

Compile piHPSDR & Fldigi, WSJTX, FreeDV from the source-code (Linux) (Desktop PC / Laptop / RaspBerry Pi)

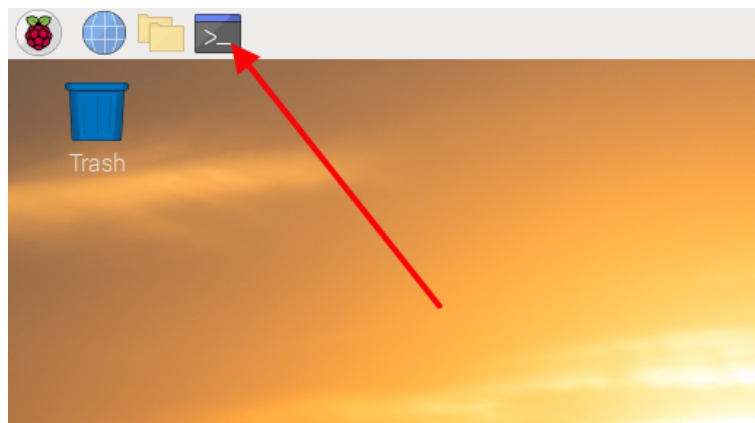
Christoph van Wüllen, DL1YCF
Kaiserslautern, Germany

... and lots of others, including my "guinea pigs" have helped to bring this into shape.

Latest change to this document: May 3, 2023

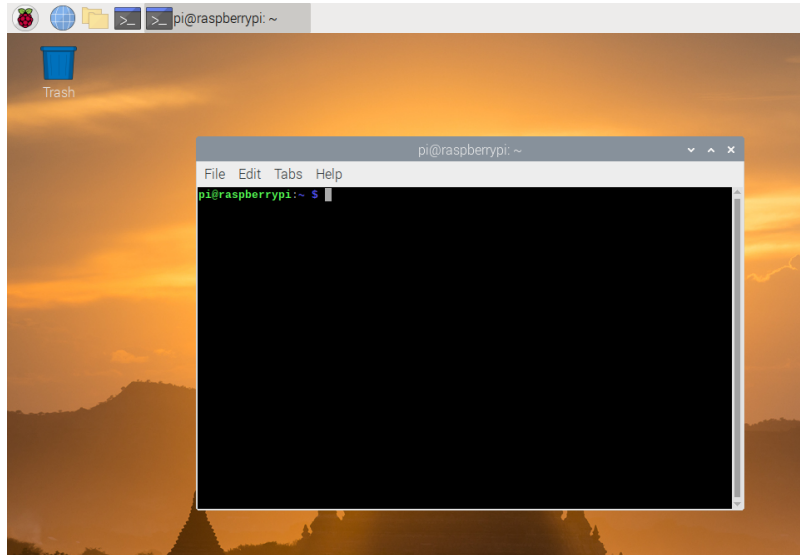
A) Hardware, Software, and Skills required to follow the instructions in this document.

- 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). The single but essential difference between the Raspberry Pi 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. Controllers or Morse Keys to be used with Desktop PCs must use MIDI.
- 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 16 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, the terminal is behind this symbol in the top menu bar (see red arrow) but can also be opened from



the Raspberry menu Accessoires ==> Terminal. A terminal window opens and looks like displayed in the figure below.

Note: The background of the screen-shots shown here may be different in your case. For example, in late 2021 it changed to a picture with water, mountains and a "burning" sky.



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. Fear not, there is very little you have to type in, because all the complicated stuff is done in so-called "script files" which come separately (you should get a file called `scripts.tar` together with this document).

As a result of the command `echo $HOME`, the name of your home directory should be printed on the screen. This is `/home/pi` for the RaspPi and `/home/user` for Desktop/Laptop Linux computers, where "user" is replaced by your Linux user name there.

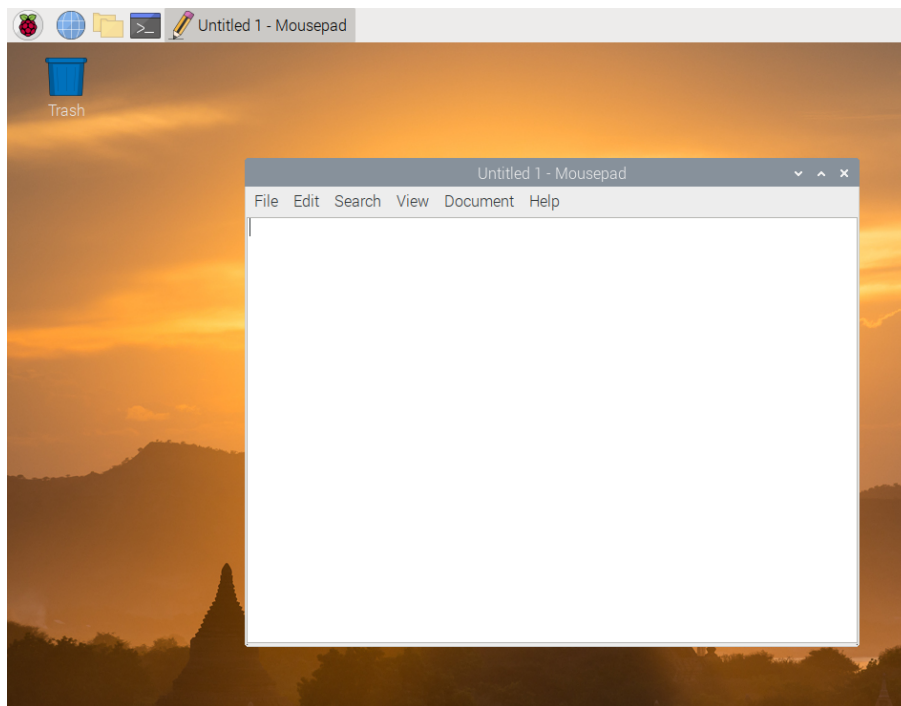
- You most likely have to install additional software components, and this requires administrator privileges. To check if you can execute a command with administrator privileges, type in the command

```
sudo ls -l
```

This should list all the files in your home directory. On the RaspPi, this usually works without further ado. 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 must 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 achieve 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).

- Although no longer required since the "scripts" do everything for you, you should be 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. So 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. 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 color and a mono-spaced font. If you type in the text, do not forget the Enter key after typing in the last line (such that the cursor jumps to the beginning of the fourth 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 MyCuteTextFile) into the line at the top (behind "Name:"). After

entering the name, the "Save" button at the bottom right of the "Save As" window becomes active and must be clicked. Close the text editor window and go back to the terminal window by clicking somewhere in its area and type in the command

```
cat MyCuteTextFile
```

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 a position following 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

```
cat MyCuteTextFile
```

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 local radio amateur meeting and ask for help. What we have done so far is just very basic Linux, if you have difficulties you cannot overcome already at this stage you won't be able to proceed further.

B) Obtain useful scripts to help with installation/compilation

In the next steps we will obtain and install software packages that may or may not be already present on your system and are needed to compile piHPSDR (and possibly other programs). To facilitate this, I have prepared a file [scripts.tar](#) which contains a bunch of so-called "shell scripts" that perform the tasks required. (Note to experts: bundling the scripts in a tar file instead of making them available as separate files circumvents problems with Windows-type end-of-line markers, missing execute permissions, etc.) You should obtain this file along with this document. Both can be downloaded from

<https://github.com/dl1ycf/pihpsdr-compile-from-sources>

but take care to use "raw" download! At the end of this section, it is described how to get this file onto the RaspPi directly from the internet.

The file [scripts.tar](#) must be placed into the home directory of your "pi" account on the RaspPi. A straightforward way to do so is to copy it onto a USB stick which is then inserted into the RPi. So copy the file [scripts.tar](#) to the USB stick on your "main" computer (no matter if it runs Windows, Linux, or MacOS), and then insert the USB stick into the RaspPi. A window pops up where you can choose to open the USB stick in the file manager (do so!). Then, with a mouse drag+drop the file [scripts.tar](#) into the "Home Folder". To check whether this has succeeded, open a terminal window and type in

```
ls -l
```

This should produce a list of files/directories, with `scripts.tar` among them. If this is the case, proceed by typing in the command

```
tar xvf scripts.tar
```

which produces the list of files extracted from `scripts.tar` on the screen, which all end in ".sh". These scripts greatly facilitate what follows, since you need not type in dozens of commands (this is, of course, error-prone) but simply execute a script.

Download these instructions and the scripts file directly onto the RaspPi:

To this end, just type in the commands

```
cd $HOME
git clone https://github.com/dl1ycf/pihpsdr-compile-from-sources
tar xvf pihpsdr-compile-from-sources/scripts.tar
```

This creates a folder `pihpsdr-compile-from-sources` in your home directory, where you not only find the scripts file but also these instructions both in PDF and OpenOffice format. The third command has already extracted the shell scripts from the tar file so you can proceed with the next section.

C) Complete the download of the operating system

In Appendix A, it has been recommended to download only a "small" operating system image to be "burnt" onto the SD card. This has been chosen since we need to load some additional software components anyway, and this is automatically done by the command

```
cd $HOME
./packages.sh
```

This takes some time, depending on the speed of your internet connection and your SD card, since quite a lot of data is downloaded from the internet and placed in your file system on your SD card. With a fast internet connection, this step requires less than 10 minutes.

