# Table of Contents

## Part One: Overview

I Introduction and Overview .................................................................................................................. 4  
A) What is a TV Server? .......................................................................................................................... 4  
B) Tuning the TV Server ....................................................................................................................... 5  
C) Transcoding Methods ....................................................................................................................... 5  
1) Repackaging the Stream .................................................................................................................. 5  
2) Transcoding Video ........................................................................................................................... 5  
3) Scaling the Video Image Size .......................................................................................................... 6  
4) Transcoding Audio ........................................................................................................................... 6  
5) Deinterlacing ..................................................................................................................................... 6  
D) How it All Works ............................................................................................................................... 7  
1) gstreamer-1.0 and http-launch ....................................................................................................... 7  
2) rtranscode ....................................................................................................................................... 7  
3) The Channel Database ..................................................................................................................... 8  
4) Kweb GUI ......................................................................................................................................... 8  
5) Web Interface ................................................................................................................................... 9  
6) TV Server Backends .......................................................................................................................... 10  
E) Installation ......................................................................................................................................... 10  
1) Required Hardware ......................................................................................................................... 10  
2) Required Operation System and Software ...................................................................................... 11  
3) Download and Install rtranscode .................................................................................................... 11  
4) Package Contents ............................................................................................................................. 12  
5) Recommended Software .................................................................................................................. 13  
F) About this Software Package .......................................................................................................... 13  
1) History ............................................................................................................................................. 13  
2) Caveats ............................................................................................................................................ 14  
3) Support ........................................................................................................................................... 15  

## Part Two: Reference

II Using rtranscode .................................................................................................................................. 16  
A) Simple mode ....................................................................................................................................... 16  
1) Using 3 arguments and any number of options ............................................................................... 16  
2) Getting arguments from a file ......................................................................................................... 17  
3) Starting with a named channel ........................................................................................................ 17  
4) Analyze stream uri .......................................................................................................................... 17  
5) Analyze stream uri and start the stream ......................................................................................... 18  
6) Add channels to a channel database from a text file containing stream URLs ........................... 18  
7) Getting help ..................................................................................................................................... 19  
B) Menu mode ....................................................................................................................................... 19  
1) Main and streaming menu .............................................................................................................. 19  
2) Recording transcoded streams ....................................................................................................... 20  
3) Sub-menus ....................................................................................................................................... 21  
C) Program Options ............................................................................................................................. 23  
1) Selecting a configuration file .......................................................................................................... 23  
2) Selecting a channel database ......................................................................................................... 24  
3) Set audio bitrate ............................................................................................................................. 24  
4) Set video bitrate ............................................................................................................................... 25  
5) Set output image size for SD videos ............................................................................................... 26  
6) Set output image size for HD videos .............................................................................................. 26
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>VII Create a kweb GUI</td>
<td>51</td>
</tr>
<tr>
<td>VIII Tutorial: Create a TV Server Backend</td>
<td>55</td>
</tr>
<tr>
<td>A) Introduction</td>
<td>55</td>
</tr>
<tr>
<td>1) Overview</td>
<td>55</td>
</tr>
<tr>
<td>2) Required Components</td>
<td>56</td>
</tr>
<tr>
<td>3) Exploring your DVB hardware</td>
<td>56</td>
</tr>
<tr>
<td>B) Channel Search</td>
<td>57</td>
</tr>
<tr>
<td>C) Create a GnuTV Backend</td>
<td>59</td>
</tr>
<tr>
<td>1) Create the GUI</td>
<td>59</td>
</tr>
<tr>
<td>2) Create a Channel Database for rtranscode</td>
<td>62</td>
</tr>
<tr>
<td>D) Create a MuMuDVB Backend</td>
<td>63</td>
</tr>
<tr>
<td>1) Create the MuMuDVB Configuration files</td>
<td>63</td>
</tr>
<tr>
<td>2) Create the GUI</td>
<td>67</td>
</tr>
<tr>
<td>3) Create a Channel Database for rtranscode</td>
<td>70</td>
</tr>
<tr>
<td>E) Organizing Your Channel Database</td>
<td>73</td>
</tr>
<tr>
<td>Appendix</td>
<td>74</td>
</tr>
<tr>
<td>IX Copyright, License and Source Code</td>
<td>74</td>
</tr>
</tbody>
</table>
I Introduction and Overview

My rtranscode V. 4 TV package for the Raspberry Pi contains tools to transcode HTTP or UDP TS video streams from a local TV server to a lower bitrate (video and audio) h264 encoded HTTP MKV stream (optionally reducing the image size as well), suitable for sending across a low bandwidth connection (through your internet connection, for example). You might use it to watch your home TV from your working place or on your smartphone or let your friends share your TV viewing pleasure. You can also use it to record transcoded streams.

It makes use of the hardware decoders and encoders built into the VC4 GPU.

A) What is a TV Server?

A TV server is a combination of hardware (TV receiver and tuner) and software, which takes the output of the (tuned) receiver and sends it as a http or upd stream (other formats are also possible) across your network. These streams can be watched with a lot of different media players running on any computer on your local network. Some TV servers can send all channels of a transponder (frequency) at the same time, which means that you can watch different channels at the same time on your network. On a Raspberry Pi you can use omxplayer, omxplayerGUI (part of my kweb suite), kodi or a hardware accelerated version of VLC to watch the TV streams.

The source may be terrestrial (DVB-T(2), ATSC), cable (DVB-C) or satellite (DVB-S(2)) digital TV. Analogue TV is not supported, except if the receiver converts it to a digital MPEG TS stream.

A TV server can be realized as a dedicated hardware device (Eingma2 Boxes, HD Homerun systems ...) or run on any computer which contains a TV tuner card or is connected to a USB-DVB-T/C/S receiver. The required software can be a complex all-in-one solution like tvheadend or VDR or a more simple backend like GnuTV or MuMuDVB.

A Raspberry Pi (2/3) is well suited as a TV server if you connect it to a suitable (linux compatible) USB-DVB device. The rtranscode V4 TV package contains a number of tools, which let you easily build your own server backend using either GnuTV or MuMuDVB.

The TV server and the transcoding server can run on the same Raspberry Pi.
B) Tuning the TV Server

The transcoding server requires an already running DVB TS stream as input. This means, that the TV server has to be tuned to a certain channel (or group of channels) and has started to deliver the stream.

Some systems (Enigma2-Boxes, tvheadend, VDR) support auto-tuning. If you access the TV server with a certain URL (matching the required channel), the server will tune the channel and send the stream on the same HTTP connection or redirect to the real stream. The transcoding server will usually work with such systems and if it doesn't work reliably you can use a “pre-run” option.

Other systems (HD Homerun) require a separate tuning command. Simple backends like GnuTV and MuMuDVB require the backend to be started as a separate process, before it can be used as the source of the transcoding server. Rtranscode V. 4 supports auto-tuning by additional user scripts. Scripts for GnuTV and MuMuDVB backends are supplied with the package.

C) Transcoding Methods

MPEG TS TV streams may use bandwidths of up to 15/16 Mbit/sec. If they contain multiple programs (streaming a whole transponder), the bandwidth may reach up to 80 Mbits/sec. This is far too much to send the stream across most internet connections or to send a stream to multiple clients on a shaky WiFi network. The transcoding server helps to reduce the bandwidth using different methods.

1) Repackaging the Stream

TS streams often contain multiple audio streams and additional streams like EPG information, Videotext etc. The transcoding server can reduce this to a more simple stream containing only the video and one audio stream (no real transcoding involved). This method can also be used to convert a unicast-udp stream (e. g. from GnuTV) to a http stream which can be accessed by multiple clients.

2) Transcoding Video

Many TV SD streams use MPEG2 encoding. Transcoding this to to H264 will require much less bandwidth for the same quality. The video bandwidth can be reduced by a factor of 4 – 10 this way.
But even if the original stream is encoded in H264 (some SD streams and most HD streams), transcoding to H264 with a lower bitrate is possible without too much loss in quality.
3) Scaling the Video Image Size

A further bandwidth reduction can be achieved, if the video image area is scaled down to a lower resolution, which in turn requires a lower bandwidth.

1080i HD video has always to be scaled down, because the VC4 GPU cannot decode and encode these streams to the original image size. This is a limitation of the hardware.

**Important Note:** `rtranscode 4.0` will run on both Raspbian Jessie and Stretch (and even on Wheezy, I suppose), but there is a huge difference. On Jessie, hardware accelerated scaling by the GPU can be used, which is missing in the matching gstreamer-omx module on Stretch, where only software scaling can be used and this is heavy work for the CPU, especially for HD streams, and limits what can be done. Transcoding will never be as effective on Stretch as it was on Jessie, except if perhaps some good C++ programmer implements HW scaling again.

4) Transcoding Audio

You always have the choice to include one of the original audio streams or to transcode it to a lower bitrate as low as 32 Kbit (mono) or 64 Kbit (stereo), which will further reduce the overall bandwidth.

By combining all these methods, stream bitrates even below 512 Kbit/s are possible.

5) Deinterlacing

SD streams and 1080i HD streams are usually interlaced, which means, that two “half images” are sent with the double FPS rate, one containing the even and the other containing the odd lines of the image. The transcoded stream is always a progressive stream. Therefore it is possible to get interlace effects, especially with fast moving content (e. g. on sport channels).

The transcoding server can optionally deinterlace the original stream using different algorithms. If you use a “double” interlacing method, the FPS is effectively doubled. A 25 FPS stream is sent as 50 FPS progressive stream built from the “half images”. But this is only possible for SD channels. For interlaced HD channels only “single” methods are possible, which effectively reduce the vertical resolution.

**Important note:** Deinterlacing always requires software scaling and both together put a high strain on the CPU. It may not work in all combinations.
On movie channels deinterlacing is usually not required because the original material is not interlaced at all.

D) How it All Works

1) gstreamer-1.0 and http-launch

The transcoding server is based on gstreamer-1.0, a lot of its plugins and also needs a matching gstreamer-omx module to use the GPU of the Raspberry Pi for video decoding and encoding.

http-launch is a small application originally written by Sebastian Droege with a few additions by myself. It takes a specially constructed gstreamer pipeline as input and serves its result as a http stream. A typical http-launch command may look like this:

```
http-launch 9080 /xyz.mkv video/x-matroska silent souphttpsrc
location="http://192.168.0.34:9082/bysid/28106" is-live=true keep-alive=true do-timestamp=true retries=10 typefind=true blocksize=16384 ! tsdemux parse-private-segments=false program-number=-1 name=demux demux.audio_0066 ! queue ! mpegaudioparse ! mpg123audiodec ! audioconvert dithering=0 ! audio/x-raw,channels=2 ! voaacenc bitrate=65536 ! matroskamux name=stream streamable=true demux. ! queue ! mpegvideoparse ! omxmpeg2videodec ! video/x-raw,width=720,height=576 ! deinterlace mode=interlaced fields=all method=greedy1 tff=tff ! omxh264enc target-bitrate=1310720 control-rate=true ! h264parse ! queue ! stream.
```

It will serve the resulting video stream on http://localhost:9080/xyz.mkv

You can access it from any computer on your network if you replace “localhost” with the IP or host name of your Raspberry Pi.

Of course nobody wants to use it in this way and the gstreamer pipeline has to be specifically constructed for each TV channel source. That’s what rtranscode can do for you.

2) rtranscode

rtranscode is a kind of Swiss knife for transcoding. To start the transcoding server for the same stream uri as above you can simply run:

```
rtranscode -t=http://192.168.0.34:9082/bysid/28106
```

This will analyze the stream, build the gstreamer pipeline and finally start the transcoder with the default settings. In the Terminal you will see the following:

```
Das Erste=http://192.168.0.34:9082/bysid/28106 sd1 mpeg:0x66,mpeg:0x67,ac3:0x6a
Starting to transcode
Size: 720x576  VBR: 1.1M  ABR: original MPEG
Deinterlace: off Scaler: OMX
Listening on http://127.0.0.1:9080/xyz.mkv
Starting pipeline
```
If you only need the transcoder from time to time, this may be the way to use it. But rtranscode can do a lot more. It's a command line program, but can run as a curses menu application in which you can set all possible options and select the channel to transcode interactively.

