Getting Started with Radiohat Digital Modes

Assuming you've got everything in the "Getting Started with Radiohat" note working, you can try experimenting with digital modes using the rest of the proof of concept software packages you installed.

Of course, this is nothing like practical software - it's just intended to exercise the hardware! It involves a lot of manual operations and attempts at automating it can be fragile. Despite that, it can be operated and I do operate it for all my HF contacts at the moment to gather experience that will allow me to write a complete software package.

To use digital modes, we will use the Jack Audio Connection kit to pipe audio between Gnuradio, the radio hat audio driver, and digital mode applications.

Unfortunately, despite its appearance in the Audio menu, WSJTX does NOT seem to support operation with Jack! To make it work we will use Pulse audio as an intermediary and it will create Jack devices.

Raspbian's Pulse audio is a badly documented system full of surprising ad-hoc quirks that are frequently added and removed by Raspbian updates. As a result, it rarely behaves in a predictable fashion for more than a few months at a time. "Invisible" patches are put in that make ALSA and other audio devices disappear and reappear seemingly randomly. This is very frustrating and annoying, but it's the only game in town as all major Linux distributions have now adopted it as the top level of their audio systems.

Here's what we have to work with:

1. We load "generic" audio drivers provided with the Rasbian at boot time by using a DeviceTree system overlay. This is triggered by the line "dtoverlay=genericstereoaudiocodec" we put into /boot/config.txt. It loads an open source overlay of that name we copied to

the boot volume.

That overlay describes a full duplex 32 bit 48khz I2S alsa audio device by referencing drivers already built into Raspbian. After they are loaded, the hardware is present yet not prepared to respond to the I2S DMA requests. The hardware is waiting patiently for the first signals.

2. When we run code in the experimental control program called “pitrans” it initializes the hardware and sets things in motion. The audio card hardware then needs no further attention to perform as a sound card for devices currently named “GenericStereoAudioCodec” (the name will be changed soon). When invoked with the option “-i”, pitrans can initialize the audio system and then immediately exit. This is useful in shell scripts.

   Later on, we will use other pitrans features to control the onboard mixers, muting, DSP and other options in the chip. Pitrans changes the configuration for analog transmit, analog receive and external io. pitrans can also monitor the pi’s serial port RTS GPIO line to detect talk requests from digital programs. WSJTX and FLDIGI both be configured to toggle this signal when they want to transmit.

   Until one of those “client” programs is loaded, the pi’s RTS line may be in the WRONG state. This would cause the transmitter to be stuck “on” all the time. Pitrans has been programmed to ignore RTS requests except when we enable honoring them by typing the “C” command (for [C]all Control Unit - the unix device driver that controls the serial port). The “C” (or “c”) command toggles digital mode control of the transmitter on and off. It should be enabled only after WSJT-X or FLDIGI is loaded and has initialized the RTS line.

3. We will need to set up a “magic” Pulse audio redirector that allows WSJT-X to use two special jack devices in its audio dialogs. These are not present in standard installations of Raspbian. We manually installed the needed Pulse redirector module using apt-get in the first guide.

Loading these redirectors is best done before loading the Jack Daemon. If the redirectors are left in the system all the time, they break ALSA audio so we can't use it for other purposes. We must remove them when we are done. The redirectors are loaded and unloaded by the paired commands:

pacmd load-module module-jack-source
pacmd load-module module-jack-sink

and

pacmd unload-module module-jack-source
pacmd unload-module module-jack-sink

For some unknown reason, they can take an unpredictable amount of time to take effect - as long as 4 seconds!

4. QJackctl is an interactive program used to configure, load and operate the jack audio connection system. Note that this is not the only way to do these things. Often an instance of jackd (the jack daemon) may have been loaded and run without you realizing it! It's a good idea to make sure that this is not the case by executing "killall jackd" and ignoring any errors that say jackd was not found before proceeding.

   You must run Qjackctl in background to keep it around for further use and once started, it needs to be configured to use our radiohat alsa input and output devices as a 32 bit 48khz full duplex audio server running with a real time priority of 6 or so. The server should be set to automatically start, and we will eventually have it automatically control the "patch" used to direct audio to and from devices that know how to use it.

