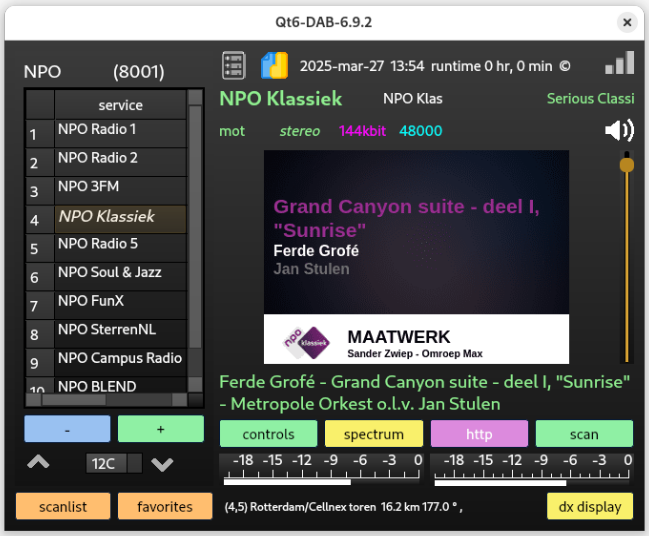
Using Qt-DAB 6.9[*]

Jan van Katwijk, Lazy Chair Computing
The Netherlands

March 29, 2025



Contents

1 Introduction

Qt-DAB is built with the idea that your favorite settings are not necessarily my favorite settings. Therefore Qt-DAB provides a high amount of settings. most settings have a reasonable default value and the software will work fine without altering these. It is obvious that changing the color of buttons or the color of scopes does not influence the working of the program.

This user's guide assumes no knowledge of Qt-DAB, in the first section the installation and the first stat up of the program are discussed. In later sections, more details are given on the (man) buttons and checkboxes.

2 Installing and running Qt-DAB

2.1 Installing Qt-DAB

For both Windows and Linux (x64) precompiled versions are available. These can be found in the releases section of the github repository.

`https://github.com/JvanKatwijk/qt-dab/releases/tag/Qt-6.9.2`

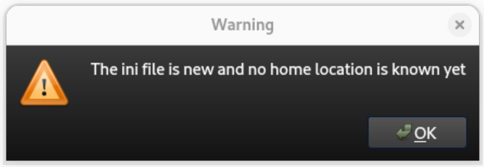
- For Linux, a so-called AppImage is available, after downloading it, it can be run as a normal program;
- For Windows, two installers are available, the versions differ in the support for RTLSDR based dsticks. Downloading and installing will create a link to the executabble program

Note that for the Linux version does not contain the device libraries, you have to install the library for the device(s) you use, yourself. In section 8 control of the configured devices is discussed, and some comments are made how and where to obtain the support libraries.

The window installers contain the device libraries for the configured devices, apart from the device library for the SDRplay devices, and the Adalm Pluto, These must be installed from the supplier of the device.

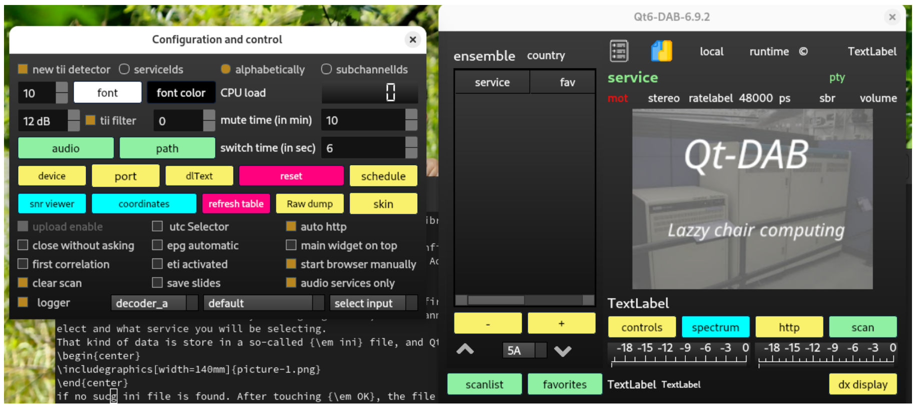
2.2 Starting for the first time

It will be obvious that on starting the program for the very first time, the software does not know what device you are going to use, what channel you want to select and what service you will be selecting. That kind of data is store in a so-called *ini* file, and Qt-DAB shows a small widget if no such file is found.



The ini file is new and no home location is known yet

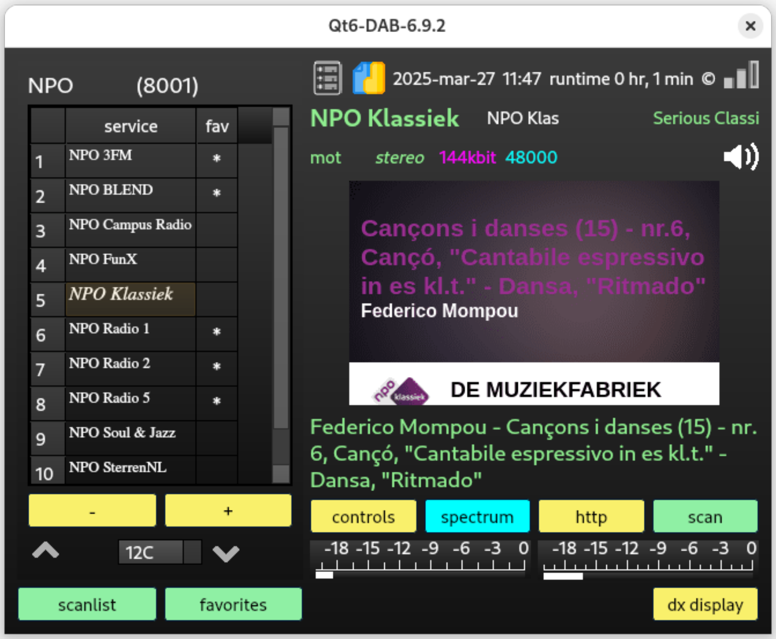
After clicking *OK*, such an ini file is created and two widgets are shown, the "main" widget and the so-called configuration widget. The latter, the configuration widget, left on the picture, has on the bottom line right (see picture below), a device selector, labeled *select input*, with which a device can be selected (the selection is recorded and shown the next invocation).



```
elect and what service you will be selecting.
That kind of data is store in a so-called {\em ini} file, and Q
\begin{center}
\includegraphics[width=140mm]{picture-1.png}
\end{center}
if no succ ini file is found. After touching {\em OK}, the file
```

After selecting a device, select a channel where DAB data is transmitted. On the picture below, channel 12C is selected. In most cases you will have to adjust the gain settings of the device before the channel data can be decoded.

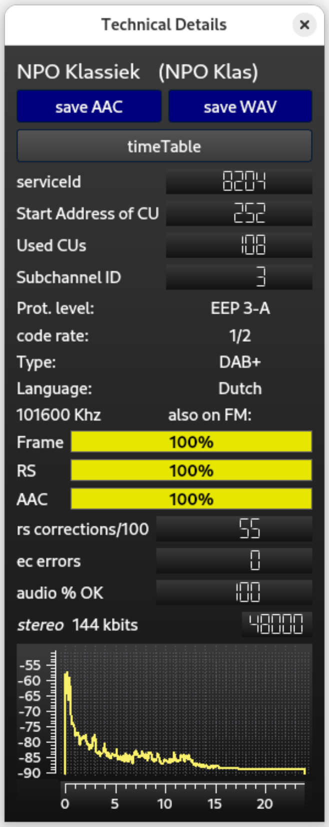
After selecting a device and a channel and setting the gain, you might see a list of services appear on the main widget. Select one by clicking on its name.



	service	fav
1	NPO 3FM	*
2	NPO BLEND	*
3	NPO Campus Radio	
4	NPO FunX	
5	*NPO Klassiek*	
6	NPO Radio 1	*
7	NPO Radio 2	*
8	NPO Radio 5	*
9	NPO Soul & Jazz	
10	NPO SterrenNL	

Federico Mompou - Cançons i danses (15) - nr. 6, Cançó, "Cantabile espressivo in es kl.t." - Dansa, "Ritmado"

It might happen that a service is selected but sound is not heard. If that happens, select the technical widget by clicking on the icon in the picture above the letter 'N' in *NPO KlassieK NPO Klas*. That widget, shown in the picture below, shows at the bottom the spectrum of the audio signal.



