libradiohat - experimental version and tools

Added 1-March-2022

introducing *libradiohat*

This is a refactored set of libraries for accessing the *RadioHat 1.0* hardware. It has many clean-ups and improvements and may now be built as a dynamic library.

This is primarily intended for use with *Bullseye*. It will also work on *Buster*, but other things may require fixup to go backwards!

This folder includes an improved and rewritten version of *pitrans*, called *transceiver* that uses the dynamic library for all *RadioHat* access.

There's a primitive *make* file included that builds the library and test program but does not install it. For now, progams that use it must access things by absolute locations, or you must move the dynamic library to somewhere on the load library path.

You'll also need to ensure that you've installed *libgpiod-dev* and *libncurses5-dev* before building. A *make clean* is probably also a good idea the first time.

Two other experimental programs are included that expect to find the libradiohat.so in *~/radiohat/libradiohat* for now.

Note that these programs probably will not work on *Buster* because the earlier version of Gnu Radio it supports is not file or program compatible with newer versions. The same is true of the *Quisk* it offers.

The earlier GRC programs in .~/radiohat should still be functional using *transceiver* instead of *pitrans* as they have not been disturbed by the new *GRC*. As usual ALSA device names may need touching up.

Other Tools

In the *GRC* folder, *transceiver.grc* / *transceiver.py* is an experimental gnuradio flow graph that attempts to put all the things needed to experiment with and test *RadioHat* in one spot. It also includes some experimental PTT and VFO access from the flow graph. Unfortunately the ways needed to do this at the moment add a lot of overhead, and the program is unstable when run in the GRC environment. It seems to work OK by running the .py file, however. It also required Gnu Radio Version 3.8 or later.

Be aware, however, that the 1ms polling rate GRC uses for exporting variables as messages adds a lot of overhead and as a result this flow graph is occasional getting out of sync and losing alternate sideband rejection - usually if a lot of VFO tuning around is done at high speed. It's still fun and very useful for quick testing, but stick to the simpler flow graphs for serious measurements!

I'm also seeing a lot of crazy behaviour caused by *Pulse Audio* and the new *PipeWire* server. Neither one of them can be disabled any more and after a while they start distorting the IQ signals causing loss of functionality. Sometimes killing pulse audio (it will restart) will fix it - other times only a reboot will do it. Note that the "Bullseye" needs rebooting - not radiohat! The new audio servers break Jack and prevent it's use with digital modes, as well.

Experimental *quisk* interface

QUISK is a folder containg a hardware configuration file for Quisk. There are two copies present under different names. It's generally operating OK with the QUISK version in the Bullseye apt repository, but it's still very experimental. It almost certainly will not work under the version included with *Buster*. QUISK is very hard to configure and get working, but if you want to play with it, you may have some luck by moving the *radiohat* folder inside QUISK to the quisk configuration directory and setting quisk up as a radio of type "Softrock fixed" with one of these config files chosen.

All these programs are intended for use with the default Alsa device and I suggest using a usb plug-in dongle. This simplifies understanding them and eliminates problems with t/r switching the codec hardware audio paths. You can also use the built-in Pi headset sound or the HDMI sound for receiver and tone testing, since they both include dummy input devices.

We'll return to using the built-in headset jack and digital modes very soon, but for now these simplifications are speeding development a great deal and ensuring a solid foundation for the work to come.