Compile pihpsdr from the source-code (Linux) (Desktop PC / Laptop / RaspBerry Pi / Tinker Board)

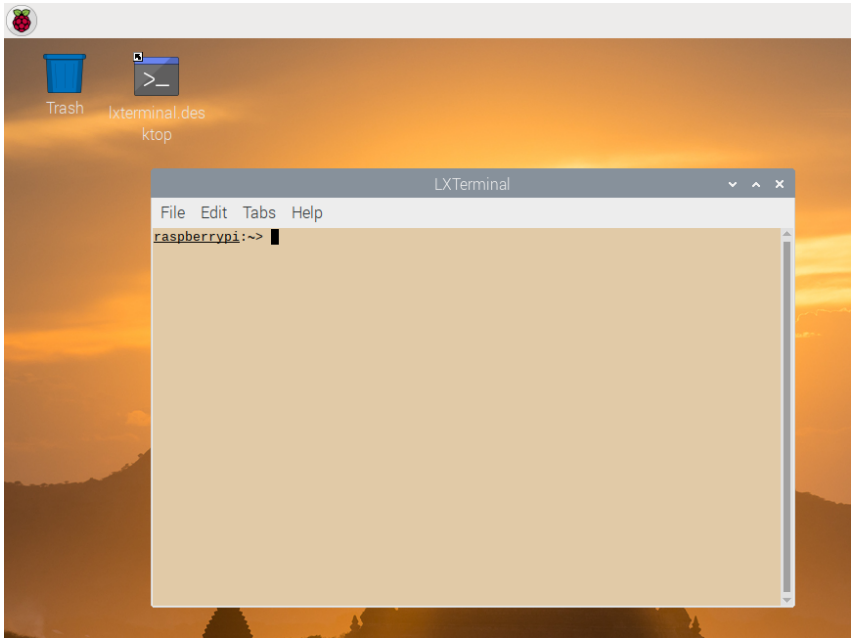
A) Hardware, Software, and Skills required to follow the instructions in this document.

- An obvious prerequisite is that you have **a computer running the Linux operating system** (OS) in order to compile and run piHPSDR there. This can be a Desktop or laptop PC, or a small single-board computer (SBC) such as the RaspberryPi (RaspPi) or the ASUS TinkerBoard. The single but essential difference between the Raspberry Pi and the TinkerBoard on one side, and a Linux installation on a Desktop PC or laptop on the other hand, is that only the SBCs have general-purpose input/output (GPIO) connections that can be used for connecting push-buttons, rotary encoders, Morse keys etc. So it should be clear that all instructions concerning GPIO only apply to SBCs.

- Since we need to install additional software components, the computer **needs an internet connection**, no matter whether this connection is realized by an ethernet cable or using a WLAN. At the very end, when piHPSDR is up and running, you most likely need an ethernet cable network connection to connect to the radio.

- It will be necessary that you have the Linux OS running on this computer. In **Appendix A** some instructions how to obtain and install Linux from the internet are given. Installing Linux is actually the most complicated part of the whole business here, but in most cases you do not need it: if you want to use piHPSDR on a Linux PC, then most likely you already have one, and if you buy or have bought a RaspPi, the vendor almost certainly also offers SD cards with a pre-installed Linux OS that you simply have to insert in the SD card slot. Take care to use an SD card which holds at least 8 GByte.

- The commands given in this document must be entered in a terminal window, so you must know how to open such a window. On a RaspPi, go into the menus behind the raspberry symbol in the top left corner of the screen and select "Accessoires->Terminal". A terminal window opens and looks like displayed in the figure below. Because we will need the terminal window quite often, it is a good idea to create a Desktop icon (the result is shown in the figure) such that we can open new terminal windows by double-clicking this icon. On a RaspPi, this is done by navigating to the terminal application in the Raspberry menu and hitting the right mouse button.



In this terminal window, you can now type in commands. Begin with typing in the command

```
echo $HOME
```

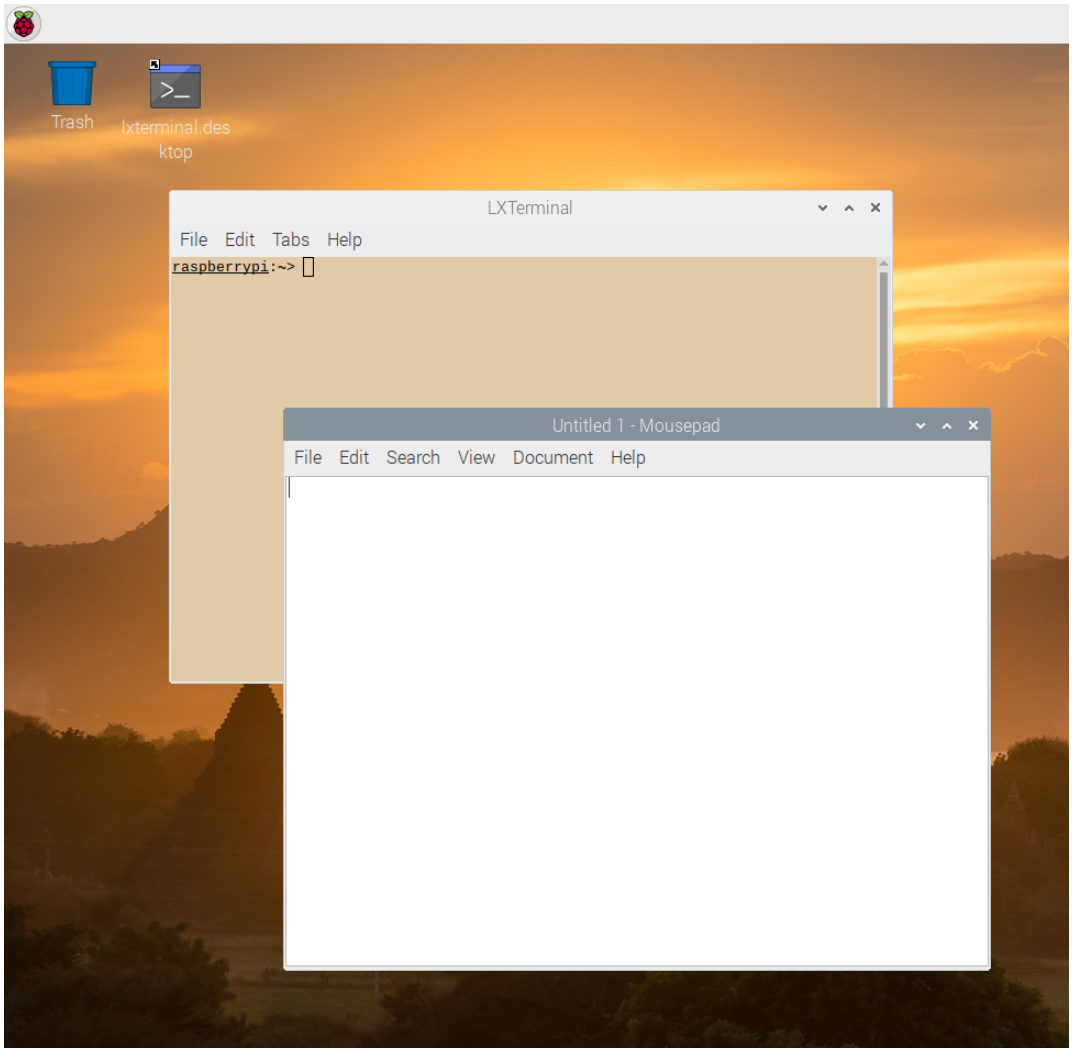
Throughout this manual, commands to be typed into a terminal window are set in blue colour with a monospaced font. You have to type the command either exactly as printed here, open the "LibreOffice Writer program" (behind the Raspberry in the "Office" menu) and open this document (the OpenDocument version, that is the file name ending in .odt), then you can copy-and-paste the commands directly from this document into the terminal window.
As a results of this command, the name of your home directory should be printed on the screen. This is /home/pi for the RaspPi, /home/linaro for the TinkerBoard and /home/user for Desktop/Laptop Linux computers, where "user" is replaced by your Linux user name there.