3) The Channel Database

To use rtranscode in interactive curses mode or control transcoding from a (k)web interface, you have to create a channel database file first. This is a simple text file which contains lines like this:

```
Das Erste=http://192.168.0.34:9082/bysid/28106 sd1 mpeg:0x66,mpeg:0x67,ac3:0x6a
```

rtranscode includes a number of tools which help you to create the channel database. Running for example

```
rtranscode -g=http://192.168.0.34:9082/bysid/28106
```

will return something like the line above.

If your TV server backend can deliver an m3u playlist of all its channels, you can download this playlist and create a channels database from it:

```
rtranscode -i=playlist.m3u -o=channels.dat
```

4) Kweb GUI

Once you have created a channel database, you can run “create_kpages.py”, which will create a kweb project.

This is a special kind of local application interface, which runs in the kweb browser (part of my kweb suite, which also includes omxplayerGUI).
Kweb is a very special web browser and application interface, which can be used to create simple HTML interfaces for local programs without having to create and run a server backend. It can be used as a kind of a simple GUI builder. Kweb is also deeply connected with omxplayerGUI: Clicking on any link to a media file or stream or playlist, will automatically open the video player.

An image sometimes says more than a thousand words and the following image shows the kweb transcoder interface and the video player in action:

5) Web Interface

While the kweb GUI can only be used locally on the Raspberry Pi running the transcoder software, the “real” web interface can be accessed from any computer on your network or even from the internet, if you use port forwarding and a dynamic DNS address.

This requires a real web server which is written in Python and uses the Karrigell framework. It supports user management for different user types. If you connect from your local network, you have full control: Select a channel, set all stream options, start (optionally including tuning), restart and stop the transcoding server. If you connect from the Internet, the server will request user name and password. Depending on the user type it will give limited (only watch or watch and restart transcoder streams) or full control.
If your configuration contains a user script for tuning your TV server backend, the server contains additional tuning options, which you can use to tune and watch the original streams. This is only available on the local network (but auto-tuning will also work from the internet).

6) TV Server Backends

If you don't have a TV server backend (hardware and/or software) running already or want to try something new, you can build your own server backend using either GnuTV (from the linux dvb-apps package) or MuMuDVB. This requires a working USB-DVB receiver.

As a first step you have to create a channels config file for your device using w-scan. Then you will run a number of scripts supplied by the rtranscode V4 package, which will create all configuration files, a kweb interface, a channel database for rtranscode and user scripts for automatic tuning.

For both backends this manual contains a complete tutorial.

**Important note:** If you are planning to add a USB TV receiver to your Raspberry Pi, you should carefully check, if the device is supported by the current Linux kernel. [https://linuxtv.org/wiki/index.php/Hardware_device_information](https://linuxtv.org/wiki/index.php/Hardware_device_information) is always a good starting point.

E) Installation

1) Required Hardware

rtranscode requires a Raspberry Pi 2 or 3 (highly recommended). It won't work on any other hardware.

You need some kind of TV server backend. This may be a dedicated hardware device running on your local network (Enigma2-Box, HD Homerun etc.) or a computer with a TV receiver card or USB-DVB receiver and some kind of server software.

The easiest way is to add a USB-DVB device to your Raspberry Pi. It is powerful enough to run both the TV server backend and the transcoding server.
2) Required Operation System and Software

To use all possibilities of this software collection it has to be installed on a full desktop version of Raspbian Jessie or Stretch. If you are using a Raspberry Pi 2 or plan to transcode 1080i sport streams I highly recommend using Jessie.

After you have successfully tested the software and created a channel database, you can also transfer the main components to a headless system. I have not tested it on a Raspbian Light installation, but you should be able to get it to work.

Rtranscode needs a full installation of gstreamer-1.0 and most of its plugins. Some are already installed on a Raspbian Desktop version. To make sure that everything is in place and the missing plugins are added, run:

```
sudo apt-get install gstreamer1.0-libav gstreamer1.0-plugins-bad gstreamer1.0-plugins-base gstreamer1.0-plugins-base-apps gstreamer1.0-plugins-good gstreamer1.0-plugins-ugly gstreamer1.0-tools
```

The next step installs the GPU support for gstreamer and is different for Raspbian Jessie and Stretch.

On Jessie run

```
sudo apt-get install gstreamer1.0-omx
```

I have created a new version of the gstreamer-omx plugin, based on the latest 1.10.5 source release and added the patch from 6by9 and two more upstream patches. It is included in the rtranscode V4 package and you can use it alternatively on Stretch. You'll find more details in the next chapter.

If you want to use the default 1.10.4 version on Stretch run:

```
sudo apt-get install gstreamer1.0-omx-rpi gstreamer1.0-omx-rpi-config
```

Rtranscode also needs a working installation of omxplayer (which is used for analyzing the stream), but this is already installed on any Raspbian Desktop release.

*Important Note: To be able to transcode MPEG2 SD streams, you have to buy the MPEG2 decoder license from the Raspberry Pi Foundation.*

3) Download and Install rtranscode

From a terminal run:
wget http://steinerdatenbank.de/software/rtranscode4.tar.gz

tar -xzvf rtranscode4

cd rtranscode4

This will install a few binaries and scripts in /usr/local/bin and create the folder
/usr/local/share/rtranscode containing a configuration file and and a default (empty)
channel database.

If you are using Raspbian Stretch and want to use my gstreamer-omx 1.10.5 module
instead of the official one run:

```
sudo ./install-omx
```

Do not install any other gstreamer-omx related packages!

For best performance you should add the following lines to /boot/config.txt
```
gpu_freq=500
force_turbo=1
gpu_mem=192
```
(requires a reboot).

Now we are ready to use rtranscode.

4) Package Contents

The main directory contains the install scripts, this manual and a number of sub-
directories:

`installation`: contains the binaries, scripts and some additional Debian packages.

`sources`: contains the sources of all compiled binaries, including all modifications to
http-launch and gstreamer-omx. To compile the Python sources you need Nuitka.

`web`: contains the web server and all its components

`kweb`: contains the Python script create_kpages.py, which can be used to create a
local kweb GUI.

`gnutv`: contains the scripts to create a fully working GnuTV backend and a matching
channels database for rtranscode.

`mumudvb`: contains the scripts and tools to create a fully working MuMuDVB
backend and a matching channels database for rtranscode.
5) **Recommended Software**

In order to use some (optional) components of the rtranscode package, you have to install my kweb suite. This will not only be used to create GUIs for rtranscode, the GnuTV and the MuMuDVB backends, but also contains omxplayerGUI, which doesn't only add a GUI to omxplayer (the best media player for the Raspberry Pi), but also extends its possibilities.

Because of the tight integration of omxplayerGUI, kweb is also the ideal browser client for the rtranscode web interface if you use it from the Raspberry Pi.


The manuals for kweb and omxplayerGUI are also available online:

Four your convenience I have added an install script, that will download and install the latest version for Jessie or Stretch:

```
./install_kweb
```

(without “sudo”!)

If you want to build your own TV server backend using MuMuDVB, you need an actual version. For your convenience I've added one to the package and you can install it with

```
sudo ./install_mumudvb
```

F) **About this Software Package**

1) **History**

I have been using hardware accelerated transcoding in my own TV server application for a few years and have been asked by a number of people, if I could make it available as a separate application to be used with different backends.

Transcoder 1.0 was published in October 2015. Transcoder 2.0 followed one year later.

For version 3, the package was renamed to rtranscode, to avoid conflicts with a Debian package named “transcode”. It also contained a real manual, not just a simple Readme file. It was published in March, 2017.
Raspbian Stretch required a new version, rtranscode V. 4, which has been rewritten in large parts and offers a number of additional features. It can be used on both Jessie and Stretch

2) Caveats

Although I have been using it (different versions) for a number of years and quite a lot of users have been successfully using it with different backends, I still consider rtranscode to be experimental software for a number of reasons:

rtranscode depends on a largeumber of software packages (especially from the gstreamer family) for which I'm not responsible in any way. These packages may (and do) contain bugs, have memory leaks or might simply not work correctly in some circumstances. That's outside of my control.

I have no chance to test rtranscode with all possible kinds of server backends (hardware and/or software) and cannot guarantee that it will work in any combination. Before I published rtranscode V. 3, I ran a number of extensive tests with tvheadend, because I knew that a lot of people were using it and would like to add the transcoding possibilities. After spending a number of days testing different versions, reading through a completely outdated documentation with a lot of missing chapters, I gave up and swore to myself, never to install this crap on any of my computers again. In my manual I stated, that rtranscode doesn't work with tvheadend. But a while later a user on the Rpi forum reported that he had got it to work with tvheadend. So I won't state it any more, that it doesn't work, but I do not guarantee it.

Some combinations of transcoding features may simply overload the GPU or the CPU of the Raspberry Pi. I have disabled all feature combinations, which definitely do not work (like double deinterlacing of 1080i streams), but there are some combinations left that may work in certain circumstances and not in others. For example, I do not recommend using audio transcoding in combination with deinterlacing 1080i channels, but it may work, if you reduce the image size. It's up to the user to test if some combinations work for him or not. It won't cause any harm. The transcoding process may simply become unstable or crash.

The transcoding server may also stop working, if your original TV stream becomes unstable, corrupted or contains too many errors (as I experienced on my satellite receiver during a snow storm). There's nothing I can do about that.
3) **Support**


If I decide to start a new thread, I’ll add a link to it in the OP of this thread.
II Using rtranscode

The command line program "rtranscode" is the basic tool of the rtranscode package. It can be used in a number of ways, using different modes.

A) Simple mode

In simple mode rtranscode will set up the arguments and then start http-launch, replacing its own process by it. Once the stream is running, entering "CTRL+c" is the only way to stop it.

rtranscode also offers some utility functions, like analyzing a stream and providing the arguments required for a stream. It can also create a channel database from a text file (e. g. an m3u playlist file) containing stream URLs.

1) Using 3 arguments and any number of options

rtranscode [options] uri videomode audiomode:audiopid[,audiomode:audiopid, ...]

uri = URL of your local TV http server stream (MPEG TS streams only!)

videomode must be one of the following:
sd1 for MPEG 720x576i SD video
sd2 for H264 720x576i SD video
sd3 for MPEG 720x480i SD video
sd4 for H264 720x480i SD video
sd5 for MPEG 704x576i SD video
sd6 for MPEG 544x576i SD video
sd7 for MPEG 480x576i SD video
sd8 for MPEG 352x576i SD video
hd1 for H264 1280x720p HD video
hd2 for H264 1920x1080i HD video
hd3 for MPEG 1280x720p HD video
hd4 for MPEG 1920x1080i HD video
hd5 for H264 1440x1080i HD video
hd6 for H264 1280x1080i HD video

audiomode must be either 'mpeg', 'ac3' or 'aac'.

audiopid must match a valid audio pid from your TS stream (decimal number or hexadecimal starting with '0x'). The selected audio pid must match the format set in audio mode. If the stream only contains one audio pid, you can also use "-1".
You can use multiple audiomode:audiopid pairs, separated by a ‘,’. The program will use the first one or the first “ac3” mode, if a certain option is set and ac3 is available.

Example:
```
rtranscode http://localhost:9082/bysid/11110 hd1 mpeg:0x17e8
```

The transcoding server will be started immediately, if the original stream is available. To stop transcoding, press CTRL+c.

2) Getting arguments from a file

```
rtranscode [options] -f=file_path
```

file_path must either be an absolute path or the name of a file in the current directory. If the path name contains spaces, it must be quoted. The file must be a text file containing one line of text like this
```
uri videomode audiomode:audiopid
```
(arguments like in 1)

Example:
```
rtranscode [options] -f="/home/pi/ZDF HD.txt"
```

The transcoding server will be started immediately, if the original stream is available. To stop transcoding, press CTRL+c.

3) Starting with a named channel

```
rtranscode [options] -n=channel_name
```

This requires a channel database. By default it uses /usr/local/share/rtranscode/channels.dat, but you can set another database file with the -d option (see below). If the channel name contains spaces, it must be quoted.

