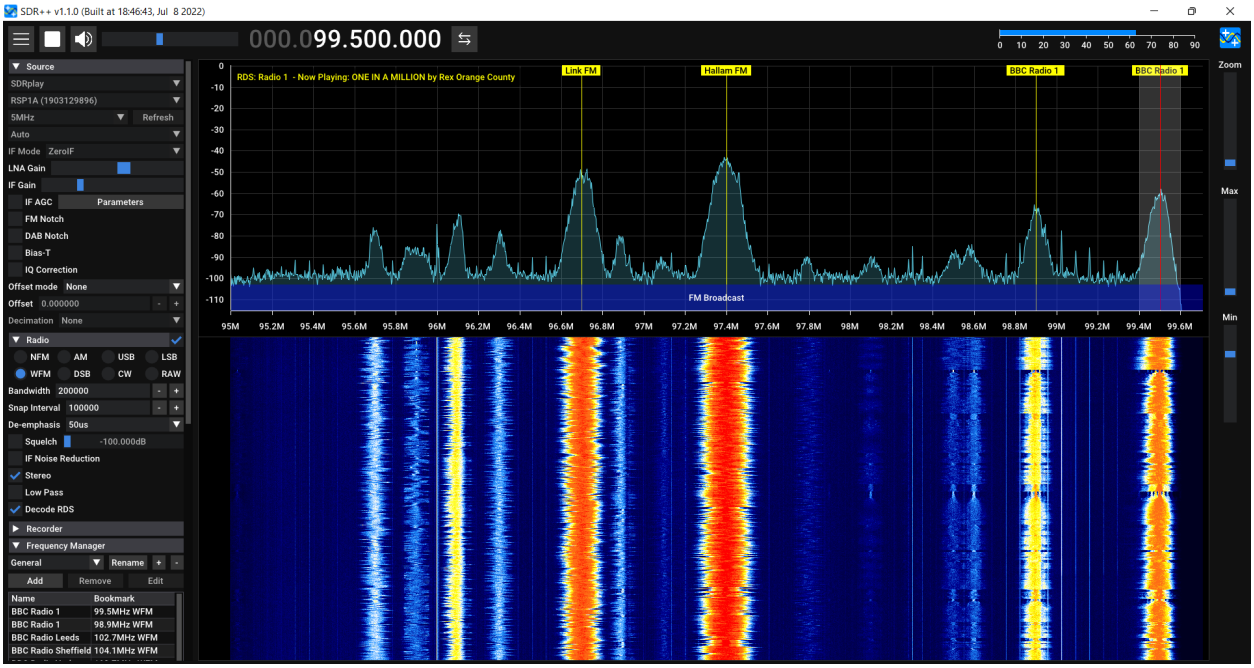
SDR++ USER GUIDE

For SDR++ up to version 1.1. November 2022

Download from https://sdrpp.org



Contents

Click to select links

Introduction

SDR++ is a cross-platform and open source software defined radio (SDR) program that will work with many different SDR receivers. Its aim is to be bloat-free and simple to use.

In this user guide we take the same approach - we'll not only tell you about its features, but also explain what these do and how it can help your SDR listening experience.

You'll also find explanations of some of the more advanced features, so you can move on to those as you get more experienced.

Features

- Wide hardware support
- Cross-platform (Windows, Linux, Mac, BSD - plus Android in beta)
- Wide hardware support including version for the Raspberry Pi
- Full waterfall
- Modular design with community plugins
- Multiple VFO's so you can listen to more than one frequency at the same time - within the receive bandwidth - using a single SDR receiver
- Built in server to operate your SDR receiver remotely
- Range Scanner

Supported SDR receivers

- Adalm Pluto SDR
- Airspy - all models
- BladeRF
- CaribouLite (planned)
- Ettus USRPs (in beta)
- HackRF (see troubleshooting section about removing soapy source)
- Hermes Lite 2 (experimental)
- Lime SDRs
- Per Vices Noctar (planned)

- RFSpace devices (in beta)
- RTL-SDR dongles
- SDRPlay receivers - providing the API is installed and running
- Spectran v6 (Coming soon)

Other receivers supported by SoapySDR, when installed, are supported but may need additional modules added to Soapy and further configuration e.g. the FUNcube dongle pro+.

SDR++ will also connect to remote receivers using SDR++ Server or SpyServer.

You may need to install the correct drivers for your SDR receiver, but once that's done SDR++ should pick it up when you select it as the source.

New in version 1.1.0

- Complete re-write of the digital signal processing (DSP) code for clearer audio, better CPU and memory performance and higher selectivity
- Range scanner - scans a frequency range for active signals (experimental)
- Invert IQ option - when operating SDR++ as a panadapter on radios with IF out connections that require this
- Rigctl client - allows use of SDR++ as panadapter (not yet in nightly releases but may be built)
- Radio Data System (RDS) decoding on broadcast FM signals
- IF noise reduction to pull out weak FM signals
- Improvements to noise blanker
- Ability to adjust more automatic gain control parameters
- FM low pass filter is now selectable, rather than set as ‘on’

Installing SDR++

Hardware requirements for PCs and MACs

If you are installing on a PC, it needs to have a graphics card or chip - and up to date drivers - that support OpenGL 2.1. Without it SDR++ will not run in graphical mode - only as a server.

You can check what version of OpenGL you are running on

- Linux by using the following command in a terminal:

```
glxinfo | grep "OpenGL version"
```

- On MAC and Windows by downloading and installing the free OpenGL Extensions Viewer

The processing performed by SDR++ can be CPU intensive, so some older and less powerful hardware may struggle. You should have no problems, though, with a PC or laptop manufactured in the last 5 years, and SDR++ has been known to run on some machines that are 10 years old. To minimise the load, you can

- close other programs
- reduce the bandwidth you are sampling
- use decimation where available
- turn off other SDR++ features that use the CPU, such as stereo sound, noise reduction or the noise blanker
- run a single VFO only

Downloading

You can download SDR++ from sdrpp.org. You can use the 'release' version, but it's best to use the nightly build - it has the latest features and is just as stable. The latest release may be an earlier version number but this manual covers additional features only found in the nightly release.

If you want to go direct to download SDR++, then you can now download a bang up to date nightly build of SDR++ for any supported operating system, without needing to create a GitHub account. Go to https://github.com/Alexa.../SDRPlusPlus/releases/tag/nightly.

This release is automatically updated every time there's a new nightly build.

Alternatively, you can download the nightly build of SDR++ from the main SDR++ Github page If you have a Github account. Github accounts are free - join Github. On the nightly builds page, scroll down to 'Artifacts' and download the correct file for your operating system. The nightly builds have the latest features, and are usually as stable as the last release.

Windows

Download `sdrpp_windows_x64.zip`

Extract it to the directory of your choice. We'd recommend you create a folder on your main hard drive such as `C:\SDRPP` or one in your user folders.

To create a desktop shortcut, right click on the sdrpp.exe file and

- In Windows 10, select Send to -> Desktop (create shortcut)
- In Windows 11, select 'more options', then Send to -> Desktop (create shortcut),

You can then rename the shortcut on the desktop to whatever you want.

Linux

Debian-based (Ubuntu, Mint, etc)

There are different packages for different versions, so you need to know, for instance, which version of Ubuntu you are running.

You can find this out by opening a terminal and typing:

```
lsb_release -a
```

Extract the compressed file to the directory of your choice.

When you have downloaded the correct release, in a terminal, run:

```
sudo apt install libfftw3-dev libglfw3-dev libglew-dev libvolk2-dev
libsoapysdr-dev libairspyhf-dev libiio-dev libad9361-dev librtaudio-dev
libhackrf-dev zstd
```

If libvolk2-dev is not available, use libvolk1-dev. This installs some dependencies that you might not have.

Then you can install SDR++. You can do this from a terminal (where you'd replace the .deb file with the one you've chosen):

```
sudo dpkg -i sdrpp_debian_amd64.deb
```

This step can be left out if you want to use a package installer to install the program file. For instance you could use Gdebi to install it.

Alternatively, if you have a spare PC or laptop, you can install Dragon OS as its operating system. This is an SDR focused Ubuntu-based distribution that comes with a nightly release of SDR++ already loaded and configured.

Arch-based

Install the latest release from the sdrpp-git AUR package

Raspberry Pi

SDR++ has a package file for the Raspberry Pi operating system 32 bit OS.

It will operate best on the Pi 4 and 400, and satisfactorily on the Pi 3. It is not recommended for earlier models or the Pi Zero.

Download `sdrpp_raspios_bullseye_armhf` from the nightly releases. This contains a '.deb' file that you can extract from the file and install. In a terminal, navigate to the folder where you extracted it, and install it with the command

```
sudo apt install ./sdrpp_debian_armhf.deb
```

This will also create a shortcut to the program in the menu under 'Other'.

Other distributions

There are currently no existing packages for other Linux distributions. For these systems you'll have to build from source.

You may find issues with some distros when self-building. In particular, on Mint based distros there is an audio problem. You can find support for some common issues in the discussions or issues sections of the SDR++ Github pages.

SDR++ will compile and run on the Pine phone although support is experimental at the moment.

MacOS

Some MacOS devices may be able to use the installer downloadable from the Releases page. You should install the zstd library first. To do this, first install Brew and then, in a terminal:

```
brew install zstd
```

For instructions on building SDR++ yourself, go to the SDR++ Github instructions. This is not straightforward, so we would not recommend it except for developers.

Android

For the Android version you need

- **Android 9.0 or later**
- OpenGL 2.1or later - most phones with Android 9 or better should have this or a later version. You can check by installing OpenGL Extensions Viewer from the Play Store. This will also tell you what version of Android you are running.

Currently, only these SDR receivers and protocols are supported under the SDR++ Android app:

Airspy	Airspy HF+	HackRF	PlutoSDR (network only)
RFspace	RTL-SDR	RTL-TCP	SDR++ Server
SpyServer			

Instructions:

1. Download and install the SDR++ Android version. It should have a name like `sdrpp-1.1.0a.apk`. If the file has '.zip' after this then rename it to delete .zip so Android can recognise and install it. It's not yet available from the Play Store. When you select the downloaded file, your device should prompt you to open it with the Package Installer. You may have to change your Android settings to allow installing from this source.

2. Download and install the Android drivers if you have an SDRPlay device. Drivers for RTL-SDR devices are built into SDR++.

3. Connect your SDR receiver before starting SDR++. As most SDR receivers are connected through full size USB connectors, you will need an 'On the Go' (OTG) adapter to connect it. If you use a USB hub with this, you'll get best performance on a powered hub. If you are just connecting to an SDR++ server (or Spyserver) you don't need to do anything.

4. You may need to permit Android to load the drivers on first use. SDR++ should have necessary permissions, including network access, but you can grant or adjust these if your device settings are different.

5. You may be asked to allow SDR++ to access the receiver. If so, approve this and ensure you 'refresh' in the SDR source menu in SDR++. You may need to re-start some drivers or grant access again if you restart your phone or SDR++ crashes.

6. If you are reconnecting your SDR receiver you MUST press 'refresh' in the SDR source before pressing play or SDR++ may crash.

The SDR++ app works in the same way as other versions, in almost all respects. Resizing the menu and waterfall may, of course, take more precision with a finger or stylus on smaller touch screens.

Since phones usually have a high screen resolution, set the High DPI scaling in the 'Display' section of the menu to an appropriate level for your device. You'll need to restart SDR++ after this for the rescale to take effect.

Because you are using a power intensive app, and connecting an SDR receiver which draws power from your phone or Android device, this can draw a lot of power. This will reduce your battery level more quickly than usual.

If you have them, you can connect a powered usb hub and mouse to the OTG cable. You'll get more accurate control of the menu items and controls in this way. Also, battery drain will be less because the SDR receiver will draw power from the hub.

Known issues:

- the user interface has some problems
- there is no easy way to select a path for recording or a file for playback, and importing bookmarks to (and exporting them from) frequency manager is not possible
- the sink on Android may have higher latency
- SDR++ doesn't suspend power saving and, for instance, audio may become choppy if the display is blanked

- menus sometimes close when the app is in the background
- at some menu sizes, the app may crash - if this happens, start SDR++ with the Android device in landscape
- on Samsung devices, the keyboard doesn't always work.

Building it yourself

The SDR++ page on Github has instructions on building SDR++ from source on different platforms. SDR++ has been built on an older Chromebook using the rpi_install.sh script and changing libvolk1-dev to libvolk2-dev.

Installing your SDR receiver and selecting Source

Install your SDR receiver

SDR++ works with a wide range of SDR receivers (often called 'dongles'). If your SDR receiver is supported then, once you've installed the driver, you can simply connect it and select it as the source.

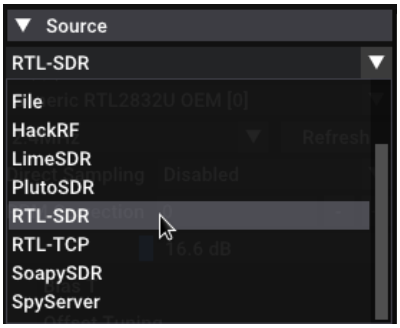
In some cases you may need to take other steps to make your receiver work. Doing this is beyond the scope of this manual, and you should look for help or drivers on the website of the manufacturer. For instance in the Linux operating system, though many receivers are supported without additional drivers, you may need to add modules to SoapySDR for some receivers to be recognised.

Get started with SDR++

Next, you need to launch SDR++. On Android you should have your SDR receiver plugged in before you start SDR++. On other operating systems, if you don't have it plugged in you can press the 'Refresh' button for SDR to detect it. You can launch SDR++ from any shortcut you've created for it, there may be a menu entry (in Linux) or you can launch it directly from the sdrpp.exe program in the folder you created in Windows.

'Source' - Selecting the SDR receiver you use

You'll then have to select the type of SDR receiver you are using from the '**Source**' module tab in the menu. If you can't see the menu, you can toggle it on and off by clicking on the menu icon in the top left. By default this will be on the top left hand side.



First select the make or type of SDR receiver by clicking on the arrow to see the drop down list.