Technical Details

NPO Klassiek (NPO Klas)

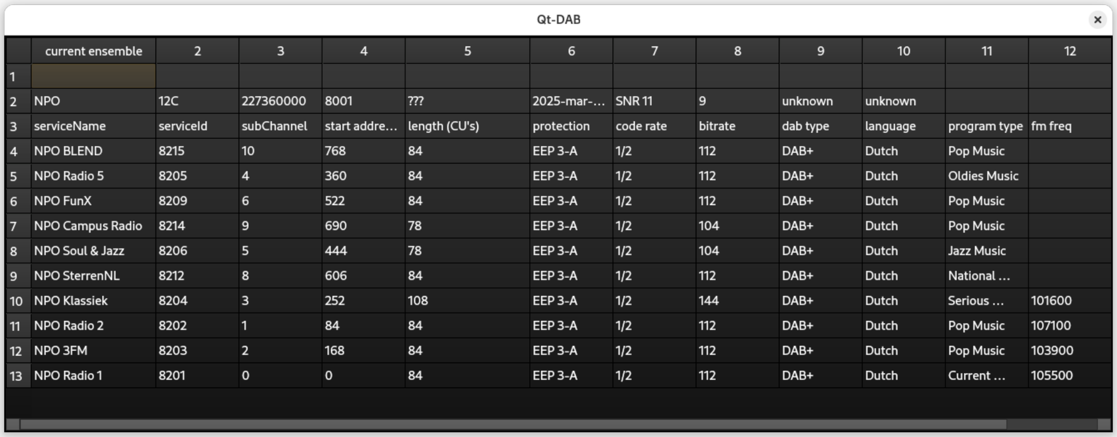
save AAC | save WAV

timeTable

serviceId	8204
Start Address of CU	252
Used CUs	108
Subchannel ID	3
Prot. level:	EEP 3-A
code rate:	1/2
Type:	DAB+
Language:	Dutch
101600 Khz	also on FM:
Frame	100%
RS	100%
AAC	100%
rs corrections/100	55
ec errors	0
audio % OK	100
stereo 144 kbits	48000

If no sound can be heard, but a spectrum is shown, then select a different audio channel on the configuration widget, bottom linem combobox labeled *default*. If no spectrum is shown, decoding the audio failed and the signal may be too weak. It sometimes helps to adjust the gain for the selected device.

To get an overview of the details of the services detected in the currently selected ensemble, touching the *name* of the ensemble above of the list of services (NPO (8001) in the pictures, shows the details on the so-called *content widget*.

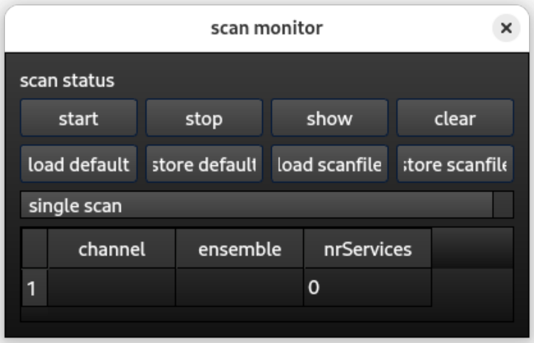


	current ensemble	2	3	4	5	6	7	8	9	10	11	12
1												
2	NPO	12C	227360000	8001	???	2025-mar-...	SNR 11	9	unknown	unknown		
3	serviceName	serviceId	subChannel	start addre...	length (CU's)	protection	code rate	bitrate	dab type	language	program type	fm freq
4	NPO BLEND	8215	10	768	84	EEP 3-A	1/2	112	DAB+	Dutch	Pop Music	
5	NPO Radio 5	8205	4	360	84	EEP 3-A	1/2	112	DAB+	Dutch	Oldies Music	
6	NPO FunX	8209	6	522	84	EEP 3-A	1/2	112	DAB+	Dutch	Pop Music	
7	NPO Campus Radio	8214	9	690	78	EEP 3-A	1/2	104	DAB+	Dutch	Pop Music	
8	NPO Soul & Jazz	8206	5	444	78	EEP 3-A	1/2	104	DAB+	Dutch	Jazz Music	
9	NPO SterrenNL	8212	8	606	84	EEP 3-A	1/2	112	DAB+	Dutch	National ...	
10	NPO Klassiek	8204	3	252	108	EEP 3-A	1/2	144	DAB+	Dutch	Serious ...	101600
11	NPO Radio 2	8202	1	84	84	EEP 3-A	1/2	112	DAB+	Dutch	Pop Music	107100
12	NPO 3FM	8203	2	168	84	EEP 3-A	1/2	112	DAB+	Dutch	Pop Music	103900
13	NPO Radio 1	8201	0	0	84	EEP 3-A	1/2	112	DAB+	Dutch	Current ...	105500

The content of the content widget can be saved as *.csv file* by double clicking on the widget (of course, single clicking on a service on that widget selects that service.

2.3 Scanning and favorites

The previous section showed how to start up for the very first time. If that succeeded, we know there are services. Of course, there may be more than a single channel with DAB data. In my environment I receive with a small whip antenna, DAB data on 8 channels, in total well over 60 services. Qt-DAB provides an excellent scanning service. Touch the button "scan" on the main widget. A small widget will appear that controls the scanning.



	channel	ensemble	nrServices
1			0

Scanning is in one of three modes, scanning is over all channels (although there are options to specify channels that are excluded):

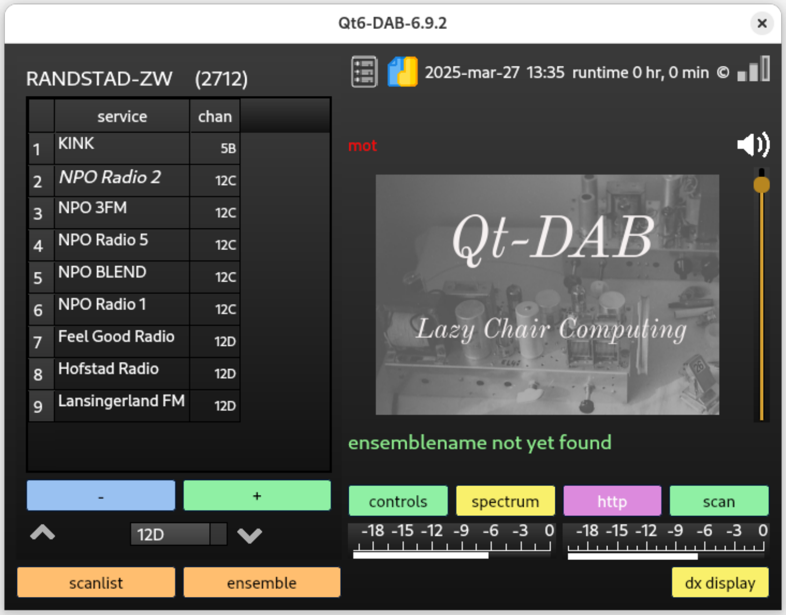
- single scan, does what the name suggests, it scans one time all channels, starting with channel 5A and up to 13F.
- Scan until data, scanning continues until a channel is detected that contains DAB data;
- continuous scanning, dies what the name suggests, it starts scanning until it is programmatically stopped.