- You most likely have to install additional software components, and this requires administrator privileges. To check if you can execute a command with administrator privileges, type in the command

```
sudo ls -l
```

This should list all the files in your home directory. On the RaspPi, this usually works without further a-do. On other systems, you might get asked the administrator password before the command is executed (if you do not know this password, you cannot manage the system and ask the person who installed the Linux). On some Desktop/Laptop Linux systems, the security level is even higher and you must be explicitly entitled to use the "sudo" command. Ask a local Linux guru how to achive this (most likely, your user name must be included in the "sudoers" file, but this may depend on the system and/or the security level imposed there).

- The final prerequisite is that you are able to create and modify text files. I usually do this from within a terminal window using the "vi" command, but this is really old-school since I am working with Linux/Unix systems for more than 30 years. I guess if you are reading these instructions this means that you have *not* been working with Unix/Linux since the 1980s, and then the learning curve for mastering the vi program is rather steep. Therefore these instructions are made such that you can equally well use a text editor with a graphical user interface (GUI). On the RaspPi, a very simple such text editor ("Mousepad") can be found behind the Raspberry: "Accessories->Text Editor". This opens a new window such that the screen looks like this:



In the white area, you can type in text, but you can also copy-and-paste text there from this document, if opened in LibreOffice Writer. Type in the text

```
Now is the time
for all brave men
to come to the party.
```

Throughout this document, contents of text files to be created, or parts of text files that need to be modified, are shown in green colour and a monospaced font. If you

type in the text, do not forget the Enter key after typing in the last line! If the text has been entered, you can go to the menu "File->Save As" of the text editor, and enter a name for the file (take the name `TextFile`). In the standard setup, the file saving dialog should be in your home directory, else you have to move there. Close the text editor window and go back to the terminal window by clicking somewhere in its area and type in the command

```
echo TextFile
```

This should produce as output the text shown in green above. As the last step, we must be able to modify existing text files. To test if we can do this, open the text editor again and choose the file named TextFile through the "File->Open" menu by double-clicking the file. Now you can use the mouse but also the arrow keys on the keyboard to navigate, say, to the word "men" and add the words " and women" at the end of that line. Save the modified file through the "File->Save" menu and close the text editor window. Then activate the terminal window again and type in the command

```
echo TextFile
```

again. Now the modified text should be printed on the screen.

If you do not succeed in performing these tasks so far, it makes no sense to continue reading. It is strongly recommended to go to the next regular radio amateur meeting and ask for help. What we have done so far is just very basic Linux, below this you won't be able to proceed further.

- In rare cases (using the GPIO on the RaspPi4, or permanently creating audio loopback interfaces) it will be necessary to modify a text file somewhere deep in the system, where you need administrator privileges to do so. In this case you usually can open the file with the text editor but cannot save it. To circumvent this problem, I suggest the following three-step procedure to modify such a file:

  a) copy the file to your home directory
  b) modify it using the text editor
  c) copy the file back to its proper location

  For example, to modify the file /boot/config.txt (which you might need if you want to use GPIO input lines on the RaspPi4, see below), step a) is performed by the command

  ```
  cp /boot/config.txt $HOME
  ```

  (note no "sudo" here!). Thereafter, a file named config.txt is in you home directory and you can modify it with the text editor (step b). Then copy it back (step c) with the command

  ```
  sudo cp $HOME/config.txt /boot
  ```

B) Install required software packages

In this step we will obtain and install software packages that may or may not be already present on your system and are needed to compile piHPSDR. This is done by a sequence of commands to by typed into the terminal window. It is a good idea to start with updating all existing packages to the most recent version. This is done with

```
sudo apt-get update
sudo apt-get upgrade
```

It is a good idea to reboot the system at this stage, especially if the kernel has been updated.

Standard tools for compiling a program. Since we want to compile a program, the standard tools for building a program (such as a compiler, a linker, etc.) are needed. These tools are often installed on a standard Linux system, but for the Desktop Linux installation I tried, this was not the case. To install packages, the computer needs an internet connection. The core command for installing software is the `apt` command, so install the standard tools with the following commands:

```
sudo apt-get install make
sudo apt-get install gcc
sudo apt-get install git
sudo apt-get install pkg-config
sudo apt-get install cmake
```

Note that a large number of packages is installed with these commands, because the installation of a software package automatically triggers the installation of all prerequisite package.

Libraries and packages necessary to compile piHPSDR. In addition to the packages virtually necessary to compile any program, we need a set of libraries that piHPSDR is built on. These are installed with the commands

```
sudo apt-get install libfftw3-dev
sudo apt-get install libgtk-3-dev
sudo apt-get install libasound2-dev
sudo apt-get install libcurl4-openssl-dev
sudo apt-get install libusb-1.0-0-dev
sudo apt-get install libi2c-dev
sudo apt-get install wiringpi
```

The last library, wiringpi, is only availabe and needed on SBCs which have GPIO connectors, so the last line should not be entered for a Desktop/Laptop system (although it does no harm there except producing an error message).

Note on RaspPi-4 and GPIO:

If you want to use the piHPSDR controller and/or a Morse key attached to the GPIO, you need the latest version of the wiringPi library. If you do not intend to use GPIO lines, nothing needs be done here.
The version of the wiringPi library can be checked with the command

```
gpio -v
```

and it should be at least version 2.52. To install the latest version from the wiringPi web site, use the commands

```
cd /tmp
wget https://project-downloads.drogon.net/wiringpi-latest.deb
sudo dpkg -i wiringpi-latest.deb
```

But it is still not working, because (at least in version 2.52) the wiringPi library does not set the pull-up options correctly for the new BroadCom chip used in the RPi-4. So you have to set either manually or permanently at startup the GPIO pin modes.

To do so, you first you to disable all Interfaces except SSH and I2C. This is done by opening in the "Raspberry Preferences->RaspberryPi Configuration" and select the "Interfaces" tab in the window that opens.

To fix the problems of WiringPi not being able to set the GPIO pullups correctly you can add the following 2 lines to `/boot/config.txt`

```
# setup GPIO for piHPSDR Controller 2
gpio=4-27=ip,pu
```

by the same three-step procedure. Now reboot the system and all should be OK. The latest version of the "V2" controller realizes the pullup resistors in hardware so this is possibly not necessary there. As a general recipe, the list in the line starting with "`gpio=`" should contain all GPIO lines used by the program for input. Note that this command uses the GPIO numbering scheme while in piHPSDR, you specify the lines by their WiringPi numbers.