   There's one annoying problem with the pulse audio jack modules, however. They automatically connect themselves DIRECTLY to the "system" input and output devices (radioHat) - ***bypassing any and all other patches we may want to use to handle the audio***. This can be defeated by proper configuration of the Pulse system, but

it’s simpler for now to just remove the bogus patches manually while setting up our own patches.

In ~/radiohead/ we installed a patch file named “WSJTX.xml” that QJackctl can use to deal with all this more easily. It can add all the desired connections and keep them active. Unfortunately Pulse audio will ***still*** add its bogus patches. They need to be removed manually each time you set up the digital routing configuration.

5. To get GnuRadio to use audio from “jackd” as source and sink devices for prototyping we have another problem. GnuRadio does not know how to select audio systems except by modifying its initialization file contents! The documentation ***says*** it should do this automatically, but the gnuradio “auto” audio selection option currently does not allow what we are trying to do. It always picks ALSA if enabled. It might as well be called the “ALSA-ONLY” audio option. The gnuradio initialization file contents must be changed to specify “jack” instead of “auto” in order to use gnu radio with the jack audio system.

   Fortunately, you can do this without tampering with the master copy of the file by adding the needed command in a local file named “config.conf” to a directory called .gnuradio/ in your own home directory. Removing that file is a simple way to restore the GnuRadio behaviour to the “auto” (remember, this actually means ALSA) configuration. Note that we need the alsa configuration working for our other, simpler gnuradio DSP tasks ! If they are not working - try deleting ~/.gnuradio/config.conf! That won’t hurt anything when you are not trying to run FLdigi or WSJT-x, but it will return to the “auto” configuration once our custom file is removed.

   For our purposes, the local ~/.config.conf file needs only to contain the following:

   ```
   [audio]

   audio_module = jack
   ```

Changing the line that says “jack” to “auto” in this file is another way to restore normal operation with ALSA sources and sinks.

6. We also installed a GnuRadio DSP source program called WSJTX.grc which is also present in standalone form as WSJTX.py. This program runs a pair of simple DSP streams that convert ordinary audio to and from Single Sideband I and Q signals.

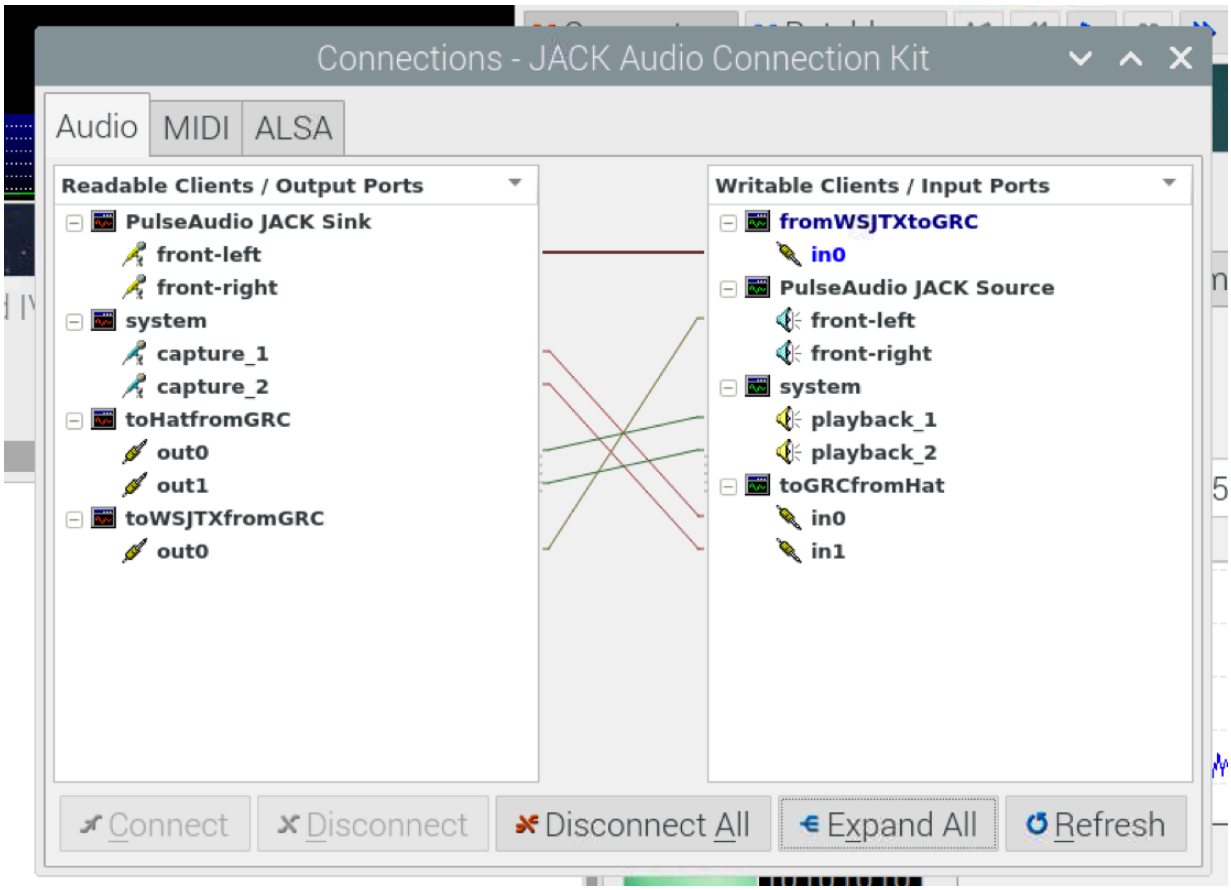
   One flow converts I and Q from a pair of jack inputs called “toGRCfromHat” and outputs mono audio on a single output called “toWSJTXfromGRC”.

