sinks on my Raspberry Pi:

```
* index: 0
        name: <alsa_output.usb-Burr-Brown_from_TI_USB_Audio_CODEC-00.analog-stereo>
              device.string = "front:1"
  index: 1
        name: <alsa_output.platform-soc_audio.analog-mono>
              device.string = "hw:0"
```

We can see that the output USB device to the IC-7200 is named "alsa_output.usb-Burr-Brown_from_TI_USB_Audio_CODEC-00.analog-stereo".

Writing a Linux Shell Script to Redirect PulseAudio Streams

Now that you've identified the audio sources and sinks associated with your radio, you are now ready to write a short Linux shell script to redirect these streams to and from NoMachine.

First, we will configure the output from your radio's USB soundcard as the default source on your Raspberry Pi:

```
pacmd set-default-source alsa_input.usb-Burr-Brown_from_TI_USB_Audio_CODEC-00.analog-stereo
```

We will now redirect the radio's output to NoMachine to be streamed back to you on your remote system. NoMachine will always send audio received by the default sink on the Raspberry Pi to your remote computer, so we must find a way for your radio's output audio to be sent to the default sink. Remember, a typical use case for NoMachine is to connect to a voice conferencing system where the audio output of the conferencing system is streamed back to your computer. Instead of the output of the conferencing system, we want the output audio of your radio streamed back to you.

We can accomplish this by creating a placeholder sink named “dummy” and making this our default sink. We will then connect the output of our USB soundcard to this placeholder sink. NoMachine will latch onto the dummy sink because we’ve made it the default sink and the audio will be streamed by NoMachine back to your remote system.

```
pactl load-module module-null-sink sink_name=dummy
pacmd set-default-sink dummy
pacmd load-module module-loopback source=alsa_input.usb-Burr-
Brown_from_TI_USB_Audio_CODEC-00.analog-stereo sink=dummy
```

Our final step will be to connect your remote computer’s mic to the USB input of your radio. NoMachine sends audio to the Raspberry Pi using a source named “nx_voice_out”. Our Linux shell command to redirect this to the radio’s USB input is:

```
pactl load-module module-loopback source=nx_voice_out.monitor sink=alsa_output.usb-Burr-
Brown_from_TI_USB_Audio_CODEC-00.analog-stereo
```

Putting it all together, here is my shell script which I call audio_icom.sh. I run this script whenever I log onto my Raspberry Pi remotely for the first time after powering up my radio. It is important that your run this only when logging on remotely because the NoMachine audio streams are active only when you are actually connecting using NoMachine.

```
pacmd set-default-source alsa_input.usb-Burr-Brown_from_TI_USB_Audio_CODEC-00.analog-stereo
pactl load-module module-null-sink sink_name=dummy
pacmd set-default-sink dummy
pacmd load-module module-loopback source=alsa_input.usb-Burr-
Brown_from_TI_USB_Audio_CODEC-00.analog-stereo sink=dummy
pactl load-module module-loopback source=nx_voice_out.monitor sink=alsa_output.usb-Burr-
Brown_from_TI_USB_Audio_CODEC-00.analog-stereo
```

If you ever wish to completely reset the PulseAudio settings, perform the following Linux shell command:

```
pulseaudio -k
```

Log out of your Raspberry Pi, and then log in again. You can now run your script to redirect PulseAudio audio streams to NoMachine.

Connecting to a NoMachine server

An excellent guide to connecting using NoMachine may be found at:
https://www.nomachine.com/getting-started-with-nomachine

To determine the IP address and port on your Raspberry Pi for your NoMachine connection, run the following commands:

sudo /etc/NX/nxserver –upnpmap
sudo /etc/NX/nxserver –upnpstatus

You will see output like the following:
Local IP 192.168.0.116
Gateway IP 192.168.0.1
External IP 173.89.234.197
NX port 4000 mapped to: 173.89.234.xxx:24560

Use the local IP address when you're on your local network, the NX address with the external PI address and port mapping, when you're outside your network.