Example:
```
rtranscode [options] -n="ZDF HD"
```

If the channel name is found in the database, the transcoding server will be started immediately, if the original stream is available. To stop transcoding, press CTRL+c.

4) Analyze stream uri

```
rtranscode [options] -g=URI
```

where URI is the URL of your original stream.
rtranscode will analyze the output of "omxplayer -i URI" and if successful, will print
the results like this:
channel_name=uri videomode audiomode:audiopid,audiomode:audiopid,...
suitable for inclusion into a database file.

Example:
`rtranscode -g=http://localhost:9082/bysid/11110`
Result:
```
ZDF HD=http://localhost:9082/bysid/11110 hd1 mpeg:0x17e8,mpeg:0x17e9,ac3:0x17ea,mpeg:0x17eb
```

Note: You can use a redirection to append the result to a channel database:
`rtranscode -g=http://localhost:9082/bysid/11110 >> /home/pi/mychannels.dat`

5) Analyze stream uri and start the stream

`rtranscode [options] -t=URI`

where URI is the URL of your original stream.

rtranscode will analyze the output of "omxplayer -i URI" and if successful, will print
the results like this:
channel_name=uri videomode audiomode:audiopid,audiomode:audiopid,...
suitable for inclusion into a database file. It will also start the transcoding server
immediately. To stop transcoding, press CTRL+c.

6) Add channels to a channel database from a text file containig stream URLs

`rtranscode [options] -i=infile -o=outfile`

infile must be the path to a text file containing stream links. This can also be an m3u
file with stream links. infile must be either a complete path starting with '/' or a file
in your current directory. If the path contains spaces, it must be quoted.

outfile must be the path to a channel database file. If it doesn't exist, it will be
created. outfile must be either a complete path starting with '/' or a file in your
current directory. If the path contains spaces, it must be quoted.

rtranscode will try to analyse any stream URL found in infile and add the channel to
outfile, if successfull.

Example
`rtranscode -i=playlist.m3u -o=mychannels.dat`
7) Getting help

rtranscode -h
or
rtranscode --help

will print a simple usage page, showing all different modes and options.

B) Menu mode

1) Main and streaming menu

rtranscode [options]

If started without arguments, rtranscode will run in menu mode. This requires a working channel database. By default it uses /usr/local/share/rtranscode/channels.dat but you can set another database file with the -d option, e.g.

rtranscode -d=/home/pi/mychannels.dat

rtranscode uses a curses menu and falls back to a simple menu, if certain problems occur (e.g. too many channels in the database without using grouping).

Note: executing commands always requires a text entry followed by hitting the Return or Enter key.

Here is an example of the menu as you might see it in your terminal:

```
ABR: original (a) VBR: medium (v) SD-Size: 576p (s)
HD-Size: 512p (h) Deinterlace: off (d) Scaler: OMX (n)
Group: ARD ZDF HD (g) Channels:
    Das Erste HD (0) ZDF HD (1) zdf_neo HD (2)
    ZDFinfo HD (3) 3sat HD (4) tagesschau24 HD (5)
ONE HD (6) KiKA HD (7) SWR BW HD (8)
    SWR RP HD (9) NDR FS NDS HD (10) NDR FS HH HD (11)
    NDR FS SH HD (12) NDR FS MV HD (13) hr-fernsehen HD (14)
    MDR S-Anhalt HD (15) MDR Sachsen HD (16) MDR Thuringen HD (17)
    rbb Berlin HD (18) rbb Brandenburg HD (19) BR Sued HD (20)
    BR Nord HD (21) PHOENIX HD (22) arte HD (23)
    WDR HD Koeln (24)
Enter a channel number, 'a','v','s','h','d','n','g' or 'q' to quit:
```

The first two lines show the active settings:

ABR = current audio bit rate for the output stream
enter 'a' and hit "Return" to select another one from a separate menu page.

19
VBR = currrent video bit rate for the output stream
enter 'v' and hit "Return" to select another one from a separate menu page.

SD-Size = current output size for SD channels.
enter 's' and hit "Return" to select another one from a separate menu page.

HD-Size = current output size for HD channels.
enter 'h' and hit "Return" to select another one from a separate menu page.

Deinterlace = currently selected deinterlacing method
enter 'd' and hit "Return" to select another one from a separate menu page.

Scaler = currently selected video scaling method
enter 'n' and hit "Return" to select another one from a separate menu page.

The "Group:" entry in the third line will only be shown, if you use grouping in your
channel database.
Enter 'g' and hit "Return" to select another group of channels from a separate menu page.

To quit the program enter 'q' in the main menu and hit "Return".

To start transcoding a channel, enter the channel number and hit "Return". The
stream menu will be shown. It might look like this:

```
Streaming ZDF HD to http://127.0.0.1:9080/xyz.mkv
Size: 910x512  VBR: 1.85M  ABR: original MPEG
Deinterlace: off  Scaler: OMX

Recording time: Unlimited

Commands:
's' = stop , 'x' = restart transcoding
'r' = record stream, 't' = set recording time

Enter command and hit Return:
```

If the stream blocks for some reason, you can enter 'x' and hit "Return" to restart it
with the same settings. Entering 's' will stop the stream and return to the main
menu.

2) **Recording transcoded streams**

It's also possible to record the transcoded stream by entering the "r" command. If
you want to record for a specific duration, you can set this using the 't' command
(before you start recording), which will open a submenu:

Current recording time is set to: 0 = unlimited
You can set the new recording time in seconds
or as 'h:m', e. g. '1:30'
or set it to '0' for unlimited recording
Enter new recording time:

If you have started recording, the streaming menu will change and may look like this:

Streaming ZDF HD to http://127.0.0.1:9080/xyz.mkv
Size: 910x512  VBR: 1.85M  ABR: original MPEG
Deinterlace: off Scaler: OMX
Recording 2018-01-29-12-48-35-ZDF-HD.mkv until Mon Jan 29 14:18:35 2018
Enter 's' to stop recording:

You can stop the recording at any time by entering 's' and hitting Return.

Note: If you have set up a recording time, the menu will not change when the recording stops. By entering an empty command (just hit Return) you can check, if the program is still recording. If not, it will show the stream menu again.

3) Sub-menus

Groups menu (example):

Current Group = ARD ZDF HD
Available Groups:
ARD ZDF HD (0)  ARD ZDF SD (1)  News (2)
Privat TV DE (3)  Sport (4)
Enter a new group number and hit 'Enter':

Audio bit rates:

Current Audio-Bitrate = original
Available Bitrates:
original (0)  32K (1)  64K (2)  96K (3)  128K (4)  160K (5)
192K (6)  256K (7)  320K (8)  384K (9)  448K (10)  512K (11)
640K (12)
Enter a new bitrate number and hit 'Enter':

Selecting '0' disables audio transcoding.
video bit rates:

Current Video-Bitrate = medium

Available Bitrates:
original (0) low (1) medium (2) high (3) super (4) 192K (5) 288K (6) 384K (7) 448K (8) 512K (9) 640K (10) 768K (11) 896K (12) 1M (13) 1.1M (14) 1.25M (15) 1.5M (16) 1.75M (17) 2M (18) 3M (19) 4M (20) 5M (21) 6M (22) 7M (23) 8M (24)

Enter a new bitrate number and hit 'Enter':

Selecting '0' disables all transcoding for video and audio and just repackages the stream.

If you select 1, 2, 3 or 4, the video bit rate will be calculated based on the image size and the frames per second, using different quality settings.

SD video sizes:

Current SD video size = 576p

Available SD video sizes:
96p (0) 144p (1) 192p (2) 240p (3) 288p (4) 384p (5) 480p (6) 576p (7)

Enter a new SD video size number and hit 'Enter':

HD video sizes:

Current HD video size = 512p

Available HD video sizes:
144p (0) 288p (1) 362p (2) 432p (3) 512p (4) 544p (5) 576p (6) 640p (7) 720p (8)

Enter a new HD video size number and hit 'Enter':

Both menus show the number of video lines in the progressive output stream. The image width will be calculated from the original size (defined in the video mode).
Deinterlacing methods:

```
Current deinterlacing = off

Available deinterlace methods:
off (0)
linear,single (1)
linear,double (2)
scalerbob,single (3)
scalerbob,double (4)
greedy1,single (5)
greedy1,double (6)
half-scale (7)
```

Enter a new deinterlace number and hit 'Enter':

Deinterlacing is automatically disabled for progressive (720p) source channels.
If you select a “double” interlacing method, the frame rate will be doubled for interlaced SD channels.
If the source stream is an interlaced HD stream, each ‘double’ method is automatically replaced by the corresponding ‘single’ method.

Scaling methods:

```
Current videoscaler = OMX

Available videoscaler qualities:
OMX (0)
Soft,NN (1)
Soft,HQ (2)
```

Enter a new videoscaler number and hit 'Enter':

Method '0' (OMX hardware scaling) is not available on Stretch and will be automatically replaced by method '1' (software scaling, nearest neighbour).

Method '2' (software scaling, bilinear) is only available for SD channels (all combinations) and for interlaced HD channels, if deinterlacing is disabled.

Activating any deinterlacing method will always switch to software scaling, even on Jessie.

**C) Program Options**

1) *Selecting a configuration file*

- `c=config_file`

By default rtranscode will use `/usr/local/share/rtranscode/rt_config.py` for configuration. You can create your own configuration files and use this option to use one of them instead of the default configuration file.
Note:
config_file must be a valid python file, which will be executed, when the program
starts. See chapter IV for details if you want to create your own configuration file(s).

config_file must be either a complete file path or simply a file name. In the second
case it will be first searched in '/usr/local/share/rtranscode' and then in your current
working directory. If found, rtranscode will try to use it instead of the default
configuration file.

2) Selecting a channel database

-d=channels_file

By default rtranscode uses '/usr/local/share/rtranscode/channels.dat' for menu mode
or when used with the '-n=channel_name' command line option. This file is empty at
installation and you have to add channels by editing it.

You may also create your own channel database files and use the '-d' option to select
them.

channels_file must be either a complete file path or simply a file name. In the
second case it will be first searched in '/usr/local/share/rtranscode' and then in your
current working directory. If found, rtranscode will try to use it instead of the
default channel database.

See chapter III for more details about channel database files.

3) Set audio bitrate

-a=audio bitrate number
sets the audio bit rate for the stream or the default audio bit rate in menu mode
number = 0..12, default = 0 (use original stream, no audio transcoding)

Audio Bitrates:
0  use original stream
1  32K  32768
2  64K  65536
3  96K  98304
4  128K 131072
5  160K 163840
6  192K 196608
### 4) Set video bitrate

- \( v \)=video bitrate number

sets the video bit rate for the stream or the default video bit rate in menu mode
number = 0..24, default = 2 (calculate medium quality)

<table>
<thead>
<tr>
<th>Number</th>
<th>Video Bitrate</th>
<th>Bitrate (K)</th>
<th>Conversion (Bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>original stream, no transcoding</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>low</td>
<td>12 (divider)</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>medium</td>
<td>9 (divider)</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>high</td>
<td>6 (divider)</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>super</td>
<td>4 (divider)</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>192K</td>
<td>196608</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>288K</td>
<td>294912</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>384K</td>
<td>393216</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>448K</td>
<td>458752</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>512K</td>
<td>524288</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>640K</td>
<td>655360</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>768K</td>
<td>786432</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>896K</td>
<td>917504</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>1M</td>
<td>1048576</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>1.1M</td>
<td>1153432</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>1.25M</td>
<td>1310720</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>1.5M</td>
<td>1572864</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>1.75M</td>
<td>1835008</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>2M</td>
<td>2097152</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>3M</td>
<td>3145728</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>4M</td>
<td>4194304</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>5M</td>
<td>5242880</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>6M</td>
<td>6291456</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>7M</td>
<td>7340032</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>8M</td>
<td>8388608</td>
<td></td>
</tr>
</tbody>
</table>
5) **Set output image size for SD videos**

-s=SD video size number

Sets the output image size for SD video.  
number = 0..7, default = 7 (576p)

SD Video Sizes:
96p (0)
144p (1)
192p (2)
240p (3)
288p (4)
384p (5)
480p (6)
576p (7)

6) **Set output image size for HD videos**

-h=HD video size number

Sets the output image size for HD video  
number = 0..8, default = 4 (512p)

HD Video Sizes:
144p (0)
288p (1)
362p (2)
432p (3)
512p (4)
544p (5)
576p (6)
640p (7)
720p (8)

7) **Set deinterlacing method**

-e=deinterlacing method number

number = 0..6, default = 0 (off)

Deinterlacing methods:
off (0)
linear,single (1)
linear,double (2)  
scalerbob,single (3)  
scalerbob,double (4)  
greedyL,single (5)  
greedyL,double (6)  
half-scale (7)  

Deinterlacing is automatically disabled for progressive (720p) source channels. If you select a “double” interlacing method, the frame rate will be doubled for interlaced SD channels.