The *single scan* option records all services detected in a *scanlist*. Touching the button labeled *scanlist* on the main widget shows the list of all services found controls the visibility of the list. Obviously, touching - with the mouse - an element in the list instructs the software to start the channel for that service and activate the service.

As said earlier, I receive well over 60 services but I'm only interested in a few of them (roughly speaking, about 8 services). That is why Qt-DAB supports the notion of *favorites* a shortlist of services that are (in this case my) favorites and can be selected by a simple mouse click.

Adding a service from the scanList is by clicking with the right hand mouse button on the name. Adding a service from the service list of a selected channel follows the same procedure.

The button *ensemble* on the bottom line of the main widget lets you choose between showing the list of services of the currently selected ensemble or the list of favorites (see picture below). If displaying the *favorites* is selected, the button is labeled with *ensemble*.



RANDSTAD-ZW (2712)

2025-mar-27 13:35 runtime 0 hr, 0 min

	service	chan
1	KINK	5B
2	*NPO Radio 2*	12C
3	NPO 3FM	12C
4	NPO Radio 5	12C
5	NPO BLEND	12C
6	NPO Radio 1	12C
7	Feel Good Radio	12D
8	Hofstad Radio	12D
9	Lansingerland FM	12D

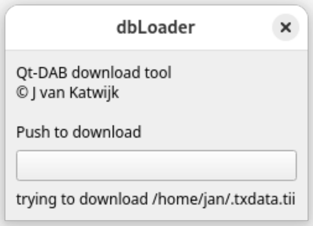
ensemblename not yet found

Of course selecting a service in the list of favorites might imply the need to switch to a different channel. When switching a channel, the software has to be resynchronized, and it will take a few seconds before the selected service is activated.

2.4 TII values and a map

As probably known, most DAB transmissions are done by more than a single transmitter, transmitting the same data. To distinguish between transmitters, the DAB signal contains usually a small amount of information identifying the transmitter whose signal is received and decoded. In order to decode that signal and show some details on the transmitter, a database needs to be available.

The download page referred to earlier contains a loader for the database, again, precompiled versions for Linux (x64) and Windows.



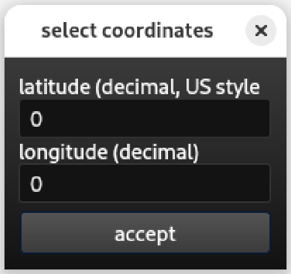
Qt-DAB download tool
© J van Katwijk

Push to download

trying to download /home/jan/.txdata.tii

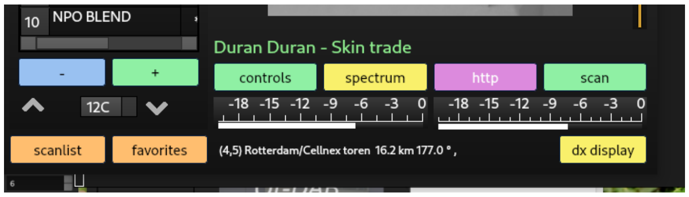
The loader has a single button, touching it will load the database.

Now recall that on the very first startup of the program, the message said that *”no home location is known yet”*. The software needs to know the coordinates of the home location to be able to compute azimuth and distance of the received transmitter(s) to your computer. The *configuration widget* contains a button (blue, labeled *coordinates*), that when touched shows a small widget where the coordinates can be entered.



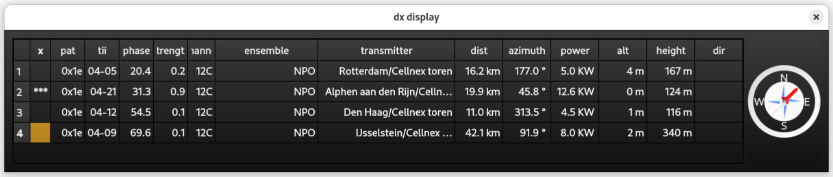
latitude (decimal, US style
0
longitude (decimal)
0

Once a database is loaded, and the home coordinates are filled in, Qt-DAB is able to show some data on the received transmitter, see picture below



(4,5) Rotterdam/Cellnex toren 16.2 km 177.0 ° ,

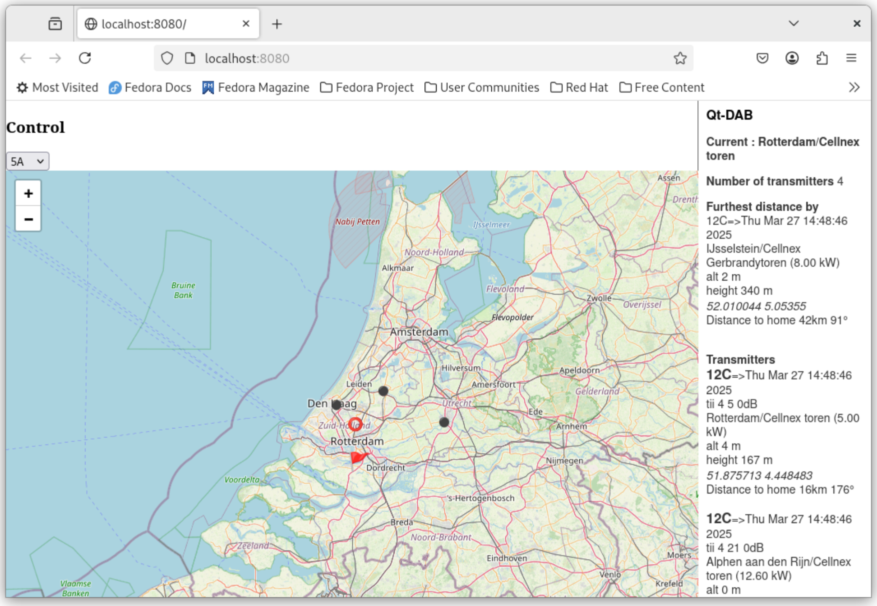
Since - depending on the ensemble and the receiver - data from more than a single transmitter is received, Qt-DAB tries to decode the TII data from each of them. Touching the button *dx display* shows (or hide) a separate widget where data from all receivers is shown, see below



	x	pat	tii	phase	trengt	ıann	ensemble	transmitter	dist	azimuth	power	alt	height	dir
1		0x1e	04-05	20.4	0.2	12C	NPO	Rotterdam/Cellnex toren	16.2 km	177.0 °	5.0 KW	4 m	167 m	
2	***	0x1e	04-21	31.3	0.9	12C	NPO	Alphen aan den Rijn/Celln...	19.9 km	45.8 °	12.6 KW	0 m	124 m	
3		0x1e	04-12	54.5	0.1	12C	NPO	Den Haag/Cellnex toren	11.0 km	313.5 °	4.5 KW	1 m	116 m	
4	🟧	0x1e	04-09	69.6	0.1	12C	NPO	IJsselstein/Cellnex ...	42.1 km	91.9 °	8.0 KW	2 m	340 m	

The picture shows that - here with a simple whip - data is received from 4 transmitters. The "***" in the first column shows the transmitter that is strongest.

Qt-DAB provides the possibility to show the locations of the transmitters on a map. Touching the *http* button on the main widget starts a small server and a webbrowser on the local machine.



Qt-DAB

Current : Rotterdam/Cellnex toren

Number of transmitters 4

Furthest distance by
12C=>Thu Mar 27 14:48:46 2025
IJsselstein/Cellnex Gerbrandytoren (8.00 kW)
alt 2 m
height 340 m
52.010044 5.05355
Distance to home 42km 91°

Transmitters
12C=>Thu Mar 27 14:48:46 2025
tii 4 5 0dB
Rotterdam/Cellnex toren (5.00 kW)
alt 4 m
height 167 m
51.875713 4.448483
Distance to home 16km 176°

12C=>Thu Mar 27 14:48:46 2025
tii 4 21 0dB
Alphen aan den Rijn/Cellnex toren (12.60 kW)
alt 0 m

The transmitters are clearly shown, as is my home location.