You can also obtain this info by right-clicking on the NoMachine icon on the Raspberry Pi's task bar and selecting the "Show the server status" menu.

Note that it might take a few minutes after starting the NoMachine server for the External IP and port mapping to appear.

If you are behind a firewall, please look at the following NoMachine Knowledge Base article:
https://www.nomachine.com/AR11L00827

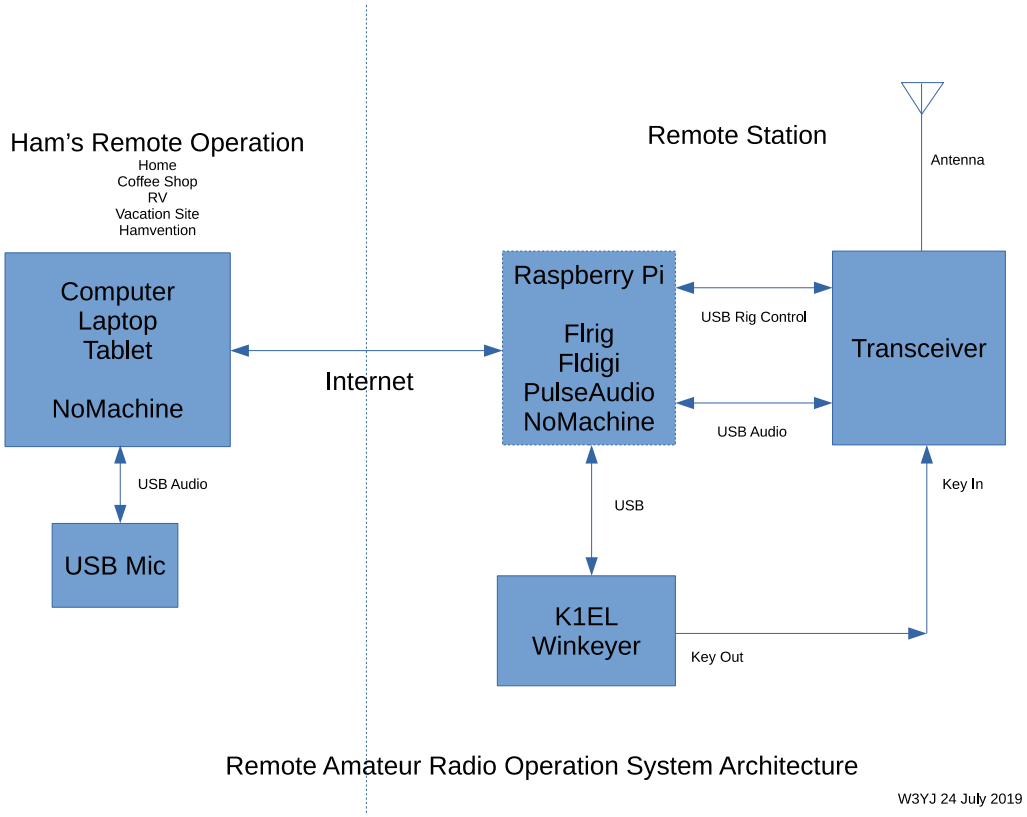
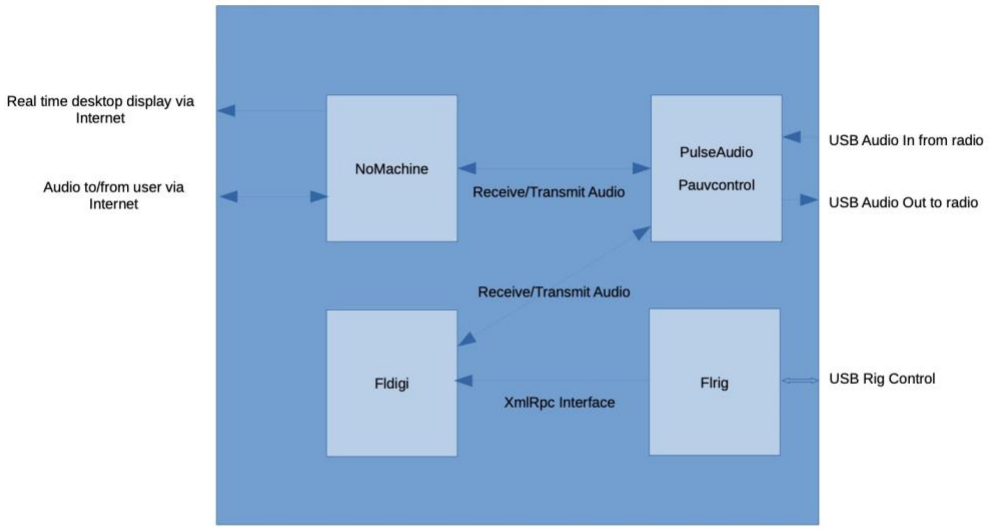