C) download, compile and install WDSP

The WDSP library is the core engine of piHPSDR. It is not available on the standard Linux archive, we have to compile and install it from the sources. To this end, type the following commands into a terminal window:

```
cd $HOME
git clone https://github.com/g0orx/wdsp
cd $HOME/wdsp
make clean
make -j 4
sudo make install
```

Note that the option "`-j 4`" to the `make` command indicates that the compilation can proceed on four CPU cores in parallel – you can omit this option then compilation takes considerably more time. Compiling and installing WDSP should proceed without any error messages.

D) download/adjust/compile piHPSDR

Go to your home directory and download piHPSDR using these commands:

```
cd $HOME
git clone https://github.com/g0orx/pihpsdr
```

The Makefile needs some adjustment before you start compilation. You can use the text editor you have already used before. Open the Makefile by the "File->Open" menu, navigate from the home directory to the "pihpsdr" folder and select the file named "Makefile". It is recommended to widen the window of the text editor such that the text lines in the Makefile do not get wrapped around. At the beginning of the Makefile, there are many lines that assume one of the two forms

```
#NAME=TAG
NAME=TAG
```

The first form is the "deactivated" form, the second one the "activated" form. Inserting/deleting the hash tag in the first columns switches between the two forms. These lines enable/disable features to be built into the piHPSDR program.

Here I give a comprehensive list of all features that I have enabled in my sample installation:

```
GPIO_INCLUDE=GPIO
PURESIGNAL_INCLUDE=PURESIGNAL
LOCALCW_INCLUDE=LOCALCW
STEMLAB_DISCOVERY=STEMLAB_DISCOVERY_NOAVAHI
MIDI_INCLUDE=MIDI
PTT_INCLUDE=PTT
```

This means that the program is, for example, compiled with MIDI support which does not harm if you do not plan to use MIDI input devices.

Note 1: STEMLAB_DISCOVERY is an option to support RedPitaya-based SDRs where the SDR program on the RedPitaya has to be started through a web interface.

Note 2: On desktop or laptop computers running Linux, the lines containing GPIO_INCLUDE and PTT_INCLUDE must be deactivated and thus read

```
#GPIO_INCLUDE=GPIO
#PTT_INCLUDE=PTT
```

Note 3: There is support for using devices that are supported by SoapySDR. These include the LimeSDR, PlutoSDR and RTLSDR (for the latter: receive only).

To compile support for SoapySDR the line in the Make file should read:

```
SOAPYSDR_INCLUDE=SOAPYSDR
```

If support for SoapySDR is not required the line in the Makefile should read:

```
#SOAPYSDR_INCLUDE=SOAPYSDR
```

It is recommended to compile all the components required for SoapySDR support rather than install them from a Linux archive because the latest versions on the master (source code) repository have several improvements and bug fixes. See **Appendix B** at the end of this document for details of downloading, compiling and installing these components. If you have not successfully installed these components but still have the SOAPYSDR feature activated, compilation of piHPSDR will fail.

After having adjusted the Makefile, the piHPSDR program can be compiled by the commands

```
cd $HOME/pihpsdr
make clean
make -j 4
```

The program should be built without any errors.

E) Initial run of piHPSDR

To test the compilation, we make an initial run of piHPSDR without any radios connected to the computer.

a) If we are running pihpsdr on a SBC, we start with the commands

```
cd $HOME/pihpsdr
sudo chown root pihpsdr
sudo chmod u+s pihpsdr
```

This makes piHPDR a "setuid root" application, which is (or rather used to be) recommended for programs that have access to the GPIO. While it seems no longer neccessary on recent versions of the RapsberryPi OS, it might be necessar on other SBCs. **On Desktop/Laptop Linux systems, the last two commands need and should not be given.**

b) As a next step, we copy the HPSDR logo into the "pihpsdr" directory within our home directory. This enables pihpsdr to "find" this logo and display it in the top left of the initial screen. The command to do so is

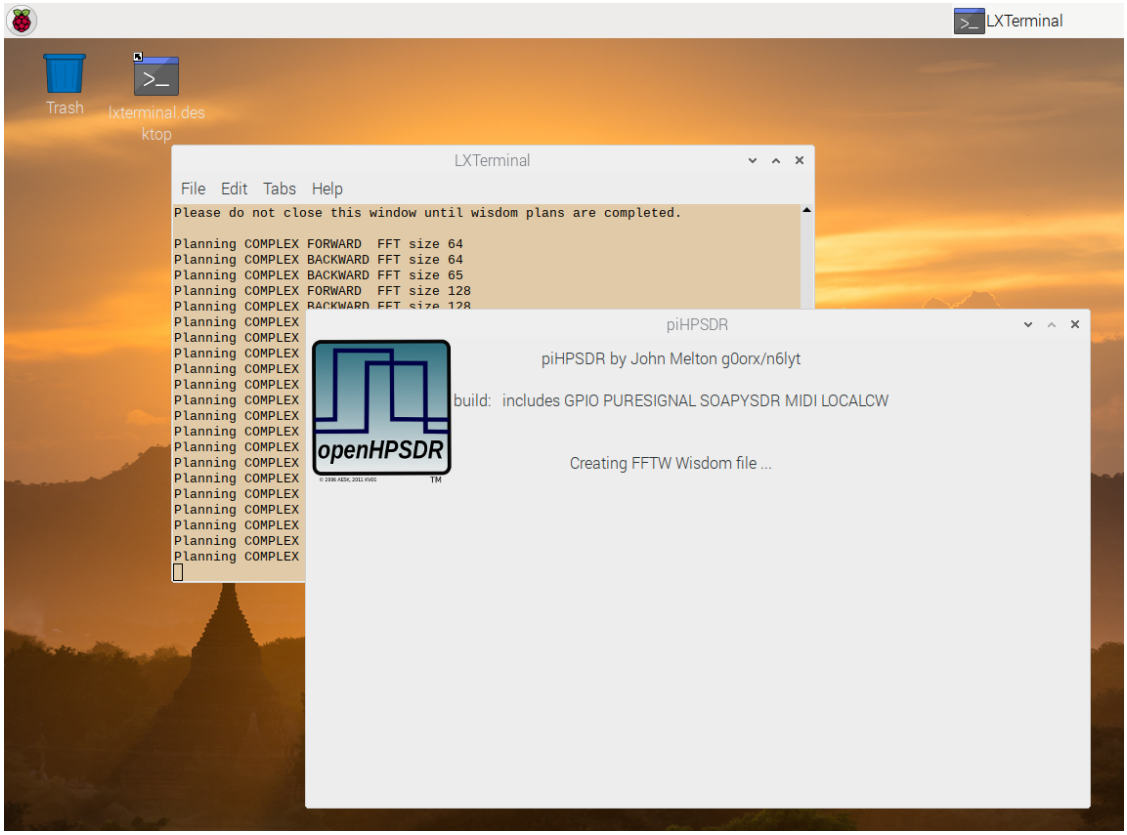
```
cp release/pihpsdr/hpsdr.png .
```

(the closing point is part of the command!).