   A second path through the same module converts a jack mono audio input called “fromWSJTXtoGRC” to Single Sideband I and Q signals on an output called “toHatfromGRC”.

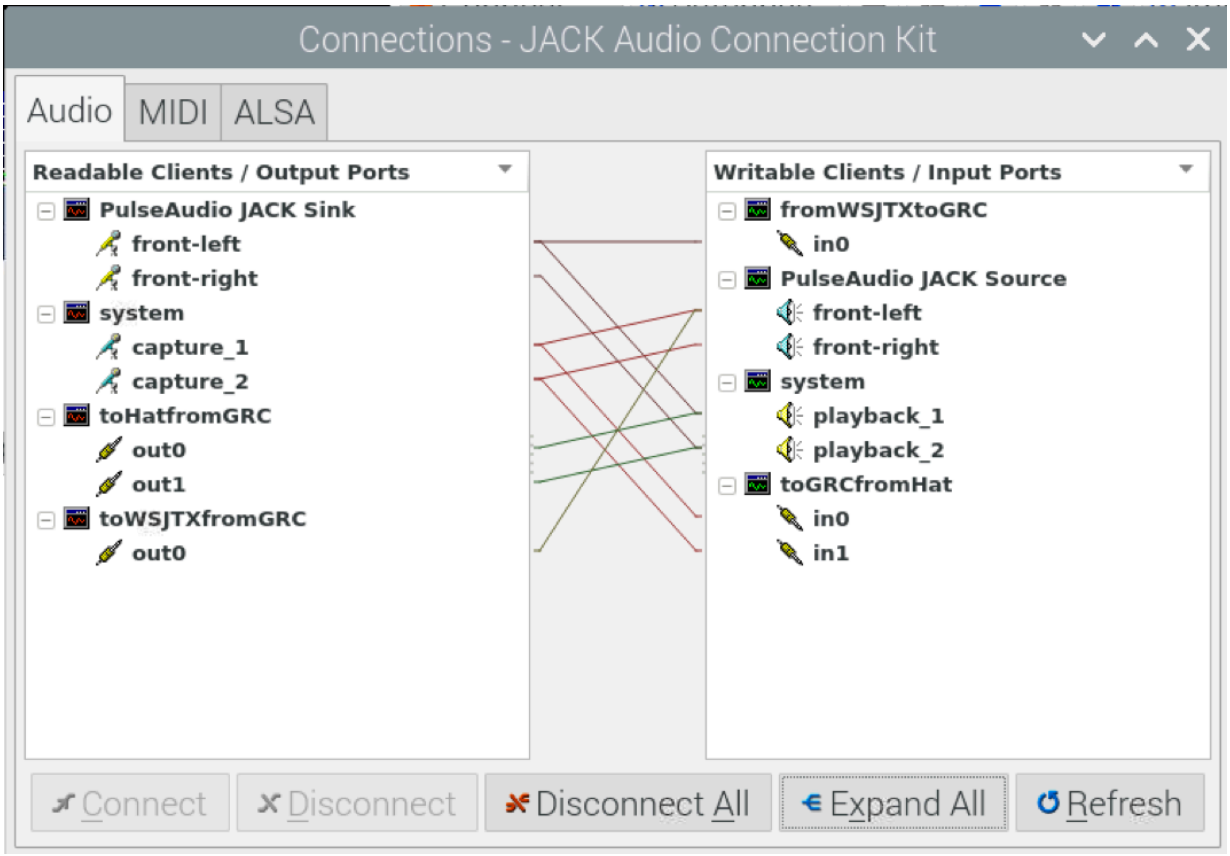
   The pulse audio redirectors to and from jack are called “Pulse Audio Jack Source” and “Pulse Audio Jack Sink”.