D) Some post-install modifications of the computer setup

Most of this is only necessary in special situations. If you make any changes described in this section, you should re-boot the system before proceeding with the next section.

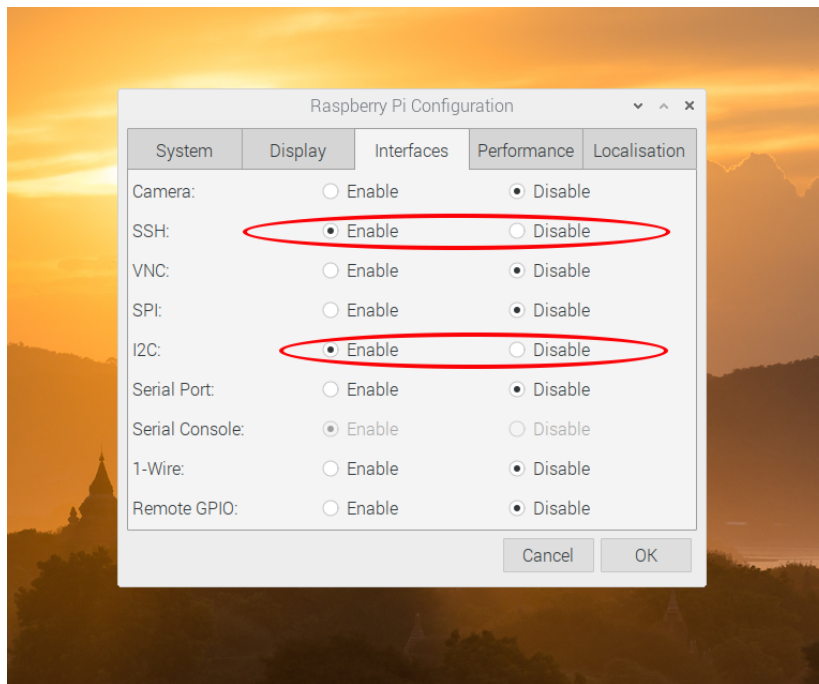
D.1) RaspPi only: configure interfaces

Note: Enabling SSH is only necessary if you want to log in into your RaspPi from another computer or copy files to/from your RaspPi using "scp" or "putty". Enabling I2C is only relevant if you use a new (Version 2) piHPSDR controller. If neither of this applies, you may well skip D.1).

Use the mouse, and chose from the "Raspberry" menu

Raspberry ==> Preferences ==> RaspberryPi Configuration

You can configure dozens of things there (e.g. the keyboard), but for this I refer to the internet. What is important here is the "Interfaces" tab, and if you click there, you get the following window, in which you can enable/disable features by clicking the radio buttons. I suggest to disable all features except (possibly) SSH and I2C. SSH is needed if you want to "log in" on your RaspPi from outside (e.g. from another PC), while I2C is needed if you want to connect the new versions of the "piHPSDR controllers".



D.2) Keyboard settings

Note. This probably only applies to users in Germany who have a German keyboard but nevertheless use the English language on the RaspPi. The problem may also occur in other non-english-speaking countries.

If the keyboard layout does not fit, goto to "Raspberry" ==> "Preferences" ==> "Keyboard and Mouse" and select the "Keyboard" tab. In this tab click "Keyboard Layout..." and select the "German" (or whatever layout you have) layout. The change becomes effective immediately.

D.3) Setting a fixed IP address

Note. This is only necessary if you plan to connect the computer and the radio directly by an ethernet cable, without a router in-between.

Personally, I like connecting a computer running the SDR program and the actual SDR hardware *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 on my RaspPi and 192.168.1.99/24 on my SDR. 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). This is simply performed by the command

```
cd $HOME
./ip.sh
```

This will set 192.168.1.50 as fixed IP address on your RaspPi after the next reboot. You can choose other addresses by editing the file ip.sh, it should be clear how to do. Note that even when using a fixed IP address, the RaspPi will use DHCP and obtain an address from the router if connected. This means, even if you use "direct connection" from computer to radio, you can, from time to time, connect the RaspPi with the router and so some work (e.g. updating the operating system or "pulling" the latest updates to piHPSDR) that requires internet connection.

D.4) Fixing some GPIO problems (RaspPi only)

*Note. This step **must** be skipped if you run piHPSDR on a desktop/laptop, and **can** be skipped if you do not intend to use the RaspPi GPIO lines (that is, you do not intend to connect a piHPSDR controller or CW keys or a PTT switch to the GPIO).*

When the RaspPi4 replaced the RaspPi3, a new Broadcom I/O chip has been introduced and some libraries still have problems to correctly configure the GPIO lines. Therefore a script is provided that lets the RaspPi on each system start configure the GPIO such that GPIO lines 4–27 are programmed as input lines with internal pull-up resistor. To do so, simply use the command

```
cd $HOME
./gpio.sh
```

which puts appropriate instructions into the file /boot/config.txt but remember to do so **only** on a Raspberry Pi.

D.5) Instrumenting the computer for audio connection between piHPSDR and digimode programs (WSJTX, Fldigi, FreeDV)

Note. This can all be skipped if you do not plan to run piHPSDR along with a digimode program (such as WSJTX or Fldigi or FreeDV) on the same computer.

The command is simply

```
cd $HOME
./pulseaudio.sh
```

This creates a file `$HOME/.config/pulse/default.pa` which configures the pulseaudio sound system. Two additional so-called "null-sink" audio output devices with name "SDR-RX" and "SDR-TX" are created which can be used to transport audio data from one application to the other.

Furthermore, the default input and output devices for pulseaudio are set to "SDR-TX" (default output device) and "SDR-RX.monitor" (default input device). This is necessary since both Fldigi and FreeDV can use pulseaudio, but they do not allow to select a device so they automatically use the default input and output device.

Hint. It has been reported that pulseaudio may do odd things if piHPSDR is running for a long time. The cure was to locate the following line in the file named `/etc/pulse/default.pa`

and locate the line

```
load-module module-suspend-on-idle
```

and either delete it or deactivate it by inserting a number sign "#" in the first column. Since the file can only be modified by users with administrator privileges, it is easiest to do so in three steps, namely copying the file to your home directory first, modifying it with your favorite text editor, and then copying back:

```
cat /etc/pulse/default.pa > $HOME/default.pa
<edit file $HOME/default.pa>
sudo cp $HOME/default.pa /etc/pulse/default.pa
```

E) Compiling the programs

Since we have made changes to the operating system in the preceding section, it is a good idea to reboot the system at this point.

E.1) Download "ham radio" software

The "ham radio" software is downloaded by the commands

```
cd $HOME
./hamradio.sh
```


This command essentially fetches all required software from the internet, and then creates icons for piHPSSDR, Fldigi, WSJTX and FreeDV on the desktop, such that you can start these programs (after they have been compiled) just by double-clicking the icon. The amount of data to load is much less than for "packages", so this command should only take few minutes.

ATTENTION: the command "hamradio.sh" deletes all directories before it loads them from the internet. If you have made any modifications (e.g. of Makefiles) and run the "hamradio.sh" script again all your modifications are lost. Normally, use this command only once after setting up the system.

E.2) Configure piHPSSDR

Note: piHPSSDR comes with some options that one can activate or deactivate at compile time. Most users will **not need to anything here and my skip D.2).**

There are only three cases where you have to do something, namely

- a) you run piHPSSDR on a desktop/laptop computer (no GPIO available)
- b) you want to include the SoapySDR module for running piHPSSDR with an AdalmPLUTO radio
- c) you want to run piHPSSDR with RedPitaya-based radios where the SDR application has to be started via a web interface, and you want piHPSSDR to do this for you.

To configure piHPSSDR, you have to modify the file Makefile inside the piHPSSDR directory in your home directory, e.g. using the "Mousepad" editor as described in section A). You have to locate certain lines in the Makefile, change them, and save the Makefile. In the following I will print the line as it stands in the Makefile in blue, and how it must look like in green. The recipe is such, that an option that is active can be deactivated by inserting a number sign in the first column of that line, and an option that is inactive can be activated by deleting the number sign in the first column.

Case a): your computer does not have GPIO input/output lines, or you are using the existing GPIO lines for a different purpose and do not want piHPSSDR to use them. In either case, locate the blue line below and replace it by the green line following

```
GPIO_INCLUDE=GPIO
#GPIO_INCLUDE=GPIO
```

Case b): you want to use SoapySDR radios. Currently, only the AdalmPLUTO is supported. Locate the blue line below and replace it by the green line following

```
#SOAPYSDR_INCLUDE=SOAPYSDR
SOAPYSDR_INCLUDE=SOAPYSDR
```

Case c): you want to use RedPitaya based radios. Locate the blue line below and replace it by the green line following

```
#STEMLAB_DISCOVERY=STEMLAB_DISCOVERY_NOVAHI
STEMLAB_DISCOVERY=STEMLAB_DISCOVERY_NOVAHI
```

Note on the AdalmPluto: according to one user feedback, the Pluto does not run too well with a RaspPi Model 3. I have not verified this (I use a Pi-4 for testing), but a reason might be that the USB ports on the Pi-4 are more powerful. It is also true that the very high sample rate of the Pluto (768k) produces a substantial CPU load. For this you have to know that the RPi3 is **much** slower than the RPi4 if it comes to floating point operations.