3 The main widget: Some more details

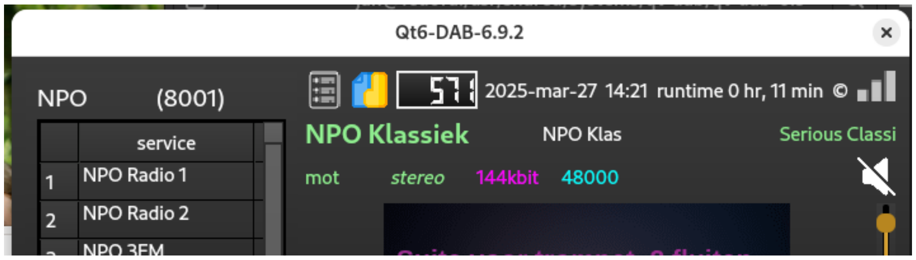
While we have seen some buttons on the main and the configuration widget, there are amy more.

Selecting services Below the list of services, the buttons labeled - and + can be used to selected the previous resp. next service in the list.

Channel selection The *up* and *down* marker left resp. right of the channel selector can be used to skip to the next (up) or previous (down) channel

Showing the directory where files are saved Touching the blue/yellow colored icon left to the date indication shows the folder (directory or map) where Qt-DAB stores the files it reads. Note that the earlier mentioned *.ini* file is stored in the user's home directory.

Muting sound and sound control The icon showing a speaker, when touched, tells the software to mute the audio output. If touched a small indicator appears that shows the amount of time the muting will continue.



The default duration for muting is 600 seconds, the value can be set on the configuration widget.

The volume of the audiooutput is controlled by the volume control slider, just below the "speaker" symbon.

The button labeled "controls" controls the visibility of the configuration widget. That widget was shown already and is discussed in detail in a later section.

The button labeled "spectrum" controls the visibility of the spectrum widget. That widget is described in a next section.

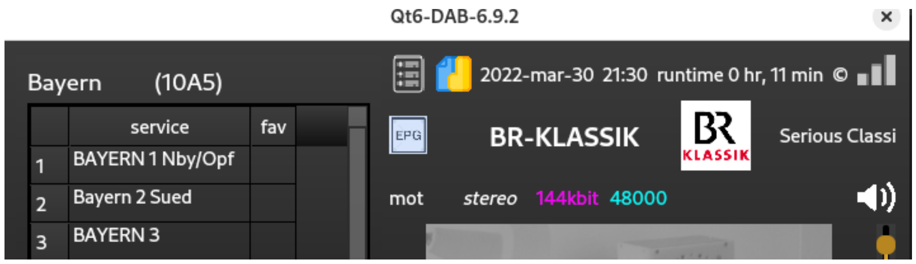
Removing favories Previously is described how to add a favorite to the list, deleting one is simply by clicking on the "*" to the right of the name of the service in the ensemble list.

Font selection and colors . Clicking, using the right mouse button on the name of the service as displayed above the slide window, shows (depending on the system being used) a selector for the colow with which the selected service name and the dynamic label text are displayed. The configuration widget contains selectors for the font, the font size and font color used for displaying the services in the service list.

Clicking, using the right hand mouse button on any of the buttons on the main widget displays a selector for (a) the background color of the button and (b) the text color on the button.

4 Icons and time tables

In some countries, the transmitted ensemble contains an SPI or EPG service. Such a service is - if detected within a specified number of seconds - start automatically as background service. If such a service is active[1], a small icon shows on the main widget as shown in the picture below

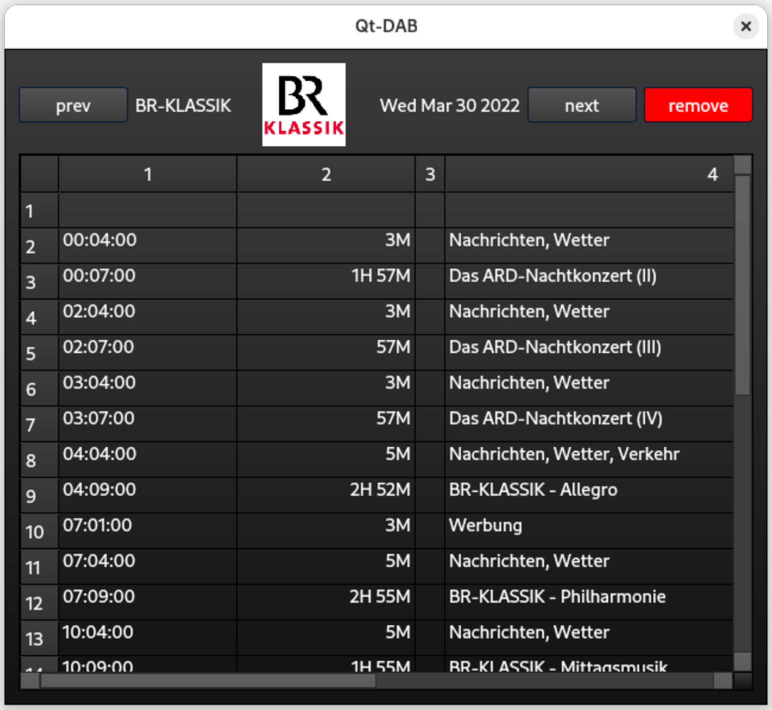


SPI data contains (usually) two components, icons as well as time tables for the services. The picture above shows that an icon was found earlier.

Assuming that a transmission was receiver earlier (or for a long enough time now), the EPG service might have recorded time table data. Touch the *timetable* button on the *technical widget* to show it.

The widget dynamically reads in the data for the service and the corresponding date. The *next* and *prev* buttons allow to change the date, one day up or down at the time. The *currently selected* page can be removed.

[1]Since the NPO does not send an SPI or EPG service in its ensemble, I am using recordings in this section

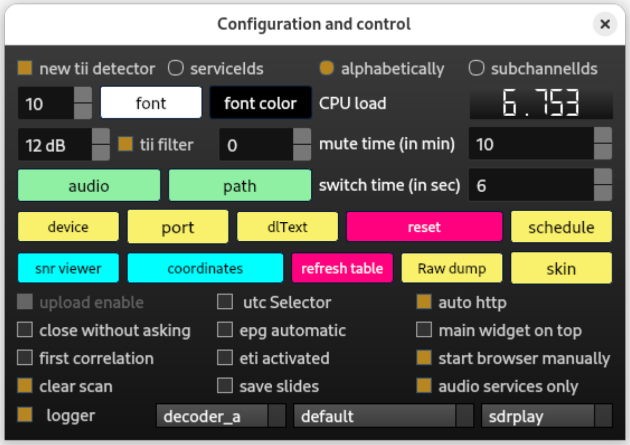

Qt-DAB

prev | BR-KLASSIK | Wed Mar 30 2022 | next | remove

	1	2	3	4
1				
2	00:04:00	3M		Nachrichten, Wetter
3	00:07:00	1H 57M		Das ARD-Nachtkonzert (II)
4	02:04:00	3M		Nachrichten, Wetter
5	02:07:00	57M		Das ARD-Nachtkonzert (III)
6	03:04:00	3M		Nachrichten, Wetter
7	03:07:00	57M		Das ARD-Nachtkonzert (IV)
8	04:04:00	5M		Nachrichten, Wetter, Verkehr
9	04:09:00	2H 52M		BR-KLASSIK - Allegro
10	07:01:00	3M		Werbung
11	07:04:00	5M		Nachrichten, Wetter
12	07:09:00	2H 55M		BR-KLASSIK - Philharmonie
13	10:04:00	5M		Nachrichten, Wetter
14	10:09:00	1H 55M		BR-KLASSIK - Mittagsmusik

5 Details on the configuration widget

Qt-DAB records (almost) all settings, and stores them in a file *.qt-dab-6.8.ini*.


