"From scratch" install of piHPSDR on a RaspBerry Pi

In at least one case, the compilation of piHPSDR failed in the very beginning since the `make` command was not found. I looked around and did not find even a "minimal" Raspian image that does not contain core utilities such as `make`, `gcc`, `git`, and `pkg-config`. Although it is certainly possible to install all required packages, the existence of such a "tweaked" system possibly has the reason that the whole system must fit on a small (4 GByte) micro-SD card, so I strongly recommend to remove this card and store it in a safe place and buy a new 16-GByte micro-SD card (costs less than 10 bucks) and install a new system. I acutally did this for documentation purposes.

Here you find a step-by-step protocol how to initialize the card, download the necessary software, compile WDSP and piHPSDR, and finally running it.

Step A: Obtain Raspian Image

From the web page `https://www.raspberrypi.org/downloads/raspbian`, obtain the file `2019-09-26-raspbian-buster.zip` which is deonoted 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.

Step B: "Burn" this image file on-to a micro-SD 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 Windows can be found on the internet. Here I document how I did this on my Macintosh computer, and the procedure should also work for a LINUX machine:

I attached an USB card reader to my Macintosh and inserted a fresh card, then a volume with name „NO NAME“ appeared. With the command `df` (in the terminal window) I obtained a list of all mounted file systems, and the relevant line in the output read

```
/dev/disk3s1      . . . .  100%   /Volumes/NO NAME
```

which shows that the device associated with the SD card is /dev/disk3. Finally, using the commands

```
sudo umount -f "/Volumes/NO NAME"
sudo dd if=2019-09-26-raspbian-buster.img of=/dev/disk3 bs=8192k
```

the volume was un-mounted and the image written to the SD card. Since the IO-speed was about 10 MByte/sec, this took about six minutes.

Step C: First-time boot of the 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.

In the following, it is implied that you open a terminal window to type the LINUX commands shown. The commands are indented and shown in blue.

Step D: Install required software packages

Although this has just been done, one should keep the system up-to-date before installing any *additional* packages, so issue the commands

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

It is a good idea to restart the system at this place, especially if the kernel has been updated. As a result, you have a "plain vanilla virgin" Raspian system with all installed software upgraded to the latest version.

For compiling WDSP, you need the fftw3 library, so issue the command

```
sudo apt-get install libfftw3-dev
```

For compiling piHPSDR, you need tons of additional libraries. Fortunately, installing a component automatically installs are prerequisites, so we need only very few commands:

```
sudo apt-get install libgtk-3-dev
sudo apt-get install libasound2-dev
sudo apt-get install libcurl4-openssl-dev
```

Step E: download, compile and install WDSP

Go to your home directory and dowload WDSP using these commands:

```
cd
git clone https://github.com/dl1ycf/wdsp
```

Then, go to the wdsp directory just created and compile and install:

```
cd wdsp
make -j 4
sudo make install
```

Note: you could likewise obtain the WDSP library from John's github repository at `https://github.com/g0orx/wdsp`, both versions should be identical

Step F: download/adjust/compile piHPSDR

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

```
cd
git clone https://github.com/dl1ycf/pihpsdr
cd pihpsdr
```

The Makefile needs some adjustment before you start compilation. You need a text editor of your choice (I am old-school and always use `vi`). There are many lines that start with a number sign (#) and contain an equal sign (=) thus setting some variables that determine which features are compiled into the 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
```

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: `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. If you do not use RedPitaya-based SDRs, leave the number sign in the first column of the two lines containing the key word `STEMLAB_DISCOVERY`, since this speeds up program start by about 15 seconds (since it is not tried to detect the STEMlab web server).

After having modified the Makefile, the piHPSDR program can be compiled by the command

```
make -j 4
```

(note that the option "`-j 4`" indicates that the compilation can proceed on four CPU cores in parallel – you can omit this option then compilation takes considerably more time).

The program should be built without any errors.

Step G: creating Desktop icon

Normally you will want to have the piHPSDR program as a "click-able" program on the Desktop. To do so, start with the following commands