7. Once a program like WSJT-x that is configured to use the pulse audio redirectors (called jack-in and jack-out) in the WSJT-x Audio dialog) is started, it can now use the gnuradio dsp filters via the Jack Audio Connection kit.

This all looks something like this in the Qjackctl “Connections” dialog when properly configured with WSJT-x running :



Here's another screen shot showing the same thing with the incorrect redirection patch Pulse audio installs. Note that patches go AROUND all of our DSP and make a real mess if put on the air! You can see "PulseAudio JACK Sink" incorrectly connected directly to "system:playback_1 and 2" in addition to our own connection to those ports. Similarly "system:capture_1 and 2" are incorrectly patched directly to "PulseAudio JACK Source".



These need to be removed manually. If a QJackctl patch file like “WSJTX.xml” has been configured, it will keep all the patches we specified to be kept correctly configured, but it will still allow the new bad patches to be ADDED to the configuration.

We can use this behavior to quickly restore the correct configuration by clicking on the button “Disconnect All” to remove ALL the patches. We will be warned that QJackctl will automatically reload our desired WSJTX.xml patches, but that’s what we want to have happen. The result will be the correct patching shown above.

Are we having fun yet?

To sum up, here’s everything we need to do:

1. Make sure things are the way we expect when we start:

   killall jackd
   pacmd unload-module module-jack-source
   pacmd unload-module module-jack-sink

2. Make sure the RadioHat audio card hardware has been initialized and is running:

   ~/radiohead/pitrans1/pitrans -i

3. Start the Jack Audio system and set up our patches (assuming QJackctl is already configured)

   qjackctl &

4. Start WSJT-X

   wsjtx &

5. Switch GnuRadio to be configured for jack IO (we've pre-installed protoype versions of config.conf we can just copy to .gnuradio/):

   cp ~/radiohead/jackscripts/config.conf.jack ~/.gnuradio/config.conf

6. Load the GnuRadio DSP program:

   ~/radiohead/WSJTX.py

7. Load the pitrans program again so we can use it to control the radio:

   ~/radiohead/pitrans1/pitrans

8. Manually restore the correct patching for jack by using the "Disconnect" button in QJackctl's "Connections" Dialog. Then

checking that the patch is right.

9. Make sure the VFO is set where we want it and set to USB in pitrans.

10. Make sure WSJT-X is set to the correct band so that the logging records match the QSO frequencies.

11. Type “c” to enable CCU control of the transmitter

12. Rearrange and hide the mess of window’s we’ve just created (without accidentally killing any of the stuff we just set up!)