E.3) Compile all the programs

This is all done by the command

```
cd $HOME
./compile.sh
```

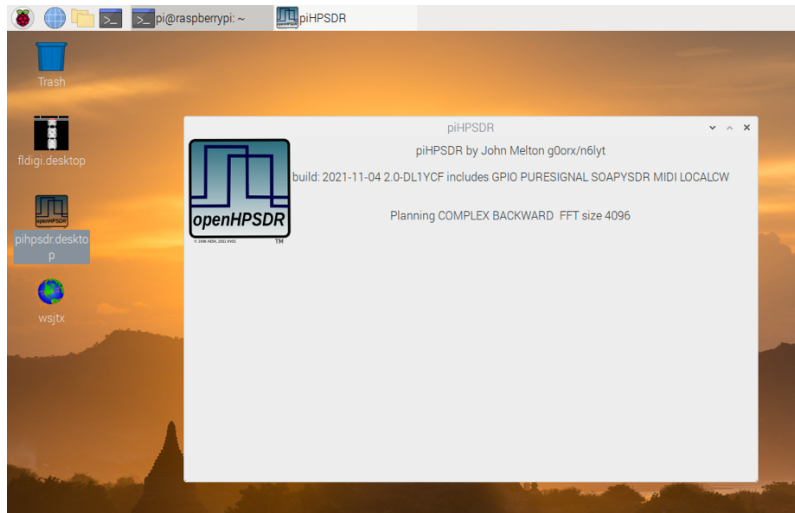
This takes about 50 min on my RaspPi4, a large part thereof is required to compile Fldigi which can only be compiled using a single CPU core due to excessive memory demand for the compilation of one of the source code files. Note everything is put into this script. That is, it not only compiles piHPSDR, Fldigi, WSJTX and the FreeDV program, but also all necessary support libraries including those for SoapySDR, even if they are not needed or wanted. This is to keep things simple.

NOTES (as of May 3, 2023)

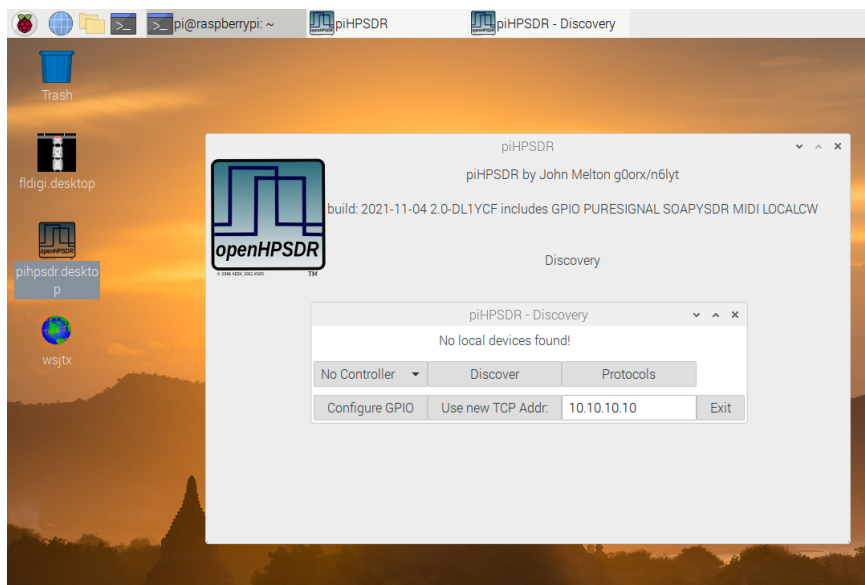
A) the current version of Fldigi requires that the version of the "autoconf" package is at least at version 2.71 (released Jan 2021), however, the RPi OS still sticks to an outdated version 2.69. It is, of course, possible (and not overly complicated) to compile the autoconf package in an up-to-date version from the, but this probably beyond the scope of this document which is targeted at Linux beginners, so for the time being I can only suggest to wait for RPi OS switching to a less outdated autoconf version. Note that autoconf changes 2.69 ==> 2.71 are quite substantial, and it is difficult to maintain software packages that work with both versions, so Fldigi was right in doing the transition, and RaspberryPi-OS is to blame for not offering a current autoconf version in the repository.

F) Initial run of piHPSDR

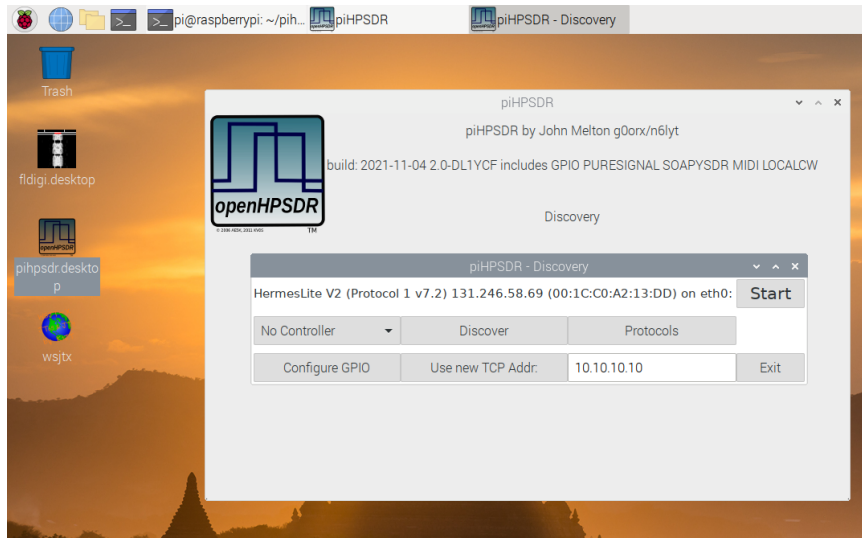
To test the compilation, we make an initial run of piHPSDR without any radios connected to the computer. To do so, just double-click the piHPSDR icon on the Desktop. If a window pops up asking you how the program should be executed click the first tab "Execute" (see section F.2). Then, the piHPSDR window should open and the screen should look like this



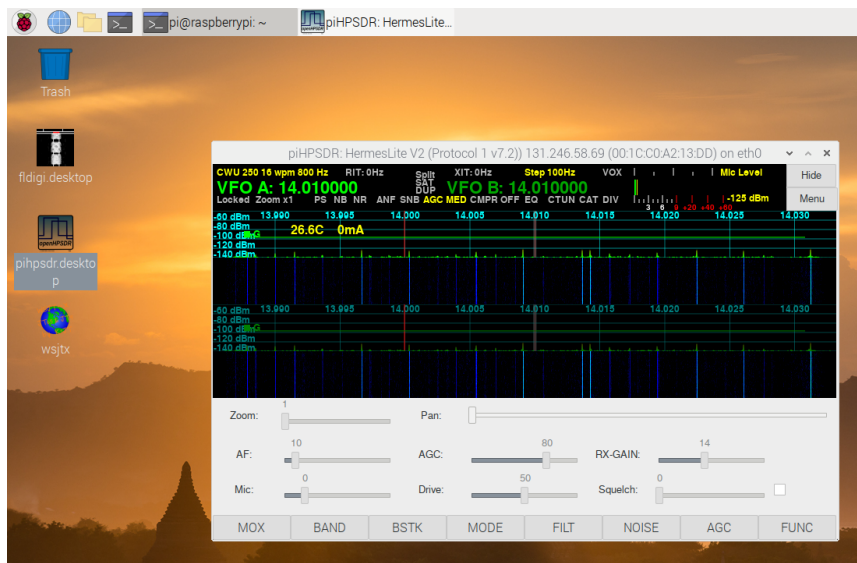
Because it is the first time you started the program, the WDSP library determines (once and for all) the optimum way to do the fast-Fourier-transforms (this will take few minutes, there is a progress report on the screen). After this time, the piHPSDR window looked like this:



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 shown, an HermesLite-II has been connected via Ethernet:

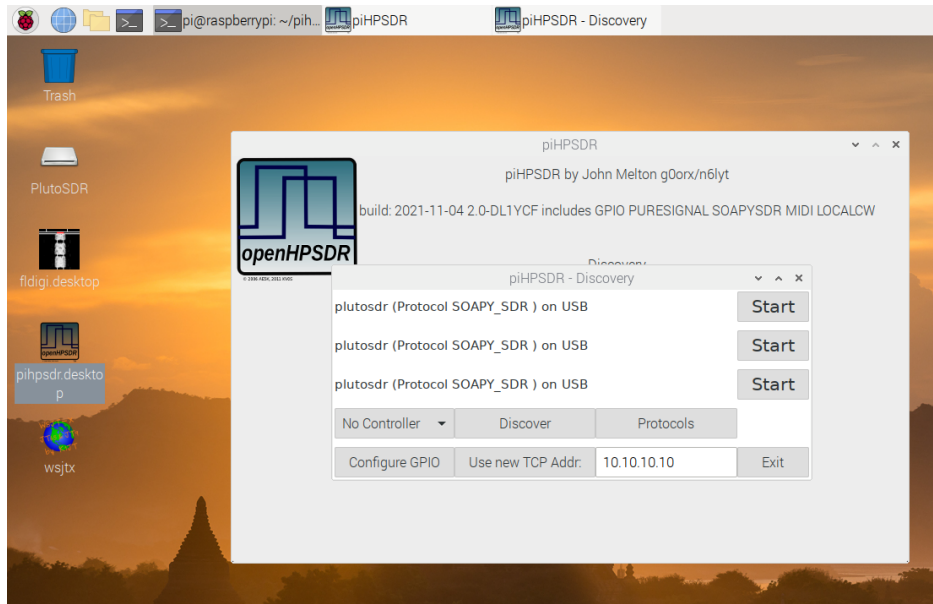


It takes some time to arrive here, because piHPSPDR tries to "discover" HPSPDR protocol1, HPSPDR protocol2, and SoapySDR devices. Using the "Protocols" menu by clicking the tab with that name, one can disable those protocols for which no hardware is present. Clicking the "Start" button then leads to the following

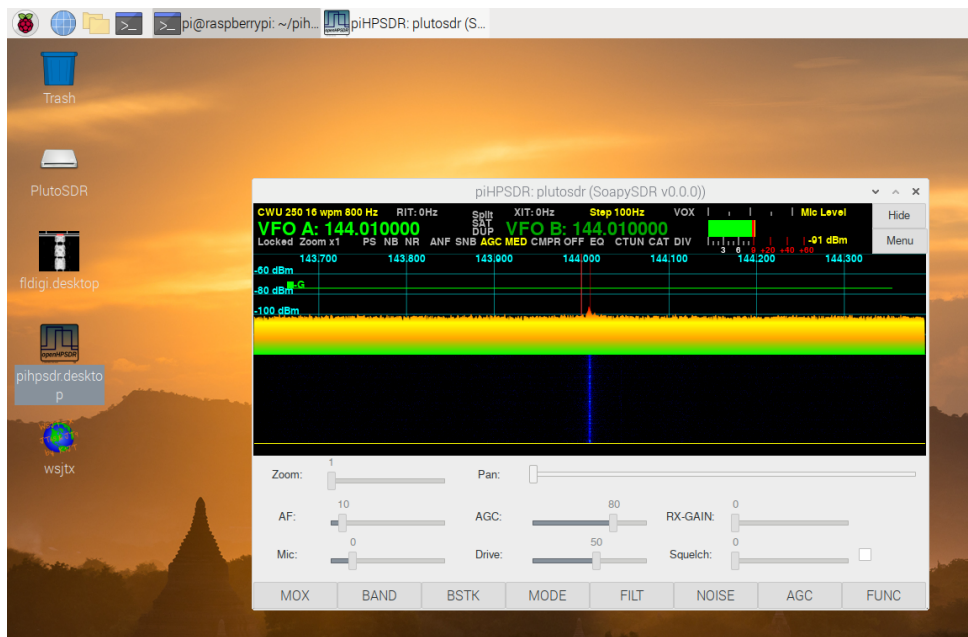


As you can see, the radio starts by default with two receivers, and both receivers have the panadapter and the waterfall on display. This can be configured of course within piHPSPDR and is stored in a local file so the next time you start piHPSPDR, these settings are restored.

To complete the test, the piHPSPDR program was left (through the Menu==>Exit button in the top right corner), the Hermeslite-II disconnected and instead, an AdalmPluto was connected to the RaspPI via an USB cable. Starting piHPSPDR again (by double-clicking its icon) then leads to the following



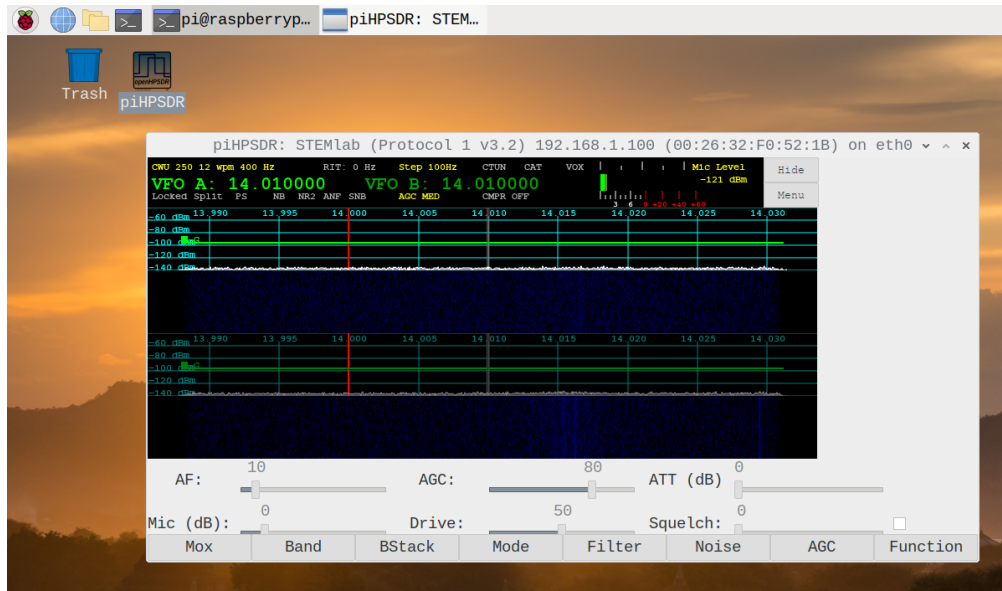
and I cannot say why the same device has been "discovered" three times. This was reported by the Soapy library and is most likely related to how Raspian handles internet interfaces. Anyway, clicking the topmost "Start" button then starts the Pluto radio (I have modified mine such that it can do 144 Mhz)



G) Trouble-shooting some issues that occasionally arise

G.1) Too large font sizes (only RaspPi):

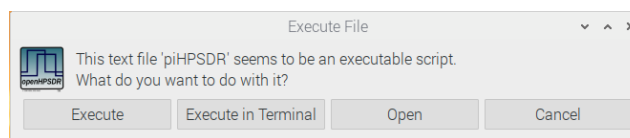
Some RaspPi users have reported that the radio window is messed up and looks like this:



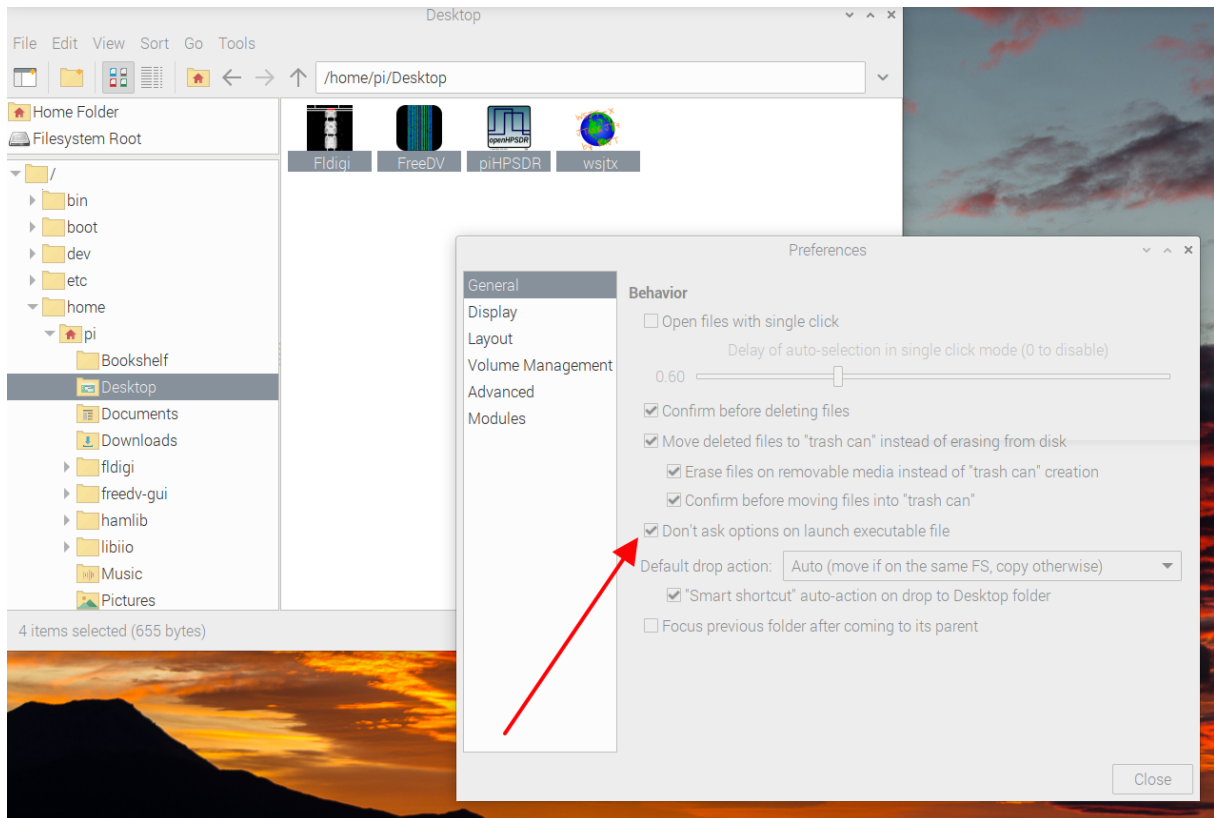
This happens especially when using a large monitor. The reason is, that the system may automatically choose a large font when using a large monitor, which is not reasonable for piHPSPDR since it is using a fixed-size window. This is easily fixed from the Raspberry -> Preferences -> Appearance Settings 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 piHPSPDR window looks OK.