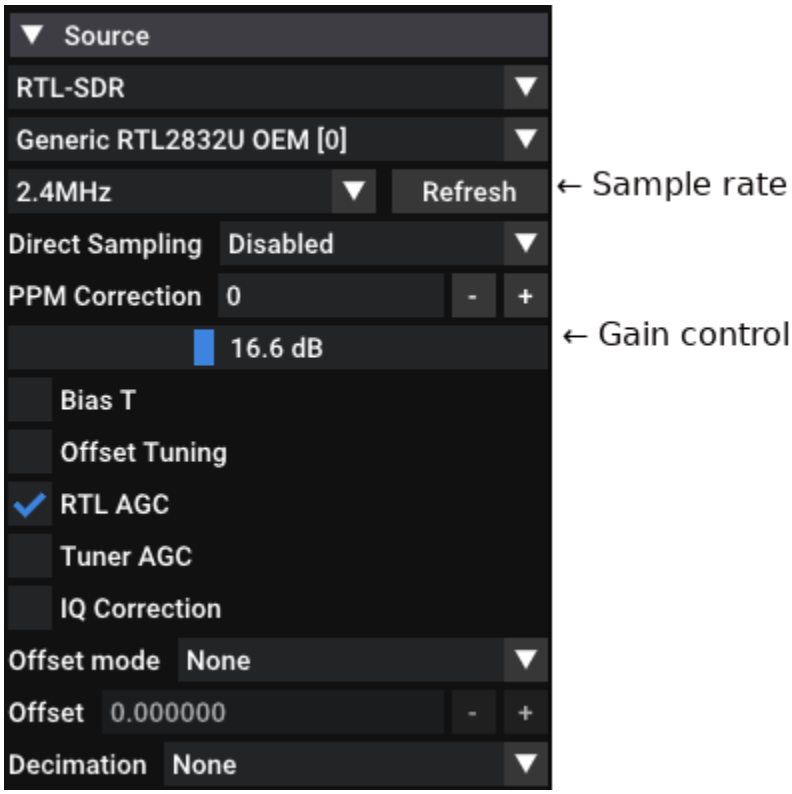
If the SDR receiver is set up properly, you should also see it appear, if it's been detected by SDR++. Look for the model name and/or receiver serial number. If you have more than

one SDR receiver of a particular make then you need to select the one you wish to use. For instance for SDRPlay you might need to choose between an RSP1A and RSPduo or between two RSP1As.

In most cases that should be enough to get the SDR receiver working. You can now use the play button to start receiving. On its first run, SDR++ should open in the FM broadcast band. Adjust the gain - if your receiver has a manual gain control (see below) - and get listening!

There's more information on using SDR++ to connect to an SDR++ server, to an Airspy SpyServer, or to one on the SpyServer network in the Advanced Features section.

The source menu



The options in this menu will vary according to which SDR receiver you use.

Source options

To adjust some of these options you may need to stop SDR++ from receiving. These options will be 'greyed out' whilst receiving.

Gain control (or LNA Gain)

SDR receivers have different ways of controlling the 'gain' applied to a signal. Some will have a manual control, others will have an automatic gain control (AGC -see below), and others may have both. If you have AGC switched on, you won't be able to adjust the manual gain.

With manual gain, you need to adjust the 'gain control' to get the right signal level. This adjusts the amount by which the signal is boosted and shows a reading in decibels - dB. Drag the blue bar to the right to increase the gain, and to the left to reduce gain.

If you tune to a known clear signal on the FM broadcast band, you should make a rough adjustment to the gain control to get a hiss free signal. Sometimes, surprisingly, reducing the gain may bring up a signal and make it clearer.

When you have listened to more signals you should adjust it again. Using a weak signal, adjust the gain to get the best 'signal to noise ratio' - where there is the biggest difference between the signal you want and the background hiss or other noise.

Some SDR receivers will also have an IF gain control which adjusts the gain at a different stage of the receive circuitry.

If SDR++ is working then you might want to skip ahead to the section on using SDR++ and the main panel. The rest of this section looks at the settings you can adjust to get the best from your SDR receiver. It includes some that you'll only see if you've got a particular model of SDR receiver.

AGC ("automatic gain control")

Whilst your gain control adjusts the level of radio signal (plus noise) coming into your SDR, the AGC automatically adjusts this to make weak signals stronger and to reduce strong signals so the overall volume is comfortable.

Depending on your SDR receiver you may have additional options to either choose between AGC built into the hardware or to specify the level. Some SDR receivers will have fixed levels of AGC, for instance: off, low and high on the Airspy HF+ Discovery.

Sample rate

The 'sample rate' determines how much of the radio frequency spectrum your SDR receiver can show at the same time on the main screen. This is limited by your SDR receiver's hardware.

For instance, the SDRplay RSP1A can show signals across up to 10 MHz on the spectrum display, whilst RTL-SDR dongles can manage 2 MHz, and the Airspy HF+ Discovery can show just 768kHz. The useful amount of the radio spectrum may be less than this - for instance about 610kHz for the HF+ Discovery.

A lower figure is not necessarily bad - with the HF+ Discovery it means that the hardware can reject strong singles on a nearby frequency that could swamp weak singles within the sample width.

It also takes a more powerful computer to process a higher sample rate. If you reduce the sample rate it can help a less powerful computer cope without interruption or glitches. Not all sample rates shown may be supported by the SDR receiver.

Bias-T

A Bias-T can generate an electric current that can travel down coaxial cable to power a remote RF amplifier, without interfering with the radio signal travelling the other way. These are available as a separately powered unit - such as with the MLA30+ magnetic loop - but some SDR receivers have their own Bias-T built in. Examples are the SDRplay range, BladeRF and the RTL-SDR Blog v3.

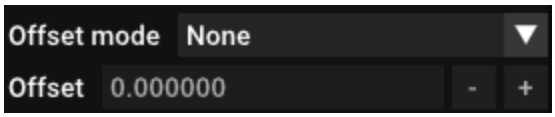
With those receivers you can tick the box in SDR++ to switch on their Bias-T.

You should refer to the manual and specifications of your SDR receiver to ensure that their internal Bias-T is compatible and has sufficient power for the attached amplifier. Incorrect use may damage your SDR receiver.

Offset tuning for up-converters and down-converters

You can have additional hardware connected to your SDR receiver that shifts a band of frequencies into the range of frequencies the SDR receiver can receive.

Select and unselect the 'Offset Tuning' box to activate or deactivate this once you have set the correct offset mode.



For instance, if your SDR receiver only covers VHF (30-300 MHz) and UHF frequencies (300 MHz upwards) and you want to listen to HF (3-30 MHz) or MF (300KHz to 3 MHz) frequencies, then you might use the Ham it Up converter. This is an 'up converter' and delivers signals raised in frequency so your SDR receiver can listen to them within its normal frequency coverage. It raises the signals up by 125 MHz so you'd hear, for instance, 7MHz by tuning to 132MHz in SDR++.

Of course you want to see the true frequency of the signal displayed in SDR++ so it actually displays (in the example given) 7MHz in the frequency readout. You can do this by selecting the type of converter in '**Offset Mode**' which covers:

- Spyverter
- Ham it Up
- DK5AV X band converter
- Ku band satellite LNA's (down converters)

This will set the right offset for those converters.

If you are not using a converter, the offset mode should be unticked or set to 'None'.

You can also set a custom offset. The figure in **'Offset'** can be set or adjusted by the *minus* '-' and *plus* '+' signs, or by typing in the offset figure in hertz. So '125000000' is for a 125 MHz *up*converter. Precede this figure with a *minus* if you are using a *down* converter.

If the frequency reading is still not correct, use a transmission on a known frequency to adjust it until it's correct. You could use a time signal like WWV or a strong VOLMET signal, or compare it with another receiver. On some less stable SDR receivers you may also need to correct the frequency using the PPM Correction setting.

IQ Correction

This setting will correct imbalances in the radio hardware that can lead to effects such as a 'DC spike' showing on the frequency display, and imbalances that can particularly affect the audio or the reception of data modes. Adjust the value until the spike or imbalance disappears or moves outside the VFO's passband.

Decimation

Decimation is a way of reducing the number of times a sample is taken of the radio signals coming into your SDR receiver. A setting of 2 means that 1 in every 2 samples will be taken, 4 means 1 in 4 and so on. To start with, leave it at zero.

Introducing decimation can increase the 'dynamic range' of the SDR receiver - the range of signal strengths over which it can successfully receive the required signals. Try different settings with your receiver and watch out for it stuttering.

Decimation reduces the frequency range sampled. For instance changing decimation to 4 on an SDRplay RSP1A reduces the frequency width you can view from 10 MHz to approximately 1.2 MHz. Increasing decimation may also reduce the quality of the received signal.

Source settings for specific SDR receivers

There are settings options which are unique, or significantly different, for various supported SDR receivers.

The source settings for SDR++ also covers

- listening to recordings of the RF Spectrum (also called the IQ signal): see the part of this guide on the 'Recorder' to learn more

- connecting to an SDR++ server at a remote location: see the part of this guide on 'SDR++ Server'
- connecting to an Airspy remote server: see the part of this guide on 'Connecting to remote SDRs using SpyServer'.

RTL-SDR only source settings

Direct sampling

RTL-SDR's mostly cover VHF and UHF frequencies. Some RTL-SDR (and other) dongles can go to lower frequencies by using 'direct sampling'. This works but you can expect:

- poorer performance than on it's design frequencies
- poorer performance than an up converter like Ham it Up
- less resistance to strong signals on nearby frequencies
- radio transmissions - 'images' - appearing at an apparently different frequency from their actual frequency. For instance on the RTL-SDR Blog v3, you might hear a USB signal below 14 MHz appearing above 14 MHz, and in LSB.

To receive the lower frequencies - usually 500 kHz – 24 MHz - the RTL-SDR dongle must use direct sampling. Use the Source menu setting to do this - usually by selecting the **Q branch**. You can then use frequencies in the HF range, but VHF and UHF signals will no longer be present.

RTL AGC and Tuner AGC

These apply automatic gain control at the sampler stage or the tuner stage of the RTL-SDR dongle respectively. These hardware controls tend to be more broadband and less effective than using software gain control for these dongles.

PPM Correction

Inexpensive RTL SDR dongles sometimes do not have a stable or well aligned oscillator that is used for setting the frequency accurately. Some better ones such as the RTL-SDR blog v3 will have a temperature controlled crystal oscillator (TCXO) in the circuit which can prevent this.

Other generic ones do not, which can mean the frequency passed to the SDR software is incorrect. If this is the case you will need to calibrate the receiver through SDR++ by entering a figure for the correction in 'PPM' - parts per million.

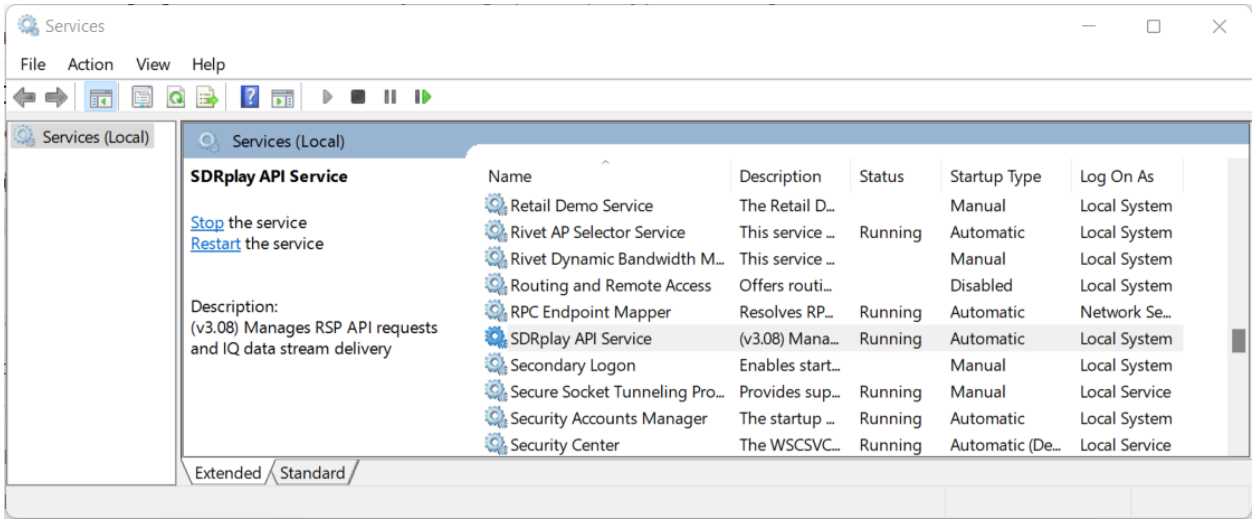
To calibrate your dongle and find the PPM figure use the `rtl_test` utility. This will have been included when you installed the drivers for your rtl-sdr dongle, and is available in Windows, MAC and Linux.

More about using rtl_test.

SDR Play only source settings

On Windows the SDRPlay API service that is required for SDR++ to recognise an SDR Play device may sometimes fail to start or have stopped. If this happens SDRPlay will not appear in the list of sources, or you may get a message that no device was found. First, wait a few seconds then press 'Refresh' in the source menu. If this does not work, you can re-start the service directly either by

- starting, then closing, SDRUno before you run SDR++
- changing the services settings to start the SDRplay API service, and change it to run automatically. You can find these settings by typing 'services' into the search function. If it's already set to automatic, re-start the service.



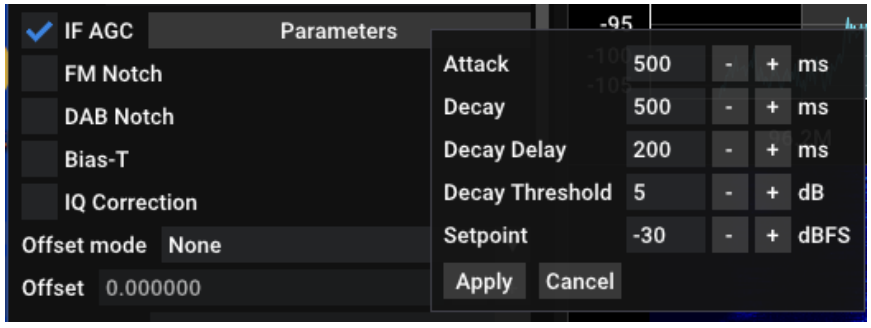
IF mode

This sets the 'intermediate frequency' used by an SDRPlay receiver. ZeroIF should be sufficient for most purposes. The 'Low IF' mode can be used for specialist uses but may generate an artificial 'spike' and signal on the spectrum display.