If the source stream is an interlaced HD stream, each ‘double’ method is automatically replaced by the corresponding ‘single’ method.

“half-scale” first scales the image size to exactly half the original image size before scaling to the final size. It should only be used for 1080i channels and sizes up to 544p.

8) Set scaler method

-x=videoscaler quality

numbers = 0..2, default = 0 (OMX)  
Scaling methods:
OMX (0)  
Soft,NN (1)  
Soft,HQ (2)  

Method ‘0’ (OMX hardware scaling) is not available on Stretch and will be automatically replaced by method ‘1’ (software scaling, nearest neighbour). Method ‘2’ (software scaling, bilinear) is only available for SD channels (all combinations) and for interlaced HD channels, if deinterlacing is disabled. Activating any deinterlacing method will always switch to software scaling, even on Jessie.

9) Set path for stream URL

-path=path

Sets output path for stream URL, must start with a ‘/’ and end with ‘.mkv’, default = ”/xyz.mkv“
10) **Set port for transcoding server**

-port=port

Sets output port for stream server, must be greater than 1024, default = '9080'

11) **Selecting the audio output format**

-u=audio_output_format

allowed values are 'aac', 'ac3' or 'both'

If audio transcoding is enabled, transcode to this format. If using 'both', AAC will be used for MPEG and AAC audio input, AC3 for for AC3 input.

If 'ac3' is selected, the transcoder will prefer AC3 input streams to MPEG audio streams.

12) **Pre-run original stream**

-r=delay
delay = 0..10, default = 0

If not zero, access the original stream for 'delay' seconds to make up for tuning time. This will be used for streaming and analyzing.

13) **Set recording directory**

-l=record_path

Must be a full path to an existing directory, default = current directory

14) **Select user script for automatic tuning**

-z=user_script

Enable automatic tuning by pointing to proper tuner (Python) script file, default = /usr/local/share/rtranscode/tuner.py (non-existent).

user_script must be either a complete file path or simply a file name. In the second case it will be first searched in '/usr/local/share/rtranscode' and then in your current working directory.
The user script must strictly follow the guidelines explained later. Tuner scripts for the GnuTV and MuMuDVB backends are supplied or created when running the creator scripts.

15) Print http-launch command line

-\texttt{sm}

print the http-launch command line with all arguments instead of starting the transcoder (can be used with all commands which directly start the transcoding server).

16) Run in quiet mode

-\texttt{q}

Both rtranscode and http-launch will not print anything to the command line. Cannot be used for menu mode. Useful if you call rtranscode from other applications and want to run it in the background.

D) Remote Control

rtranscode runs in a terminal (command line, lxterminal, SSH connection). If you close the terminal, all programs started from it will be closed and this will also stop transcoding.

If you want to manage transcoding from the internet using rtranscode, you must establish an SSH connection first. You can run the transcoding server, but it will be stopped, if you close the SSH connection. This may not be what you want, but there is a simple solution.

To continue the programs started from a terminal or SSH connection even when the connection is closed, you can use an application named “screen”. You may have to install it first:

\texttt{sudo apt-get install screen.}

Now you can start rtranscode in the following way:

\texttt{screen rtranscode [options]}

You can use the program as usual. To detach from the program, press “CTRL+a”,

29
followed by 'd'. The screen session is closed and the program will continue to run in the background. You can now safely close the terminal or SSH connection.

To connect to the running program again later on (from a new terminal or SSH connection), enter:
  screen -r

If you stop the program, the screen application will also close.

Note: The web interface offers a more comfortable solution for remote control.
III Channel Databases

A) The default database

For menu mode, named streams (-n=name) and the web interface you need a channel database. By default ‘/usr/local/share/rtranscode/channels.dat’ will be used. This file is created during installation and does not contain anything yet.

Channel database files are simple text files, which are easy to create and edit. To use the default channel database, you have to add some content first (as root). Using nano:

```
sudo nano /usr/local/share/transcode/channels.dat
```

If you prefer a desktop program:

```
gksudo leafpad /usr/local/share/transcode/channels.dat
```

Enter lines of the following form:

channel name=uri videomode audiomode:audiopid, audiomode:audiopid,...

for example:

```
ZDF=http://192.168.0.34:9082/bysid/28006 sd1 mpeg:0x78,mpeg:0x79,mpeg:0x7a,ac3:0x7d
```

rtranscode now supports multiple audio streams in the database (separated by a comma). By default, the first audio stream is used. If the option “-u=ac3” is used, rtranscode will search for the first ac3 stream. If it doesn't find one, it will take the first audio stream. In the web interface you can select the audio stream to be used for transcoding.

You can use rtranscode's analyze function to get the content of the lines:

```
rtranscode -g=uri
```

Empty lines or lines starting with a '#' (comments) are ignored.

After saving the file it can be used and the channels will appear in the main menu.

**Important note:** The format of the channel database has changed in version 4, but channel database files from version 3 can still be used.

B) Adding groups

To make the menu more comfortable to use, you can group your channels. This is really required if your database file contains more than about 40 channels (because the curses menu will break otherwise and will be replaced by a simple menu which might require scrolling).

To add a group, add a line like this:

```
31
```
All channel entries following this line, belong to this group. To switch to another group, add another group line with a different name. You can use the same group multiple times, which means that you can switch back to a group already used before.

If there is no group entry before the first channel line, all following channels will be added to the "default" group, until a new group declaration is found.

If you have more than one group in your database, the "Group:" entry will be shown in the menu and you can switch between groups using the 'g' command.

C) Create your own channel databases

Instead of using the default database, you can create your own database(s) and select them using the "-d=database" option.

You can create and edit a text file manually or use rtranscode to help set it up, as shown in chapters II,A,4 and 6.

I want to give a simple example how this can be used with mumudvb. I have started mumudvb with a config file to stream the whole Astra S19.2E transponder on frequency 12188 MHz (German TV channels of the RTL group). mumudvb's web interface runs on port 4242. I download the channel m3u playlist with the following command:

```
wget -O rtl.m3u http://localhost:4242/playlist.m3u
```

Now I use rtranscode's database create function:
```
rtranscode -i=rtl.m3u -o=rtl.dat
```

The program will show the streams it is analyzing and finish with the message:
```
added 12 entries to /home/pi/rtl.dat
```

Now I can use it for rtranscode in menu mode:
```
rtranscode -d=rtl.dat
```

The menu will show the following channels for transcoding:

```
RTL Television (0)  RTL Regional NRW (1)  RTL HB NDS (2)
RTL FS (3)  RTL2 (4)  TOGGO plus (5)
SUPER RTL (6)  VOX (7)  RTL-NITRO (8)
RTLplus (9)  n-tv (10)  RTL HH SH (11)
```
If your TV server backend offers you a m3u playlist of all available channels, you can download it and use it to create a complete channel database in one run (but this may take a while).

If you build your own backends with either GnuTV or MuMuDVB, using the tutorials in chapters VIII, you can also create a complete channel database using the supplied scripts.
**IV The Configuration File**

Rtranscode, create_kpages.py and the web interface use a configuration file to overwrite all global settings. The default configuration file is `/usr/local/share/rtranscode/rt_config.py`. You can edit it (as root!) or create a copy of this file and modify this copy.

Note: The configuration file is a Python file which will be executed by the main programs. If you break the Python syntax, it will not work any more and it may even crash the main program. Therefore I recommend to create your own config file in the following way:

Create a copy in your user directory, e. g.
```bash
cp /usr/local/share/rtranscode/rt_config.py /home/pi/myconfig.py
```

Open this file with Idle for editing. After you have finished editing it, run "Check module" from Idle's "Run" menu. This will first save the file and then show the result (nothing) in the Python console window. If the Python syntax has been broken by your editing it will throw an error and show you the line with the error in the program window.

If no errors are reported, you can test your config file
```bash
rtranscode -c=/home/pi/myconfig.py ...
```

If all runs well, you can make it your default configuration:
```bash
sudo cp /home/pi/myconfig.py /usr/local/share/rtranscode/rt_config.py
```

**A) Setting default values**

The configuration file consists of two parts: The first part contains some heavy stuff which may deeply influence the way the program works (and also break everything). This part should only be modified by people who really know what they are doing. The second part contains some default settings, which are used, if you run rtranscode in simple mode (and no options are specified). They are also the default values used in menu mode or by the kweb and web interfaces. They are easy to modify and can be changed by anyone. We will start with this part.

```python
rt_port = '9080'
```

This is the port on which the streaming server runs. It must be a string (included within single quotes). You may not use a value below '1024'.

34
rt_path = '/xyz.mkv'

http-launch will reject any access which doesn't use the right URL, which is created by the path value. If you make your streams accessible from the internet (using port forwarding in your router), you will prevent anybody else besides you and your friends from accessing the stream. The only way to do this is to use a long and cryptic path, something like '/rvh6uw87hkj37.mkv'

It must start with a slash and end with "\.mkv" (included within single quotes).

rt_video_bitrate = '2'
rt_sd_video_size = '7'
rt_hd_video_size = '4'
rt_audio_bitrate = '0'
rt_deinterlacer = '0'
rt_videoscaler = '0'

These values set the default values for video bit rate, SD and HD video size, audio bit rate, deinterlacing and scaling method. The values are index values (numbers as text within single quotes) of the dictionaries in part 1. Run "rtranscode -h" to get a list of allowed values. You will find a full list of all possible values and their meaning in chapter II,C,3-8

**Note: setting a non-existant key may even break the program!**

rt_ch6limit = '9'
Start value for using 6ch ac3 audio out, by default '9' (384K) and should not be lower than '6' (192K). This value is an index value (number as text within single quotes) of the audio_bitrates dictionary in part 1. Run "rtranscode -h" to get a list of allowed values for audio bitrates. You will find a full list of all possible values and their meaning in chapter II,C,3.

rt_audiooutput = 'both'
The value may be either 'both', 'aac' or 'ac3' (within single quotes). If set to 'both', audio output format will be AC3 if the input format is AC3 and AAC otherwise.

rt_delayed_start = False
This value (Boolean) may be either 'True' or 'False' (without quotes!). If set to "True", the streaming pipeline will start when the first client connects, otherwise it will start immediately (default).
rt_prerun = 0
The value must a number between 0 and 10 (no quotes!). If greater than 0, the stream will be accessed by a dummy module for n seconds (n = value), before starting the transcoder or the stream analyzer. This may be helpful, if the original stream is not immediately accessible, because the application needs some time to tune the channel.

rt_channels_per_line = 3
The value must a number greater 0 (no quotes!). It defines how many channels per line are shown in the menu. Values between 1 and 6 may be useful (depending on the length of the channel names in your database.

rt_simple_menu = False
This value (Boolean) may be either 'True' or 'False' (without quotes!). If set to 'True', the simple menu (not the curses menu) will be used.

rt_stop_delay = 0.2
rt_start_delay = 0.5
These values (floating point without quotes) are used internally and should be left unchanged.

rt_channels_dat = '/usr/local/share/rtranscode/channels.dat'
If you want to use another channel database file by default, enter its full path here, included within single quotes.

rt_record_path = ''
By default this is empty, which means that all recordings go to the current working directory. You can set a full path to an existing directory instead, e. g. on a hard disk or a network share.

rt_user_script = '/usr/local/share/rtranscode/tuner.py'
Path to a user script used for automatic tuning of your backend. The default value points to a file which doesn't exist by default. If you have a working user script file, you may move it to this place (and rename it to “tuner.py”). Automatic tuning will only work in rtranscode menu mode or from the web interface. The server backends described in the tutorials contain (or create) working tuner scripts

**Important note:** Building your own tuner scripts requires a decent knowledge of Python programming. A complete description is beyond the scope of this manual. If you need to create your own tuner script you can contact me for help (on the Rpi forum support page).
rt_always_stop_tuner = False

By default, optional tuner scripts will not stop the tuner stream, when you stop the transcoder stream. If you want to change this, set the value to “True”

There is a last group of settings, which are only used by the web interface. Their meaning will be explained in chapter VI.