Configuration and control

[✓] new tii detector | ○ serviceIds | ● alphabetically | ○ subchannelIds

10 | font | font color | CPU load | 6.753

12 dB | [✓] tii filter | 0 | mute time (in min) | 10

audio | path | switch time (in sec) | 6

device | port | dlText | reset | schedule

snr viewer | coordinates | refresh table | Raw dump | skin

[ ] upload enable | [ ] utc Selector | [✓] auto http

[ ] close without asking | [ ] epg automatic | [ ] main widget on top

[ ] first correlation | [ ] eti activated | [✓] start browser manually

[✓] clear scan | [ ] save slides | [✓] audio services only

[✓] logger | decoder_a | default | sdrplay

The *configuration and control* widget provides choices for settings, some are rather trivial, other more fundamentally. The settings are discussed line by line, from left to right.

Top line

- the checkbox labeled *new tii detector*. If selected a second TII detector/decoder is chosen. This new detector is more sensitive and gives - therefore - more false reports;
- the radioboxes *serviceIds, alphabetically, subChannelIds* determine the order in which the services in the service list are displayed

Line 2

- the spinbox with the value "10" tells the chosen fontsize of displaying the services in the service list and can be set to another value;
- the button labeled *font* - when touched - allows choosing the font for displaying the services in the service list;
- the button labeled *font color* - when touched - allows choosing the color of the font for displaying the services in the service list;
- the label *CPU load* precedes the number that shows the *overall* CPU load;

Line 3

- the spinbox with value "12 dB" selects the threshold value used for the TII decoder;
- the checkbox labeled *tii filter* allows choosing to use an additional filter for preventing the TII detector in recognizing false TII data;
- the checkbox labeled *0* allows settings a subId in detecting TII values;
- the label *mute time (in min)* allows selecting a duration muting will be active when selected, the default is set to 10 minutes;

Line 4

- the label *audio* allows selection of an audio backend, either using *portaudio* or using *Qt audio* classes. The effect of change is with the next program invocation;
- the label *path* allows selection of a directory path for files created by Qt-DAB. Default is the directory *Qt-DAB-files* in the user's home directory. Changes will be effective the next program invocation;
- the label *switch time (in sec)* precedes a spinbox where the time can be set. The switch time is the time yaken by the software to decide whether or not there is DAB data in the selected channel;

Line 5

- the button labeled *device* controls the visibility of the widget for device control;
- the vutton labeled *port* allows setting a different value for the port with which Qt-DAB communicates with the map server (if active);
- the button labeled *dlText*ed - instructs the software to save the text from the dynamic label into a file;
- the button labeled *reset* does what it suggests;
- the button labeled *schedule* allows scheduling actions of Qt-DAB, such as switching to a given service, on a specified time withing the next 7 days;

Line 6

- the button labeled *snr viewer* controls the visibility of a small separate widget that shows the progress of the SNR in time;
- the button labeled *coordinates* was seen already. It allows setting the user's home coordinates, for use with TII decoding;
- the button labeled *refresh table* loads a fresh instance of the database from the user's home directory into the running Qt-DAB program;
- the button labeled *Raw dump* - when activated - causes the software to dump the (partially processed) input into an ".sdr" file, which can be used as input in Qt-DAB;
- the button labeled *skin* allows selection of a "skin" for the appearance of the running program. The effect of choosing is visible in the next program activation.

Line 7

- the checkbox labeled *upload enable* is deactivated;
- the checkbox labeled *utc Selector*, when activated, displays the time as UTC rather than as local time;
- the checkbox labeled *auto http*, when activated, starts the http handler on startup of the program (of course only if a database is present and home coordinates are filled in);

Line 8

- the checkbox labeled *close without asking*, when activated, prevents a pop-up showing up, asking consent for closing the program;
- the checkbox labeled *epg automatic*, when checked, instructs the program to look for SPI or EPG type services on selecting a channel and start the service in the background;
- the checkbox labeled *main widget on top* does just what it suggests;

Line 9

- the checkbox labeled *first correlation* needs some explanation. Input to Qt-DAB may come from more than a single transmitter. Activating this checkbox ensures that syncing is with the first arriving data stream with a sufficient strong signal;
- the checkbox labeled *eti activated* starts the eti subsystem, that allows ETI files to be generated;
- the checkbox labeled *start browser manually*, when activated, prevents the system to start a webbrowser automatically;

Line 10

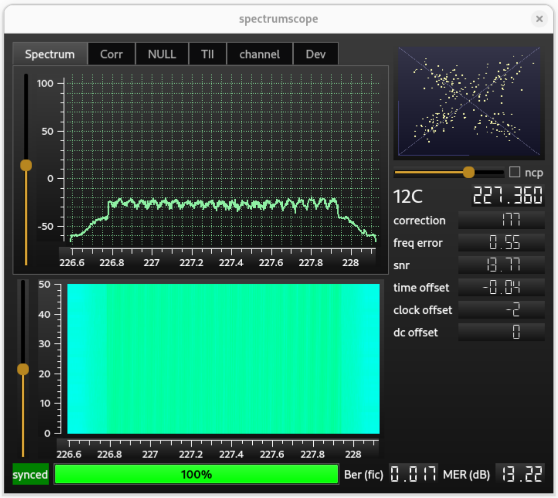
- the checkbox labeled *clear scan* clears the scan list when scanning is activated;
- the checkbox labeled *save slides*, when activated, instructs the system to save a copy of all slides received;
- the checkbox labeled *audio services only*, when activated, tells the system to show only audio services in the services list. If an EPG or SPI service is found, it will be listed as well.

Line 11

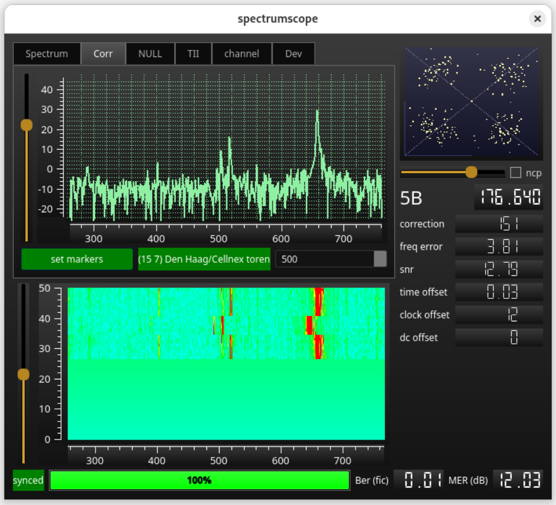
- the checkbox labeled *logger*, when activated, tells the system to start logging data in a log file;
- the combobox labeled *decoder_a* is currently de-activated;
- the combobox labeled *default* allows setting an audio channel;
- the combobox labeled *sdrplay* allows selecting a device.

6 The spectrum widget

As mentioned above, the main widget contains a button labeled *spectrum* that controls the visibility of the so-called *spectrum widget*. The spectrum widget (see figure below), contains tabs to switch between 6 views on (elements of) the incoming signal.



12C 227.360
correction 177
freq error 0.55
snr 13.77
time offset -0.04
clock offset -2
dc offset 0
Ber (fic) 0.017 MER (dB) 13.22



(15 7) Den Haag/Cellnex toren 500
5B 176.640
correction 151
freq error 3.81
snr 12.79
time offset 0.03
clock offset 12
dc offset 0
Ber (fic) 0.01 MER (dB) 12.03

The picture left shows - with this tab setting - the spectrum of the incoming signal, and - top right - the dots of the signal that result from decoding, ideally one sees 4 dots, one in each quadrant. The picture right shows - with another tab setting - the result of the correlation.

At the right hand side of the pictures one sees the channel and the channel frequency, and below that some values telling some things about the signal quality.