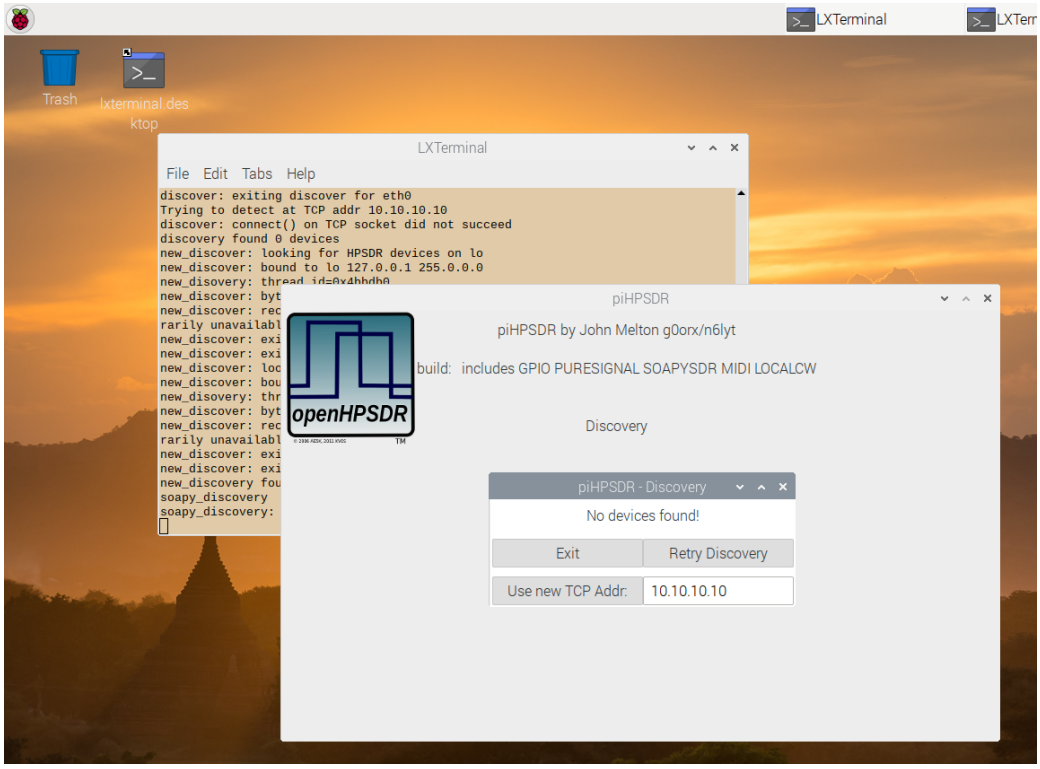
c) Start pihpsdr just by entering the command

```
./pihpsdr
```

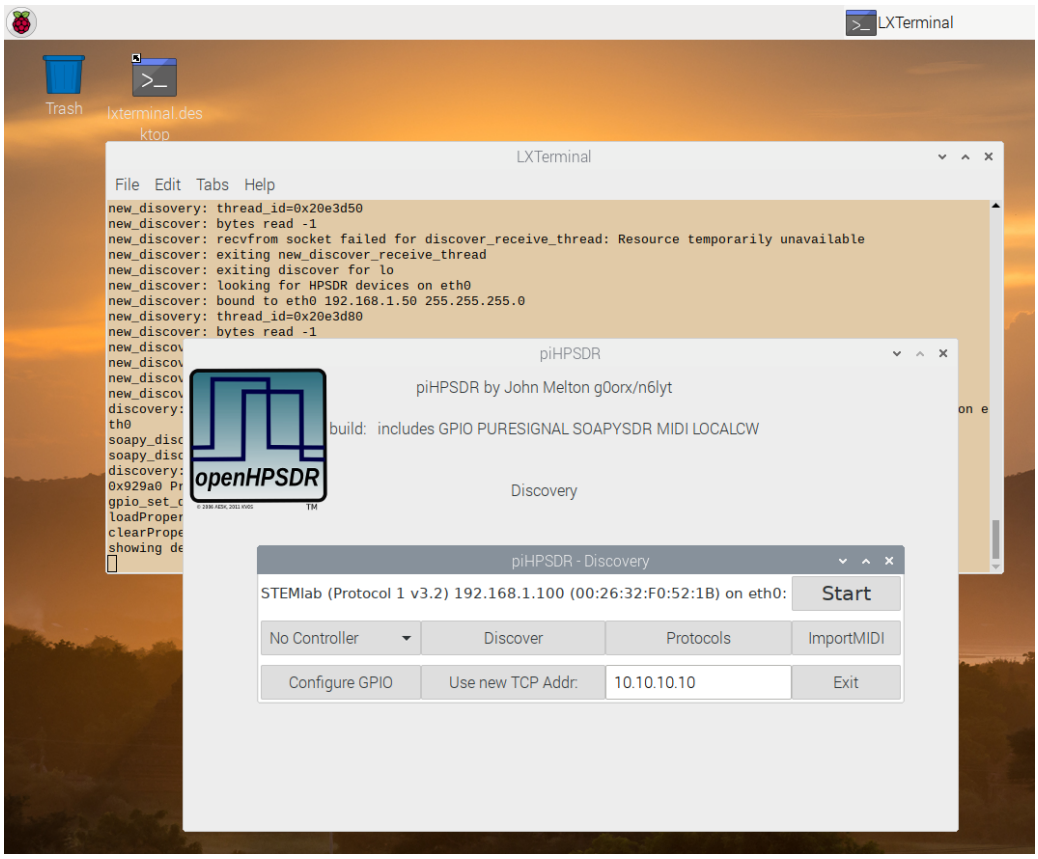
The piHPSDR window should open and the screen should look like this



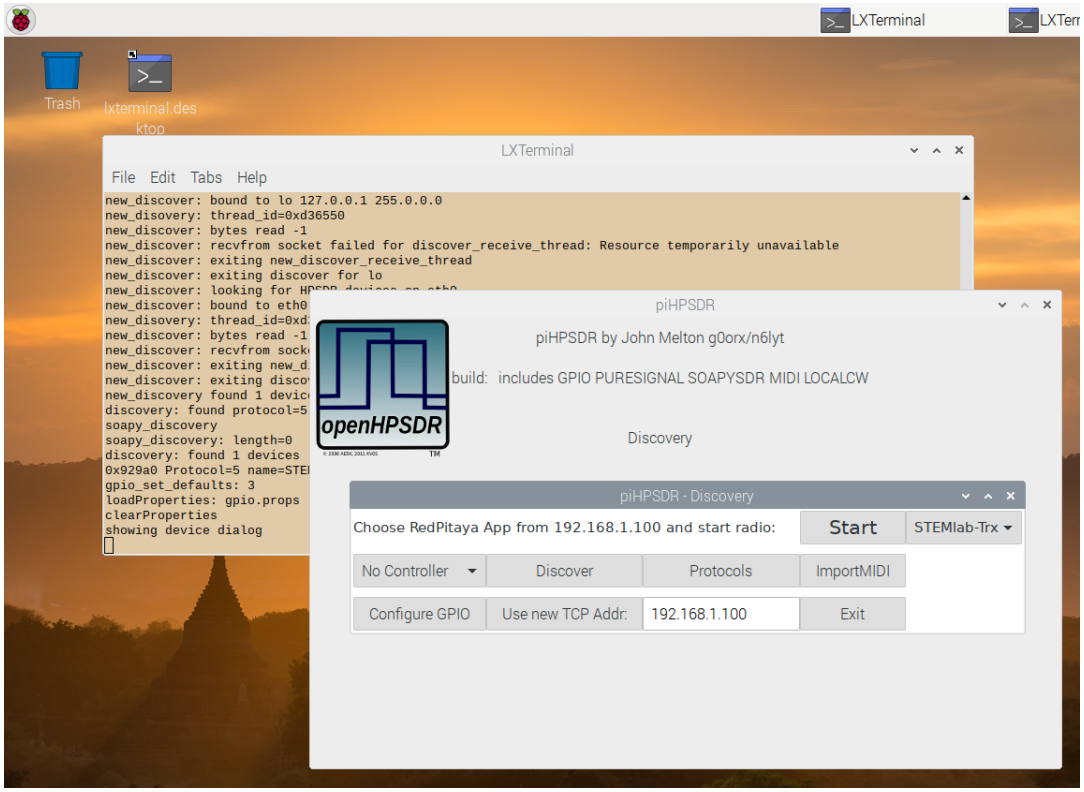
Because it is the first time you started the program, the WDSP library determines (once and for all) the optimum way to do the fast-Fourier-transforms (this will take few minutes). After this time, the piHPSDR window looked like this:



Since we have no radios connected (and therefore no devices have been found), clicking the "Exit" button is the only thing we can do at the moment. If we had connected radios (for example SOAPY devices such as the Adalm-Pluto via USB, or an ANAN radio via an ethernet cable) these devices should be "discovered" and the radio can be started via a "Start" button.. In the next picture, this situation is show, a RedPitaya based radio (STEMlab) has been connected to the RaspPi:



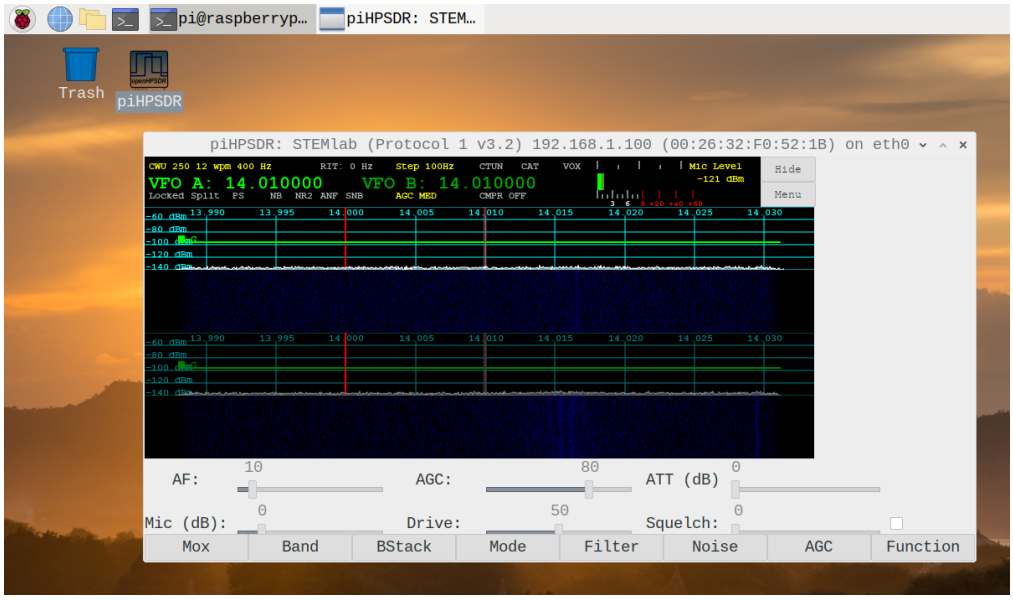
However, to make this work one either has to modify the software on the RedPitaya such that the SDR application automatically starts upon powering up, or one has to connect to the RedPitaya after powering up through a web browser and manually start the SDR application. However piHPSDR is able to do this for you! To this end, you must know the IP address of the RedPitaya (the one you are using when connecting to it with a browser). In my case the STEMlab had the IP address 192.168.1.100, and this address can be entered in the field right to the text "Use new TCP Addr:", followed by clicking that text button (you have to type in the IP address only once, it is saved and restored the next time piHPSDR is started). Then after a short while, the piHPSDR screen looks like this:



You see that piHPSDR has detected the RedPitaya through its web interface. The button reading STEMlab-Trx might be a menu with several entries, if more than on SDR application is found on the RedPitaya (in this case you have the choice which one to use). Clicking the Start button starts the SDR app on the RedPitaya and then connects piHPSDR to this radio.

Some possible trouble with the font sizes (only RaspPi):

Some RaspPi users have reported that the radio window is messed up and looks like this:



This happens especially when using a large monitor. The reason is, that the system may automatically choose a large font when using a large monitor, which is not reasonable for piHPSDR since it is using a fixed-size window. This is easily fixed from the `Raspberry -> Preferences -> Appearance Settings` menu, in the window that opens you click the `System` bar and change the font to a small one, e.g. `FreeSans` with font size 10. Then immediately the piHPSDR window looks OK.

F) Create a desktop icon

Normally one wants to start pihpsdr by clicking an icon on the Desktop. This eliminates the need to open a terminal window, go to the pihipsdr directory and issue the pihpsdr command. To so so, we have to create, in our home directory, a file named `pi.desktop` with the following content:

```
[Desktop Entry]
Name=piHPSDR
Icon=/home/pi/pihpsdr/release/pihpsdr/hpsdr_icon.png
Exec=/home/pi/pihpsdr/pihpsdr.sh
Type=Application
Terminal=false
```

Now we have to copy the file into appropriate locations, this we do with the commands

```
cd $HOME
mkdir -p $HOME/.local/share/applications
cp pi.desktop $HOME/Desktop
cp pi.desktop $HOME/.local/share/applications
```

You might have noted that the desktop entry file does not specify the program `pihpsdr` as the executable, but something named `pihpsdr.sh`. Therefore, we have to create this file.
Simply create this file in your home directory with contents

```
#!/bin/sh
cd $HOME/pihpsdr
./pihpsdr >pihpsdr.log 2>&1
```

and issue the commands

```
cd $HOME
chmod 755 pihpsdr.sh
mv pihpsdr.sh $HOME/pihpsdr
```

The reason for using the file pihpsr.sh is that this mechanism puts lots of debugging and logging messages into the file /home/pihpsdr/pihpsdr.log, and this may help to resolve the issue if something goes wrong. Furthermore, the wrapper takes care that regardless of whether you start pihpsdr by clicking the desktop icon or by starting it from a terminal window, the same working directory is used, and this is important since pihpsdr stores the file with its internal settings there.

TinkerOS: For TinkerOS, the procedure is essentially the same. Replace "/home/pi" everywhere by "/home/linaro".

Desktop/Laptop: Again, the procedure is the same, but you have to replace "/home/pi" by "/home/user", where user is the name of the "normal user" you created when installing Linux.