IF AGC

This controls the gain or amplification at the IF stage of the receiver automatically. There's more about how 'automatic gain control' (AGC) works in the section above .

You can alter how the IF AGC works on an SDRPlay receiver by selecting 'parameters'. You can control how quickly AGC kicks in to reduce strong signals or boost weak ones through the attack and decay settings, and adjust the signal level at which it will operate through the setpoint. Usually, the default settings will work well.



FM Notch / DAB Notch

These switch the notch filters for these bands on and off. This is to prevent your receiver being swamped by strong signals from these broadcast bands.

Antenna

This allows you to choose between the antenna inputs (Ports A, B, C) in the RSPdx and RSPduo.

Airspy HF+ Discovery-only source settings

Attenuation

This switches in the receiver's internal attenuation which reduces the received signal strength. It works only when the AGC is switched off. Attenuation can be useful where you have a very strong signal. You can also increase the attenuation where you are receiving a lot of noise, as this can reduce it and will sometimes allow you to hear the wanted signal better.

HF LNA

The HF+ Discovery has an LNA or 'low noise amplifier'. The LNA will amplify (increase) the received signal (but also the noise).

Lime SDR only source settings

Antenna

The Lime SDR is normally sold with between 2 and 10 RF inputs. This control allows you to select the appropriate input.

Pluto SDR

IP

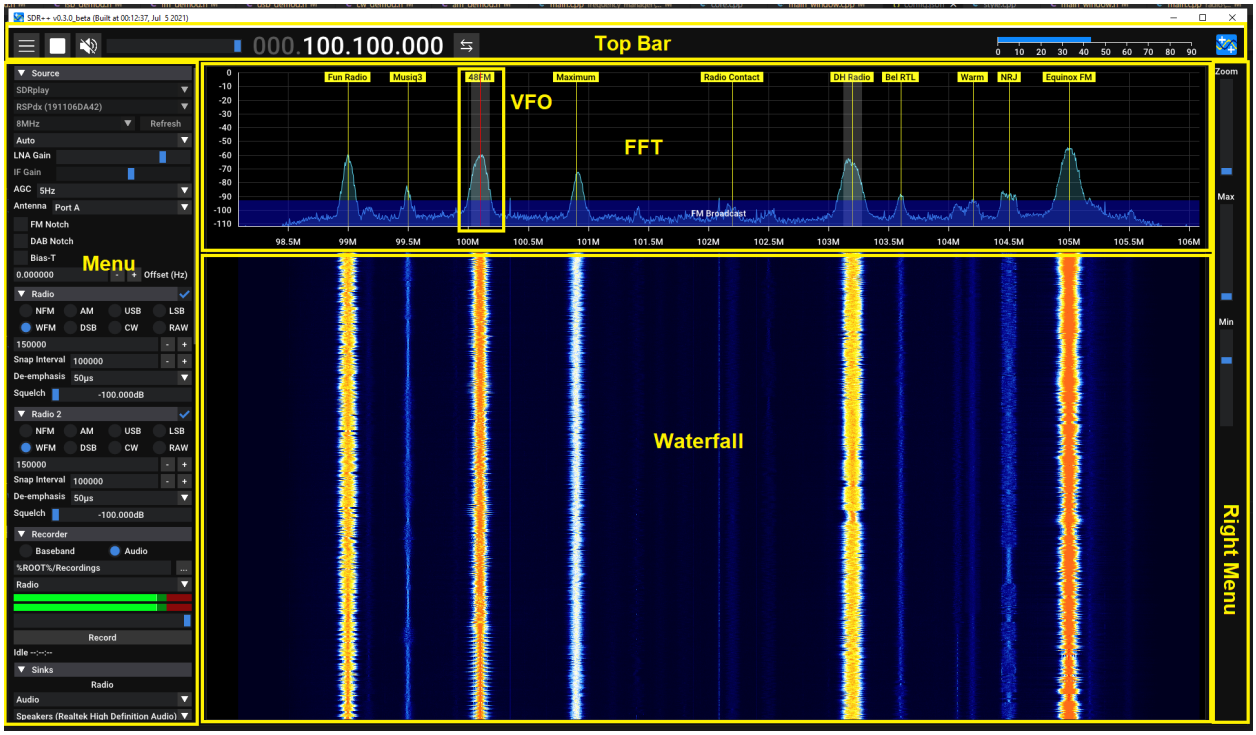
The USB connection to the Adalm Pluto is used as a serial console for sending commands and configuring the Pluto, but SDR programs access the receiver through the network connection it creates. By default this is 192.168.2.1, but you may have changed this when you configured the Pluto or when adding a USB ethernet adaptor. You insert the IP address in this field to access the Pluto.

Gain mode and PGA Gain

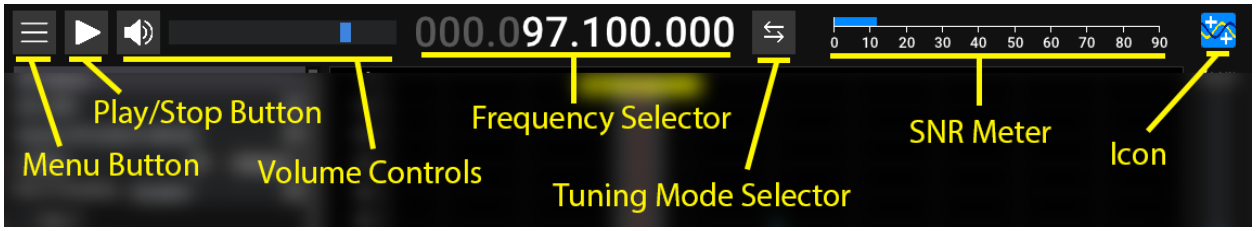
This allows you to set a fixed level of semi-automatic gain control. Choose between: manual, fast attack, slow attack and hybrid. If you choose manual, the gain will be set by the programmable-gain amplifier setting - the 'PGA' slider. The PGA slider is not accessible with the other settings.

Using SDR++

The main screen



Top Bar



The top bar contains the most important controls.

Menu button and menu

The menu button reveals the side menu to show all the modules and allow you to make changes, or hides it so you have more screen space for the spectrum display and waterfall.

The menu shows all the different settings and modules in SDR++. If you want the modules in a different order you can drag and drop them into a different place in the menu.

Start button

Play button - use this to start the receiver.

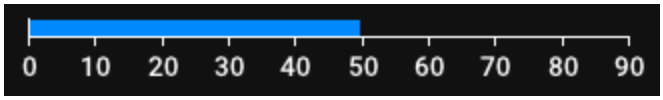
When started, a white square stop button replaces it. Press this to stop receiving.

Volume control

Drag the cursor to adjust the volume level or click on the bar at the level you want. You can click on the speaker symbol to mute and un-mute the audio. If you have multiple VFO's this will adjust the volume for the selected VFO. Otherwise you can adjust the volume for each VFO independently using the control for it's 'sink'.

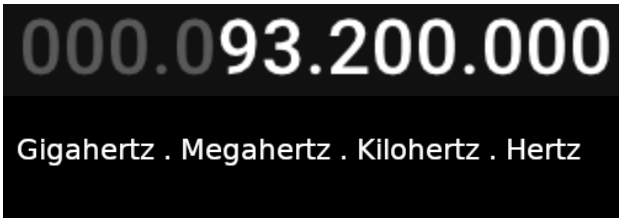
SNR Meter

The signal meter provides a measure of the relative strength of signals above the background noise - the 'signal to noise ratio' or 'SNR'. It is not a calibrated S-meter like you would find on an amateur transceiver or professional receivers, as it is designed to work with a wide range of SDRs, with varying settings.



Frequency Selector and tuning

The frequency selector or display shows the frequency of the received signal in hertz. The decimal points distinguish between (from left to right):



So, in the illustration above, it shows 93.2 MHz or 93,200 kHz or 93,200,000 Hertz.

The frequency being received by SDR++ is shown by the VFO band on the FFT spectrum display and waterfall. You can have multiple VFOs.

In some circumstances you might have no VFOs running - for instance if you have unticked the box for the radio VFO. In these circumstances the centre frequency of the SDR will be shown.

You can tune your receiver in several different ways:

Using the frequency selector

- Place your mouse's cursor over the figure in the frequency display that you want to change and use the **scroll wheel** to move it up or down
- Click (or tap if you have a touch screen) on the upper part of the figure to increase it by 1, or on the lower part to decrease it by 1
- To input a specific frequency hover your mouse or tap on the largest digit you need to change then type in the whole frequency. The cursor will move across to each digit you change as you do this. So, to change 93.2 to 101.5 in the picture above you would place the mouse over the '0' before the '9' and type '1015'.
- Hover the mouse cursor over the spectrum display or waterfall and use the scroll wheel. One 'notch' of the wheel will increase or decrease the frequency by the amount set as the 'Snap interval' in the radio menu.

Other ways of tuning include:

- Click on the FFT spectrum display or waterfall on the frequency you want. Usually this will be when it shows a signal of interest - more about these below.
- Drag the frequency scale at the bottom of the FFT spectrum display.
- Drag the VFO band to the frequency you want.

You will also have to set the correct modulation, for instance ‘AM’, in the Radio module (see below). The Frequency Manager module can also tune the receiver directly to bookmarked frequencies you have saved.

Tuning Mode

Next to the frequency display is a button that changes the behaviour of the VFO band. The button toggles between:

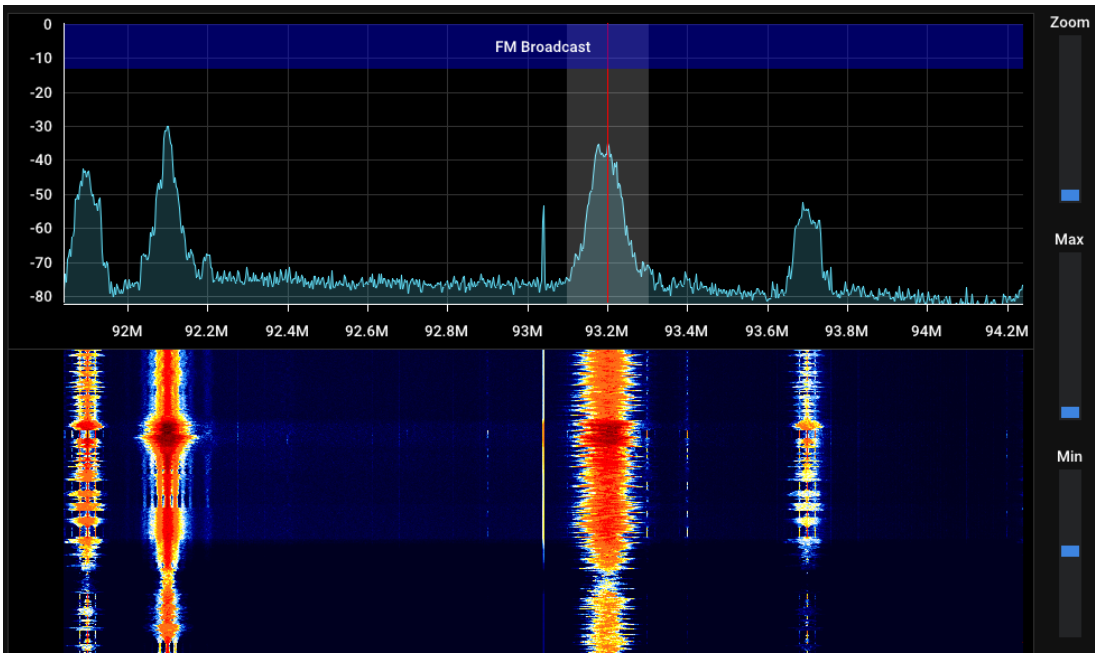
‘Centre tuning mode’ - if you click or tap on the waterfall or spectrum to tune then the selected frequency ‘snaps’ to the centre of the display, and the spectrum width is centred on this frequency

‘Normal tuning’ mode - you can click or tap to tune anywhere on the spectrum display and waterfall and they stay where they are. You will then see a continuous line on the waterfall for the chosen signal and other signals that you might be monitoring.

Icon

The icon shows the credits and build version of the software when clicked.

Spectrum display (FFT) and waterfall



Using a spectrum display and waterfall is the best way that a modern SDR receiver can show a chunk of the radio spectrum. You can see several transmissions at once and select which one to investigate.

You can see the strength of each signal through the depth of the color and the width of the transmission. You can see some of the characteristics of the transmission such as the sidebands and even recognise visually whether a signal is CW, upper or lower sideband or data.

The FFT or spectrum display

This is at the top. When receiving, it will show a part of the radio frequency spectrum. It can show the width of the radio frequency spectrum that your SDR receiver is set to receive. For instance, on an SDRplay SDR this can be up to 10 MHz. This can usually be set in the 'source' module.

The vertical axis shows the strength of the signal - or noise[1].

The horizontal axis shows the frequency.

Signals - or indeed some man made or natural noise and interference - will create peaks because they are louder or stronger than the base noise level.

You can click or tap on the peaks to tune to that frequency and hear what the signals are.

As already mentioned, further ways of tuning the receiver include

- dragging the frequency scale at the base of the spectrum display so that the cursor is over the desired frequency.
- dragging the VFO band to the desired frequency

The waterfall

This shows the same information as the FFT/spectrum display but over time - hence it moves. You can see this on a CW (morse) signal where the peaks of the dots and dashes will create a track on the waterfall, as though somebody is writing them out on paper but vertically.

The waterfall helps you identify signals of interest because it will catch intermittent transmissions like 2-way voice traffic. These show up as a track of interrupted lines on the waterfall. You can click on the line to tune to it and hear the transmission.

Using the FFT/spectrum display and waterfall together

You can adjust the size of each by dragging the dividing line up or down to make one bigger and the other smaller.