Ham's Remote Operation
Home
Coffee Shop
RV
Vacation Site
Hamvention
Remote Station
Antenna
Computer
Laptop
Tablet
NoMachine
Internet
Raspberry Pi
Flrig
Fldigi
PulseAudio
NoMachine
USB Rig Control
USB Audio
Transceiver
USB Audio
USB Mic
USB
Key In
K1EL
Winkeyer
Key Out
Remote Amateur Radio Operation System Architecture
W3YJ 24 July 2019

Figure 1

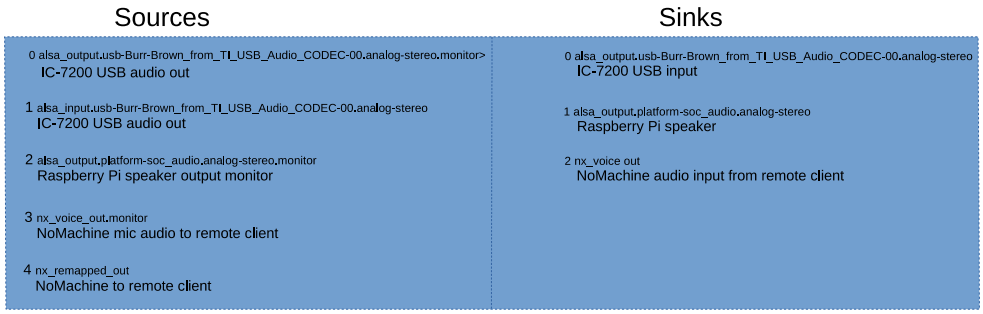


PulseAudio sources/destinations/levels controlled by Pauvcontrol

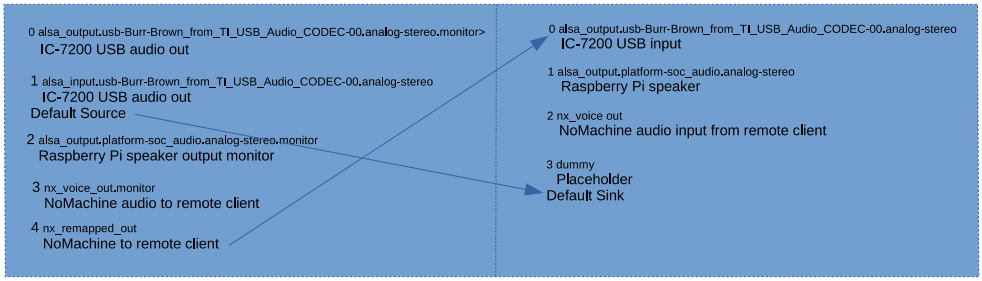
Raspberry Pi Software Architecture

W3YJ 24 July 2018

Figure 2

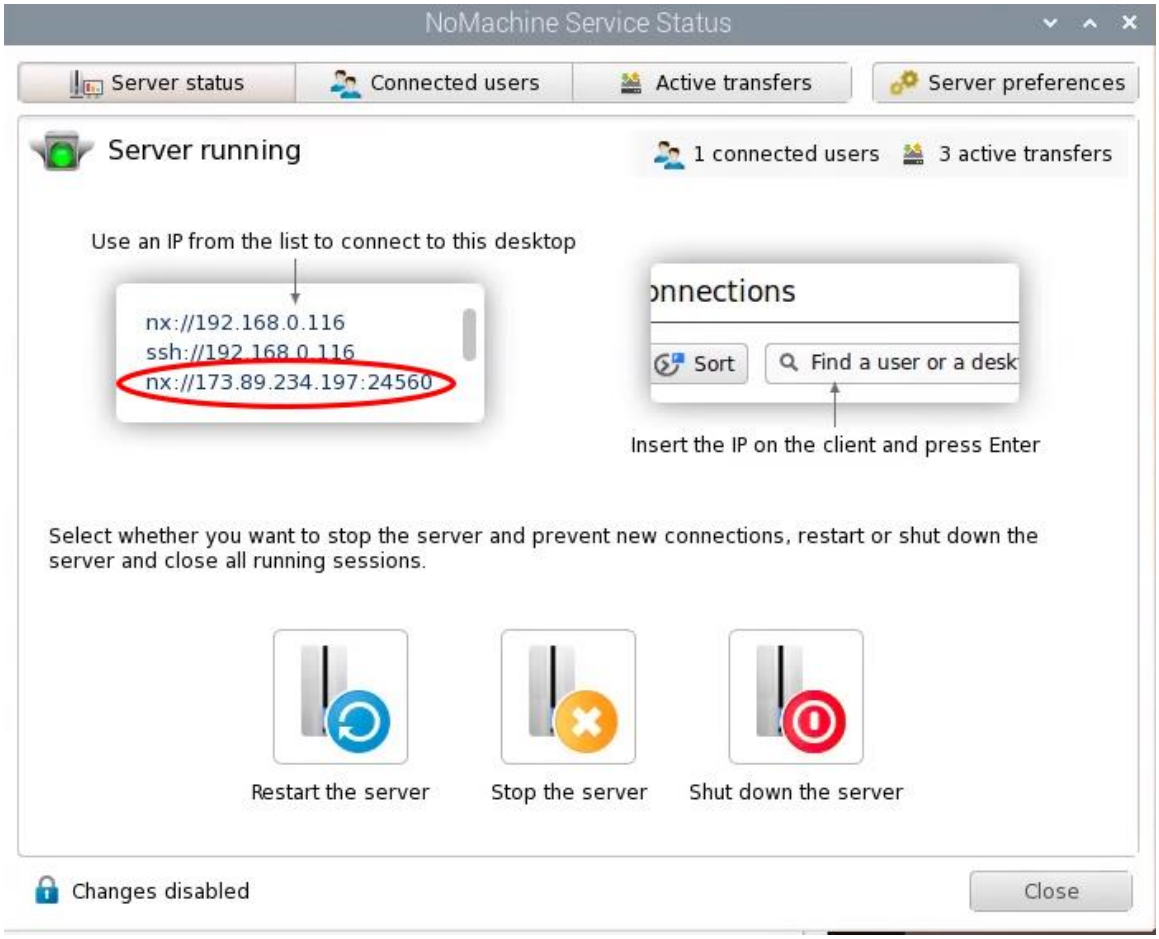


Initial sinks/sources



After running sink/source configuration script

```
pacmd set-default-source alsa_input.usb-Burr-Brown_from_TI_USB_Audio_CODEC-00.analog-stereo
pactl load-module module-null-sink sink_name=dummy
pacmd set-default-sink dummy
pacmd load-module module-loopback source=alsa_input.usb-Burr-Brown_from_TI_USB_Audio_CODEC-00.analog-stereo sink=dummy
pactl load-module module-loopback source=nx_voice_out.monitor sink=alsa_output.usb-Burr-Brown_from_TI_USB_Audio_CODEC-00.analog-stereo
```



NoMachine status showing global IP address and port number