B) The really heavy stuff

The data structures in part 1 of the config file (mostly dictionaries) define which arguments are sent to http-launch and how the gstreamer pipelines for http-launch are built. Changing anything here requires a deeper understanding of both Python data structures and of building gstreamer pipelines. I won’t go into too many details here, just give some hints and examples for experiments.

The pipelines are built from three templates, rt_av_template (for both video and audio conversion), rt_v_template (video conversion only, using one of the original audio streams) and rt_rep_template (no transcoding, just repacking of the stream). All templates include a lot of variables, included between "$" characters, which are replaced by the program at run time, depending on the mode (SD or HD modes), audio and video bit rates and other settings. The av_template, for example looks like this:

'http-launch $port$ $path$ video/x-matroska $runmode$ $source$ ! tsdemux parse-private-sections=false program-number=-1 name=demux demux.audio$apid$ ! queue ! $audioparser$ ! $audiodecoder$ ! audioconvert dithering=0 ! audio/x-raw,channels=$channels$ ! $audioencoder$ bitrate=$abr$ ! matroskamux name=stream streamable=true demux. ! queue ! $videoparser$ ! $videodecoder$ $deinterlacer$$videoscaler$! omxh264enc target-bitrate=$vbr$ control-rate=variable ! h264parse ! queue ! stream.'

The building of the real pipeline depends on the following structures: rt_sources, rt_sd_modes, rt_hd_modes, rt_audio_decoders, rt_aac_encoder, rt_ac3_encoder, rt_video_bitrates, rt_audio_bitrates, rt_sd_video_sizes, rt_hd_video_sizes, rt_deinterlacers and rt_videoscalers.

This is the result of three years of experiments and experience with HW accelerated transcoding using gstreamer on the Raspberry Pi. Changing something will often result in disaster.
C) Adding modes

rtranscode V. 4 supports quite a number of video modes now: 8 for SD and 6 for HD streams, including some really obscure formats which I have found on the Astra S19.2E satellite.

Basically there are only two classes of video modes: they use either MPEG2 or H264 encoding. The only other difference between the different modes is the original image dimension (width,height) which is required to build the scaling and deinterlacing components.

If rtranscode's stream analyzer doesn't recognize a TV stream, you should analyze the stream with omxplayer:

```
ompxplayer -i stream-uri
```

Let's assume, that it shows you, that it contains an MPEG2 encoded video stream with an image size of 640x480, which is not supported by rtranscode. Than it's quite easy to add a new mode to rt_sd_modes:

Add a ',' to the line starting with: 'sd8' and add a new line below:

```
'sd9': {'videoparser': 'mpegvideoparse', 'videodecoder': 'omxmpeg2videodec',
        'dimension': (640, 480), 'check': ['mpeg2video', '640x480'],
        'desc': 'for MPEG 640x480i SD video'}
```

Tip: use copy and paste from a similar mode and just replace all dimension settings (in three places!).

If you add a progressive (non-interlaced) HD mode, you must add its key (e.g. "hd7") to the rt_progressive_modes list like this:

```
rt_progressive_modes = ['hd1', 'hd3', 'hd7']
```
V Using http-launch Directly

You may want to experiment with using http-launch directly. It can be called with:
http-launch PORT PATH MIMETYPE RUNMODE <launch line>

PORT = http server port
PATH = URL path, e.g. "/xyz.mkv"
MIMETYPE = the mime type matching your stream
RUNMODE must be one of the following:
default | silent | delayed | silent-delayed

<launch line> must be a gstreamer1.0 tool chain as you can use it with gst-launch-1.0 with two specialties:
The final muxer must be named "stream" and the final argument should be "stream." Here is an example from rtranscode itself:

http-launch 9080 /xyz.mkv video/x-matroska silent souphttpsrc
location="http://192.168.0.34:9082/bysid/10301" is-live=true keep-alive=true do-timestamp=true retries=10 typefind=true blocksize=16384 ! tsdemux parse-private-sections=false program-number=-1 name=demux demux.audio_13ee ! queue ! mpegaudioparse ! mpg123audiodec ! audioconvert dithering=0 ! audio/x-raw,channels=2 ! voaacenc bitrate=65536 ! matroskamux name=stream streamable=true demux. ! queue ! h264parse ! video/x-h264,alignment=au ! omxh264dec ! video/x-
raw,width=910,height=512 ! omxh264enc target-bitrate=1941333 control-rate=variable ! h264parse ! queue ! stream.
VI The Web Interface

A) Introduction

The web interface provides the most comfortable way to manage transcoding of your TV channels using a web browser on any computer on your local network or optionally from the internet. Before you can use it, you have to create a channel database. Basically it provides the same functionality as rtranscode in menu mode with a few exceptions:

- You can select an audio channel for transcoding (if your channel database supports it).
- Crashed streams will be automatically restarted.
- If a rtranscode user script is available, you can also control your TV server backend (see chapter VIII).
- It is much more intuitive and easier to use as everything can be controlled with a few mouse clicks.
- You can make the streams available to your family or friends from a simple web page.
- Recording transcoded streams is not supported.

The web interface is realized as a standalone threaded web server written in Python, based on the Karrigell 3.1.1 framework which contains (among many other things) a refined user management providing different kinds of user types. The web server manages http-launch directly through its builtin rtranscode module (for convenience named rtc4).

B) Using the Web Interface

To use the web interface you have to start the web server. Open a terminal, cd into to web directory inside the rtranscode directory and run:

```
./serve.py
```

If the web application doesn't find a channel database (by default: /usr/local/share/rtranscode/channels.dat), it will immediately terminate with an
error message. Otherwise it will simply run without displaying anything.

Alternatively you can also double click the serve.py in a file browser window and select “Execute” or “Execute in Terminal”.

Now connect to http://localhost:9079/ with your browser. The interface will work with any browser, but on the Raspberry Pi I highly recommend to use kweb, as it needs much less resources and offers a much better media integration. But if you use kweb, you have to enable Javascript and Cookies. For your convenience you will find a ‘start_kweb’ command script in the web directory, which you can double click from the file manager (select “Execute”) to start a kweb window showing the rtranscode web page.

The web page will look like in the image above; the lower part, starting from the second horizontal line, will only be shown, if you have added tuner control by a rtranscode user script.

Now you can select a group and a channel (the select button has only to be used, if Javascript is disabled), set the transcoding options and then click the “Transcode” button. If you have included a tuner backend control or if you have set the pre-run option, a message (e.g. tuning ZDF … ) will appear for a few seconds. Otherwise the transcoding will start immediately. And the web page will look like shown in the image.

The “Stream” button is nothing but a direct link to the transcoded video stream. The “Playlist” button links to a M3U playlist which is generated by the server. If and how you can use them to access the video stream, depends on the browser you are using:

One method will always work: right click on the “Stream” button, select “Copy Link Location” (or similar, depending on your browser) and paste the link into the appropriate entry field of a video player of your choice.

If you are on a Raspberry Pi and use kweb, it's really simple: Clicking on either “Stream” or “Playlist” will immediately open omxplayerGUI and play the video.

On Android it is similar: If you click on the “Stream” button you will be asked which video player you want to use and then the stream will be played with the selected player.
If you are using Firefox, it can be set to download the M3U playlist and open it with a specific application. The appropriate mime type is ‘audio/x-mpegurl’. On the Raspberry Pi this doesn't work with omxplayerGUI, but it works with (a HW accelerated version of) VLC.

Older versions of Chromium used to open M3U playlists with a default player which is set in the system. This seems to have been removed in newer versions.

Clicking on “Stop Transcoding” will stop the transcoding server (http-launch) and return to the default page, where you can select another source for transcoding.

It is important to understand, that only one user can have full control of the transcoding process at the same time. On the local network this is handled dynamically. If you access your server from the internet, you have to login with an admin account to get full control. Other users will see a simpler web page, depending on the current state. Part D) of this chapter will explain this in more detail.

C) The Configuration Page

The “Configuration” button is only available, if you are in full control of the web interface, either on your local network or after logging in with an admin account. Clicking it will open the configuration page.

The first three options let you control the appearance of the web interface:

**Web Style:** Select one of the predefined styles using different color schemes and font sizes.

*Important Note: A new selection will take immediate effect, if Javascript is enabled. You only have to click the “Save” button, if it is disabled. This is true for all selection menus.*

**Link Look:** Here you can define if links are shown as buttons or as normal links.

**“Selection Box Size”:** Here you can set the height of the group and channel selection boxes on the main page. If you select “0”, the box size will be equal to the
number of your groups. If you set it to “1”, a selection pop-up menu will
be displayed. Any other number will give the selection boxes a definite height (number
of entries shown).

The next two options are only available if no transcoding stream is running.

**Streaming Port:** Here you can change the port on which the stream will be served.
Only values between 1024 and 65535 are allowed. After changing the value, click
the “Save” button.

**Streaming Path:** Here you can set the path part of the URL for the transcoded
stream. It should start with a “/” and end with “.mkv”. If you make your stream
accessible from the internet, you should use a long and cryptic path, because this is
the only way to protect your stream access from casual users. Click the “Save”
button after modifying it. If the field is empty, when you click “Save”, a random path
will be generated.

The last five options let you override options set in the configuration file
(rt_config.py).

**Audio Output Format:** You can select between “AAC” (always use AAC mono or
stereo), ”AC3” (always use AC3 mono, stereo or 6 channels) or “both” (AAC for
original MPEG or AAC audio streams, AC3 for original AC3 AC3 streams).

**6 Channel Audio Limit:** AC3 output streams can use 6 channel mode, if the audio
bit rate is set to a certain minimal value, which can be selected here.

**Pre-Run Time:** If not zero, access the original stream for this number of seconds to
make up for tuning time. It won’t have any effect, if you have implemented a tuner
backend with a rtranscode user script.

**Stream Restarts:** This value defines, how often a stream will be started. If the
value is greater than one, the transcoder stream will be restarted, if it crashes, until
the maximum number is reached.

**Poll Time:** Sets the number of seconds the web page will wait before checking for
status update (e. g. stream started, running or stopped).

The button “Login” is only shown, if you access the web interface from the local
network and you have not logged in with an admin account. Clicking it will open
Karrigell’s login page.
After logging in with an admin account, the lower part of the configuration page will look differently. Instead of the “Login” button you now have a “Logout” button.

Clicking the button “User Management” will open Karrigell's user database managing page, which will be described in the next part.

Clicking the “Stop Server” button will terminate the web server. You should always stop the server before shutting down or rebooting your Raspberry Pi to avoid delayed shut down.

D) User Types and User Management

By default any client on your local network will be able to access and fully use the rtranscode web interface. No login is required. This behaviour can be changed in rt_config.py (described in part E).

Although this default behaviour is quite comfortable, it may lead to conflicts, when multiple local clients connect to the server at the same time. Only one of them can be the local admin. The first client which connects to the server will become the local admin, but if he isn't doing anything for a certain time (three times the polling time), another client may take control and become the local admin. If he connects while the first client is still in control, he will see the “No stream available” message. But when he hits the “Refresh Status” button after the timeout has elapsed, he will become the new local admin.

If one client has started the transcoder, he will remain the local admin as long as the stream is running and the web page is kept open. All other clients will see the streaming page and can only access the stream (and possibly restart transcoding if the stream has crashed or is blocking).