Zoom

This sliding control on the right menu allows you to zoom in and out. It narrows or widens the frequency range that the spectrum and waterfall covers. This means the peak and

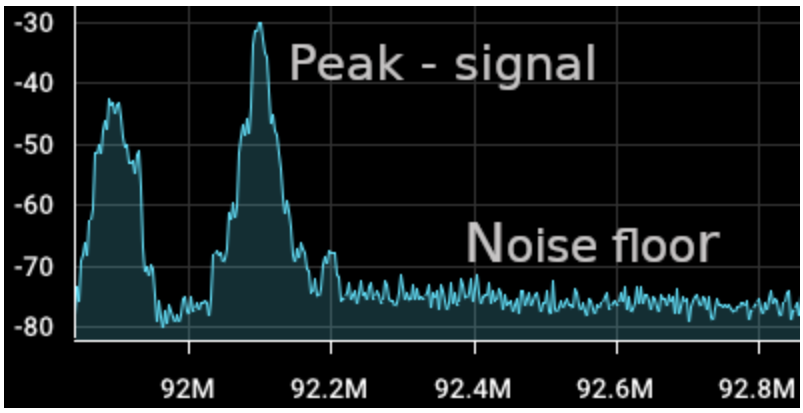
[1] this is in dBFS and not dBm and is not calibrated. It is just relative to the maximum level of the digitization

track of the signal is bigger. You'll be able to see more details of the sidebands and click more accurately on the signal so it is 'in tune'

Min and **Max**

These sliding controls select the high and low points for the signal strength shown on the spectrum - effectively the top and bottom range.

Start by adjusting the 'min' slider so that the base noise floor - the valleys and plains below any peaks - is brought down to the bottom of the spectrum display like this:



By doing this you'll see that the waterfall background - except where there are signals - changes from an intense color to a more or less uniform dull or black area. This means that you can better see the brighter tracks from weak signals.

The 'max' slider will be at 0dB when fully down. You can move it upwards if you need to show weaker signals on the waterfall.

Play with these controls until you have the right contrast on the waterfall that works for you. You may need to adjust these when you change frequency because the noise floor and signal strengths will be different.

Tuning using the FFT/spectrum and waterfall

You can tune to frequencies shown on the spectrum and waterfall by pointing and clicking. This will set the VFO to that frequency and the VFO band will be shown. You can tune up and down by using the scroll wheel on your mouse, if it is hovering over either the FFT/spectrum display or waterfall. It will tune in increments set by the 'snap interval' set in the Radio module, for instance 100kHz for FM broadcast or 100Hz when in single sideband (LSB or USB).

The VFO (variable frequency oscillator) tunes the SDR to a particular frequency. It's shown here as a band or line going straight through both the spectrum display and the waterfall. This is like the tuning needle or cursor you'd get on an old radio. The band/line is as wide as the filter width the VFO is set to. You can increase or decrease the filter width by dragging the edge of the VFO band outwards or inwards.

If you hover your mouse pointer over the VFO band it will show 'radio' or whatever name you have given to that VFO. If you do the same with the ctrl key pressed it will show the frequency, bandwidth, whether the bandwidth is locked and the signal to noise ratio (which will change dynamically).

Other uses for the spectrum and waterfall

If you have a manual **antenna tuner**, you can adjust it to maximise the signal strength. Technically it's matching the antenna impedance with that of the receiver to produce more signal into your SDR.

As you adjust the inductance and capacitance to tune the antenna you'll see the noise floor rise and sweep across the spectrum. You can adjust one control (usually capacitance) and then the other to get the peak at the frequency you want to hear.

You can create **bookmarks** at particular frequencies using the **Frequency Manager module**. These will display on the spectrum display. The **Bandplan** module will display the allocation of blocks of frequencies on the spectrum. More about these below.

Keyboard controls

F11 Toggle fullscreen.

Home Toggle waterfall.

End Toggles between play and stop. In Linux this may close SDR++.

< > Left Arrow / Right arrow - Tunes SDR++ up or down in frequency by selected step ('snap interval'). If you have your mouse over the frequency selector it will instead move between the digits of the frequency, which you can then adjust with the up/down arrows.

tab Moves down the menu between variable slider controls, so that you can then alter them instead by directly entering the desired figure in the box. For instance, it might move between the source gain control to the Radio VFO bandwidth and with another press to the snap interval below it. Type the figure you want to use e.g. 2400 followed by either enter (which will us the chosen figure) or another press on tab to move you to the next box. shift + tab will instead move you up the menu between boxes.

page up and page down keys - move from on VFO to another, if multiple VFOs are enabled.

You can find a full list of user interface key controls on the SDR++ github pages.

SDR++ command line related parameters

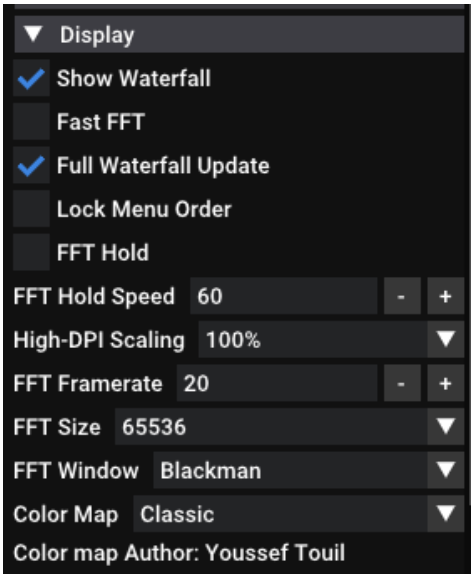
You can use these in the command line interface e.g. in Windows after sdrpp.exe.

Short	Verbose	Description
`-a`	`--addr`	In server mode this specifies the IP address of the network interface accepting connections
	`--autostart`	Autostart the SDR when SDR++ starts up. This may be useful for those using SDR++ as a permanent display. It will only work for sources that do not require connection prior to starting. Only available from later version 1.06 nightly builds.
`-c`	`--con`	Show console (-s on earlier versions)
`-h`	`--help`	Show help. Only available from version 1.06 or nightly builds.

Short	Verbose	Description
`-p`	`--port`	Server mode port. Insert port address after parameter.
`-r`	`--root`	Root directory where all files are saved. Insert path after the parameter.
`-s`	`--server`	Run in server mode. More details on using these commands with SDR++ Server.

Display menu

This controls the configuration of the waterfall. 'FFT' stands for 'fast fourier transform'. FFT is the algorithm used by the computer to convert the data from the signals in the SDR into visual representations - the spectrum display and waterfall.

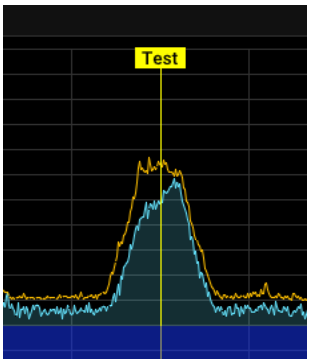


Show waterfall - switches the waterfall on or off

Fast FFT - using this will mean that a faster but less precise rendering will be used. This will show less detail on the waterfall and spectrum display. [Removed from release version 1.1.0 onwards].

Full waterfall update - when enabled, the history of the waterfall is updated when viewing parameters like zoom and min/max are changed.

FFT Hold - this provides a trace across the spectrum display showing the maximum signal level that has been reached.



You can adjust the amount of time for which the trace is displayed using **FFT Hold Speed**. A larger value means the trace reduces quicker.

Lock menu order - enabling this prevents you from moving the menu section. It is enabled by default in Android to prevent accidental moves whilst dragging scroll bars. If you try to move a menu section whilst this is set, an error message will be displayed

High-DPI scaling - this allows you to adjust the size of the controls, borders, scroll bars and text in SDR++. You might use it if you have a very high resolution monitor, or an Android device like a phone where you need these to be bigger so they can be seen, or to make the controls large enough to be easily moved by a finger or stylus on a touch screen. You'll need to close and then re-open SDR++ for the settings to take effect.

FFT Frame rate - SDR++ runs an algorithm to generate the FFT/spectrum. When this is run more times each second it produces a more detailed FFT/Spectrum display, but at the expense of frequency resolution. As a result the waterfall descends faster. Higher is quicker. The default is 20.

FFT size - This increases or decreases resolution - the number of signal data points that are shown on the spectrum display and waterfall. Higher is better, and will show weaker signal traces, but will take more processing power. A slower CPU may not handle the higher rates and produce some clipping or stuttering in the audio output.

FFT Window - This setting determines the algorithms applied to data before being run through the FFT, and will affect the look of display signal peaks on the spectrum display and waterfall. You have 2 choices - 'Rectangular' or 'Blackman' and, from version 1.1.0, Nuttall. Nuttall is the new default.

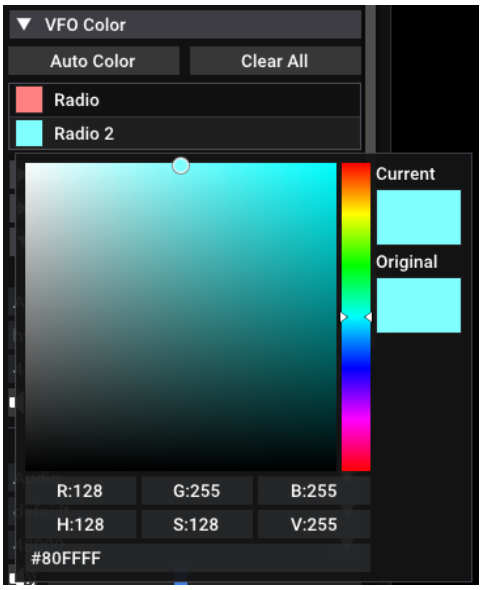
Color Map - You can select one of a number of 'maps' that change the color and intensity shown on the waterfall for the range of signals from weak to strong.

This allows you to change the color of the VFO ‘band’. The VFO band shows the frequency to which the SDR is tuned across the spectrum display and waterfall, and is as wide as the filter width to which it is set.

A color can be chosen which best contrasts with the background.

If you have set up multiple VFOs, you can set the colors of each to help you identify them.

VFO Color



This simple control allows you to select a color to use for the VFO band. This is particularly useful if you have multiple VFOs.

Click on the colored square next to the VFO that you want to change, then either select the color from the tool or specify the color in hex code.

You can reset the colors by selecting ‘Clear All’ or ‘Auto Color’ to have SDR++ allocate colors automatically.

Bandplans

The allocation of bands of radio frequencies, particularly at VHF and UHF, varies between different countries and regions. For instance the allocation of frequencies on the 60 meter amateur band varies, as does the existence of a 4 metre (70 MHz) amateur band.

SDR++ contains a number of bandplans for different geographical areas e.g. UK or USA. At a minimum they will show the blocks of frequencies used by radio amateurs and broadcasters. Some, such as the UK band plan, will also cover HF air and marine allocations, and major usage on VHF and UHF, such as private mobile radio.

By default SDR++ will open with the 'Worldwide bandplan'. You can change it from the drop down list to another available bandplan.

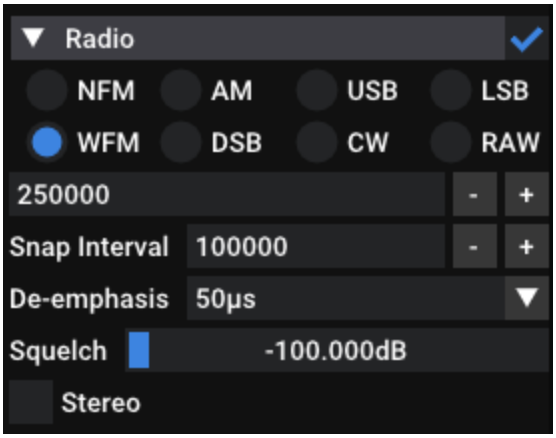
You can choose whether the bandplan is displayed, and whether it is shown at the top or bottom of the Spectrum monitor.

You can alter the bandplan by editing the appropriate file .json file [in the /res subfolder] in a text editor. You should arrange the entries in frequency order and follow the same format. There are preset 'types' of allocations, such as "amateur" that will each be shown in the same color. If you want to create a new type or alter the colors used, then you will need to edit the entries under "bandColors" in the main `config.json` file in the root folder.

Themes

There are a number of themes which you can select to choose the overall look of SDR++. It will change the colors of the background, spectrum and waterfall, text and menu boxes.

Radio module



←Filter width

This is the main module for setting the type of 'modulation' you can receive.

You can remove the tick in the box next to 'Radio' to disable this module. SDR++ will continue to show signals in the spectrum display and waterfall but there will be no sound sent to the speakers or other outputs.

Modes

SDR++ can receive all the main modes of modulation applied to signals. You can select these by clicking on the button for the mode.

Mode	Name	Use	Default Filter
NFM	Narrow band Frequency Modulation	From 27 MHz upward for Citizens Band, most business radio and on amateur radio bands. Voice, and V/UHF data transmissions such as DMR and POCSAG/Flex are modulated in NFM	12.5 khz
WFM	Wideband Frequency Modulation	Broadcast band 87-108 MHz. You can also choose the de-emphasis used, and switch stereo reception on and off.	150 kHz
AM	Amplitude Modulation	Long, medium and short wave broadcast stations, Citizens Band, Airband.	10 kHz
DSB	Double Sideband	Used as a method of using upper and lower sidebands to better receive AM shortwave broadcast stations	4.6 kHz
USB	Upper sideband	Amateur bands above 10MHz and on 60m, most HF utility stations (Air, marine, military)	2.8 kHz
LSB	Lower sideband	Amateur bands below 10 MHz	2.8 kHz
CW	Continuous Wave	Morse code transmissions. Basically LSB with a narrow filter	200 Hz
RAW		This enables the full sample width of the SDR to be output	

Filter width

Different modes of transmission use different bandwidths. SDR++ has default filters to match the bandwidth (see table above), but you can alter these by typing in the value or clicking on the '-' and '+' symbols to narrow or widen the filter. You can also adjust this by dragging the edge of the VFO band - shown on the spectrum display - in or out. You might do this to make the audio more pleasant, cut out interference from signals that are close in frequency or to ensure all the information on a data signal is received.

Snap interval

This is used to set the frequency step (or 'increment') that the receiver will move when tuning.

For instance, for FM broadcast stations you'd want this set as 100 kHz. It would then move from, for instance 93.1 to 93.2, then to 93.3 MHz for each movement of the scroll wheel on your mouse (when the cursor is over the frequency display or waterfall).