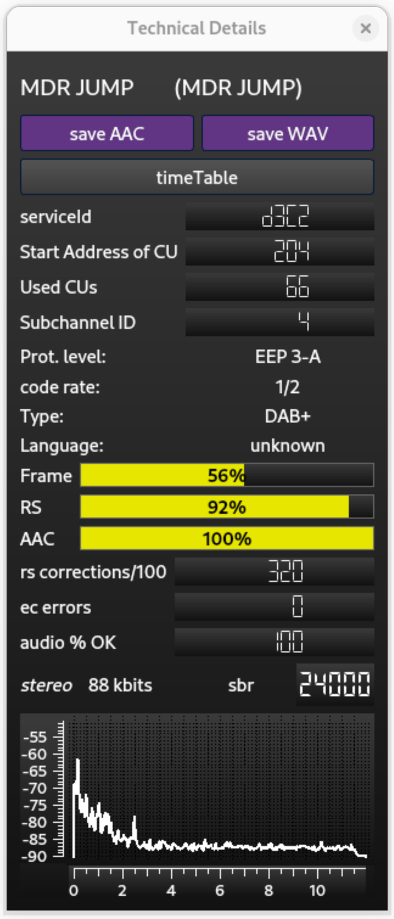
The bottom line contains also quality indications, where the green bar tells that decoding the catalog data (FIC data) could be handled well.

Selecting another tab gives a different view, views are:

- Spectrum view, as seen above;
- Correlation view, as seen above;
- the NULL view, showing the (time domain) samples at the end on the NULL period and the beginning of the first datablock;
- the TII view, showing (part of) the spectrum of the NULL period where the encoding of the TII data shows;
- the channel view, showing the effect of the channel on the incoming data;
- the deviation view, showing the deviation of the decoded elements compared to the ideal position.

7 The Technical Details Widget

The *Technical Details* widget was already seen before. It is used to show all kinds of details of the audio component of an audio service.



MDR JUMP (MDR JUMP)

serviceId	d3C2
Start Address of CU	204
Used CUs	66
Subchannel ID	4
Prot. level:	EEP 3-A
code rate:	1/2
Type:	DAB+
Language:	unknown
Frame	56%
RS	92%
AAC	100%
rs corrections/100	320
ec errors	0
audio % OK	100

stereo 88 kbits sbr 24000

The buttons right below the name of the selected service are

- the button labeled *save AAC* can be used to save the AAC intermediate output of the audio into a file. Such a file can then be processed by other programs;
- the button labeled *save WAV* can be used to store the audio output as a ".wav" file; the button labeled *timeTable*, we have seen before.

The next 4 numbers specify the identity of the service and tell the location in the DAB datastream where to find the data for that service; The next three tokens (EEP-3A, 1/2 and DAB+) tell how to interpret the data.

The three yellow bars tell about the actual decoding, basically, if they are all 100% yellow, it was possible to translate the DAB data into sound.

the three numbers below the yellow bars tell about the number of corrected errors, and the completeness of the audiostream.

This picture shows that the audio was in stereo, where the input data was 88 kbis, sbr (Spectral Band Replication) was used to enhance the signal, that was output on 24000 S/s (but converted by Qt-DAB to 48000 S/s)

8 Supported input devices

Qt-DAB supports a variety of input devices, the Adalm Pluto, the AIRspy, the hackrf, the LimeSDR, RT2832 based sticks and SDRplay RSP devices. Furthermore, there is support for the rtl_tcp server, support for the spyserver, for file input (raw, wav and xml), and for devices for which a *Soapy* interface library exists (Linux only).

Both the *appImage* and the *Windows installer* are configured with (almost) the whole range of devices: SDRplay RSP (different versions for the 2.13 and 3.XX library versions), the Adalm Pluto, the AIRspy, the hackrf, the LimeSDR, and - of course - the RT2832 based dabsticks.

8.1 The SDRplay RSP

Qt-DAB provides two different device handlers for the RSP's, one for devices using the 2.13 SDRplay interface library, the other one supports devices using the 3.XX SDRplay interface library. Note that under Windows, if API 3.10 or up is installed on Windows, the 2.13 library is not accessible. Note furthermore that the recently announced RSP 1B is supported, and the new RspdxR2 is supported when the library version 3.15 (or up) is installed (the picture shows that 3.14 was installed). For the RSDduo, the 3.XX software contains a switch for selecting the tuner on the RSPduo.

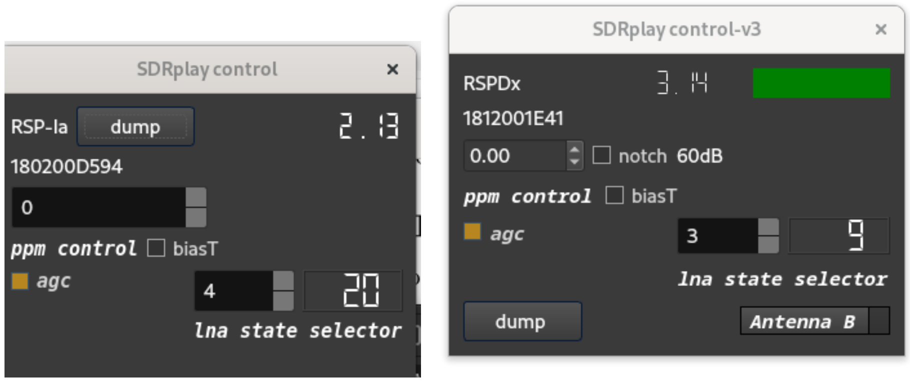


Figure 1: Qt-DAB: The two control widgets for the SDRplay

As figure 1 shows, the control widgets for the two different versions resemble each other, their implementation differs considerably though. Both have spinboxes for setting the *if gain reduction*, the *lna state* and a *ppm offset*.

Since Qt-DAB is capable of correcting the frequency, there is no actual need for setting a value here.

The spinbox for the *if gain reduction* is programmed to support the range of values between 20 and 59. The range of values for the *lna state* depends on the model of the RSP. The software detects the model and fills in the range accordingly.

If the *agc* is selected, the *if gain reduction* spinbox will be hidden, its value is then irrelevant.

The RSPdx (and its predecessor, the RSP II) has two (actually 3) slots for connecting an antenna. If an RSPdx or RSP II is detected, a combobox will be made visible for *antenna selection*.

Finally, both versions of the control widget contain a *dump* button. If touched, the raw input from the connected device will be stored in a so-called *xml formatted* file. First a menu is shown for selecting a filename, a suggestion for the name of the file *device name - date* is given. Touching the button again will stop dumping and the file will be closed.

Where to get the driver library The driver library for the SDRplay devices is proprietary software, the binary can be doenloaded from the sdrplay site (www.sdrplay.com).

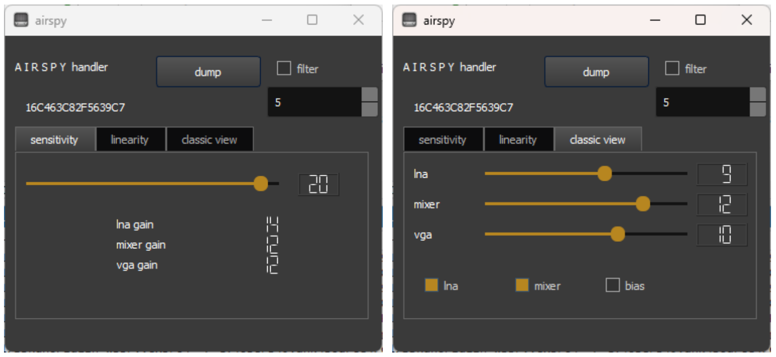


Figure 2: Qt-DAB: Widgets for AIRspy control

8.2 The AIRSpy

The control widget for the AIRspy (figure 2, right) contains three sliders and a push button. The sliders are to control the *lna gain*, the *mixer gain* and the *vga gain*.