G.2) Desktop icons not working properly

When you double-click the one of the icons on your desktop, it may happen that the following dialogue pops up:



This can be suppressed. Simply invoke the file manager (the icon in the top bar to the right of the browser "earth globe" icon, navigate to the "Desktop" folder in your home directory and select all the desktop entries. Then go to the menu Edit --> Preferences which looks like this:



Simply check the box at the beginning of the line "Don't ask options..." (indicated by the red arrow, this has already been checked in the picture), close the menu and close the file manager. That's it, you have to do this only once. In the (unlikely) case that you have no file manager icon in the top menu bar, you can open a terminal window and enter the command

`pcmanfm`

to start the file manager.

H) Running piHPSPDR along with digimode programs (WSJTX, Fldigi, FreeDV) on the same computer

Note: If your computer has a very small screen (say, less than 1024*768 pixels), then it makes not much sense to run piHPSPDR along with anything, since the screen space is just too small. This is the case if you have the "piHPSPDR controller" or a similar device, where the RaspPi is put into a small box together with switches and knobs and a 7-inch touch screen.

In such a case, you may use the VNC software (search the internet). VNC is built into the RaspPi OS but you need a client for your main (desktop) computer. Then you can add a second (virtual) screen to your RaspPi and display its contents on your main computer. If your controller contains a recent RaspPi, changes are also good that you can connect a second (external HDMI) monitor to it. If you want to do RTTY for example, you also need to connect a keyboard to the RaspPi.

To run piHPSPDR with a digimode program, we need to "connect" piHPSPDR and the digimode program in two ways. The first "connection" is rig control, that is, the digimode

program can change piHPSPDR's VFO frequency and the mode, induce RX/TX and TX/RX transitions in piHPSPDR, and so forth. Then we need "audio transport", that is, RX audio samples (that would normally end up at your headphone) must go to the digimode program, and audio samples created by the digimode application must go to piHPSPDR and treated within piHPSPDR as if they came from a microphone. All necessary ingredients are already there, so we just give screen shots how to adjust things.

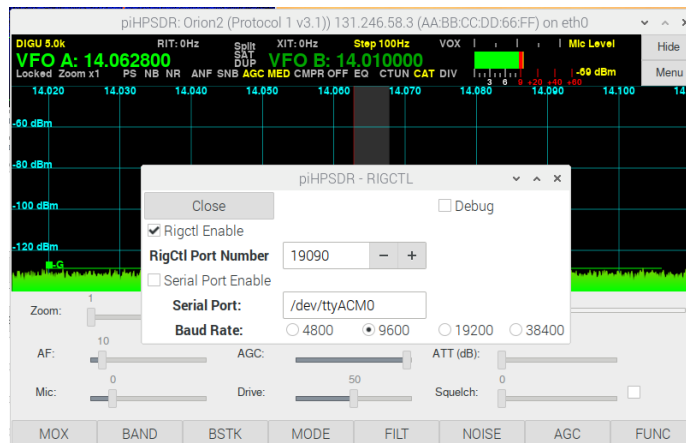
Note: The FreeDV program actually needs **four** audio connections. Two audio connections are used for transporting audio samples between piHPSPDR and FreeDV in both directions, and this is the same as for the other digimode programs (Fldigi, WSJTX). In addition, to operate with FreeDV one also needs a "true" head-phone for your ears and a "true" microphone for your voice. For testing and producing the screen shots, I connected a USB sound adapter named "iMic USB Audio" to the RaspPi, and connected a headphone and a microphone to that adapter.

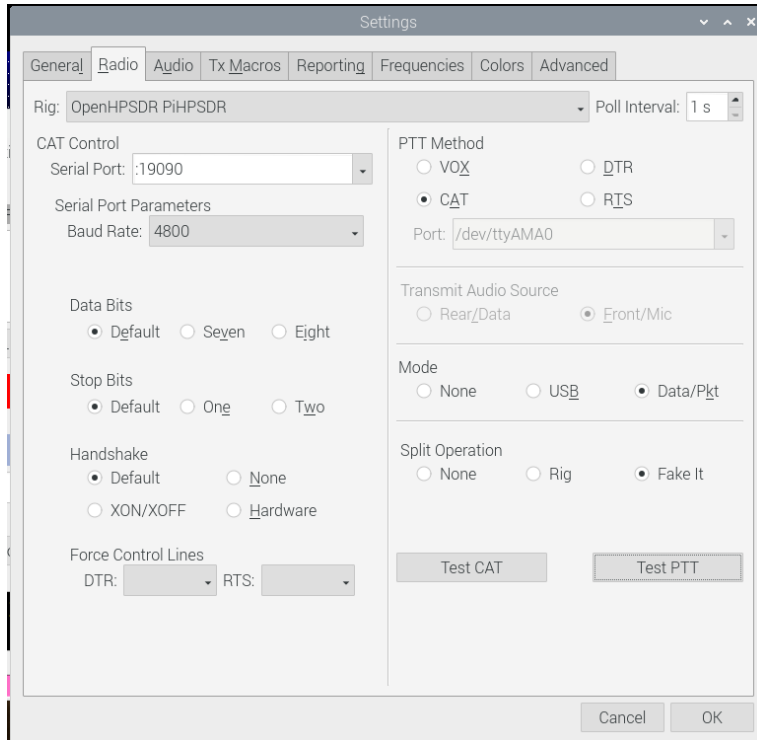
NOTE on headphone/microphone connection: If your (SDR) radio has an audio codec (that is, you can connect headphone and/or microphone to your radio), this is useless when operating FreeDV. You *have* to use a sound card for connecting Headphone/Microphone in any case. Take care this sound card (or this head-set) supports a sample rate of 48000 Hz since FreeDV seems to require this.

NOTE on HDMI audio output: With the latest ("Bullseye") operating system, I was not able to use digital HDMI audio output for the "headphone" connection. Connecting the headphone either to the built-in 3.5mm stereo jack (built-in analog output) or to an external USB sound case was no problem. This seems to be a minor issue since you need such an external sound card anyway for microphone input (HDMI does not provide this).

H.1) Rig Control

In piHPSPDR, click the RIGCTL tab in the main menu and check the "Rigctl Enable" box: Make sure that the "port number" is 19090 since we need this number in the other programs.

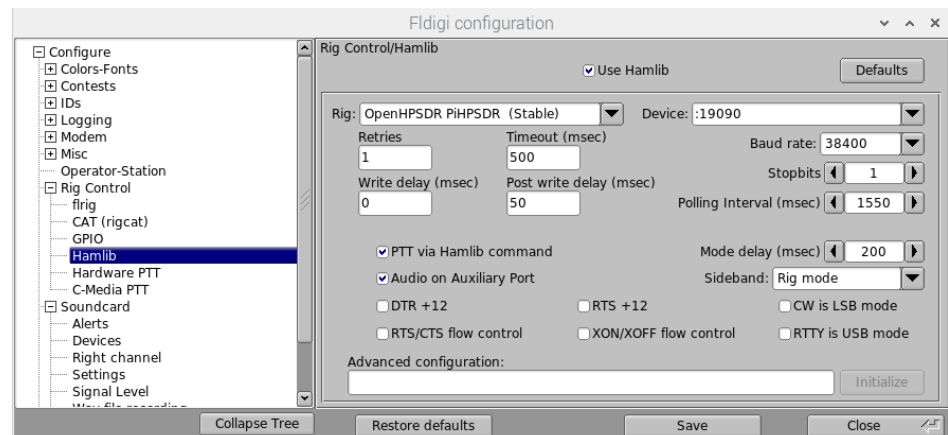




In WSJT-X, go to "File ==> Preferences" and click the "Radio" tab. Choose "OpenHPSDR PiHPSDR" as the rig, enter ":19090" as the serial port (note the colon at the beginning of the string) and enable "CAT" for the PTT method. Choose "Data/Pkt" for the mode (this means that the DIGU mode is chosen in piHPSDR). If you want to TX at audio frequencies below 150 or above 2850 Hz, you also have to choose "Fake It" in the "Split Operation" field.

Note the other fields (Serial port parameters, Data Bits, Stop Bits, Handshake etc. have no meaning when using TCP connection and can be left "as is".

For FLDigi, go to "Configuration ==> Config menu" and expand the collapsed list such that it shows the "Rig Control ==> Hamlib" screen: Again, choose the Rig and the Device (as in



wsjtx and as shown in the figure). Then you must check the box at the top "Use Hamlib" and then can hit the "Initialize" button at the bottom right. Do not forget to "Save" the configuration then you can "Close" the window. When starting Fldigi for the first time, you automatically arrive at the "Hamlib" screen during setup within the Fldigi configuration wizard.