If the current local admin uses the login function and logs in with a “real” admin account, he will remain admin, even if he is not doing anything. In some rare cases he may lose control, but he can always get it back by clicking the “Take Control” button on the configuration page (which is only shown, if a real admin has lost control).

If you believe that this is too complicated, you can set “rt_web_alwayslogin” to “True” in rt_config.py. Then each client on your local network will have to login with a user name and password and only users with admin rights will be able to control the transcoder.
If you connect to your rtranscode server from the internet (using appropriate port forwarding in your router), a login with user name and password is always required.

**Important Note: Do not make the server available on the internet before you have created a new admin account and deleted the old one!**

User management is built into the Karrigell framework. The rtranscode web interface is delivered with a user database containing one user named “admin” with the password “raspberry”. We can use this account to login (with admin rights). Clicking on the “Login” button on the configuration page will open Karrigell’s login page.

After you have logged in with an admin account, you will find a new button on the configuration page “User Management”. Clicking it will open Karrigell’s user database managing page.

The first thing we will do now is to create a new user with admin rights. Click on “New User”. Enter a new login name and a secure password. Write the password down – there is no way to retrieve it, if you can't remember it any more. The E-Mail field is optional and can be left empty. You must select “admin” in the Role selection and then you can click on 'OK'. Click the “logout” link and login again with the new user name and password. If that works you can delete the original “admin” account. Click on 'edit” beside the “admin” account and then on “delete”. You will be asked again, if you want to delete the account, and have to confirm it.

Now you can continue to add other (non-admin) users. The procedure is the same, but you have to use other roles, either “visit” or “edit”. The difference between these user types is really small: Users with the “edit” role can restart crashed or blocking streams. Otherwise users of both types can only wait for a transmission to start and then access the transcoded stream.
Note: If you ever forget your admin login name or password, you can restore the original database file: copy the file “users.pdl” from the installation folder to .../web/data/www.

E) Configuration Options in rt_config.py

In chapter IV you have been shown how to edit the rtranscode configuration file named “rt_config.py” but without the special settings for the web interface. The web interface will first check for a configuration file inside the “web” directory (by default there is none) and then in /usr/local/share/rtranscode. If you want to use a separate configuration file for the web server, you can copy rt_config.py from the installation directory into the web directory. It is easier to edit, as you do not have to use sudo.

The settings for the web interface are at the bottom of the configuration file. I’ll explain them one by one.

rt_web_style = 'grey.css'

This sets the default style of the web interface. For a list of available styles, check the .../web/www/styles directory. You may only use file names from this directory. This style can be changed at run time on the configuration page.

rt_web_button = 'button'

Defines if links are shown as buttons or normal links. Possible values are either 'button' or 'link'. This can be changed at run time on the configuration page.

rt_web_auto_restart = 3

Defines how often a stream should be started. If this is set to 1, a stream will be started only once. The default value is 3, which means that the transcoder stream will be restarted twice more, if http-launch has crashed. The value can be changed at run time on the configuration page.

rt_web_polling = 10

Defines how often a client will check the server for a status change. The default value is 10 (seconds). The minimal value you can use is 2. The value can be changed at run time on the configuration page.
rt_web_greeter = 'rtranscode 4.0 Web Interface'

This defines the headline shown in the web interface.

rt_web_title = 'rtranscode 4.0'

This defines the title of the main page of the web interface.

rt_web_boxsize = 8

This value sets the height of the group and channel selection boxes in the web interface. If you set it to 0, the height will match the number of groups. If you set it to 1, you will get a normal selection popup menu. Greater values set the box size in lines. The value can be changed at run time on the configuration page.

rt_web_tunercontrol = True

If you have added a user script to tune a server backend (see Chapter VIII), this value defines, if tuner control and playlist buttons will appear in the web interface (for local users only). If you want to exclude them, set this value to 'False' (without quotes!).

rt_web_secure_playlist = False

By default, the stream playlist will be served without having to login. This makes it possible to access the playlist URL directly from a media player. To provide a certain level of protection, the playlist URL contains a random string, which is created when the server starts. If you set this value to 'True', accessing the playlist function from the internet requires a login. To use it with any kind of media player, you have to download it with your browser first. Local clients are not affected.

rt_web_alwayslogin = False

If you set this value to “True”, local client always have to login like remote clients (connecting from the internet).

F) Bits and Pieces

1) Access From the Internet

If you want to access the rtranscode web interface from the internet, you will have to set port forwarding (NAT) in your router. Check your router manual for more
details. By default, port 9079 is used for the web server and port 9080 for the transcoded stream. Most routers offer to set a range of ports in one setting, so you could use TCP 9079-9080.

This simple method of port forwarding is only possible, if your ISP provides you with a unique IPv4 address and if running any kind of server is not blocked by your ISP (mobile connections, for example). If your ISP provides an “IPv6 light stack”, for example, you share an IPv4 address with other users. Port forwarding is then only possible using external services. How to solve such problems is beyond the scope of this manual.

2) Server Backend Control

If you are using a rtranscode user script to control your server backend (see chapter VIII), simple tuning commands are added to the main web page. You can use them to tune a channel, stop the server backend or access a playlist of the original channels. This works only inside your local network.

Backend tuning will happen automatically, if you start a transcoding process and the required channel is not already tuned.

If your backend supplies multiple channels at the same time (e.g. when running mumudvb in transponder mode), the web interface provides transcoding settings for each of them (if the channel is supported in your channel database). That does not mean that you can transcode multiple channels at the same time.

3) Configuring Karrigell

It is possible to configure the Karrigell web server by modifying two different files: host_config.py inside the web directory and conf.py in web/data/www. I’ll limit this
description to the values which might be changed. Do not touch anything else!

Let's start with the first one:

port = 9079

This defines the port used by the server. You can set another port (not below 1024), if you need it.

use_ipv6 = False

You can set this to 'True', if you have to use IPv6 instead of IPv4.

silent = True

You can set this to 'False', if you want to run the server in a terminal and show its log. Do not set it to 'False' if you want to run the server in the background!

In the second file (...web/data/www/conf.py) you can set up logging which is disabled by default:

logging_file = None

#logging_file = os.path.join(karrigell_dir,"logs","access.log")

Comment out the first line and uncomment the second to enable logging of each server access to a file:

#logging_file = None
logging_file = os.path.join(karrigell_dir,"logs","access.log")

The log files can be found in ...web/karrigell/logs

4) Auto-Starting the Server

If you want to start the server at boot time, you have to create a script first. It might look like this (if the path is correct):

#!/bin/sh

cd /home/pi/transcoder4/web
.
serve.py > /dev/null 2>&1

Save it as “start_rtcserver” in your home directory and make it executable:

chmod +x start_rtcserver
Now you can add it to /etc/rc.local, at the end, just before the line “exit 0”:

```
su -l pi -c "/home/pi/start_rtcserver" &
exit 0
```

Use raspi-config, option 3 (Boot Options), B2 (Wait for Network at Boot) and enable this option.

You have to modify this, if your user name is not “pi”.

**5) Security Considerations**

The Karrigell server is a simple web server without any encryption. Anybody who can listen to your TCP stream using deep packet inspection might catch your user name and password. Firefox will always complain if you are logging in, for example, that the connection is not secure. So what? Does it do any harm? Listening into your data stream is normally illegal, if not ordered by a court. A “normal” hacker (just another client on the internet) cannot inspect your data transfer. Using a long and secure password will protect you against hacking attempts.

If you want to protect your server against packet inspection, you can set up a HTTPS server which supports a proxy mode. I have used lighttpd as a proxy for Karrigell in the past. But if you do this, you have to set:

```
rt_web_alwayslogin = True
```

in rt_config.py. The web proxy will appear as local client and login request will be disabled if you do not set this option.

You cannot proxy the transcoded stream the same way. The only protection of your stream is using a long and cryptic path.
VII Create a kweb GUI

rtranscode is a command line program with an optional simple curses menu running in a terminal. If you want a more comfortable desktop application, you can use tools from my kweb suite (Minimal Kiosk Browser, omxplayerGUI) and the program create_kpages.py to build a web interface running in kweb. Kweb is a simple, but fully functional web browser with a unique feature: it can execute any kind of program from links, buttons or web forms inside a (local) HTML file without having to run a server backend. create_kpages.py creates one or more (if you use groups) HTML files from an existing channel database:

You have run this command inside the 'kweb' directory of the rtranscode4 package;

```bash
cd kweb
./create_kpages.py [options] [channels database]
```

Without any options and arguments, this will create a channels.html file from the default database /usr/local/share/rtranscode/channels.dat in your current working directory. If your channel database contains groups, a separate page for each channel will also be created and the main page (channels.html) will contain a group menu. It will also create a script 'RT-GUI'. Double click this file from a file manager window and select “execute” to run the GUI.

This image shows an example GUI with omxplayerGUI running on top playing the transcoded stream.
The kweb GUI is quite easy to use. If your channel database contains groups, you will see a button for each group in the top line and the first group will be shown below. Each group page (or the main page, if you don't use groups) presents a web form with the following options:

- A menu button to select the video bit rate.
- A menu button to select the audio bit rate.
- A menu button to select the size for SD video.
- A menu button to select the size for HD video.
- A menu button to select the deinterlacer.
- A menu button to select the video scaler.
- A menu button to select the pre-run time (if required).

Inside a larger, scrollable menu box you will see all the channels from your database file (or group). Select a channel and click the "Transcode" button to start transcoding. A terminal will pop up showing nothing (because the transcoder is running in silent mode). I'll show you later, how to get rid of it, but for testing it is nice to have. If the terminal closes again after a moment, the source stream is not available.

If you click the "Play Transcoded Stream" button, omxplayerGUI will open and show the stream.

To stop transcoding, click the "Stop Transcoding" button.

*Note: you can start a new transcoding without clicking on "Stop Transcoding" first (the old instance will be closed automatically).*

The GUI gives you all features of running rtranscode in menu mode with one exception: auto-tuning is not supported (because rtranscode is run in simple mode). But you can add a tuning page to the GUI using the '-b' option. This may be a web interface of your TV server or a local kweb GUI which can be built following one of the tutorials in the next chapter.

To get rid of the terminal, open kweb's menu page, click on "Settings" and scroll down to the direct_commands list. Add a new line at the bottom

```
bg_rtranscode.sh
```

and click the save button beside the list.

If you have modified your channels database or your configuration settings, simply run `./create_kpages.py` again (within the same directory). It takes less than a second to create a new GUI.
To create a web interface for another channel database, just add the file name or complete path as argument, e.g.

`./create_kpages.py [options] mychannels.dat`

If you do not use a full path, the file is first searched in `/usr/local/share/rtranscode` and afterwards in the current working directory. The resulting (main) HTML file will always have the name of the channel database file with the extension ".html" instead of ".dat".

Options:

- `-c` = config file
  (default = `/usr/local/share/rtranscode/config.py`)

- `-p` = path
  output path for stream URL, must start with a '/' and end with '.mkv', default = "/xyz.mkv"

- `-s` = style
  file name of a css file from `/usr/local/share/kweb`, default = "about.css"

- `-b` = backend-uri

backend-uri must be either a HTTP URL (e.g. the web interface of your TV server backend) or the full file path of a kweb GUI main page which may be created for GnuTV or MuMuDVB as shown in the tutorial chapters.

The following image shows a rtranscode kweb GUI which includes the GnuTV tuner GUI with omxplayerGUI running on top displaying the transcoded channel.
VIII Tutorial: Create a TV Server Backend

A) Introduction

1) Overview

In this chapter I will show you step by step, how you can create a simple TV server backend using either GnuTV, a command line utility from the linux dvb-apps package, or MuMuDVB, a much more advanced DVB streaming server.

For both backends we will create a simple GUI running in kweb, a channel database for rtranscode and a user script, which enables automatic tuning in rtranscode or in the rtranscode web interface. This is what you can do with it:

Tune and watch any channel offered by your USB-DVB device from a simple GUI on your local Raspberry Pi.

(Optionally) stream the channel via udp multicast to your local network and watch it from any computer.

Make your local TV available from any place in the world using rtranscode or the rtranscode web interface.