The snap interval is retained by SDR++ for each mode. So, if you alter it to 2800Hz on USB all other USB signals will use this filter width until you alter it again.

See the section on the frequency display and tuning for more on tuning.

De-emphasis (WFM only)

On wide (broadcast) FM signals, audio frequencies - particularly in the high range - are boosted (emphasised) at the transmitter then reduced at the receiver (de-emphasis).

You can switch off the de-emphasis or adjust the amount applied. In Europe 50µs is used, but in the US it is 75µs.

Squelch

The squelch cuts out the audio unless a signal exceeds the chosen level. It's useful for muting the background noise whilst allowing you to hear the wanted signal when it goes above the noise level.

You can set this by ticking the box to activate it, then dragging the marker to the right until the audio mutes. Check that the audio breaks through when a wanted transmission occurs. Adjust if necessary.

You will need to adjust the squelch level for different frequency ranges because noise levels will vary.

AGC Attack and Decay (not NFM or WFM)

This allows you to set the parameters for automatic gain control. This is useful for when the level of the signal - or the volume of the modulated signal - changes rapidly. You can set how quickly it reacts to this.

- 'Attack' governs how quickly the level is changed to compensate for increased signal strength
- 'Decay' governs how fast the level changes to a reduced signal strength. This is usually much slower than attack to prevent noise levels being boosted during short gaps, for instance when someone pauses between sentences.

The AGC levels you set are kept for each modulation type. For instance, if you set the levels for USB, then set them again for CW, the levels for USB will not be changed, and will be there when you return to using USB..

Carrier AGC

This applies the automatic gain control adjustments according to the level of the AM carrier rather than the demodulated audio level. This may produce quieter, more pleasant and consistent audio. It doesn't work as effectively where you have a signal that is fading in and out.

IF Noise Reduction (NFM and WFM only)

You can use IF noise reduction to make a received signal clearer and cut out background noise ('QRN') and some man-made interference ('QRM'). This is available for wide (broadcast) and narrow frequency modulated signals only.

When you are in NFM mode, you have an additional setting for either

- NOAA APT weather satellite transmissions
- voice transmissions
- narrow band modes - also can be used on voice to reduce hiss.

'IF' noise reduction works at the 'intermediate frequency' stage of the SDR receiver.

Stereo (WFM only)

This toggles between stereo and mono where there is an FM stereo broadcast. Switching on stereo when you have a weak signal can add significant 'hiss'.

Low Pass (NFM and WFM only)

This applies a filter to the audio signal to half the bandwidth of the VFO. In technical terms, it removes anything above a modulation index of 1. It can help remove high frequency hiss. You can keep this on, unless it interferes with the reception of a data signal where it's not recommended.

RDS (WFM only)

SDR++ can decode digital 'Radio Data System' signals broadcast as part of an analog FM broadcast. This can include the station name and program information. 'Alternate Frequency' and 'Traffic Program' information is not supported.

The RDS data is displayed in the top left of the Spectrum Display (FFT), just under the frequency display.

RDS is very sensitive to signal strength. You'll get better results with a strong signal, RDS will work in both mono and stereo modes.

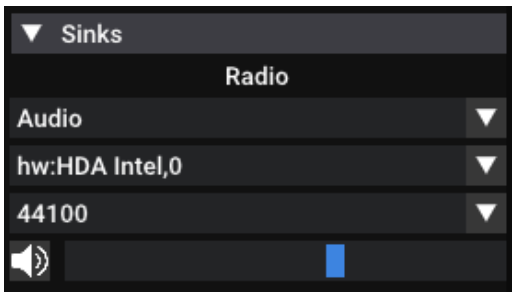
Noise blanker (USB, LSB and DSB only)

The noise blanker works against 'impulse' type noise, such as a car ignition system or lightning.

Select this if you are suffering from this type of interference. Adjust the slider until the interference is reduced, but not so far that audio quality is degraded. This feature is still being worked on, so your results may vary.

Audio and Sinks

A 'sink' is a term for anything used to output sound from a demodulator or decoder. The raw data (or IQ signal) from the SDR receiver is taken by SDR++ and turned into sound that can be sent to your soundcard or over a network. You can then hear the signal through your speakers or it can be passed to another program (for instance. to decode a data transmission) via a virtual audio cable/loopback or to another computer by UDP/TCP over a network.



If you open the 'Sinks' module in the menu, you will see a 'Radio' sub-heading. This is the default name for the output from the VFO (or 'tuner') in SDR++. The Radio audio will generally be the only working 'sink' - or final output - for the SDR receiver when you first install it. You may later have more than one sink if you set up multiple VFOs, or if you add

the M17 decoder module. You can adjust the sink to send the output to a network connection - and would select this from the drop down menu.

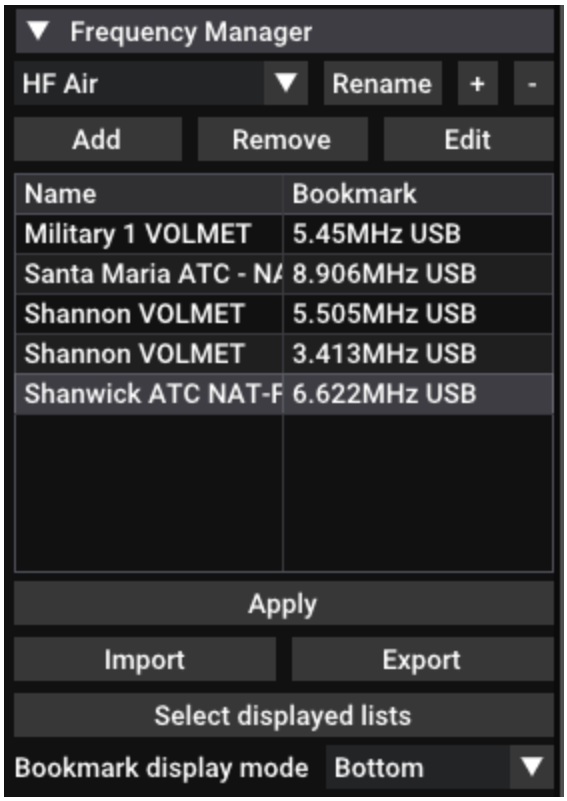
From here you can select the output for the audio. This would normally already be configured as your sound card (hw:HDA Intel,0 in the illustration above), but you may want to make a different choice if your soundcard has different outputs, for instance speakers or headphones.

Alternatively, if you have installed a virtual output such a VB-Audio Cable (in Windows), this will appear here. You can use a virtual audio cable to route your audio to a program running on your device (such as WSJT-X) to decode digital modes.

The bottom row of this module is the output level, and here duplicates the volume control on the top bar of SDR++.

You can create a sink for each additional VFO you use - see more about multiple VFOs

Frequency Manager



Frequency manager is a simple but versatile way of saving and displaying your favourite frequencies in 'bookmarks'.

You can save them, together with the bandwidth and mode, and store them in different lists. Bookmarks cannot be saved with the same name, but adding an extra character or even one or more spaces to the bookmark name is enough to enable it to be saved.

Bookmarked frequencies can be shown in the spectrum display.

You can open Frequency Manager from the menu, then click on the arrow next to it's name in the menu to reveal the full module.

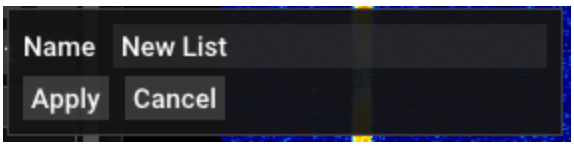
Lists

View and choose your lists

When you have created several lists just one of them will be displayed. Click on the down arrow next to that list's name to display all your lists. Click on the one you wish to choose.

You can then click on any bookmark in that list to tune SDR++ to that frequency.

Create a list



SDR++ already has one 'General List' created. To create other lists, click on the plus ('+') symbol on the row next to the currently displayed list. This will open a box where you can type in the name of your list.

You might want to create frequency lists for airband, marine, shortwave or medium wave broadcast, FT8, CB channels, local DMR, or for themes such as local mosques, rescue services and so on.

Rename a list

You can easily rename a list by clicking on the 'Rename' button.

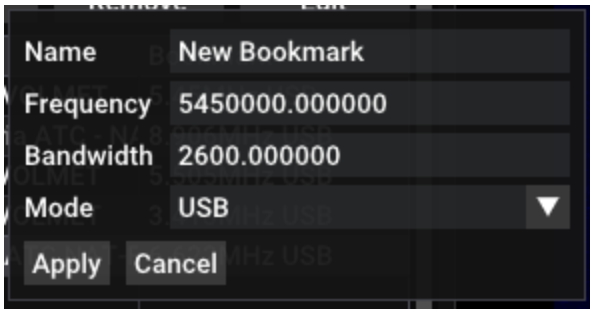
Remove a list

You can delete a list by clicking on the minus ('-') button. This will delete all the bookmarks in it.

Bookmarks

Create a bookmark

To save a frequency as a bookmark click on 'Add' This brings up a box where you can create a name for the bookmark. It will automatically pre-complete the bookmark with the frequency, bandwidth, and mode SDR++ is tuned to. However, you can edit these if you wish.



Click on 'Apply' to save.

The bookmarks will be shown in the list sorted in alphabetical order of the name. Bookmarks do not support unicode characters. Bookmarks should have a unique name, because duplicates are not permitted. If you do want to use the same name, then one or more spaces after the name will be enough for it not to be treated as a duplicate.

Tuning to the frequency of a bookmark

To do this you simply open the list to show the bookmark and double-click on it.

You can also click on a bookmark that is showing on the waterfall to tune to it.

Remove a bookmark

You can delete a bookmark by opening the list it is in, highlighting the bookmark and selecting the 'Remove' button.

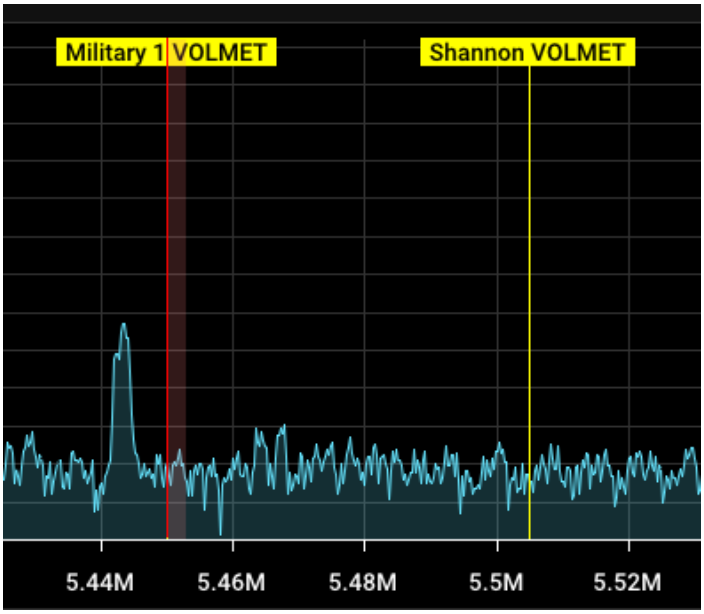
Edit a bookmark

You can edit a bookmark by opening the list it is in, highlighting the bookmark (with a single click) and clicking on the 'Edit' button. This brings up a similar box to when you created it, and the name, frequency, bandwidth and mode can all be edited.

Displaying bookmarks in the spectrum display

By default, the bookmarks in all your lists will be shown on the spectrum display. You can alter this by clicking on ' Select displayed lists'. This will show all your lists. Tick the ones you want to be displayed.

You can choose whether the names of the bookmarks are displayed at the bottom or the top of the Spectrum display. Here's an example where they're displayed at the top. SDR++ is tuned to the one on the left.



Importing and exporting bookmarks

You can import bookmarks into SDR++ that you have exported to a file from another instance of the program. For instance, if you are running SDR++ on another computer then you can copy over the saved bookmarks from there.

To export your bookmarks, you need to

- open the Frequency Manager module
- display the list that you want to export
- highlight the bookmarks you wish to export (click on each one, with the control key pressed down)
- press the export button
- Save the file with a name of your choice (for instance the 'List' name) and the file suffix `.json`

To import your saved bookmarks

- open the Frequency Manager module
- display or create the list(s) that you want to import the bookmarks into
- Select 'Import', locate the json file with the bookmarks and press OK

Bookmarks can be shared - so if you have a list of frequencies for your favourite stations you can export this and provide it to someone else with SDR++, who can then import and use them.

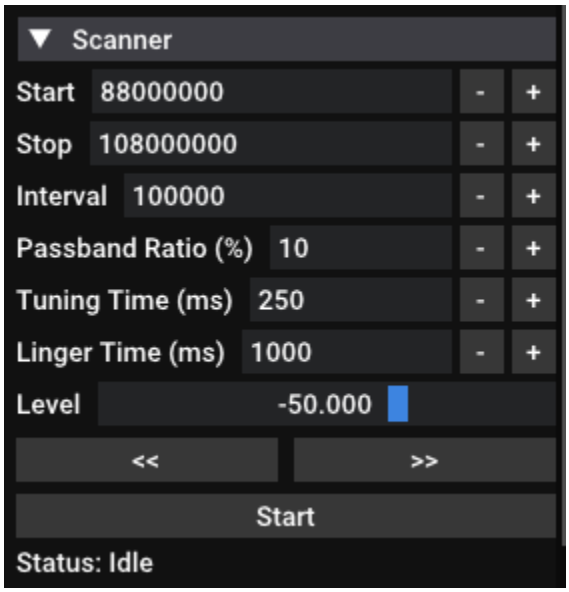
The full list of bookmarks, lists and settings are contained in the `frequency_manager_config.json` file in the directory where SDR++ is installed. If you are upgrading between versions of SDR++ you can either

- Create a copy of this file and move it into the root folder of your upgraded version
- Install SDR++ to the same folder without deleting files - and the upgraded version will use this file for frequency manager
- Export the bookmarks you want to save, so you can restore them afterwards. This will take longer than the other options if you want to keep all your bookmarks.

Scanner

This feature is available currently on nightly builds on SDR++ as a module you can add, but should be available for the version 1.1.x release.