To ease balancing the setting of the sliders, two combined settings are included in the widget, selectable by the tab *sensitivity* and *linearity*. Figure 2 left side, shows the setting at selecting the tab *sensitivity*.

Touching the button labeled *dump* instructs the software to dump the raw stream of samples into a file in the *xml format* (Note that while processing DAB requires the samplerate to be 2048000, that rate is *not* supported by the AIRspy, implying that the driver software has to do some rate conversion. The xml file though will just contain the samples on the rate before conversion).

If more than one airspy is detected, a widget will appear with which the device to use can be selected.

Where to get the driver library The driver library for most Linux systems can be found in the repositoty of the distribution.

8.3 The Hackrf

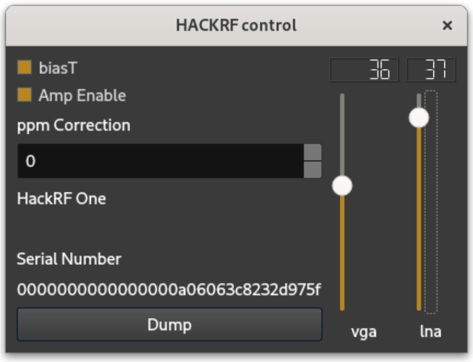


Figure 3: Qt-DAB: Widget for hackrf control

The control widget for hackrf (figure 3) shows, next to the Serial Number of the device, a few sliders, a few checkboxes, a spinbox and a push button.

- the *sliders* are there for controlling the lna and vga gain, the slider values are limited to the range of possible values;
- The *Ant Enable* checkbox is for Antenna port Power control (not used in this controller);
- The *Amp Enable* checkbox is - if enabled - for additional gain on the antenna input;
- the *ppm correction* spinbox can be set to correct the oscillator (on 227 MHz, the Qt-DAB software reports an offset of somewhat over 3 KHz);
- the *Dump* push button when pushed, starts dumping the raw input in xml file format. Touching the button again will halt the dumping and close the file.

Where to get the driver library Most Linux distribution have in their repositories a suitable driver library.

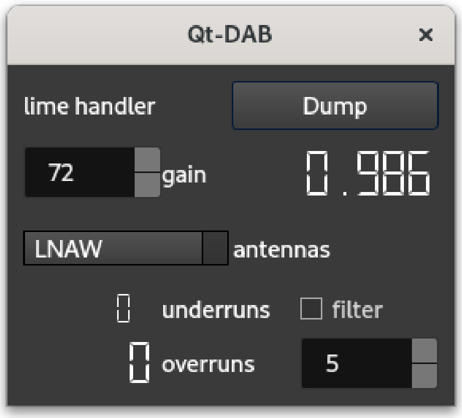


Figure 4: Qt-DAB: Widget for Lime control

8.4 The LimeSDR

On selecting the LimeSDR (if configured), a control widget for the LimeSDR is shown (figure 4). The widget contains five controls:

- *gain* control, with predefined values;
- *antennas*, where *Auto* is usually the best choice;
- *Dump*, if touched, the raw input from the connected device will be written to a file in the so-called xml format.

New is the inclusion of a filter. Note that the limeSDR reads samples with a samplerate of 2048KHz with no filtering, while the frequency distance between successive channels is 1712KHz. So, DX-ing in adjacent channels, where e.g. the first channel contains a strong DAB signal and the next one a weak, is difficult.

Therefore the control widget for the limeSDR has two additional controls,

- switching a software FIR filter on-off (the checkbox labeled *filter*),
- setting the filterdepth of the FIR filter (the spinbox below the checkbox).

Note that using the filter is not free, for a filter with a size of N, N * 2048000 complex additions and multiplications are performed. While on a modern PC that is not an issue, it certainly is on ARM based micros like the RPI 3.

Where to get the driver library The driver library is best compiled from sources, see "https://wiki.myriadrf.org/Lime_Suite" fopr details.

8.5 The RTLSDR stick

On selecting the dabstick (i.e. RT2832 based devices) (if configured), a control widget for the device appears (figure 5).

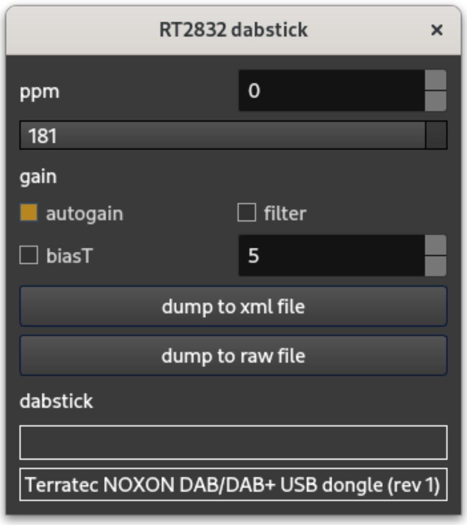


Figure 5: Qt-DAB: Widget for rtlsdr device

The widget contains just a few controls:

- a *spinbox* for setting the ppm. Note that on average the offset of the oscillator with DABsticks is (much) larger than that with devices like the SDRplay. The DAB software is able to correct frequencies to up to app 35 KHz, for some sticks the frequency error was large and correction using the ppm setting was required.
- a *combobox* for setting the gain. The support software for RT2832 based devices generates a list of allowable gain settings, these settings are stored in the ".ini" file;
- a *combobox* for setting the autogain on or off;
- a *push button* that, when touched, will instruct the software to dump the raw input in the aforementioned xml format. At first a menu appears for selecting a file. Touching the button again will stop dumping and close the file.

New is the inclusion of a *filter*. Note that the DABstick reads samples with a speed of 2048S/s for a signal with a bandwidth of app 1.536 MHz, while the frequency distance between successive channels is 1712KHz. So, DX-ing in adjacent channels, where e.g. the first channel contains a strong DAB signal and the next one a weak, is difficult.

The controller therefore contains an optional FIR filter, for which the rtlsdr control widget has two additional controls:

- switching a software filter on-off (the checkbox labeled *filter*),
- setting the size of the FIR filter (the spinbox below the checkbox).

Note that switching the filter on is not for free, for a filter with a size of N, N * 2048000 complex additions and multiplications per second are performed. While on a modern PC that is not an issue, it certainly is on ARM based micros like the RPI 3.

If more than one connected RTLSDR based device is detected, a widget appears on which the device of choice can be selected.

Where to get the driver library While distributions as e.g. Ubuntu provide a shared library for supporting the dabsticks, it is advised to compile the library from the sources. The precompiled versions in the distributions require to "blacklist" some kernel modules. See "https://osmocom.org/projects/rtl-sdr/wiki/Rtl-sdr" for details on building a shared library.

8.6 The Pluto device

When selecting *luto*, a widget (figure 6) appears with a spinbox for selecting the gain, and a checkbox for selecting the agc. If *agc* is enabled, the spinbox for the gain setting is invisible. The widget contains furthermore three buttons:

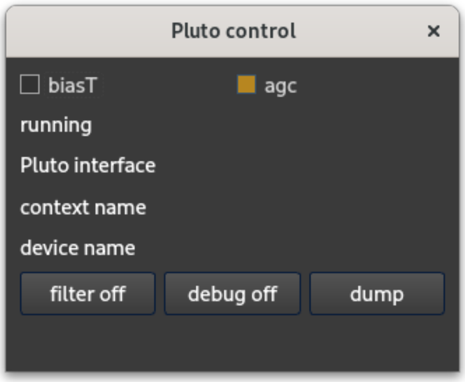


Figure 6: Qt-DAB: Widget for Adalm Pluto device

- the *debug control* button, when activated, instructs the software to show output on each step in the initialization process (note that the setting of the debug button will be maintained between invocations);

- the *dump* button will cause the original input - with a samplerate of 2100000 samples per second[2] - to be stored in an xml file.
- the *filter* button. The adalm pluto has as option specifying a fir-filter, to be executed within the Pluto device. This implementation of the controller for pluto will load a predefined filter onto the Pluto device which is enabled by default. With the filter button the filter can be disabled or enabled. Note that the button text indicates the action when touching, not the current state.