Tune any channel and access the stream from the web interface.

Additional features offered by MuMuDVB only:

Make any channel available on your network as HTTP stream.

Stream all channels belonging to a transponder at the same time via HTTP and optionally UDP multicast.

Watch encrypted channels using a CAM (Conditional Access Module) built into your DVB device or a softcam like oscam. (There also seems to be CAM support in GnuTV, but I could not test it).

It's possible to build and use both backends. They do not interfere with each other, but you can only run one at a time.

*Important Note: In some countries (Germany, for example), DVB-T2 is using H265 encoded video channels. These cannot be used by rtranscode or watched on the Raspberry Pi because the hardware does not include a H265 decoder.*
2) Required Components

A working USB-DVB-T(2)/C/S(2) adapter connected to your Raspberry Pi.

An MPEG license from the Raspberry Pi Foundation to watch (and transcode) MPEG2 encoded SD channels.

A tool for to search for available channels and to create a channel config file: w-scan

```
sudo apt-get install w-scan
```

My kweb suite (for the GUI and the media player).

For GnuTV we need the dvb-apps:

```
sudo apt-get install dvb-apps
```

For MuMuDVB:

The MuMuDVB package from the repository is very old and should not be used. I build new packages for the Raspberry Pi from time to time which are available from the MuMuDVB website: [http://mumudvb.net/download/](http://mumudvb.net/download/)

Download the latest release and install it with gdebi (to get all dependencies):

```
sudo apt-get install gdebi-core
sudo gdebi -n file-name
```

where file-name is the name of the package which you have downloaded.

For your convenience I have included the actual release (February 2018) in this package. You can install it running

```
sudo ./install_mumudvb
```

from the main rtranscode4 directory.

3) Exploring your DVB hardware

Note: If you have only one DVB-USB device connected and if this device doesn’t include multiple tuners, you can skip this section.

From a terminal run:

```
ls /dev/usb
```
It should show one directory for each device which is connected to your Raspberry Pi, e.g. “adapter0” for the first device. Now let’s explore this device:

```
ls /dev/usb/adapter0
```

If the output looks like this:

```
demux0  dvr0  frontend0  net0
```

the device only supports one tuner (frontend). But if it shows multiple frontends like “frontend1” (etc.) you should try to find out, which frontend is used for DVB-T, DVB-C etc. We will need this information for the other steps. Check the manual of your device.

If you have multiple adapters connected, you need to know, which adapter number is used for which device. Unfortunately you cannot be sure, if the adapter numbers are the same after a reboot. In this case I know only one solution: have only one adapter connected at boot time and plug the second into an USB port afterwards (when your Raspberry Pi is already running).

**B) Channel Search**

From your TV or a DVB-Tuner connected to it you may know a function usually called “search channels”. We need a similar function which is realized by the nice scanning tool “w-scan”. It will create a channels configuration file (we’ll use the name ‘channels.conf’). GnuTV requires this file in Zap/Xine format. For MuMuDVB we will use the VDR format.

Open a terminal and cd either into the gnutv directory inside the rtranscode4 directory or into the mumudvb directory

Let’s start with a simple example for gnutv:

```
w_scan -f s -s S19E2 -R 0 -E 0 -X > channels.conf
```

For MuMuDVB it’s almost the same, just without the “-X” option:

```
w_scan -f s -s S19E2 -R 0 -E 0 > channels.conf
```

This will scan the Astra S19E2 satellite(s) for free TV channels. Explanation of the options used here:

- `-f s` = scan a satellite tuner
- `-s S19E2` = select satellite Astra S19E2
- `-R 0` = exclude radio channels
- `-E 0` = exclude encrypted channels
- `-X` = create Zap/Xine format
> channels.conf = write the output to a file named channels.conf

Simply replace a satellite name, if your antenna points to another satellite. To get a
list of supported satellites, run

```
$ w_scan -s ?
```

To scan a DVB-T(2) device, use

```
$ w_scan -f t -c Country-Code -R 0 -E 0 -X > channels.conf
```

(or without the '-X' for muMuDVB)

To get a list of country codes, run

```
$ w_scan -c ?
```

To scan a DVB-C device, replace “-f t” by “-f c”.

If you want to include encrypted channels, replace “-E 0” by “-E 1”.

This is the basic command. Depending on your harware, additional options may
have to be used. Run “w_scan -h” and “w_scan -H” for additional options or visit one
these wiki pages:

https://linuxtv.org/wiki/index.php/W_scan

http://vdr-wiki.de/wiki/index.php/W_scan

Execution of the w_scan command will take some time (10 – 45 minutes). If it
finishes, you will find the channels.conf file in your directory. It contains one line
for each channel found, which may look like this for GnuTV

```
```

or like this for MuMuDVB

```
```

Note: Sometimes w-scan does not find all channels, especially channels which are
not running all the time (like some sports channels). If you need these channels, you
could run w-scan with an additional option: “-t 3”. But w-scan will run much slower
then.

The next step is important. Open channels.conf with leafpad and reduce it to the
channels you really need: just remove the lines with these channels.
The list of channels my include channels which do not really work, like “pseudo”
channels which only provide a smart (HbbTV enabled) TV with information about a channel which is streamed through the internet connection. We cannot use them with our TV server backend.

If you have activated scrambled channels during w-scan search, you should remove all channels, which are not supported by your CAM.

Note: If you are editing a channels.conf file created for MuMuDVB (VDR format), you can also comment out a line by putting a ‘#’ in front, instead of deleting the line.

If you are not sure, create a copy of the channels.conf first. This way you can add a channel later again, if you deleted it by accident.

Once you have a matching channels.conf file, you can go on with the further steps which are described in part C) for GnuTV and in part D) for MuMuDVB.

C) Create a GnuTV Backend

*Important Note: GnuTV doesn't work very well with rtranscode for HD Channels. Use MuMUDVB for better HD support.*

The gnutv directory contains a few of scripts which we will use to create a GUI and a channel database for rtranscode. Open a terminal and cd into the gnutv directory inside the rtranscode4 directory.

1) Create the GUI

If you have a simple device with only one tuner built in, run

```
./create_pages.py -t
```

If your hardware is more complex, you have to add one more option:

```
-d=adapter,frontend,demuxer
```

The arguments are simple numbers. For adapter 0, frontend 1 and demuxer 1 it is:
```
-d=0,1,1
```

-t means “test mode”. It will help us to see, if everything really works.
The command runs very fast and will create a lot of files. Open the directory in the file manager, double click “GnuTV-GUI” and select “Execute”. This will open a kweb instance in a small window running the GUI.

Clicking on a channel name will tune the channel. A terminal will pop up, which will show us gnutv (the command line utility) in action.

If it the terminal doesn't close and it looks like shown in the image we know that everything is working. If you click on “Watch TV”, omxplayerGUI will open and show TV stream. It may take a few seconds, until the stream is played, because omxplayer is not very fast when accessing a live TV stream, especially when UDP is used. Clicking on “Stop TV Server” will close the stream. If you want to tune to another channel just click the channel name; you do not have to stop the TV server first. If you click on “Quit GnuTV GUI”, the GUI will close and stop the TV stream first. If you simply close the window, the TV server will keep running.

If it doesn't work, you should try another channel first. If that also doesn't work, you may have to check your hardware and run “./create_pages.py” again with a different “-d=” option.

If everything is working, we will run “./create_pages.py” again to get rid of the test mode. But before we do this, we will first have a look at the other options supported by the script.

By default, the stream runs on “udp://localhost:5001”. Only a single client, running on the same Raspberry Pi can access this stream (rtranscode is also a client). If you want to make the stream available on each computer on your local network, you have to use udp-multicast. This can be done using the following option;
-i=multicast-address, e. g.
-i=239.100.0.1
You can also change the port (default = 5001), using the ‘-p’ option, e. g.
Note: udp multicast may not work with every router. Sometimes only SD channels work. And you should not use multicast if you are connected through WiFi. And it is also possible to stream any TV channel to your whole network via HTTP using rtranscode (with or without transcoding),

The last option selects a stylesheet to be used for the interface, e.g.
-s=black.css

You can only use style sheets from /usr/local/share/kweb. To get a list of available style sheets, run:
ls /usr/local/share/kweb/*.css

You can also create your own style sheet, but you have to move it to /usr/local/share/kweb before you can use it.

Now we will run create_pages.py again, but this time without the test mode option. If you are happy with the default options, simply run:
./create_pages.py
The following example will use udp-multicast, port 3000 and the black stylesheet:
./create_pages.py -p=3000 -i=239.100.0.1 -s=black.css

If you now open the GUI again and click a channel, a terminal will still pop up, but it will remain empty. To get rid of this terminal, we have to tell kweb to run the script “start_gnutv_web” in the background.

Open the directory “gnutv” in the file manager, right click on “start_gnutv_web” and select “copy file path”. Start kweb (from the application menu). By default it will show its menu page. Click on “Settings” and scroll down to a multi-line text area named “direct_commands”, add a new line at the bottom, paste the path into it (CTRL-V or select “Paste” from the right click menu) and click the ”Save” button.

You can run “create_pages.py” any time later again using different options. If you move the directory “gnutv” to a different place, you must run through all these steps again, so better leave it in place.

We will run one last command, which is needed for the next step;
sudo cp start_gnutv /usr/local/bin
This has to be done again, if you have run “create_pages.py” again with different options.
2) Create a Channel Database for rtranscode

The next step is to make the GnuTV backend usable for rtranscode including automatic tuning in menu mode or using the web interface and so we have to create a channel database. The script "create_chdat.py" will tune each channel, analyze it and will finally put all the results into a database file named gnutv_channels.dat.

There are no options to be set, so simply run

./create_chdat.py

but you should be aware that this will take quite a lot of time, about 10-12 seconds per channel.

If you have more than 40 channels in your channels.conf, automatic grouping is enabled. The program uses the provider names as group names. If it doesn't find one, the channel is added to the “Unknown” group. This is not an optimal solution but there is no other usable information in the channels.conf file. I recommend regrouping the channels later on, as described in part E) of this chapter.

While the program runs, it will print out to the terminal what it is doing. At the end it will tell you, how many channels it has added to the database. If analyzing of some channels failed, it will print a list of failed channels.

There a different reasons why analyzing a channel may fail:

a) The channel is not running while the program is trying to analyze it.

In this case you can try to add it later using rtranscode. You could try the following:
Tune the channel using the GUI. Check if it is running and can be watched with omplayerGUI. Now run

rtransode -g=udp://localhost:5001

if you are using the default settings. If not, you have to use a different URL, based on your (unicast) IP and port settings.

If you get a full result like this

name=URL videomode audiomode:audiopid,...

you can add it to the file gnutv_channels.dat. But do not use the name returned by rtranscode. Use the full matching name from channels.conf.

If you do not get a full result, continue with he following.

b) Analyzing fails, because some channel property is not supported by rtranscode. This may be:
1. a video encoding which is not supported (only MPEG2 and H264)
2. audio encoding(s) not supported (only MPEG 1,2,3, AC3 and AAC)
3. video resolution (width x height) is not supported by any video mode
Problems 1 and 2 are not easy to resolve, but for problem 3 you may find a solution
in chapter IV, C).

To find the cause of the problem, tune the channel using the GUI and then run
\texttt{omxplayer -i udp://localhost:5001}
(if you are using the default configuration).

You will find the relevant information in the description of the video and audio
streams.

Now it's time to test everything in \texttt{rtranscode}. I assume, that you are still inside the
gnutv directory. Run
\texttt{rtranscode -d=gnutv_channels.dat -z=rt_gnutv.py}

\texttt{rtranscode} should now run in menu mode. Select any channel for transcoding. For a
few seconds you should see a “tuning ...” message and afterwards the streaming
page. Test the stream with omxplayerGUI.