```
sudo chown root pihpsdr
sudo chmod u+s pihpsdr
```

This makes piHPDR a "setuid root" application. While (surprisingly) this was not necessary on my Raspi B3, it is is, for example, required on my TinkeBoard for programs that have access to the GPIO. Now go to your desktop folder `/home/pi/Desktop` and edit a text file which you name `pi.desktop` with the following content:

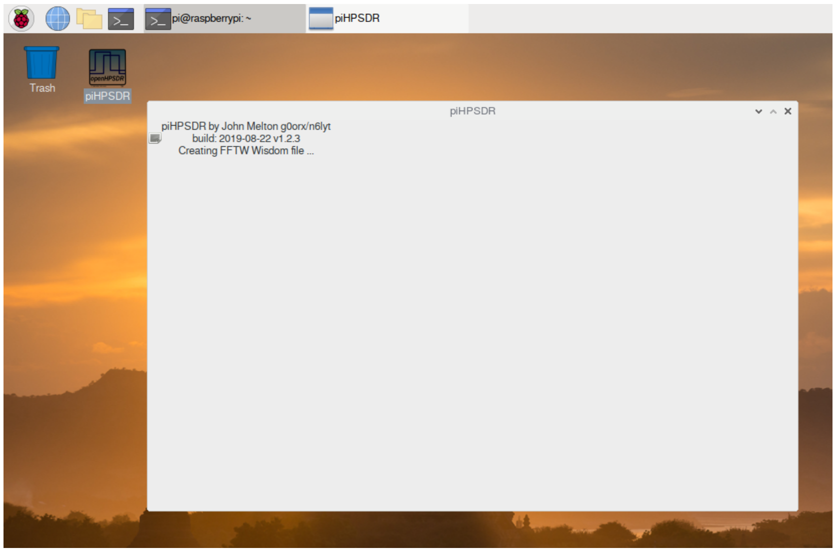
```
[Desktop Entry]
Name=piHPSDR
Icon=/home/pi/pihpsdr/release/pihpsdr/hpsdr.png
Exec=/home/pi/pihpsdr/pihpsdr
Type=Application
Terminal=false
Path=/home/pi/pihpsdr
```

As soon as you have created this file (e.g. using the `vi` editor), an icon with text `piHPSDR` and the HPSDR logo should occur on your Desktop. In order to get rid of being asked each time whether the program should be run in a terminal window, you issue the command (within a terminal window, and in the directory `$HOME/Desktop`) the command `pcmanfm`. In the window that opens, you select the file `pi.desktop` and manoeuver to the `Edit -> Preferences -> General` menu tab, and check the option "`Don't ask options on launch executable file`".

Note: The `Path` option in the file specifies that the working directory of the piHPSDR program is `/home/pi/pihpsdr`, which means that the WDSP wisom file (`wdspWisdom00`) as well as the configuration files (`*.props`) reside in that directory. This has the advantage that if you start piHPSDR from a terminal window from that directory, the widsom and props files can be re-used.

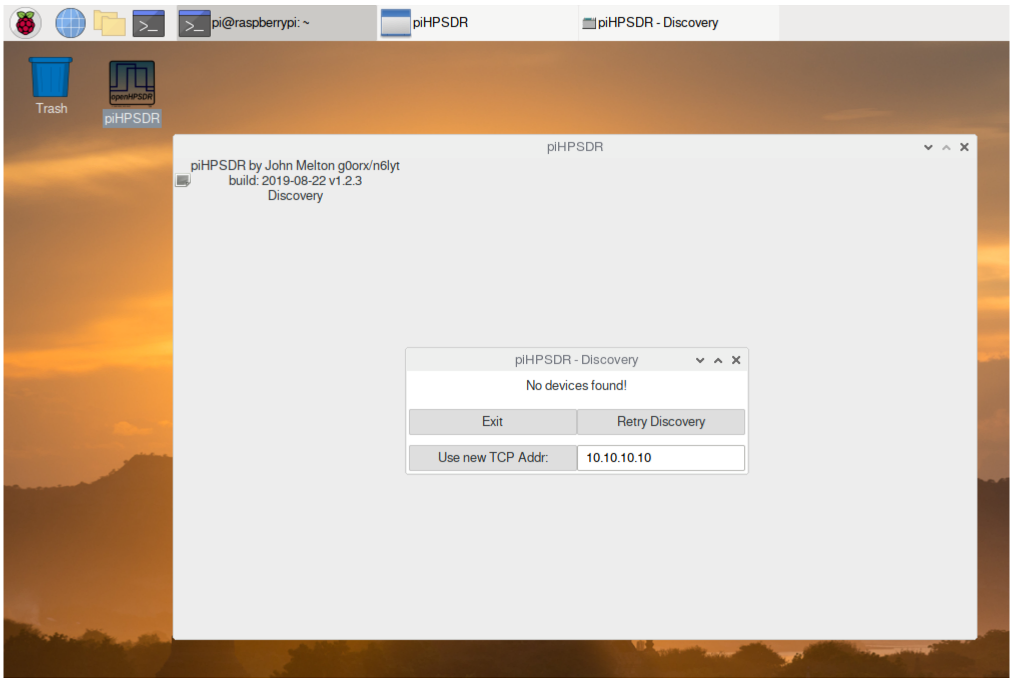
Step H: starting piHPSDR for the first time

Now you can start piHPSDR by double-clicking on the piHPSDR icon on your desktop. The piHPSDR window should open and look like this:



```
piHPSDR by John Melton g0orx/n6lyt
build: 2019-08-22 v1.2.3
Creating FFTW Wisdom file ...
```

Because it is the first time you started the program, the WDSP library determince (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:

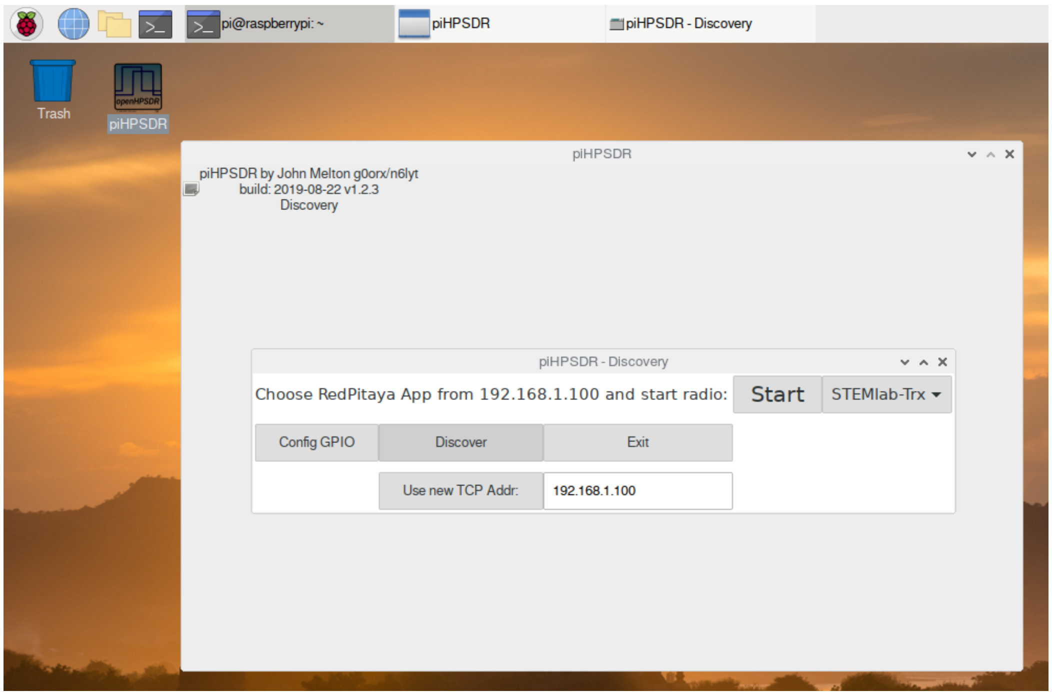


piHPSDR by John Melton g0orx/n6lyt
build: 2019-08-22 v1.2.3
Discovery

No devices found!

10.10.10.10

This is how the screen looks like if no SDR was attached. In my case, I had a RedPitaya based SDR (STEMlab/HAMlab) and for these radios, one must know its IP address. In my setup the (fixed) IP address of the HAMlab is 192.168.1.100 so I over-wrote the field reading "10.10.10.10" with that IP and clicked on "Use new TCP Addr", then the screen changed to

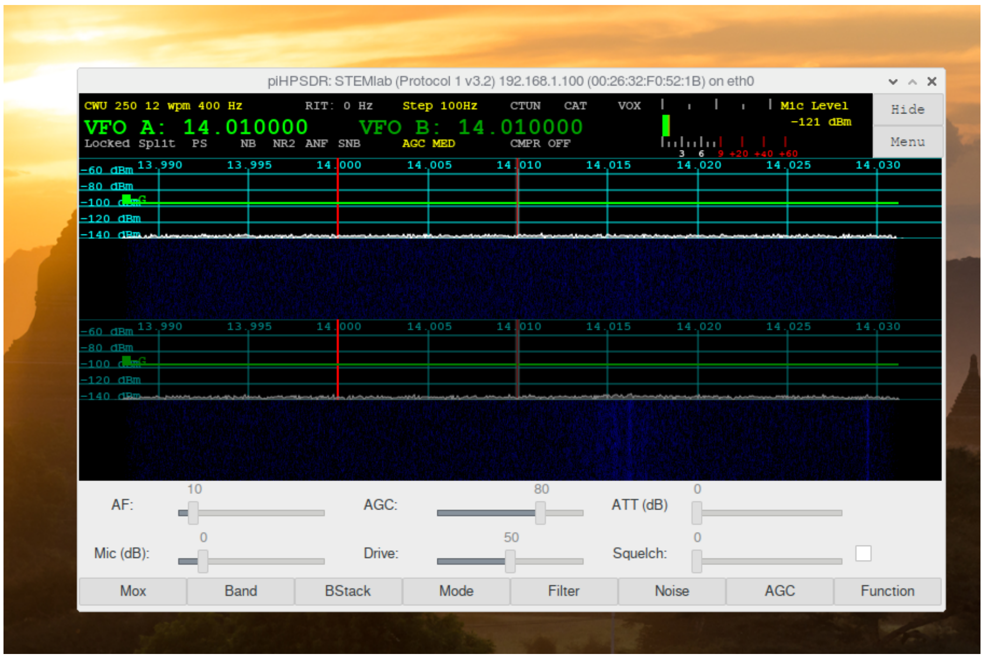


piHPSDR by John Melton g0orx/n6lyt
build: 2019-08-22 v1.2.3
Discovery

Choose RedPitaya App from 192.168.1.100 and start radio:

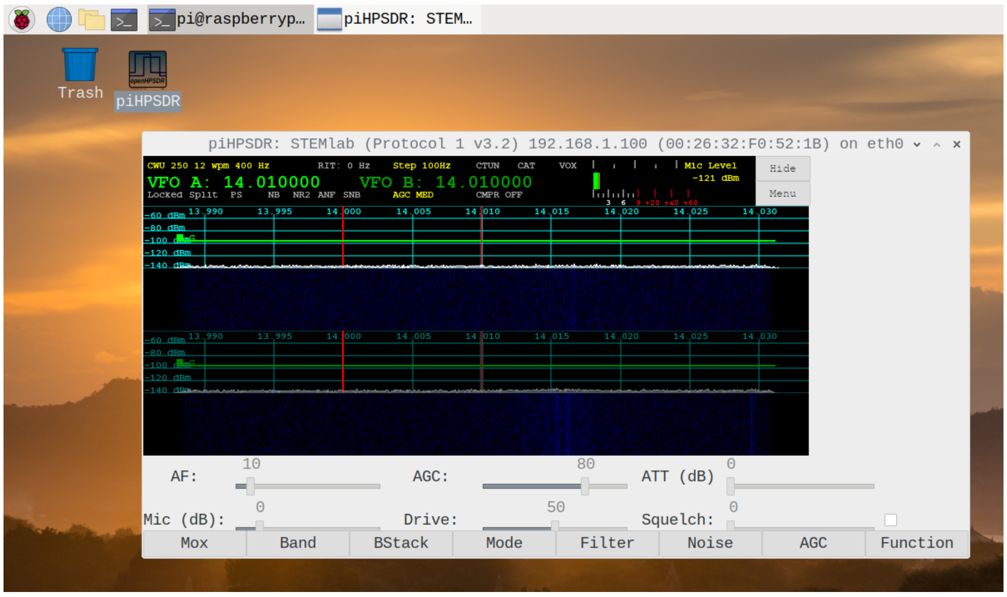
192.168.1.100

and after clicking the "Start" button, the radio finally started:



Some possible trouble with the font sizes:

Some 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.

Step I: setting a fixed IP address for the RaspPi

This step is not necessary as long you have both the RaspPi 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 on the RaspPi, edit the file `/etc/dhcpcd.conf`.

Here look for the line "`# Example static IP` configuration" followed by some lines commented out. Change this section to

```
# Example static IP configuration:
interface eth0
static ip_address=192.168.1.50/24
#static ip6_address=fd51:42f8:caae:d92e::ff/64
static routers=192.168.1.1
static domain_name_servers=192.168.1.1
```

Now the RaspPi does not depend on being connected to a DHCP server when booting, and after each re-boot *always* has the IP address 192.168.1.50.

Note for TinkerBoard users:

The procedure is essentially the same for the TinkerBoard. Of course, you need a different OS image to start with. You do not need to "burn" the image to a micro-SD card since you can directly write to the eMMC on the TinkerBoard (you simply connect the TinkerBoard to your desktop computer via a USB cable). Note further that the default user in TinkerOS is "linaro" and not "pi", therefore the `/home/pi` in all above instructions has to be replaced by `/home/linaro`.