The scanner searches across a specified range of the frequency spectrum. It monitors the whole range through accessing the FFT. This is very quick, especially if the frequency range is wholly within the bandwidth of your SDR receiver.



The scanner will move instantly to a signal it detects. It does not need to laboriously move from one frequency to another, like a conventional scanner radio.

It will move the VFO to the strongest active signal above a set level.

If you don't want to listen to the signal you can move to the next detected signal on another lower or higher frequency.

The scanner is very fast at detecting transmissions, but may also stop on strong interference ('QRM'). There is no facility to lock out a frequency in the scanned range.

At present the scanner does not scan memory banks of frequencies in Frequency Manager, nor can you save search ranges. The ranges you set are not currently persistent and will be lost when you close SDR++.

Scanner settings

Stop/Start: This sets the range of the frequency spectrum that is scanned. The setting is in hertz, so to scan the airband you might use 118000000 (118MHz) as the start frequency and 136000000 as the stop frequency.

> **Tip**: It can be better to scan smaller frequency ranges where you can see that there is no interference. You'll be surprised at how many new frequencies in use that you'll find with the SDR++ scanner as you'll not miss frequencies that a conventional scanner wouldn't reach before the transmission ended.

Interval: The scanner will be checking all the frequencies for activity but will set the channel spacing for these signals at a specific interval. So, if the frequency plan sets 12.5kHz channel spacing, you would set the interval to 12500.

When scanning, set the 'start' frequency at the first channel so the spacing for subsequent channels will be correct. The scanner will put the VFO at the centre frequency for each channel if it detects a signal within that channel's bandwidth.

You should ensure that the bandwidth in the radio module is set correctly for the type of signal for which you are searching. So, if you were scanning the broadcast FM bands, your interval would be 100kHz (100000), as stations are spaced at these intervals on the FM band, but your bandwidth for a wide FM transmission would be between 150 and 200kHz.

Passband ratio: This sets the amount of the bandwidth of a frequency (starting from the centre of the frequency) that will be checked for a modulated signal, and thus stop the scan so SDR++ can receive it. A narrow percentage will better reject adjacent channel interference

Tuning time: This sets the time for checking the FFT, once the level is exceeded, for a signal. The default is 250ms (a quarter of a second). This should be adequate for most purposes.

Linger Time: If the scanner has successfully detected and demodulated a signal, this sets the amount of time after the signal stops transmitting before the scanner abandons the frequency and resumes scanning. This delay enables there to be a pause between, for

instance, the base transmission and the reply from the mobile station, without the scanner leaving the frequency. You can thus hear the reply of the other station. The default is 1000ms (one second). You might increase this if stations are taking longer to reply to each other, or to give you longer to stop the scan so you can monitor that frequency.

Level: Conventional scanners use an audio squelch, so that they stop on a frequency where the level of signal is above the level at which the squelch is set. **SDR++ scanner does not use the audio squelch** to determine where it 'stops'. Instead it goes to signals above a set level on the FFT. Look at the noise level on the FFT and set it for something slightly above this. So, for example, if the noise is peaking at -70, start by setting it at -60.

Adjust the setting from there so the scanner stops on the signals you want to hear. You can drag the level to the right to make the scanner less sensitive by stopping only on stronger signals. You can drag it to the left to be more sensitive - but it will be more likely to stop on noise or interference.

> **Tip**: You might want to adjust the level with the squelch off, so you can hear everything that the scanner is stopping on. Adjust it by small increments until the VFO is skipping to the detected signals.

Resume scan (up or down >> or <<)

When the scanner stops on an unwanted signal or on interference, you can cause it to resume scanning and go to other signals though these controls. Obviously, >> sends it to the next identified signal higher in frequency, and the reverse with the other control.

Start/Stop: Commences or ends the scan. You will also need to have pressed the 'Play' button on the top bar to start SDR++ receiving. The scanner does not stop when the Play button is pressed again to stop SDR++ receiving.

Status indicator: This shows

- Idle - when it is not scanning
- Scanning
- Tuning - checking that a signal is present
- Receiving

Scanner tips:

- Save the frequencies you identify in Frequency Manager, so when the VFO goes to them you can quickly see what the transmission is by checking the marker displayed above it on the Spectrum Display.
- The Spectrum Display may move about as the scanner moves the VFO to different frequencies. If you find this disorienting, you can either
    - Set the tuning mode selector to centre tuning mode so that the VFO is always centre screen, but this can slow down the scanner or
    - Adjust the scan range and sampled bandwidth of the SDR receiver so that the displayed spectrum is stable and the only the VFO 'cursor' moves around. This also has the advantage that the waterfall will be stable and can show a history of signals received on these frequencies within the sampled spectrum. It will, though, reduce the number of signals you detect, and will restrict you to a very small portion of a band if you have an SDR with a small sample width such as the HF+ Discovery (about 600kHz) or an RTL-SDR (about 2MHz).

Recorder

You are able to record either the

- audio from the signal you are tuned to or
- the ‘baseband’.

The baseband is the whole of the raw IQ data from the bandwidth the SDR is processing. You can later select that file in the ‘Source’ module and play it back as if it was a live session using your SDR.

Some people record the baseband to explore their favourite band, for instance MW dxers will record the whole medium wave band through part of a night and play it over and over to find distant stations on all the different frequencies recorded.

By default the recorder will save the file in a ‘recordings’ sub-folder of the directory where SDR++ is installed. In Linux this is in ~/.config/sdrpp. Click on the 3 dots if you want to change the folder.

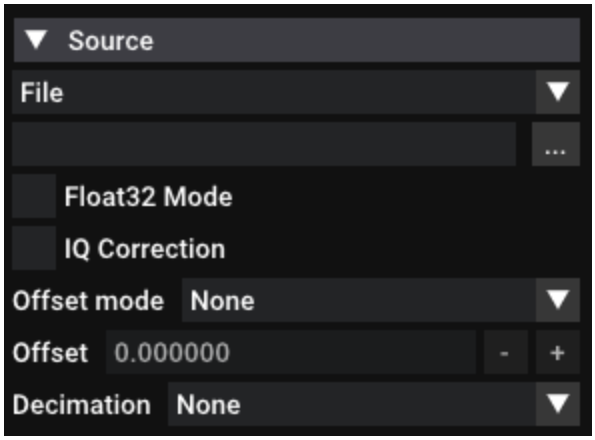
There is a signal strength meter to enable you to adjust the output so that only the strongest peaks go into the red.

You can use ‘Ignore silence’ to reduce the file size of your audio recordings - because it won’t record periods when there is no audio. This is useful when you are using the squelch to monitor and record intermittent transmissions using NFM.

Audio and baseband (IQ) recordings are in the .WAV format and have a maximum file size of 4GB. Baseband recordings are 16 bit signed samples.

When you are ready, press ‘Record’, which will change into a ‘Stop’ button whilst recording. You can use this, of course, to halt the recording.

To play back recordings of the raw IQ (baseband) data, use the ‘File’ option in Sources:



You can choose the recorded file by selecting the three dots then navigating to the saved file. As with using the SDR receiver as a source, you can apply an offset and decimation etc.

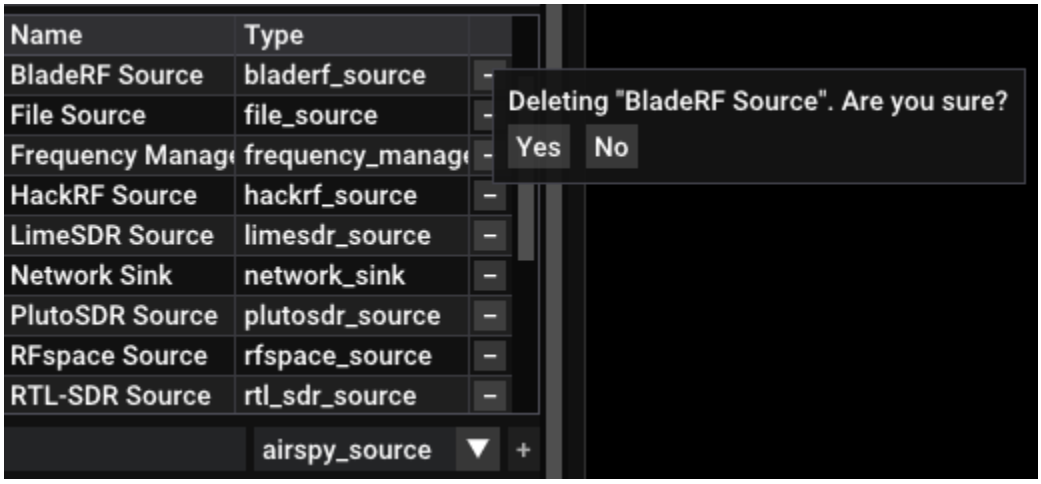
To play an audio recording, simply use the standard audio player for your operating system.

Adding and removing modules

You can add - and remove - many modules for SDR++. Adding modules can increase its functionality and flexibility. Removing modules can help you slim down the SDR++ so it operates more efficiently without unnecessary functions. You use the Module Manager to do this.

Removing a module

To remove a module you simply need to use the 'minus' symbol next to its name in Module Manager. You'll be asked to confirm this.



Adding a module

We are going to use the example of adding additional VFOs to show how to add a module.

Adding Multiple VFO's

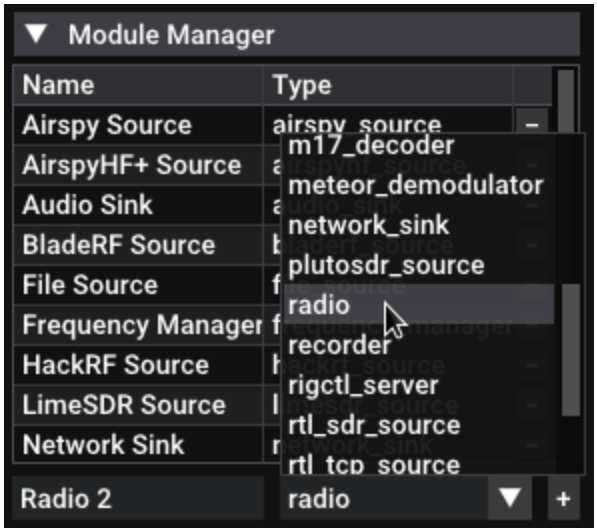
SDR++ allows you to tune to more than one frequency at the same time using the same SDR receiver.

The frequencies you choose have to be within the 'sample width' of the SDR receiver. For example if you have an RTL SDR blog dongle, it only covers about 2MHz of RF bandwidth. If you zoom out you will see all of this sample width in the spectrum display and waterfall -

for instance it might cover 6MHz to 8MHz. The 2 frequencies that you tune to must be within this - so you could tune to both a China Radio International on 7220kHz AM in the 41m broadcast band, and to a radio amateur on 7120 kHz LSB on the 40 metre amateur band.

How to set up a second VFO

Go to Module Manager and create a new 'radio Module'. To do this:



1. Type a name for the new module in the box on the bottom left - here 'Radio 2'
2. Select 'radio' from the drop down 'Type' menu on the right
3. Click on the '+' button to create.

This will create the new module which will be placed at the bottom of the menu. You can drag it into a convenient position, for instance below the other VFO entry ('Radio'). You will see a second VFO band appear in the Spectrum display in a different color which shows the frequency of the second VFO.

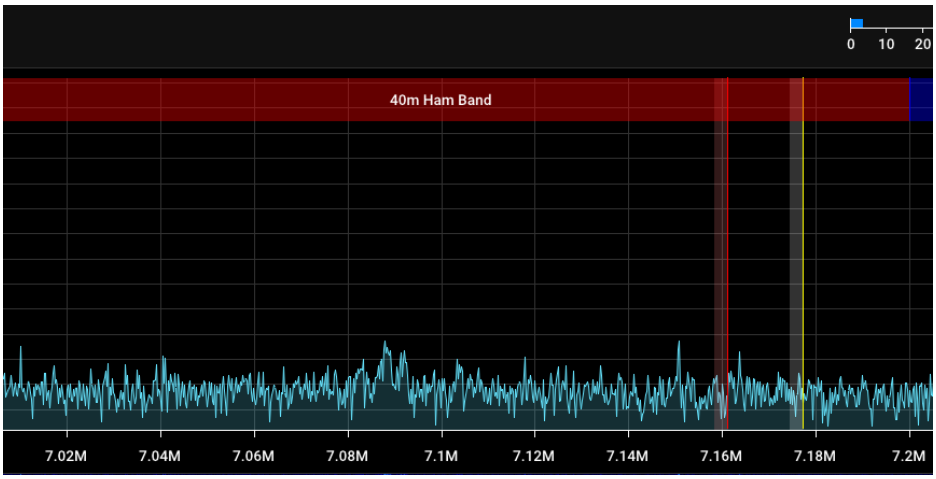
Before you can use your second VFO, you need to configure a 'sink' for its audio output to go into. This 'sink' is created automatically when you created the second VFO:

To listen to this second VFO

- click the drop down box and select ‘audio’ (you can also send it over a network)
- select the soundcard or speakers etc that you want to use
- drag the slider for audio level into the middle so the sound won’t be too loud.

Finally, to use the second VFO:

- tick the box next to its name in the module you created
- start SDR++ receiving by clicking on the ‘start’ arrow
- tune the second VFO by dragging the VFO band to a new frequency. It will change color to show that it is the VFO being controlled. You can select the modulation mode (AM, USB etc.) independently of the mode of the first VFO.



In this picture there are 2 VFOs, with the one being controlled in red, and the other in yellow.

If you want to identify a VFO, hover your mouse over the VFO band in the spectrum display and it will show the VFO’s name. The frequency ‘line’ in the VFO band will be fixed depending on whether it is being controlled or not. However, you can alter the color of the rest of the band on each one by using the VFO color control in the menu.

The frequency selector will show only the frequency of the VFO that is currently being controlled.

You can cycle between the VFOs by using the page up and page down keys.

Having multiple VFOs is very useful. You could use one VFO to send a USB audio signal to WSJT-X via a virtual audio cable to monitor FT8 transmissions, and a second to listen to radio amateurs talking on SSB, further up the same band (or on different bands if you have a large sample width). Alternatively, you might use multiple VFOs to monitor several signals on a satellite downlink, or to stand watch over several frequencies where signals are intermittent - like air traffic control.