G) Some further useful things

Fixed IP address. As long as you have your Linux computer and your ANAN radio both connected to a router which offers DHCP service, both devices automatically get an IP address assigned upon powering up, and everything works as described. But there may be situations where this is not fulfilled, for example on a field day where you are outside and want to connect the RaspPi with the ANAN simply by plugging an ethernet cable with its two ends into both devices. The same you have to do if you have only WLAN in your house and have connected the RaspPi to the Internet via WLAN but must connect it to the radio via an Ethernet cable. In this case, you have two choices:

- You assign a fixed IP address to the RaspPi and set it up as a DHCP server, such that the RaspPi provides the IP address for the ANAN

- You do not set up a DHCP server but instead assign fixed IP addresses to both the RaspPi and the ANAN which must be in the same subnet, e.g. 192.1.168.50 for the RaspPi and 192.168.1.99 for the Anan.

In **Appendix C** it is described how to set up the RaspPi with a fixed IP address.

Loopback interfaces. If you are running both piHPSDR and a digimode program such as fldigi or wsjt-x on the RaspPi, the preferred way of transporting audio data between piHPSDR and the digimode program is to do it fully in software, without the audio signal ever being converted from digital to analog. This can be done by using virtual sound cards also known as "virtual audio cables" or "loopback interfaces". Fortunately, loopback interfaces are already built into the Linux sound system, and **Appendix D** explains how to activate them and how to use them with piHPSDR and fldigi as well as wsjt-x.

If you run piHPSDR on a RaspPi but the digimode program on, say, your Desktop PC, then you do not need loopback interfaces but rather a conventional sound interface connecting the PC and the microphone/PTT/headphone jackets of the radio.

Appendix A: Installing Linux

Step 1: obtain OS image

RaspPi: From the web page https://www.raspberrypi.org/downloads/raspbian, obtain the file `2019-09-26-raspbian-buster.zip` which is denoted as the "Raspian buster with desktop" image. Note that the release date which is encoded in the file name may vary. This is a compressed disk image, so un-zip this file such that it becomes the file `2019-09-26-raspbian-buster.img` which is about 3.6 GByte long.

While I am sure that there are other sources of suitable image files, the following protocol (instructions) have been tested with exactly this one.

TinkerBoard: There is a bunch of different LINUX-type systems for the TinkerBoard. This description refers to the "official" one obtained from the web page

https://www.asus.com/uk/Single-Board-Computer/Tinker-Board/HelpDesk_Download/

From there I got the compressed image file (about 1.13 GByte long) with file name `20190821-tinker-board-linaro-stretch-alip-v2.0.11.img`. Again, the release date encoded in the file name may vary.

Desktop/laptop Linux system: Here it depends on which Linux distribution is being used. The instructions given here have been tested with the "Debian GNU Linux" distribution. To this end, a "small" CD-image file (about 350 MByte) with file name `debian-10.3.0-amd64-netinst.iso` has been obtained from the internet page https://www.debian.org/CD/netinst/ (netinst CD image for the amd64 architecture) and this file has to be "burnt" onto a CD or DVD, or onto an USB stick if the PC/laptop supports booting from an USB stick.

Step 2: Install operating system

RaspberryPi and TinkerBoard: The OS image file already contains the complete OS. It has to be written (or "burned") onto an SD card (or, in the case of the TinkerBoard, on the internal eMMC card). How to do this varies depending on which computer you are using. Detailed instructions how to "burn" an image to an SD card from, say, a computer running various operating systems can be found on the internet, see for example

https://www.raspberrypi.org/documentation/installation/installing-images

Note that "burning" can take several minutes, since the I/O speed is about 10 MB/sec on most cards. If you have "burnt" an SD card, it then has to be inserted in the SD-card slot.

Desktop/laptop computer: If you boot from this CD/DVD or USB stick, you get a Debian installation screen from which you choose "Graphical Install". Then proceed further choosing your localization etc. Because only a small boot image has been

downloaded, additional components are obtained from the internet during installation, so you clearly need internet connection for the installation.
When the "software selection" screen appears, check the boxes "desktop environment", "ssh server" and "standard system tools". For the look-and-feel of the desktop environment, there are several choices, I have checked "LXDE" because this is also the standard desktop on the RaspPi. Since more than 1000 software packages are going to be installed, the process may take some time, mainly depending on the speed of your internet connection.
During the installation, you have to specify the password for the administrator ("root") account as well as choosing the name and the password of at least one regular user.

Step 3: First-time boot

RaspPi: The micro-SD-card was then inserted in the RaspPi and the machine booted (with keyboard, mouse and monitor attached). The RaspPi should be connected to a router with a DHCP server via an Ethernet cable.
The system boots, asks for the country/timezone, and for the password of the default user "pi". It automatically connects to the internet and updates all installed software. When this is complete, the system should be restarted.

TinkerBoard: Connect the TinkerBoard to Keyboard, Mouse and Ethernet and power up. Nothing has to be done.

Desktop/laptop: The system automatically boots after the installation. Because this is a standard Linux system, it is much more restrictive concerning the allowance for users to use the `sudo` command to perform administrator tasks. Normally the file `/etc/sudoers` has to be edit to grant the "normal user" such privileges. One possiblity is to add the line

```
user   ALL=(ALL:ALL) ALL
```

to the file `/etc/sudoers` where the name of the "normal user" has to used instead of "user". This gives this user full administrator privileges so the system is potentially insecure.

Appendix B: Compiling and installing the SoapySDR library and components for specific radios

You will need "cmake" and " libusb1.0" to be installed on your system, so enter the commands (as said before: no harm can be done if these packages are already installed)

```
sudo apt install cmake
sudo apt-get install libusb-1.0-0-dev
```

If these packages are already installed on your system, issuing these commands does no harm

Step 1: Download, compile and install SoapySDR

```
cd $HOME
git clone https://github.com/pothosware/SoapySDR.git

cd $HOME/SoapySDR
mkdir build
cd build
cmake ..
make -j 4
sudo make install
sudo ldconfig
```

Step 2: For LimeSDR radios, compile and install LimeSuite (skip this step if you do not need to run LimeSDR devices)

```
cd $HOME
git clone https://github.com/myriadrf/LimeSuite.git

cd $HOME/LimeSuite
cd build
cmake ..
make -j 4

sudo make install
```

Install the udev rules:

```
cd $HOME/LimeSuite
cd udev-rules
sudo ./install.sh
```

Step 3: For Adalm-Pluto, compile and install SoapyPlutoSDR
(skip this step if you do not need to run Adalm-Pluto devices)

Install libad9361 and some more prerequisites

```
sudo apt install libad9361-dev
sudo apt install bison
sudo apt install flex
sudo apt install libxml2-dev
```

Download, compile and install libiio:

```
cd $HOME
git clone https://github.com/analogdevicesinc/libiio.git

cd $HOME/libiio
mkdir build
cd build
cmake ..
make
sudo make install
sudo ldconfig
```

Download, compile and install SoapyPlutoSDR

```
cd $HOME
git clone https://github.com/pothosware/SoapyPlutoSDR

cd $HOME/SoapyPlutoSDR
mkdir build
cd build
cmake ..
make -j 4
sudo make install
```

To install the udev rules read and follow the instructions at
https://wiki.analog.com/university/tools/pluto/drivers/linux
(that is, copy the file `53-adi-plutosdr-usb.rules` to `/etc/udev/rules.d`).