Where to get the driver library For the Adalm Pluto, libraries are available for Linux and Windows, see "https://wiki.analog.com/university/tools/pluto/users" for details.

8.7 Support for Soapy (Linux only)

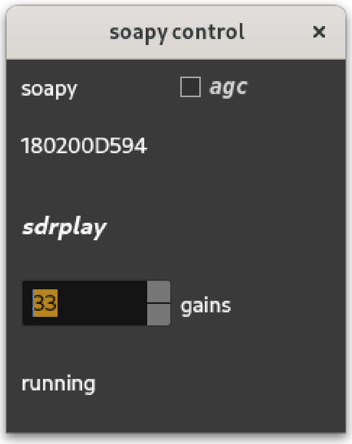


Figure 7: Qt-DAB: Widget for soapy

Soapy is a generic device interface, a kind of wrapper to provide a common interface to a whole class of devices. Qt-DAB supports Soapy, and its use is tested with the Soapy interface for SDRplay, hackrf and rtlsdr device.

The widget for soapy control (see figure 7) when applied to the soapy interface for the SDRplay contains simplified controls, compared to the regular controls for the SDRplay.

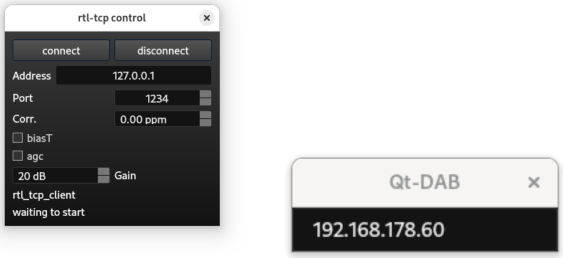
Note however, that to use Soapy for interfacing a specific device, a soapy device interface should have been installed for that device.

[2]The smallest samplerate that pluto gives is slightly larger than the required 2048000, 2100000 is chosen since it is easy to handle

8.8 rtl_tcp

rtl_tcp is a server for rtlsdr devices, delivering 8 bit IQ samples (i.e. 2 bytes per sample).

In the small widget (see figure 8) the ip address of the server can be given. Since the default port for the server is 1234, that port number is the one used by the client. After clicking the *connect* button, the client will look for a server and pass some parameters.



Address 127.0.0.1
Port 1234
Corr. 0.00 ppm
20 dB Gain
rtl_tcp_client
waiting to start

192.168.178.60

Figure 8: Qt-DAB: Widget for rtl_tcp

However, the port number can be set in the ".ini" file, by setting

```
rtl_tcp_port=XXX
```

where XXX is to be replaced by the portnumber of choice.

8.9 Input from a spyServer

Spyservers are well known, they provide a common interface to the remote use of either an AIRspy device or an RTL2832 based dabstick (see figure 9).

On starting the spy server interface asks for an IP address to be filled in, the port used is 5555.

Qt_DAB offers two versions of the spyserver, one for 8 bit output and one for 16 bit output. Of course, transmission of DAB input requires quite some bandwidth, for 16 bit data, and an input rate of 2500000 (as for the AIRspy), with 4 byte samples (2 x 2), the transmission rate is over 10 MByte/second. Since data is sent as packages, with a (small) header, the actual rate is slightly over 10 Mbyte. Of course, the using the 8 bit variant reduces it to app half.

8.10 File input

Qt-DAB supports both *writing* raw input files and *reading* them back. Writing a file as PCM file is initiated by the *Raw dump* button on the main GUI, writing a file as xml file by the *dump* button on the various device widgets. Qt-DAB differentiates between reading

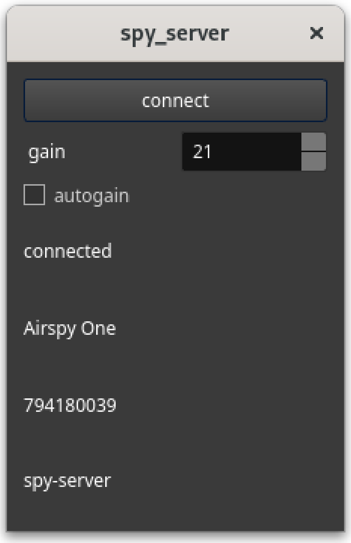


Figure 9: Widget for spyserver

- raw 8 bit files as generated by e.g. Osmocom software (usually files with an extension ".raw" or ".iq");
- PCM (i.e. ".wav") files, provided the data is 2 channels , with 16 values,and with a samplerate of 2048000, generated by Qt-DAB and with an extension ".sdr";
- xml files. The xml file format was defined by Clemens Schmidt (author of QIRX) and me and aims at saving files in the original format, so to allow easy exchange between different DAB decoder implementations. In order to support proper decoding of the contents, the data in the file is preceded by a detailed description in xml, hence the name xml file format.

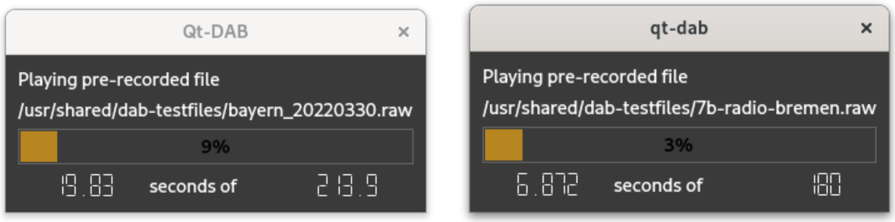
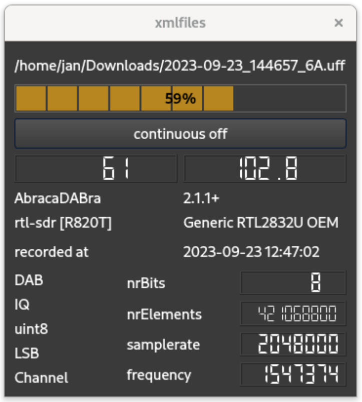

Qt-DAB
Playing pre-recorded file
/usr/shared/dab-testfiles/bayern_20220330.raw
9%
19.83 seconds of 213.9
qt-dab
Playing pre-recorded file
/usr/shared/dab-testfiles/7b-radio-bremen.raw
3%
6.872 seconds of 180

Figure 10: Qt-DAB: Widgets for file input

When selecting file input ".raw" or ".wav", a simple widget is shown (figure 10), with as indication the number of seconds the file is being played.

Since processing an xml file implies some interpretation, the widget (figure 11) for control when reading an xml file is slightly more complex. It contains - next to the progress in reading the data - a description of the contents of the file. So, the program that generated the file as well as the device used in that program are displayed, the number of bits of the samples, as well as the number of elements is displayed as is the samplerate of recording and the frequency of the recording.



/home/jan/Downloads/2023-09-23_144657_6A.uff

61 102.8

AbracaDABra	2.1.1+
rtl-sdr [R820T]	Generic RTL2832U OEM
recorded at	2023-09-23 12:47:02

DAB	nrBits	8
IQ	nrElements	421068800
uint8	samplerate	2048000
LSB		
Channel	frequency	1547374

Figure 11: Qt-DAB: Widget for xml file input

Touching the *cont* button will instruct the software to restart reading at the beginning of the segment in the file after reaching the end.

8.11 Other devices

Interfacing another device to Qt-DAB is not that complex, the interface contains a handful of functions that are to be implemented. Interfacing is described elsewhere. The sourcetree of Qt-DAB contains code for a few device handlers, these handlers are - since I do not have access to the devices for which they are written - untested. While the uhd device seemed to have been working with the included device handler, the device handlers for both the elad-s1 and the colibri are very experimental and not tested. Anyway, I am always interested to experiment further with these devices.