Advanced Features and applications

Connecting to remote SDRs

SDR++ can act as a client for remote servers over a network connection, including SDR++ server and Airspy Spyservers. SDR++ server is simple because it is built in and we would recommend that you use this when setting up a remote receiver for your own use.

Please note: if SDR++ can't connect to a remote server, it may appear to do nothing for up to 20 seconds before timing out. This delay is because some SDR receivers can take time to be started by the server or because the remote server has crashed. Occasionally, this might cause SDR++ to crash.

SDR++ Server

SDR++ will operate as a server to provide the IQ stream over a network connection to another instance of SDR++ running on a different device elsewhere on a network - including over an internet connection. You can then use the client SDR++ to work with that signal in the same way as if it was an SDR receiver on your own device. The server can be run from the command line, for instance on a headless Raspberry Pi 3 or 4.

On the device acting as server you'll need to launch SDR++ in server mode, specifying the port to output the IQ stream over the network. Make a note of the IP address of the device. On your device you'll need to open the port and alter any firewall settings.

If connecting over the internet then you'll need to alter some settings on your router: you need to open the port and direct the input on that port towards the IP address of the device hosting the SDR++ server.

You can launch the server from the command line with

```
sdrpp --server --port [port no]
```

For instance to start it on port 12001

```
sdrpp --server --port 12001
```

You can omit the port setting. SDR++ Server will start by default on port 5259, and the client will be set at that port by default, also.

You can also add `--addr` to specify which IP address SDR++ listens on, in case the server has multiple network devices or connections.

On the client device you should

- select SDR++ Server as the source
- specify the IP address and port of the server (default is 5259)
- select the sample type - 'Int8' should work and is recommended if you are having trouble with your network connection

SDR++ on your client device now has full control over the source settings of the SDR receiver on the device running on the SDR++ server.

- enter the type of SDR receiver in 'Source [REMOTE]'
- select compression if you have a more powerful CPU on you client device - this will reduce the network bandwidth used
- Select 'Connect'
- Press the start arrow

You'll now start receiving the IQ stream and, if your settings are correct, you can use this as if it were the same SDR receiver connected directly to your client device. You'll be able to alter source settings such as AGC and bandwidth.

To disconnect, stop receiving on SDR++, and press disconnect.

You can stop SDR++ server by closing the terminal or 'cmd' box, or by hitting ctrl + C keys together.

SDR++ server works best on a good network connection. The signal may be intermittent on a poor connection, particularly where a wifi connection is used for the server or client.

SpyServer

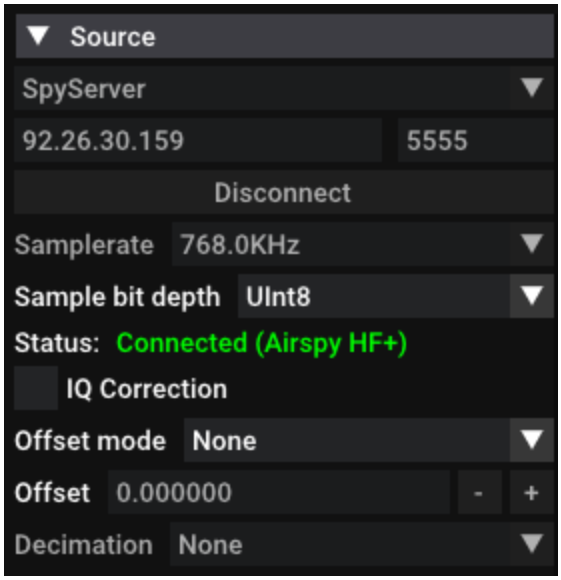
You can control SDR receivers that are not directly connected to your computer using SDR++. This includes SDR receivers using the Airspy Spyserver software.

SDR++ server may be a better choice: Spyserver only works with RTL-SDR, Airspy and Airspy HF+ receivers, whereas SDR++ server works with any SDR receiver or protocol supported by SDR++. SpyServer is not open source, and you have less control over the remote SDR receiver source settings.

You can set up the Spyserver on your own network, but this section deals with how to connect and listen rather than set this up.

You can find a tutorial on setting up a Spyserver on the RTL-SDR blog.

You need to set up a connection by selecting SpyServer in the Source menu:



To listen to an external SpyServer receiver:

- Go to the list of SDRs on the Airspy Spyserver network at https://airspy.com/directory/

- Find a remote SDR that is showing a green wireless symbol to show that it has free slots for users to connect.  Click to open the details of the SDR
- Copy the web address for the SDR e.g. sdr://92.26.30.159:5555
- Open the Source menu on SDR++
- Put the IP address of the Spyserver SDR (i.e. 92.26.30.159 in the example above) into the left hand box.
- Put the port number - shown after the colon (:) and usually 5555 - but it can vary) into the right hand box
- In the Sample bit depth try 'UInt8' - this is the lowest quality but will pass the least data across your internet connection and give you the best chance of a listenable connection. You can raise it to a higher bit depth when you are sure the connection is a reliable one, or if you are having trouble with UInt8.
- Press 'connect'
- Press the start button in SDR++

Sometimes the audio will stutter and break up to start with. It should settle but experiment with other settings if it doesn't - for instance the sample bit depth or decimation. To disconnect, stop receiving on SDR++, and press disconnect.

If the SDR receiver that you connect to uses an up-converter, such as 'Ham it up' then you will need to adjust the offset (for more details see the description in the 'Source' section earlier in this guide).

For a receiver on your home network you would enter the IP address for that server.

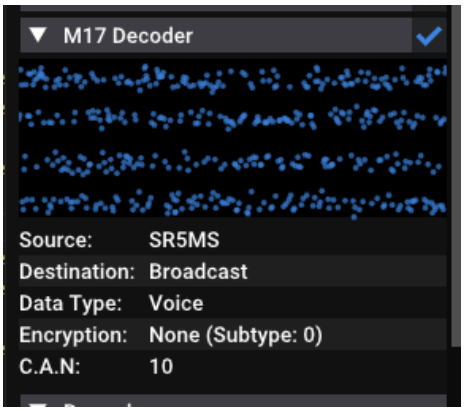
If you are testing a Spyserver on the same computer as SDR++, you would use 'localhost' or 127.0.0.1.

M17 decoder

M17 is a new open source digital audio mode that also uses open source hardware. SDR++ has a module to decode M17 that you can add from module manager. To do this, name the module M17 at the bottom of the Module Manager menu section. Select 'm17_decoder' from the drop down list then '+' to add the module to SDR++.

This adds

- a menu item for the decoder
- a dedicated M17 VFO, that will be shown on the spectrum display
- a new 'sink' for the audio output. You should ensure that the audio is directed to the sound card you want to use.



When you find an M17 transmission the M17 menu item will show the signal pattern (on which you can overlay reference lines), and the source name (usually the amateur radio operator's callsign).

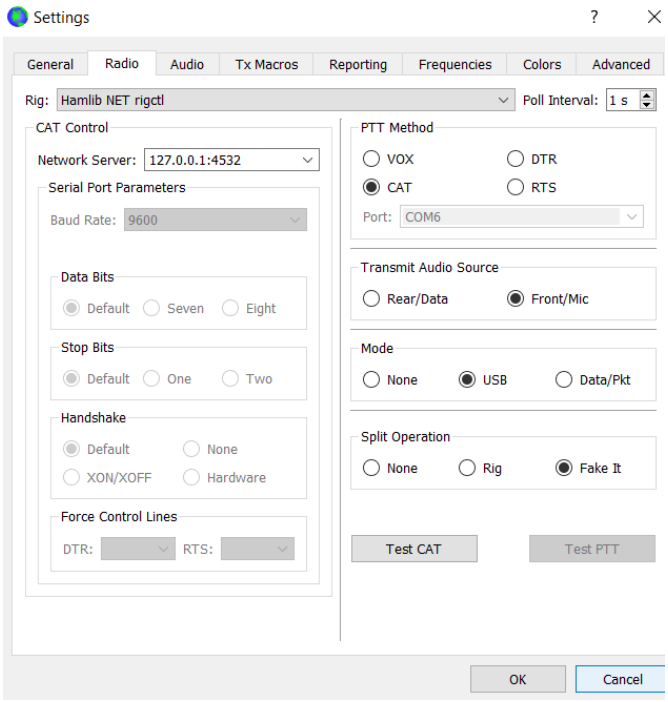
Rig control server - allowing other programs to control SDR++

The rig control server will allow SDR++ to be controlled by another program, such as gpredict, to listen on a specific frequency, and record the signal. You will need to align the server settings in SDR++ with those in the program to enable this to work.

To use Rigctl, for instance, with WSJT-X on the same PC that you are using for SDR++, then:

- In SDR++ set the rigctl server to 'localhost' and port '4532'
- If you have multiple VFOs select the one you want to be controlled by the server
- ensure 'Tuning' is ticked

- start the rigctl server in SDR++ - it will move from ‘Idle’ to ‘Listening’.
- In WSJT-X create a new configuration and call it SDR++
- In the configuration screen of WSJT-X:
    - Set Rig to ‘Hamlib Net Rigctl’
    - Enter the network server as ‘127.0.0.1:4532’
    - Select PTT method as ‘CAT’
    - Select Mode as ‘USB’
    - Select split operation as ‘Fake It’
- Press ‘Test CAT’ and check it goes green
- Check that SDR++ shows the status as ‘Connected’
- Press the ‘start’ arrow in SDR++ to begin listening

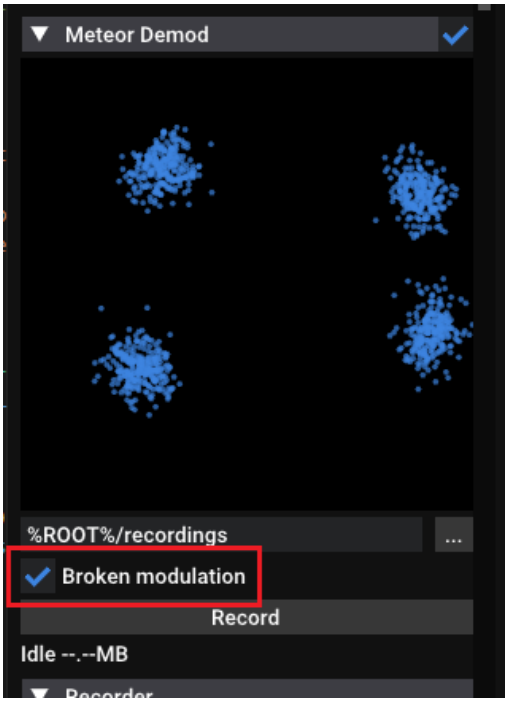


You would also then need to have

- a virtual audio cable (or Linux equivalent) installed to route the audio from SDR++ to WSJT-X
- WSJT-X audio configuration set to receive audio from that source
- the ‘sink’ in SDR++ set to route the radio audio to the virtual audio cable.

If you want the SDR++ to start the rigctl server when you open it, tick the box for 'Listen on startup'. Tick the 'Recording' box if you also want the output (usually the audio) to be recorded at the same time that SDR++ is being controlled by the external program (for instance for the duration of a satellite pass.

Meteor Demodulator



You can add this from the module manager. This extracts data from the Russian Meteor M2 weather satellite LRPT transmissions around 137MHz. The data needs a decoder such as 'SatDump' to produce a weather picture.

As of July 2022 there have been issues with the signal transmitted from the M2 Meteor satellite. SDR++ has been modified to address this. Select 'Broken modulation if Meteor is broken, or you'll get bad results. If Meteor is NOT broken, do NOT use it or you'll get bad results.

You can tell that Meteor is broken because:

1) There is a spike in the appearance of the Meteor signal in the FTT/ Spectrum display - it's supposed be smooth

2) The constellation is abnormal, as shown here.

Using external Programs with a virtual audio cable

Windows

A virtual audio cable is a piece of software that can take an audio signal 'piped' from an SDR program - that would normally go to your speakers - and instead route it to programs that can use that audio. It doesn't use any external cables.

In that way the audio containing data signals can be decoded by a more specialist program that, for instance, can decode digital DMR and M17 voice signals, CW, radioteletype (RTTY), fax, FT8 and other amateur modes, digital broadcast modes such as DAB+ and DRM, satellite data, pagers, ADSB and ACARS messages from aircraft and so on.

You can download a free virtual audio cable from https://vb-audio.com/Cable/

Linux

Linux distros do not need a virtual audio cable but can accomplish the same thing in different ways.

ALSA is the native Linux sound system. You can add 'loopbacks' to do this[2]. You need to go to a terminal and launch your favourite text editor as root. So for gedit:

```
sudo gedit /etc/modules
```

In the text editor add as the last line: -

```
snd_aloop
```

Then save the file.

You'll then have at least 2 loopback outputs to use. You can select one of these e.g. 'hw:Loopback,0' in the 'Sinks' module as the radio audio output instead of your sound card. You can then choose this as the input in any Linux based program you are using. This will not work on the standard Raspberry Pi OS, so you should use a Pulseaudio solution.

[2] (source) https://www.onetransistor.eu/2017/10/virtual-audio-cable-in-linux-ubuntu.html

Pulseaudio works on top of ALSA. You can use Pulseaudio to achieve the same thing. See this tutorial, which was written for the Raspberry Pi but can work on other Linux devices.

You can also either:

- install and use rtaudio
- use a network sink, especially if the program you send it to can use UDP/TCP.

Programs

Here are some of the applications you can use with SDR++ and a virtual audio cable. There is a more extensive list at the RTL-SDR blog website.

DMR with DSD+

DMR is a digital voice mode used on VHF and UHF (over NFM signals) by business walkie-talkies and by radio amateurs. Business users can include schools and universities, shopwatch nets, building sites, tram and bus operators.

DSD+ can take an audio signal and decode the DMR data into voice outputs and can also detect the color code and channel used. The free version of DSD+ can also decode another digital voice mode - NXDN. The free version is functional but the paid version - DSD+ Fastlane - does more and can operate as a standalone SDR program without using SDR++.

> Download DSD+ or subscribe to DSD+ Fastlane.