After you are done using digital programs you must:

1. Restore the alsa configuration for gnu radio

   cp ~/radiohead/jackscripts/config.conf.alsa ~/.gnuradio/config.conf

2. Unload the pulse audio redirectors

   pacmd unload-module module-jack-source
   pacmd unload-module module-jack-sink

3. Make sure jackd has stopped:

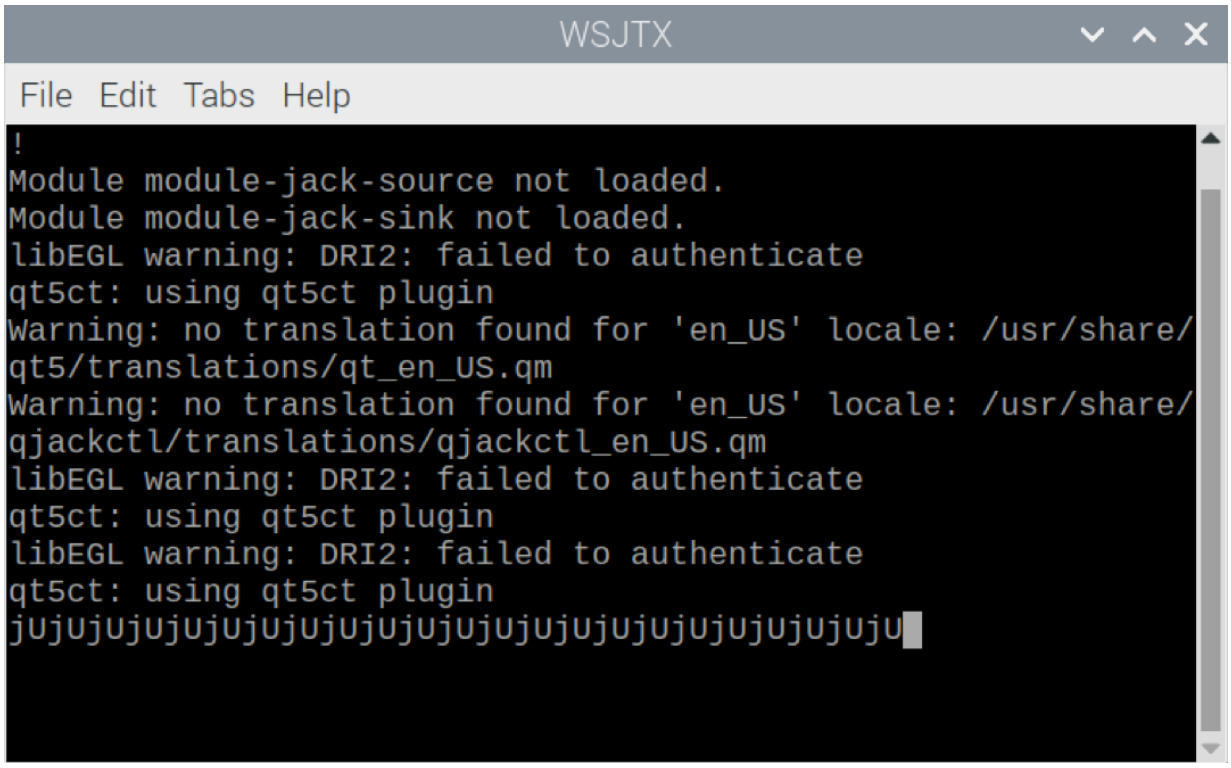
   killall jackd

An (Only Slightly) Simpler way