PTT Config

VOX PTT Settings

☐ Left Channel Vox Tone

Hamlib Settings

☒ Use Hamlib PTT

Rig Model: OpenHPSDR PiHPSDR

Serial Device (or hostname:port): :19090

Serial Rate: default

Serial Params:

Serial Port Settings

PTT Port

☐ Use Serial Port PTT

Serial Device:

☐ Use DTR ☒ Use RTS

☐ DTR = +V ☒ RTS = +V

PTT In

☐ Enable PTT Input

Serial Device:

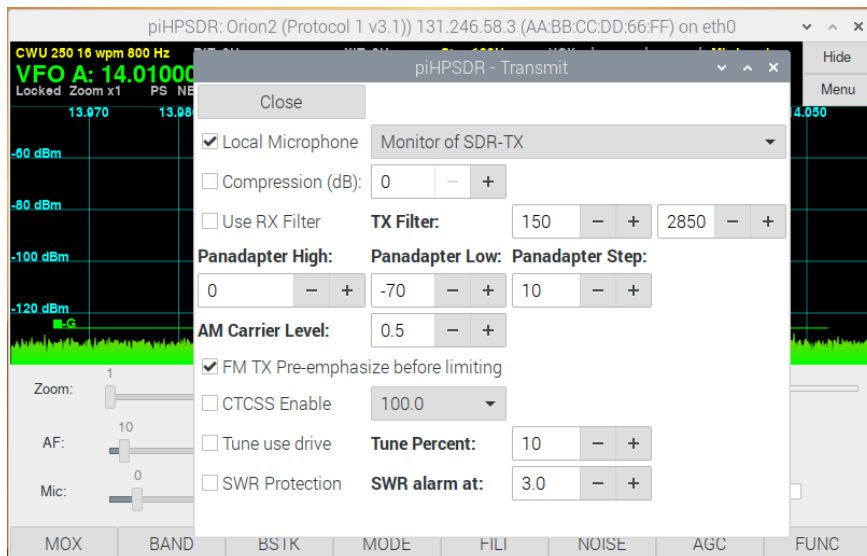
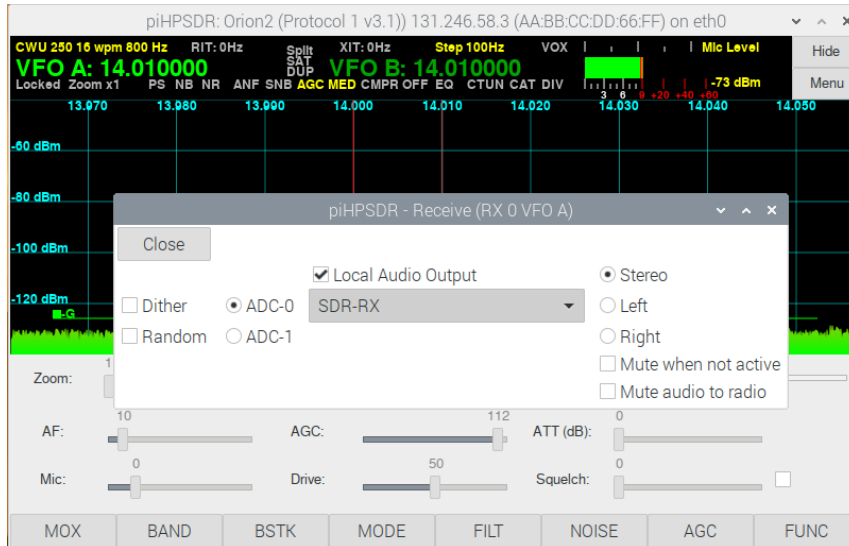
☐ CTS = +V

Test PTT OK Cancel Apply

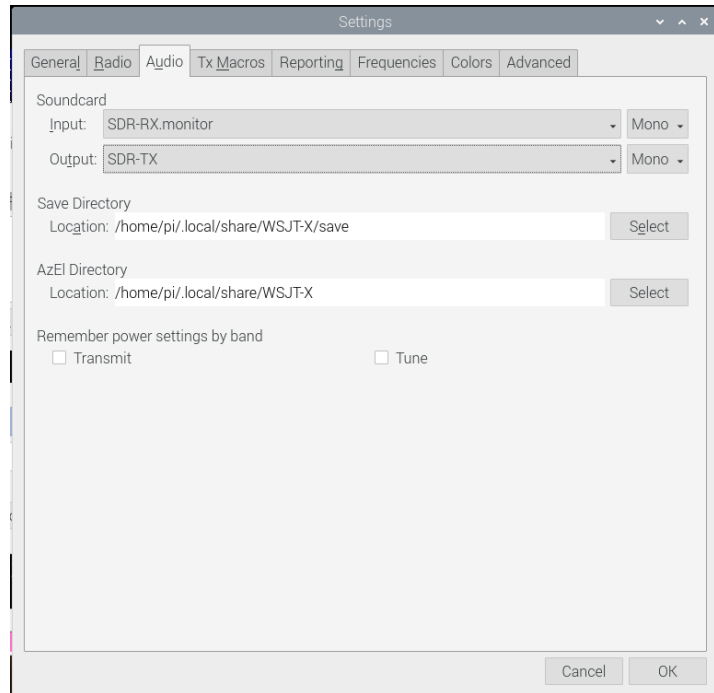
In the FreeDV program, go to "Tools ==> PTT Config". Check the box specifying that Hamlib is used, and choose the Rig model (OpenHPSDR PiHPSDR) and the TCP port (:19090, don't forget the leading colon!). With the button "Test PTT" you can verify that FreeDV can induce a RX/TX transition in piHPDSR.

H.2) Audio Transport

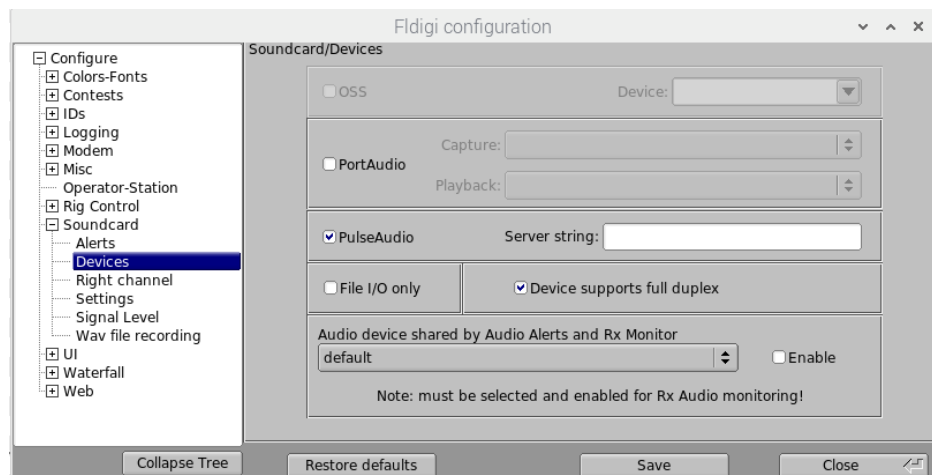
piHPSDR must be instrumented in its main RX and TX menus to use the so-called "null-sink" devices SDR-RX and SDR-TX that were created. Note "SDR-RX" is used in the RX menu, and "SDR-TX" is used in the TX menu. Since we need a sound input device in the TX menu, we must use the "monitor" device associated with SDR-TX.



For wsjtx everything is in the "Audio" tab that we can reach through the File ==> Preferences menu. Here we use "SDR-TX" as the output device since here data is sent to piHPSDR upon TX, while we use the monitor device associated with SDR-RX to capture the RX audio:

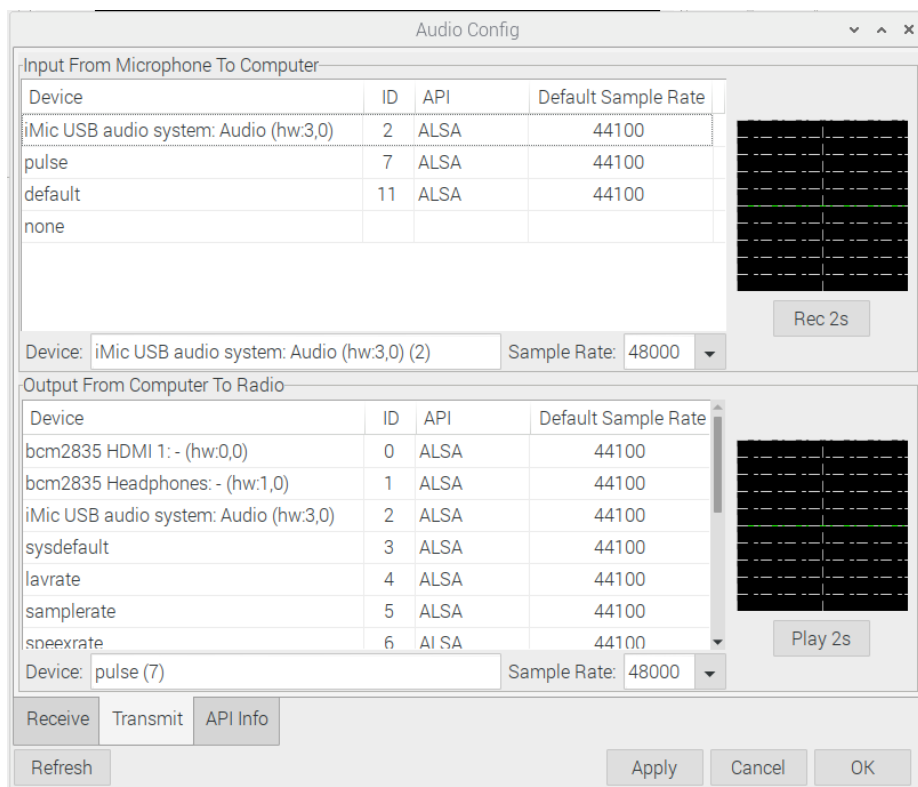
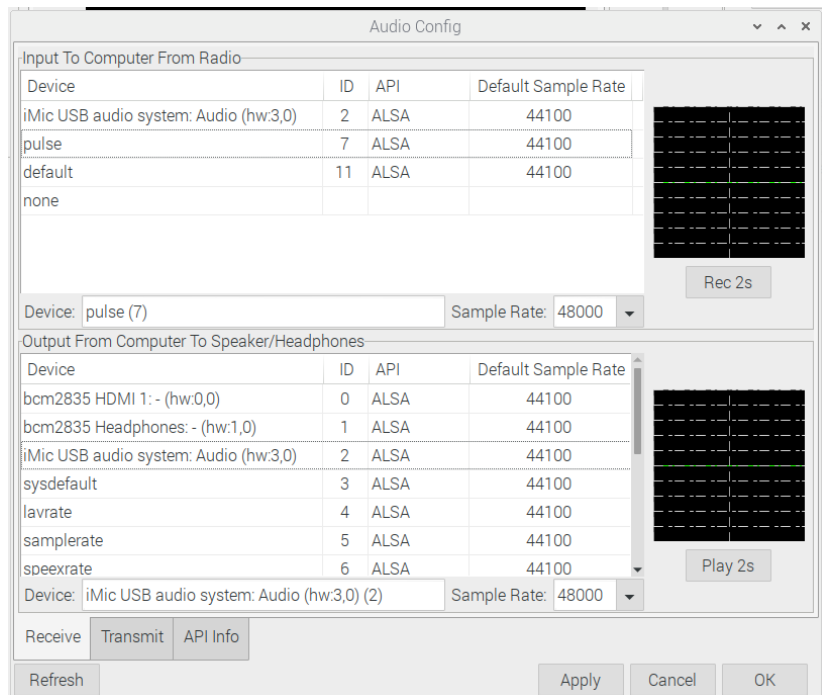


In fldigi, very little is to do. Going to the config menu we have to navigate to the Soundcard ==> Devices screen, check the PulseAudio box and leave the server string empty. Then Save & Close.



When starting Fldigi for the first time, you automatically arrive at the "Soundcard/Devices" screen during setup within the Fldigi configuration wizard.

For FreeDV, you have to go to Tools ==> Audio Config. Four audio devices have to be configured, and be sure to use a 48k sample rate for each of them. **That is, your USB headset or USB sound adapter *must* support the 48000 Hz sample rate!** In the "Receive" tab, choose "pulse" for the "Computer from Radio" section, and the name of your USB sound card ("iMic USB audio system" in my case) for the "Computer to Headphone" section:

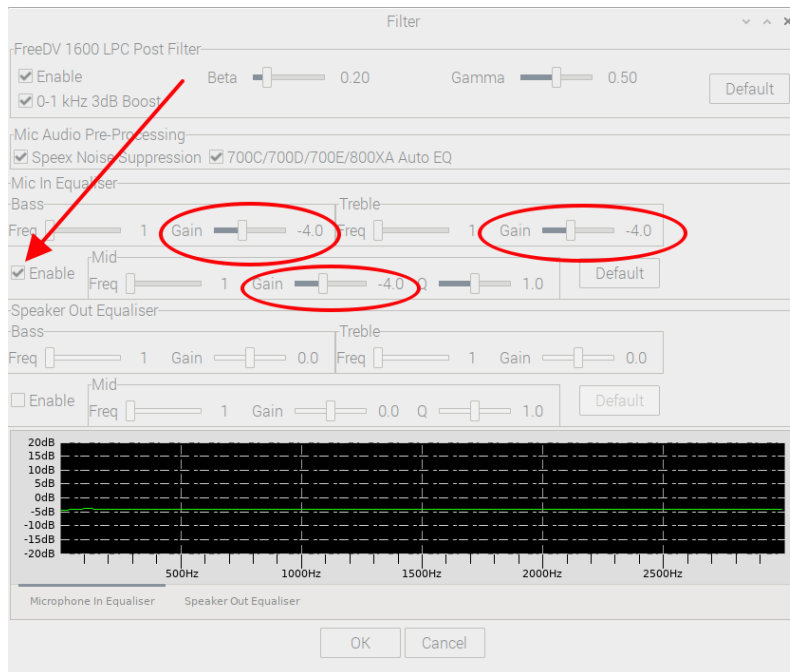
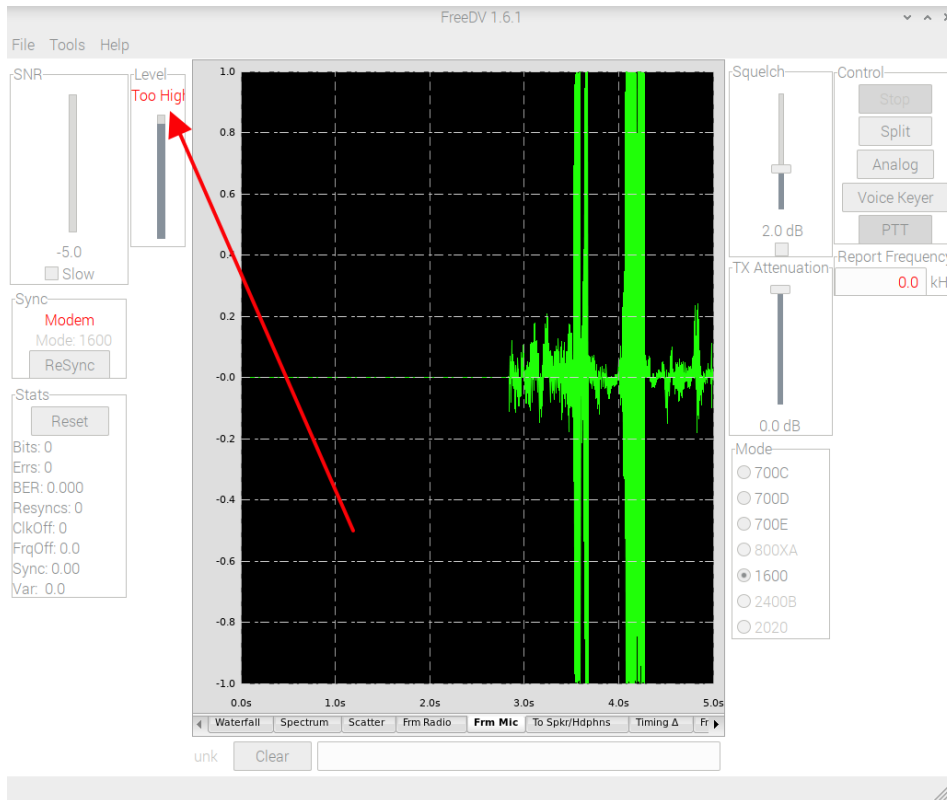


In the "Transmit" tab, choose "pulse" in the "Computer to Radio" section, and the name of your USB sound card in the "Microphone to Computer" section:

Usually the required sample rate can be selected by the drop-down menu. I experienced cases where the "Sample Rate:" tab read "NONE" and no sample rate could be selected. In this case, you have replace the string "NONE" by "48000" just by clicking in and typing into the text field to the right of "Sample Rate:".