Step 4: For RTLSDR radios, compile and install SoapyRTLSDR
(skip this step if you do not need to run SOAPY-supported RTL sticks)

Download and install librtlsdr

```
cd $HOME
git clone git://git.osmocom.org/rtl-sdr.git

cd $HOME/rtl-sdr
mkdir build
cd build
cmake ..
make
sudo make install
sudo ldconfig
```

Download, compile and install SoapyRTLSDR

```
cd $HOME
git clone https://github.com/pothosware/SoapyRTLSDR

cd $HOME/SoapyRTLSDR
mkdir build
cd build
cmake ..
make
sudo make install
```

Appendix C: setting a fixed IP address

This step is not necessary as long you have both the RaspPi/TinkerBoard and the radio (e.g. the ANAN) connected to a router which offers a DHCP service. Personally, I like connecting the RaspPi and the ANAN *directly* by an Ethernet cable, and have a fixed IP address for both (!) of them. Then I can do QSOs without having any IP routers or switches involved. For example, I use the fixed IP address 192.168.1.50/24 for my RaspPi and 192.168.1.99/24 for my ANAN. These were chosen such that the devices can also be run when connected to my router (for example, if the RaspPi should be connected to the Internet).

To enable a static fixed IP address, use the text editor to create a new text file named eth0 in your home directory with contents

```
auto eth0
 iface eth0 inet static
 address 192.168.1.50
 netmask 255.255.255.0
 gateway 192.168.1.1
 dns-nameservers 192.168.1.1
```

and then copy it to its proper location with the command

```
sudo cp eth0 /etc/network/interfaces.d
```

Reboot the machine and you will have a fixed IP address (192.1681.50 in the example). Note that there is the possibility to set a fixed IP address through a graphical user interface, but that unfortunately did not work for me for an unknown reason. For this reason I cannot exclude that the procedure outlined here does not work for you. However, any local Linux guru can help you with this.

If you need "guru assistance", you may also ask for setting up a DHCP server on the RaspPi, since this eliminates the need of having a fixed IP address on the radio.

Appendix D: creating loop-back "sound cards"

Loopback sound cards are interesting if you plan to run piHPSDR and a digimode program such as Fldigi or WSJTX on the same computer (say, both are running on a RaspPi, or both are running on a Linux laptop). In this case, you want to transfer audio from piHPSDR to the digimode program and back, and this is best done *directly*, that is, without ever going analog.

Since there is some misunderstanding around what this is and how it must be configured, imagine you have a conventional rig and a computer running a digimode program. To do so, you need two (physical) cables:

- cable #1: connects the headphone jack of the computer with the microphone jack of the TRX
- cable #2: connects the microphone jack of the computer with the headphone jack of the TRX

In reality these cables can transport audio in either direction, but you can imagine defining for each cable an "input side" and an "output side", e.g. marked with a red (input side) and a blue (output side) plug, and that you alway connect red plugs with headphone jacks and blue plugs with microphone jacks. Then, audio always flows through the cables from the red (input side) to the blue (output side) plug.
A loopback device is a virtual sound card offering a "headphone/speaker" and a "microphone" device. This forms a virtual cable where the "red plug" (input side) is always associated with the headphone device and the "blue plug" (output side) is always associated with a microphone device. Audio data written to the "input side" of the device from one application can be read by another application at the "output side".

Since we need two cables, we need to have two such loopback "sound cards", which can be created with the following command in a terminal window:

```
sudo modprobe snd-aloop enable=1,1 index=4,5 id=vac1,vac2 pcm_substreams=2,2
```

The two "virtual audio cable" (vac) sound cards will have the numbers 4 and 5 and the names vac1 and vac2. I chose the numbers 4 and 5 since this leaves some room for other audio devices already connected. On the other hand it does not matter if devices in-between (e.g. with numbers 2 and 3) do not exist. You will see them using the command

```
aplay -l
```

if you want. As the output of this command, you should see, among others (many lines deleted):

```
**** List of PLAYBACK Hardware Devices ****
card 0: ALSA [bcm2835 ALSA], device 0: bcm2835 ALSA [bcm2835 ALSA]
card 4: vac1 [Loopback], device 0: Loopback PCM [Loopback PCM]
card 4: vac1 [Loopback], device 1: Loopback PCM [Loopback PCM]
card 5: vac2 [Loopback], device 0: Loopback PCM [Loopback PCM]
card 5: vac2 [Loopback], device 1: Loopback PCM [Loopback PCM]
```

Now imagine that a digimode program and piHPSDR is running on the same computer. You will need two such virtual cables and the configuration is as follows:

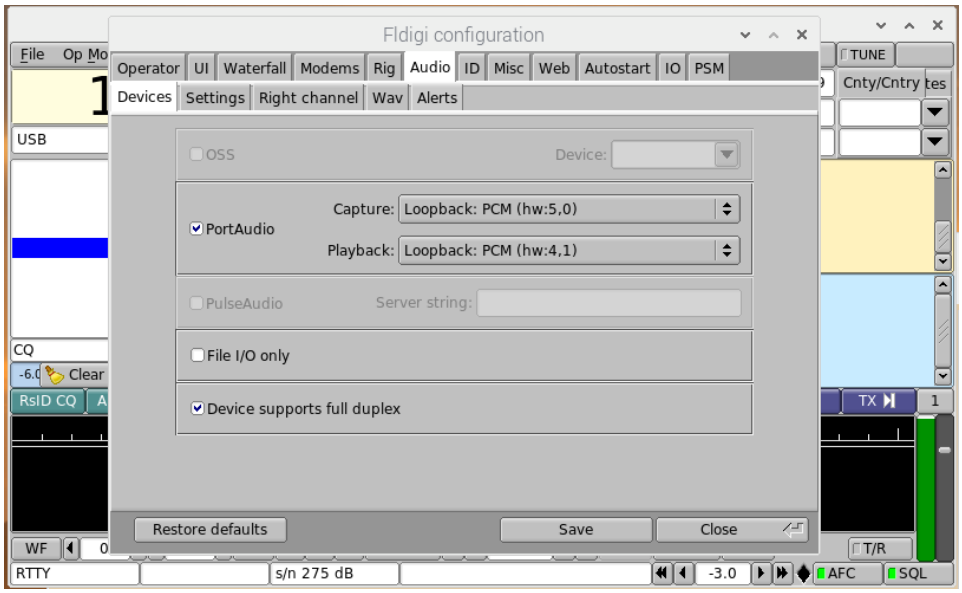
piHPSDR:
- TX local microphone activated, device: "cable #1 output side"
- RX local audio activated, device: "cable #2 input side"