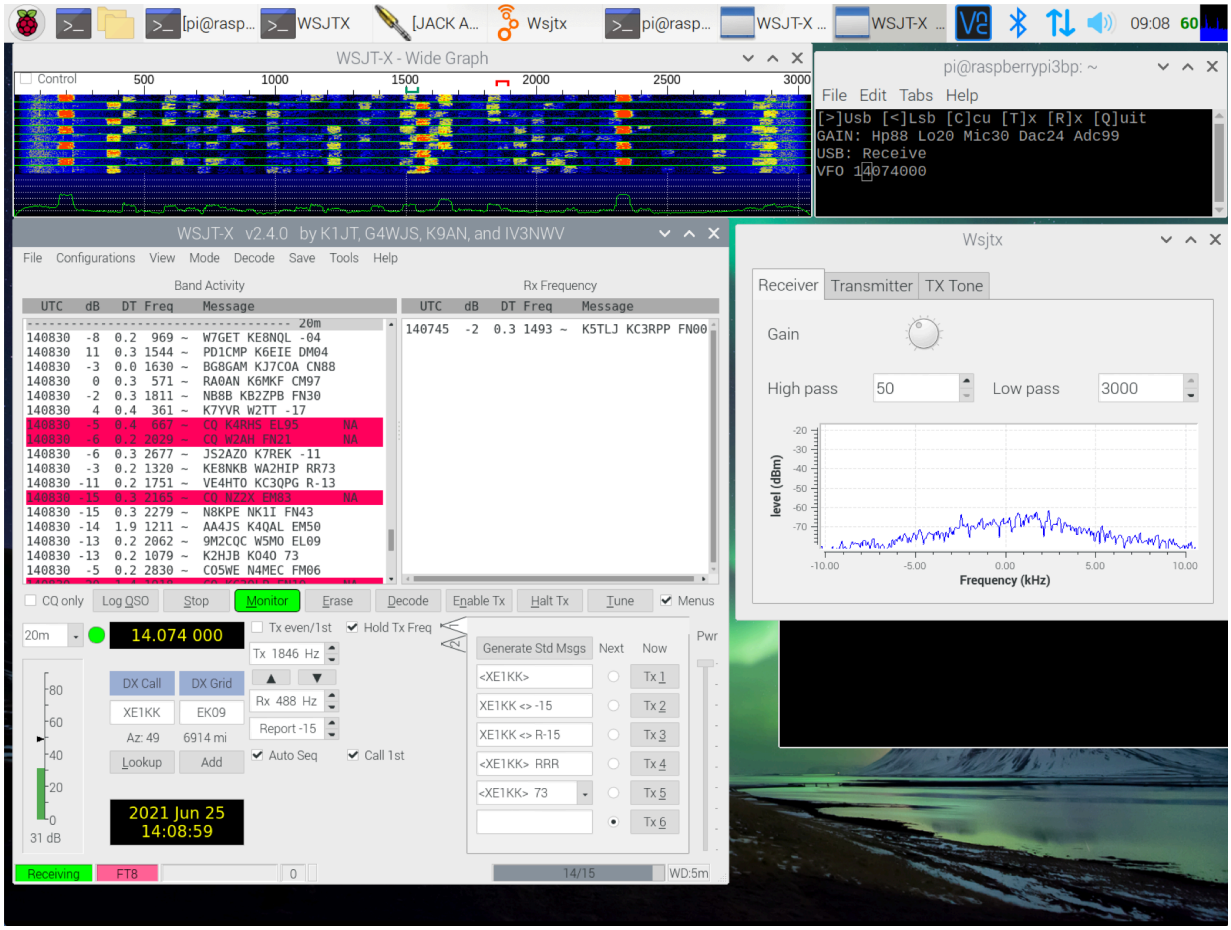
It was important to go through all that to understand how this works, but now that you’ve read all the preceding stuff you can try using the script called “WSJTX” in ~/RadioHead/ to do *most* of this. It usually succeeds, but you must still manually do a few things (remove the bogus patches, load pitrans and set the digital mode by typing “c”, and clean up the

huge mess of windows it leaves behind.