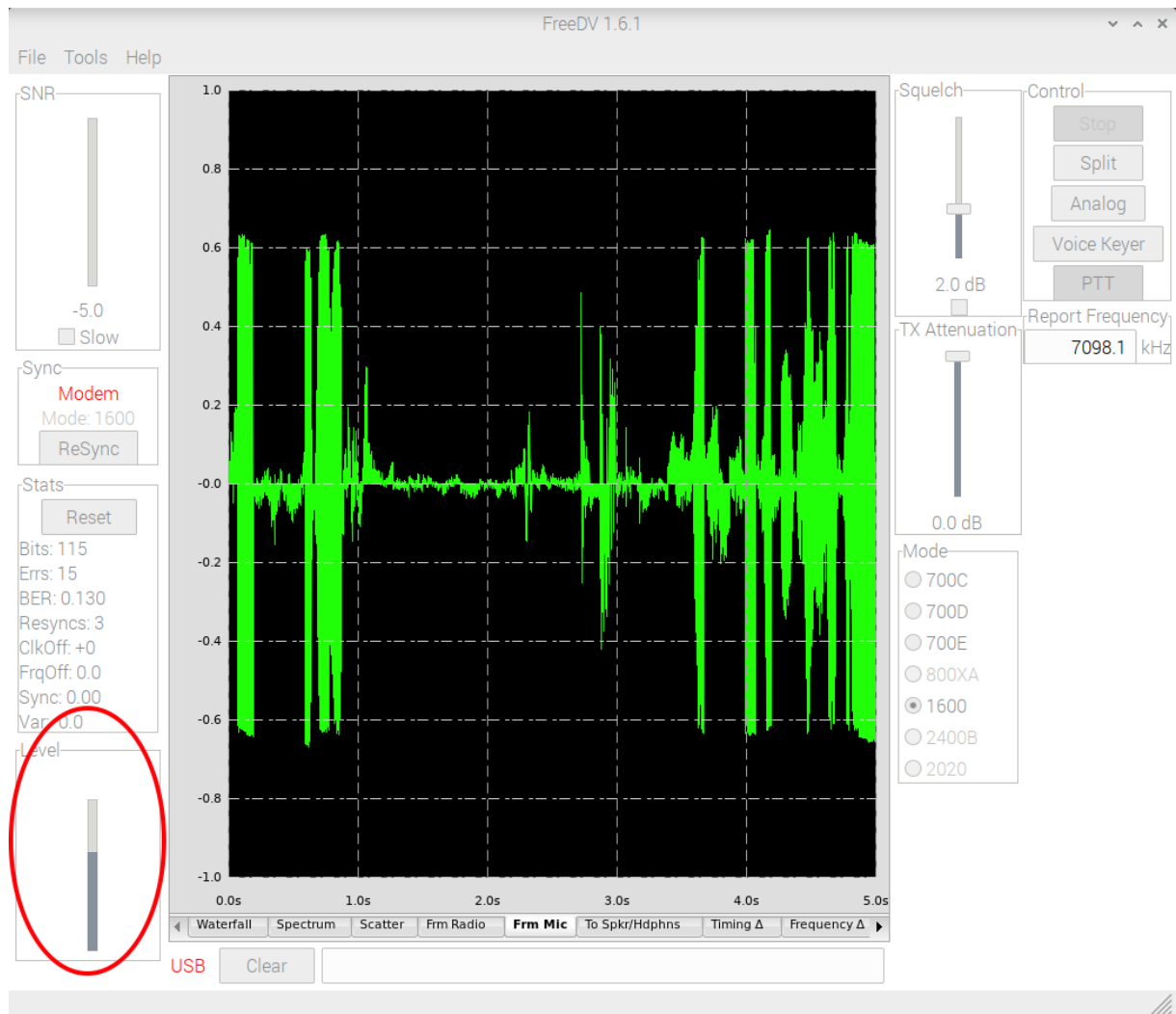
Note that FreeDV is very picky about the microphone level. During my experiment, when whistling into the microphone, the spectrum displayed a mic signal which has

obviously been chopped, and a warning (see red arrow) is on the screen indicating too high microphone level:



To correct this, go to the Tools ==> Filter menu. There you can enable the mic equalizer (red arrow) and choose the levels for "Bass", "Mid", and "Treble" (red circles). For the default center frequencies (all 1 Hz) only the "Treble" slider has much effect, to achieve a frequency-independent attenuation (see green curve at the bottom) set all three sliders to the same value (-4 dB in my case).

With this attenuation, whistling into the microphone both the spectrum looked OK (max. amplitude about 0.6) and the "Too high" indicator has vanished (red circle). Most FreeDV users even recommend lower amplitudes (up to about ± 0.4).



Appendix A: Installing Linux

Step 1: Obtain OS image

RaspPi: An operating system image can be found at the RaspberryPi official web site <https://www.raspberrypi.com/software/operating-systems/>. Scrolling down it shows (as of May 3, 2023)

Raspberry Pi OS (64-bit)

Compatible with:

3B 3B+ 3A+ 4 400

CM3 CM3+ CM4

Zero 2 W

Raspberry Pi OS with desktop

Release date: February 21st 2023
 System: 64-bit
 Kernel version: 5.15
 Debian version: 11 (bullseye)
 Size: 816MB
[Show SHA256 file integrity hash:](#)
[Release notes](#)

Download

[Download torrent](#)
[Archive](#)

Raspberry Pi OS Lite

Release date: February 21st 2023
 System: 64-bit
 Kernel version: 5.15
 Debian version: 11 (bullseye)
 Size: 307MB
[Show SHA256 file integrity hash:](#)
[Release notes](#)

Download

[Download torrent](#)
[Archive](#)

and clicking the "Download" button indicated by the red arrow ("Raspberry Pi OS with desktop) one obtains a "zipped" OS image file. The last time I tried this (May 2023), the zipped file had the name 2023-02-21-raspbios-bullseye-arm64.img.xz (about 1 GByte) and unpacking this file produced a file with 4.4 Gbyte and file name 023-02-21-raspbios-bullseye-arm64.img. **Note that I switched to a 64-bit OS!** While this does not seem necessary, it improves software compatibility in the future. For example, FreeDV does not compile on a 32-bit ARM system but it does on a 64-bit one.

These images change every six months or so, I try to keep the implication thereof "behind the scenes". Some packages have changed name when going from "buster" (before October 2021) to "bullseye" (since then) and I load them both in the script, which means that there will be error messages concerning non-existing packages (don't panic).

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-11.0-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 a USB stick if the PC/laptop supports booting from a USB stick.

Step 2: Install operating system

RaspberryPi: The OS image file already contains the complete OS. It has to be written (or "burned") onto an micro-SD card. The smallest cards you can get nowadays have about 32 GByte, this is more than enough. If you still have some older cards, use a card with at least 8 Gbyte, or else your filesystem will overflow. The steps in this document have all been tested with a 16-GByte card. How to do burn the OS image to the SD card 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, e.g. on the "getting started" page for RaspPi

<https://www.raspberrypi.com/documentation/computers/getting-started>

Note that "burning" can take several minutes, since the I/O speed is at most 10 MB/sec on most cards (this means you need about 7 minutes to write the OS image to the micro-SD card). If you have "burnt" an SD card, it then has to be inserted in the SD-card slot of the RaspPi.

Desktop/laptop computer: Normally one writes the boot image to a USB stick in the same way one "burns" an SD card, but it is also possibly to use a CD/DVD if you are "old style". Then simply boot your desktop PC/laptop off the USB stick, then 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 username and password of the default user. Choose the default "pi" as the use name, since the examples in this document imply this choice (although you might get a security warning for choosing such a user name). The system automatically connects to the internet and updates all installed software to the most recent version. When this is complete, the system must be restarted/rebooted (this option is automatically offered to you, so just click the "Restart" button as soon as it is shown).

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 edited to grant the "normal user" such privileges. One possibility is to add the line

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

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

Step 4: Upgrade operating system

For non-U.S./non-U.K. users, it is usually a good idea to switch to the local keyboard setting as described in section D.2. For example, in Germany it will be difficult to type in the minus sign that occurs in the two commands given below if you have a keyboard with German layout. The "image file" obtained in step 1 is updated on the internet in regular intervals, but normally you would like to run the most recent version of the operating system. To do so, open a terminal window and type in the two commands

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

On the RaspPi this should not be necessary (since this task is normally automatically performed upon first-time boot from the SD card) but it also can do no harm. You will get a bunch of output after either of the two commands, and for this step to work you need internet connection. If you connect the RaspPi and your internet router by a standard Ethernet cable, you should automatically get internet connection (here I assume that your router offers DHCP service, but this you also need for all other computers). Remember that after an OS upgrade, the system should always be rebooted to make the upgrade effective, so after the second command has completed, re-boot the system.

Appendix B: Standard installation in a nutshell

In the guide, we have considered many special cases which are not relevant to the largest part of the users. Therefore, here it comes in a nutshell:

a) Setup your RaspPi, preferably with a "virgin" operating system, as described in Appendix A. Transfer the file `scripts.tar` (e.g. via an USB stick) to the home directory (`/home/pi`) on your RaspPi. The file `scripts.tar` contains several shell scripts (files ending in ".sh", and to extract this file, open a terminal window and type in the commands

```
cd $HOME
tar xvf scripts.tar
```

b) These shell scripts now need to be executed. To this end, open a terminal window on the RaspPi and type in the following commands. As indicated below, the shells script `gpio.sh` should only be executed if you are on a RaspberryPi, and the script `ip.sh` is only needed if you plan to use a direct cable connection between the computer and the radio, without any routers/switches involved. After each command, there may be lots of output lines and some commands take long, but here they are one after the other:

```
cd $HOME
./packages.sh
./ip.sh                                <=== skip this if using a router between
                                       Computer and Radio
./gpio.sh                             <=== skip this if not using a RaspPi
./pulseaudio.sh
./hamradio.sh
./compile.sh
```

This should produce a "run-able" piHPDSR, Fldigi, WSJTX, and FreeDV program that can be started by double-clicking the corresponding icon on the Desktop. The desktop icons should appear almost immediately after you executed the "desktop.sh" script. Compilation takes most of the time (about 50 min on a RaspPi4). For running piHPDSR with WSJTX/Fldigi/FreeDV, see sec. H) how to set up the programs accordingly.