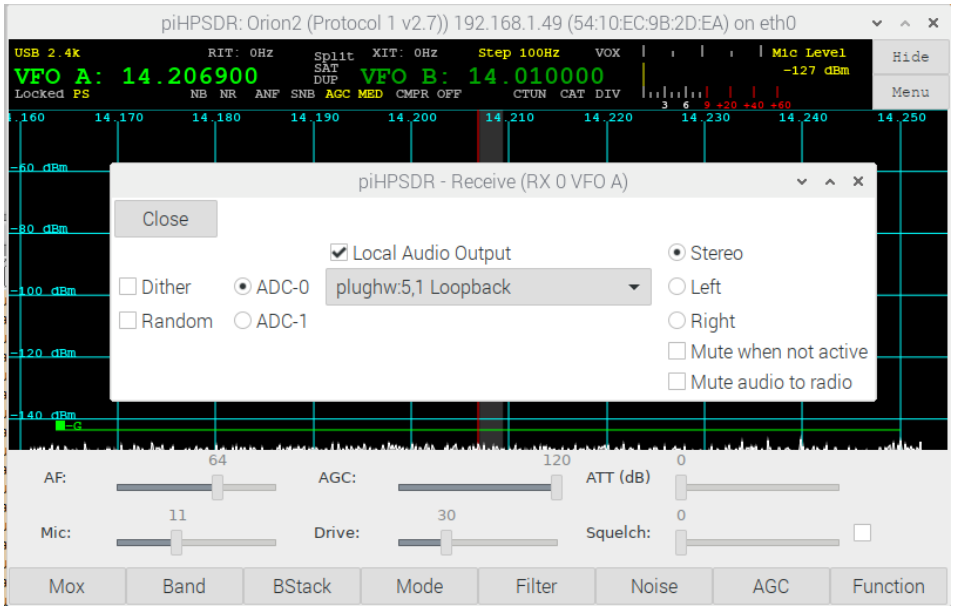
fldigi, wsjtx etc.:
- Audio output (playback) device: "cable #1 input side"
- Audio input (capture) device: "cable #2 output side"

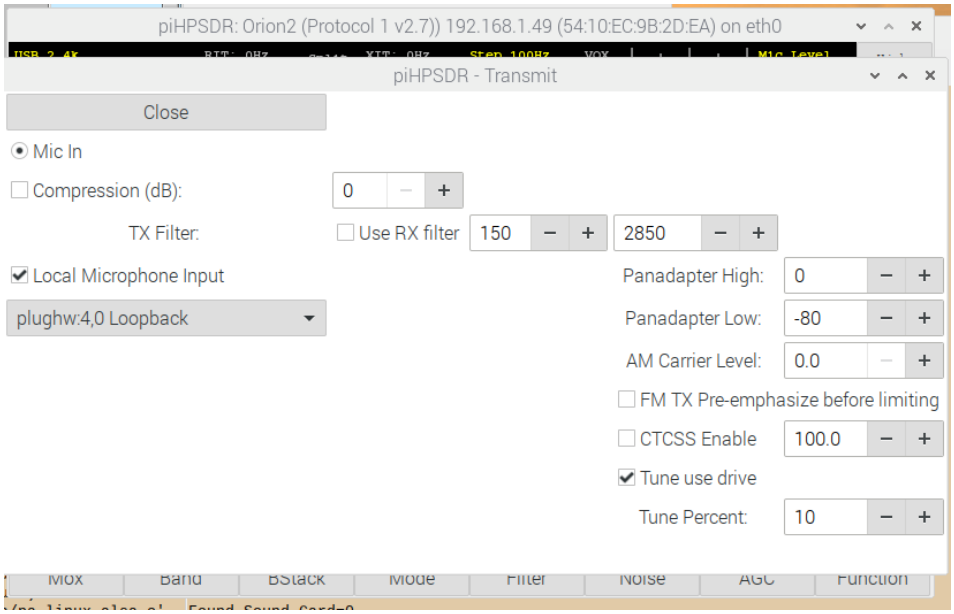
Unfortunately, the actual device names to be used for, e.g. "cable #1 input side" differ between the programs. Here is a list of the various possibilities we have seen, and we indicate the program where we have seen this:

cable#1 input side	Loopback: PCM (hw:4,1) plughw:4,1 Loopback plughw:CARD=vac1,DEV=1	Fldigi piHPSDR WSJT-X
cable#1 output side	Loopback: PCM (hw:4,0) plughw:4,0 Loopback plughw:CARD=vac1,DEV=0	Fldigi piHPSDR WSJT-X
cable#2 input side	Loopback: PCM (hw:5,1) plughw:5,1 Loopback plughw:CARD=vac2,DEV=1	Fldigi piHPSDR WSJT-X
cable#2 output side	Loopback: PCM (hw:5,0) plughw:5,0 Loopback plughw:CARD=vac2,DEV=0	Fldigi piHPSDR WSJT-X

Note that the "input side" of a cable is always connected to a program actually producing sound and thus used as a "headphone device", while the "output side" of a cable is considered as a "microphone device" connected to a program reading audio data. So for any program, use device number 0 where you expect a "microphone or capture device" and device number 1 where you expect a "headphone or playback device". As an example, we provide screen-shots from Fldigi (SoundCard panel) and piHPSDR (RX and TX panels):


Fldigi configuration
Operator
UI
Waterfall
Modems
Rig
Audio
ID
Misc
Web
Autostart
IO
PSM
Devices
Settings
Right channel
Wav
Alerts
OSS
Device:
PortAudio
Capture: Loopback: PCM (hw:5,0)
Playback: Loopback: PCM (hw:4,1)
PulseAudio
Server string:
File I/O only
Device supports full duplex
Restore defaults
Save
Close
USB
CQ
Clear
RsID CQ
TUNE
TX
WF
T/R
RTTY
s/n 275 dB
-3.0
AFC
SQL


piHPSDR: Orion2 (Protocol 1 v2.7)) 192.168.1.49 (54:10:EC:9B:2D:EA) on eth0
VFO A: 14.206900
VFO B: 14.010000
piHPSDR - Receive (RX 0 VFO A)
Close
Local Audio Output
Dither
Random
ADC-0
ADC-1
plughw:5,1 Loopback
Stereo
Left
Right
Mute when not active
Mute audio to radio
AF:
AGC:
ATT (dB)
Mic:
Drive:
Squelch:
Mox
Band
BStack
Mode
Filter
Noise
AGC
Function


piHPSDR: Orion2 (Protocol 1 v2.7)) 192.168.1.49 (54:10:EC:9B:2D:EA) on eth0
piHPSDR - Transmit
Close
Mic In
Compression (dB):
TX Filter:
Use RX filter
Local Microphone Input
plughw:4,0 Loopback
Panadapter High:
Panadapter Low:
AM Carrier Level:
FM TX Pre-emphasize before limiting
CTCSS Enable
Tune use drive
Tune Percent:

Through the "modprobe" command, the virtual sound cards were created once, and this gets lost after a re-boot. You may want to create the "virtual audio cables" automatically when powering on machine. There are different ways to do this.

If the file `/etc/rc.local` exists, it is most easy to automatically execute the modprobe command on startup by inserting a line reading

```
modprobe snd-aloop enable=1,1 index=4,5 id=vac1,vac2 pcm_substreams=2,2
```

at the bottom of the file before the final line reading "`exit 0`". This can be done with the three-step procedure for modifying files that require administrator privileges outlined above.

This worked on my RasPi4. Some Linux variants do not have the file `/etc/rc.local` but a file named `/etc/modules` and a directory named `/etc/modprobe.d` exists. In this case you have to modify the file `/etc/modules` and add a line reading

```
snd-aloop
```

and you have to create (in your home directory) a new file with name loopback.conf containing a single text line reading

```
options snd-aloop enable=1,1 index=4,5 id=vac1,vac2 pcm_substreams=2,2
```

and then copy it to /etc/modprobe.d with the commands

```
sudo cp $HOME/loopback.conf /etc/modprobe.d
```