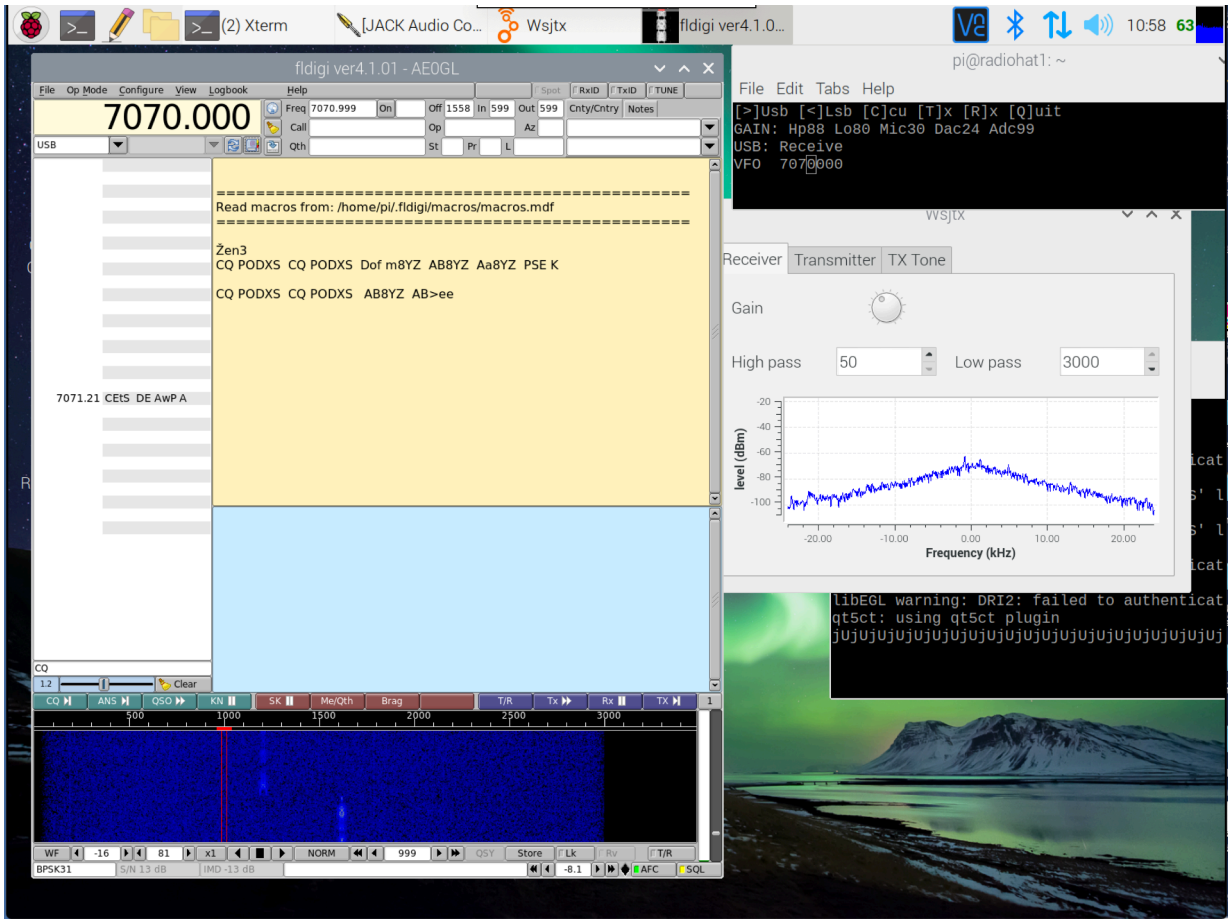
Occasionally a processor overload peak at the wrong time during loading of all this stuff will cause the GRC program to start dropping data. It shows that this is happening by endlessly typing little two letter error message abbreviations into the parent python window that launched its GUI. When this happens, the best thing is to unload everything and try again. Once it starts properly, an occasional two letter complaint may appear but that's OK and usually just means it didn't like a switching transient. Here's the normal appearance of the shell window (with the usual collection of mystifying and gratuitous error messages from all those the open source programs and modules):



Here's what the whole thing typically looks like when everything's cleaned up and I'm operating it:



Operating FLdigi is similar, but it does most of the configuration work for you and actually works better if you let it create its own Jack interface instead of using the Pulse audio one. Note that using the “WSJTX” script provided, if you simply quit from WSJT-X, you can start FLdigi from the Raspbian GUI menu (assuming it’s properly configured) and it will use the redirection that’s been set up. You can go back and forth between them as often as you like, but don’t try running both at once. Setting up FLdigi is left as an exercise for the reader, but here’s proof it works…



The redirection won't be removed until you terminate the Gnu Radio python script by closing it's GUI window.

Sometimes the shell script doesn't unwind everything properly, so I keep a shell script around called RESETAUDIO that removes the redirectors, restores the GRC init file and kills jack audio. You can find it inside the ~/Radiohead/ directory.