FT8 and other amateur digital modes with WSJT-X

WSJT-X enables you to decode a wide variety of data modes including FST4, FST4W, FT4, FT8, JT4, JT9, JT65, Q65, MSK144, WSPR and Echo (Earth-Moon-Earth signals). All of these are weak signal modes and can travel very large distances with the right equipment.

See the Rig Control section above for details on how to control the frequency SDR++ receives with WSJTX. For more help there are user guides on the download link below, and YouTube has videos on configuring and using WSJT-X.

Download WSJT-X

Other digital modes

There are a number of applications that can decode a wide variety of text based data modes from the audio signal from SDR++: including RTTY (radioteletype), CW (morse code), fax and SSTV (slow scan 'TV' - actually color pictures).

Fldigi is an open source program and has a smaller but equally impressive list of modes that it can decode.

Download Fldigi

MultiPSK - there's hardly a mode that this doesn't decode - all of the above plus DStar, DMR, ACARS, SITOR, Navtex, Hellschreiber, POCSAG, ADS-B, ACARS and more. It's quite a complex application for beginners, and is designed to allow radio amateurs to transmit as well as receive in some of those modes.

Download Multipsk

Decode pagers

Multimon-ng - is a command line tool to decode free, open source application to monitor POCSAG and FLEX - used by pagers on VHF and UHF. You can often find a version in the repositories for your Linux distro, or you can build Multimon-ng from source. It's light enough to work on a Raspberry Pi. It's also available for Windows. You can find tutorials on using it online.

Download Multimon-ng for windows

PDW is also an open source monitor for POCSAG and FLEX as well as ACARS: the data transmissions from aircraft on the AM VHF airband. It uses a graphical interface and is easy to use.

Download PDW

Optional built modules

There are unofficial external modules for SDR++ that are not included in the installation package but can be built by advanced users.

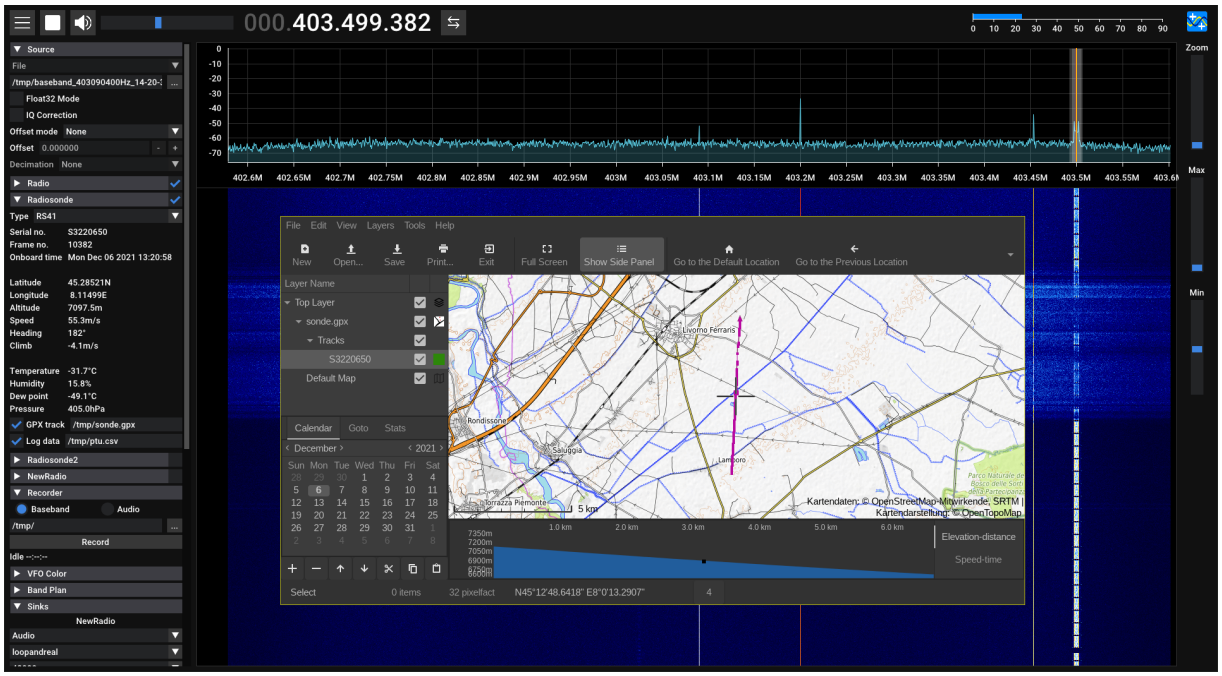
Not shown here are modules for

- Demodulating unencrypted Tetra transmissions
- Controlling SDR++ using an Arduino
- Inmarsat C demodulator

Radiosonde decoder plugin

You can include this module when building SDR++ from source in Linux. Like other community plugins, its development is not currently supported by the SDR++ team. The plugin takes sensor data broadcast from radiosondes such as weather balloons and plots the height and location.

You can obtain the instructions and source code from: https://github.com/dbdexter-dev/sdrpp_radiosonde (*Photo courtesy of Davide Belloli)*



Troubleshooting

SDR++ works without problems for the vast majority of users. Reading this manual can help you avoid getting a problem. But software defined radio can be complicated, especially for people new to it, and there are a lot of settings to get wrong! This is a collection of questions and answers about the problems some users experience, that we hope you find it helpful.

You can find further troubleshooting information in the SDR++ readme file and get help from other users on the SDR++ Discord Server help channels.

Audio issues

- I have no audio
- The sound from SDR++ 'stutters' and breaks up when I'm listening to a station
- I'm having problems with PipeWire in Linux

Look and feel

- The controls are too big/small on my Android device

Receiving

- I have audio but no signal where there should be one?
- There are signals shown in the FFT or waterfall but my VFO doesn't move to the peak when I tap/click to tune it?
- I have signals (above 14MHz) that shouldn't be there

Hardware and configuration

- SDR++ doesn't list my SDRPlay receiver as a source
- I'm having problems with SDR++ in Windows
- SDR++ won't run, and I only see a window flash for an instant
- How do I reset the SDR++ configuration back to when it was installed?

- My SDR++ has gone back to default configuration and lost its settings and/or bookmarks
- I'm worried that I will lose my stored frequency bookmarks when I update SDR++
- When I try to run SDR++ after installing it in Windows, it says vcruntime140.dll and msvcp140.dll are missing
- SDR++ crashes when starting a HackRF

I have no audio

Check:

- Is your SDR receiver connected and working?
- Have you pressed the 'Start' button?
- Is the volume slider turned up enough
- Is the AGC or gain set appropriately?
- Is the box ticked to enable your 'Radio' module or whichever VFO you are using?
- Do you have a 'sink' for the Radio/VFO you are using? Is this set to 'audio' and to the correct output e.g. the soundcard and not to a virtual audio cable?
- Is your computer audio control muted or set to the correct volume level?

The sound from SDR++ 'stutters' and breaks up when I'm listening to a station

This could be caused by a number of things:

- The resources on your device - like CPU usage - could be close to maximum. Try monitoring what the CPU usage is with Windows Task Manager - use CTRL+ALT+delete to start it. There are equivalents in other operating systems. Close down other programs or apps.
- You might be able to cure it by altering the bit rate in the 'Sink' for the VFO you are using. Try altering it to a different value like 48000 or 44100.
- Check you haven't got the squelch on.
- Check whether it does this for other transmissions - it might be the particular transmission and not SDR++.

In Debian based linux distros you'll usually need to apply this fix.

I'm having problems with PipeWire in Linux

Some people experience their audio freezing or artifacting when using PipeWire for audio on Linux distributions. If you add the line `PIPEWIRE_NOJACK=1` then this may stop this. Alternatively you can disable the Jack client altogether and use the Pulse audio layer only.

The controls are too big/small on my Android device

Use the High-DPI scaling control to adjust the settings so that these are a convenient size for you. You will need to restart SDR++ after adjusting this. If you choose a scale that makes the display too small, so it is impossible to navigate to the DPI scaling setting, then connect a mouse through an OTG cable attached to your device.

I have audio on my device/computer but no signal where there should be one?

Not all stations broadcast continuously - tune to a known continuous broadcast, for instance a local broadcast station or a 24 hour VOLMET station. If there is still no signal check the following:

- Is your antenna connected?
- Is the AGC or gain set appropriately?
- If you have an RTL-SDR dongle is direct sampling set to Q-branch (for stations below 28 MHz) or disabled (for VHF and UHF transmissions)?
- Have you selected the appropriate modulation e.g. AM or FM or lower/upper sideband etc.?
- Is your audio sink set to output to speakers?

There are signals shown in the FFT or waterfall but my VFO doesn't move to the peak when I tap/click to tune it?

Change the snap interval to a smaller one so you can use smaller tuning steps.

I have signals above 14MHz that shouldn't be there?

This is a known issue with RTL-SDR dongles operating in direct sampling mode. They repeat or 'mirror' strong stations from below 14MHz so they are still received above that

frequency. This can also reverse the sideband on SSB signals, so they are received on lower sideband in the 'image', when the original on the correct frequency is on upper sideband.

You could fit a high pass filter to cut out signals below 14 MHz, upgrade to an SDR receiver with better filtering, or buy an upconverter like the 'Ham it up'..

Alternatively, your receiver might be overloaded by strong signals from local stations, particularly from FM and AM broadcast stations. You can buy special filters to reduce these. SDRPlay receivers have these filters built in and they can be selected in SDR++.

SDR++ doesn't list my SDRPlay receiver as a source

The SDRPlay service has to be installed and running for SDRPlay devices to be picked up when SDR++ loads. If this happens:

In Windows

1. Ensure that you have SDRPlay selected in the Source menu.
2. Wait a few moments after SDR++ starts, then press 'refresh' in the Source menu to see if the receiver serial number appears. If that doesn't work...
3. Close SDR++
4. Ensure you have the SDRPlay drivers installed (see Appendix 1 for links to download these) then either
5. Start SDRUno, then close it down - this will start the SDRPlay service

   OR

6. In the Windows search box type 'Services' and open the App.
7. Find 'SDRPlay API Service' in the list. Right click on this and select 'Properties'.
8. In the Properties box
   a. change 'Startup type' to automatic and
   b. if 'service status' says it is not running, select 'Start'.

Your SDRPlay receiver should now be found when you restart SDR++.

In Linux, follow the instructions and videos for installation on the SDRPlay website. This involves both the installation of both the API and Soapy SDR .

I'm having problems with SDR++ in Windows

You can begin to analyse problems with less obvious solutions by starting SDR++ from the command line and examining the messages when SDR++ starts for the problem.

- In Windows File Explorer, navigate to the folder where you have installed SDR++
- Right click on the folder icon and select 'Open in Terminal'
- start SDR++ in logging mode `sdrpp.exe -c`

In Linux you can simply open a terminal and type the `sdrpp` command and the messages will be shown in the terminal window.

Look through the messages on screen for any obvious issues, such as modules that fail to load

SDR++ won't run and I only see a window flash for an instant

SDR++ uses Open-GL for it's display. All modern graphics drivers should support this. Update your graphics card drivers to the most up to date version.

How do I reset the SDR++ configuration back to when it was installed?

With SDR++ closed, delete any files with `config.json` in the name, that are in the directory where SDR++ is installed.

You should back up `frequency_manager_config.json` if you want to save your bookmarked frequencies.

When SDR++ restarts it will rebuild these files.

My SDR++ has gone back to default configuration and lost its settings and/or bookmarks

This can be caused if your PC or device, or the running instance of SDR++, crashes at the point that it is saving the configuration file or the frequency manager file. In those circumstances, these files can be corrupted. This is usually very rare. When SDR re-starts it sees that those files are corrupt and will overwrite them with a new default configuration.

You can either back-up those files regularly, or create regular restore points for your PC or device, to prevent loss of data.

I'm worried that I will lose my stored frequency bookmarks when I update SDR++

If you install a more up-to-date version of SDR++ in the same folder it overwrites some files from your currently installed version, but does **not** disturb the frequency manager configuration file. If you are worried you can either:

- export your frequency lists from within SDR++ to back them up, or
- Back up the `frequency_manager_config.json` file - which is in the folder in which you installed SDR++. For instance creating a copy and renaming it as something different such as `frequency_manager_old.json`

When I try to run SDR++ after installing it in Windows, it says vcruntime140.dll and msvcp140.dll are missing

These files are part of the Visual C++ Redistributable for Visual Studio and may not be present on your system. To solve this, you'll find the latest version for your system at Microsoft Visual C++ Redistributable latest supported downloads

SDR++ crashes when starting a HackRF

If you also have the SoapySDR module loaded (not necessarily enabled), this is a bug in libhackrf. It's caused by libhackrf not checking if it's already initialised. Remove the soapy source by using module manager.

Appendix 1 - Links to driver downloads for SDR receivers

Here are the links for drivers for supported SDR receivers:

- RTL-SDR dongles - https://zadig.akeo.ie/# - not needed because they are built into SDR++
- SDRPlay receivers such as the RSP1, RSP1A, RSPdx - https://www.sdrplay.com/softwarehome/
- Other receivers supported by SoapySDR
- Airspy SDR receivers including the Airspy HF+ Discovery - https://airspy.com/download/
- Adalm Pluto SDR -. https://wiki.analog.com/university/tools/pluto/drivers/windows
- RFSpace devices - http://www.rfspace.com/RFSPACE/Support.html
- BladeRF - https://www.nuand.com/win_installers/
- Lime SDRs - https://www.sdr-radio.com/limesdr There's also a guide to installing the drivers at https://wiki.myriadrf.org/LimeSDR-USB_driver_installation
- HackRF - The guides on developing and use the HackRF are at https://greatscottgadgets.com/sdr/1/

Credits and SDR++ online links

This user manual is written by John Donkersley G0OXO with contributions from Alexandre Rouma ON5RYZ. SDR++ is created by Alexandre Rouma.

SDR++ Online

https://sdrpp.org

SDR++ Github page

Contribute to Ryzerth on Patreon - get early access to modules, Android versions and information.

Follow SDR++ on social media:

Whatsthegeek on Twitter

Ryzerth YouTube channel

SDR++ also has a Discord server you can join to discuss SDR++ and all things radio