If you stop the stream and start the same channel again, no tuning message should
be displayed. If you stop the stream and start another one, you should always see a
new tuning message.
If everything works and you want to use gnutv as your default backend, you should
run the following commands:
\texttt{sudo cp gnutv_channels.dat /usr/local/share/rtranscode/channels.dat}
\texttt{sudo cp rt_gnutv.py /usr/local/share/rtranscode/tuner.py}

\textbf{D) Create a MuMuDVB Backend}

\textbf{1) Create the MuMuDVB Configuration files}

MuMuDVB is a much more advanced TV server backend than GnuTV, but it is
harder to configure. It needs a special configuration file for each channel (or
transponder). Creating these configuration files manually for hundreds of channels
is really a lot of work. And the number of possible configuration options is really
huge.

Note: If you are already using MuMuDVB and have your own configuration scripts,
you can skip this part and the next one and continue with part 3).

\texttt{rtranscode4} contains a script which will automatically create these configuration
files based on the information found in the channels configuration file which we
have created with \texttt{w-scan}. 

63
MuMuDVB supports a tuning method called “full autoconfiguration”. In this mode it extracts most of the information it needs from the stream itself, which makes configuration a lot easier. It works for more than 99% of all channels and we will use it for our system.

By default, MuMuDVB streams all channels via UDP. Optionally you can also use HTTP streams. If the HTTP server is enabled, it can also be used to communicate with MuMuDVB (get a playlist, channel page and more). This is required for our systems and so the HTTP server is enabled. I have disabled UDP for different reasons, but you can enable it, if you need it.

The MuMuDVB configuration files are built from templates which you will find inside the “templates” directory. They should just work for most users, but depending on your hardware you may have to modify or extend them. Let's have a look at one of these template files first (we’ll use the DVB-S template, because it is the most complex one and might require some modifications depending on your satellite antenna).

```
card=$card$
tuner=$tuner$
freq=$freq$
pol=$pol$
srate=$srate$
delivery_system=$delivery_system$
#sat_number=0
#switch_input=0
#diseqc_repeat=1
autoconfiguration=full
autoconf_radios=0
#cam_support=1
#scam_support=1
multicast_ipv4=0
sap=0
rewrite_sdt=0
rewrite_pat=0
unicast=1
#port_http=$port_http$
unicast_consecutive_errors_timeout=20
```

All values which are enclosed by “$” signs will be set automatically by our script so we don't have to care about them.

Some options are commented out (by a leading '#'). You may have to enable them (remove the leading '#') and add an appropriate value if you need these settings, especially the sat_number (1..4) or the switch_input (1..15) to send diseqc signals.

If your DVB device includes a CAM, you have to enable cam_support. If you are (legally!) using a softcam, you have to enable scam_support,
If you want to (additionally) use UDP multicast and your router supports it, set
\texttt{multicast_ipv4=1}

I do not recommend it, if you are connected via WiFi or if you plan to stream complete transponders, because it may really clutter up your network.

For everything else I recommend reading the detailed documentation on the MuMuDVB website: \url{http://mumudvb.net/documentation/asciidoc/mumudvb-2.1.0/README_CONF.html}

But for most users it will all work out of the box. And as the creation of (hundreds of) MuMuDVB configuration files takes almost no time, using trial and error to get it to work should be easy enough. So let's get to work:

Open a terminal and cd into the mumudvb directory inside the rtransode4 directory. Before we run the first script, we have a look at the two possible options:
\texttt{-p=port}
By default MumUmDVB uses port 4242 for its HTTP server. You can change the port with this option, but then you have to use the same option for all other scripts as well!
\texttt{-d=adapter,frontend}
adapter and frontend must be numbers, e.g.: 0,0. You only have to use this option, if you have multiple USB-DVB devices or if your device supports multiple tuners (frontends).

Now run:
\texttt{./create_configs.py}
(and add the options if you need them).
The program will create a channel configuration file for each channel inside the “channels” directory and also a configuration file for each transponder (frequency) it finds inside the “transponders” directory. It will also create a channels.list and a transponders.list file and last but not least a shell script “prepare_mumudvb”.

Before we start to test our configuration files, you should run this script once (as root):
\texttt{sudo ./prepare_mumudvb}

Now we want to test, if MuMuDVB really accepts our configs:
\texttt{cd channels}
\texttt{ls}
This should show you a lot of configuration files. Now run
\texttt{../start_mumudvb_test filename}
Replace “filename“ by any of the configuration files which have been displayed using the “ls” command.
MuMuDVB will display lots of messages on the screen. If it doesn't stop with an error message, the last lines might look similar to this:

```
Info: Autoconf: Diffusion 1 channels
Info: Autoconf: Channel number :  6,  service id 31220  name : "EURONEWS FRENCH SD"
Info: Autoconf: Unicast : Channel accessible via the master connection, 0.0.0.0:4242
Info: Autoconf: The NIT version changed, channels number could have changed !
Info: Main: Channel "EURONEWS FRENCH SD" back.Card 0
Info: Autoconf: We got the NIT, we update the channel names
```

To test the stream itself, open omxplayerGUI from the program menu, add the following into the text field at the top

```
http://localhost:4242/playlist.m3u
```

and click “Play/Stream”. A video window should open and show the TV stream after a few seconds. (If you have changed the default port number, replace “4242” by your own port number).

To stop the stream, type “CTRL+c” in the terminal.

Now we want to run the same test using a transponder configuration.

```
cd ../transponders
ls
../start_mumudvb_test filename
```

Replace “filename“ by any of the configuration files which have been displayed using the “ls” command.

Now MuMuDVB should report, that it has tuned a number of channels.

To watch the channels use omxplayerGUI as shown above. If you stop the current video by clicking on the “[]” symbol at the bottom, the video window should show you the names of all available channels and you can start watching any channel by double clicking its name.

And what should we do if it doesn't work? You should carefully check which error is shown by MuMuDVB. If it can access the hardware, but cannot tune, there are a number of problems I can think of:

Did you select the right frontend (if your device supports more than one tuner)?

Satellites only: Does your antenna require diseqc commands and which?
Are special options required which are not part of the template? This may depend on your country, provider, the satellite you are trying to access etc. You have to check the MuMuDVB configuration options for your type of device (DVB-T, DVB-S, DVB-C, ATSC ...). MuMuDVB uses some default settings which may not match your situation. See: [http://mumudvb.net/documentation/asciidoc/mumudvb-2.1.0/README_CONF.html](http://mumudvb.net/documentation/asciidoc/mumudvb-2.1.0/README_CONF.html)

Fortunately recreating the configuration files with modified options (from a modified template) takes less than a second. Or you can modify just one existing configuration file and use that for testing, until it works, and then add the required modifications to your template.

If the system is working you should run two more commands:
```
sudo cp start_mumudvb /usr/local/bin
sudo cp prepare_mumudvb /usr/local/bin
```

There is one more thing to consider, before you go on with the next steps. All other scripts will use the information from either channels.list or transponders.list. These (text) files are built using automatic grouping, either by stream provider (channels) or by transponder. It might be a good idea, to edit these files now and use your own group names (see part E) for details). But it's also possible to edit the groups later in the rtranscode channel database.

### 2) Create the GUI

The next script will create a kweb GUI for your MuMuDVB backend. Before we run the script we'll have a look at its possible options:

- `-p=port`
  only required if you have created the configuration files using your own port setting (must be the same here).

- `-t`
  Use transponders for the GUI instead of single channels.

- `-s=stylesheet`
  This option selects a style sheet to be used for the interface, e. g.

- `-s=black.css`

You can only use style sheets from /usr/local/share/kweb. To get a list of available style sheets, run:
```
ls /usr/local/share/kweb/*.css
```
You can also create your own style sheet, but you have to move it to /usr/local/share/kweb before you can use it.

Now lets create the GUI. Run ./create_pages.py optionally using the -p and -s options (do not use -t right now).

This will create a lot of files (mostly HTML). One file, “MuMuDVB-GUI” is an executable, which you can double click in the file manager (and then select “Execute”) to open the GUI.

If you click on any channel name in the scrollable box on top, a tuning message will appear for a few seconds and will be replaced by two buttons afterwards. At the same time a terminal will pop up, which doesn't display anything (because we are already running MuMuDVB in silent mode). Ignore this for now, we will get rid of it later on. If you click on “WatchTV”, an omxplayerGUI window will open and play the selected channel.

Clicking on “Stop TV Server” will stop the MuMuDVB server. If you want to stream another channel, you can simply click on it's name. It's not required to stop the server first.
If you click “Quit MuMuDVB GUI”, the window will close and the server will be stopped. If you simply close the window, the server will keep running in the background.

Now we will run the create_pages.py again but this time we will use transponder mode:
./create_pages.py -t

If we start the GUI afterwards, the content of the scrollbox will look a bit differently. Instead of channel names, it shows transponders (frequencies), followed by the first two known channel names. If you move the mouse on top of any entry, all known channels from this transponder will be shown. If you tune any transponder and
afterwards click on “Channel List”, a second kweb window will open and show the channel list page of MuMuDVB’s web server.

If you click on “Watch TV”, a playlist containing all channels will be sent to omxplayerGUI. You can switch between channels, using the up and down symbols or keys. If you click on the stop symbol, omxplayerGUI will show you a list of all available channels.

Once the MuMuDVB server is running, either in single channel or in transponder mode, the TV streams are available anywhere on your local network. The easiest way to access the streams from any
kind of player, is to use the playlist URI: \url{http://hostname:4242/playlist.m3u}.
Simply replace “hostname” by the host name or IP of your Raspberry Pi running the server. If you have set another port number, you will also have to replace “4242”.

If you are running kodi on the same or another Raspberry Pi, it is well suited to watch the video streams. Use the PVR IPTV Simple Client plugin and configure it to use the playlist URI. Kodi can switch much faster between video channels than omxplayer.

It’s up to you to decide if you want to run MuMuDVB from the GUI in single channel or transponder mode - and it's quite ease to reconfigure it at any time.

To get rid of terminal, which is popping up each time we tune a channel, we have to tell kweb to run the script “start_mumudvb” in the background.

Start kweb (from the application menu). By default it will show its menu page. Click on “Settings” and scroll down to a multi-line text area named “direct_commands”, add a new line at the bottom
```
start_mumudvb
```
and click the ”Save” button.

There’s one more thing to do, if you plan to regularly use the MuMUDVB backend. From a terminal run
```
sudo nano /etc/rc.local
```
Add a new line just before line “exit 0”:
```
prepare_mumudvb
```
and save the result with CTRL+o.

### 3) Create a Channel Database for \texttt{rtranscode}

Before we run the script to create a complete channel database for \texttt{rtranscode}, we'll have a look at the possible program options:

```
-p=port
```
If you have set the HTTP port previously you must use this argument here, too.

```
-t
```
Transponder mode: use transponder configuration files instead of single channel files.
This option is for users, which are already using MuMuDVB and have manually created their configuration files. “directory” must be a full path which points to a directory which contains your configuration files (and nothing else). It will only work, if your files use the “unicast=1” option. You may have also to use the -p option, if you are not using the default HTTP port.

This script will run for a long time. It takes approximately 12 seconds per channel. You should carefully consider, if you want to use transponder mode or not.

Ready to start? Then run

`.create_chdat.py`

(followed by the options you need).

While the program runs, it will print out to the terminal what it is doing. At the end it will tell you, how many channels it has added to the database. If analyzing of some channels failed, it will print a message at the end and create a file named “failed_channels.txt”.

This program will create a new channel database named “mumudvb_channels.dat”. It will also create a user script for rtranscode (to allow automatic tuning) named “rt_mumudvb.py”.

The database will contain the same groups as channel.list or transponder.list, which have been created together with the MuMuDVB configuration files (and which you may have edited afterwards).

Now it's time for a first test. From inside the mumudvb directory, run

`rtranscode -d= mumudvb_channels.dat -z= rt_mumudvb.py`

rtranscode should now run in menu mode. Select any channel for transcoding. For a few seconds you should see a “tuning ...” message and afterwards the streaming page. Test the stream with omxplayerGUI.

If you stop the stream and start the same channel again, no tuning message should be displayed. If you stop the stream and start another one, you should always see a new tuning message, except if you have used transponder mode. If the new channel belongs to the same transponder, no tuning message should appear.

Now let's have a look at the failed channels, if there are any. The file “failed_channels.txt” contains lines like this:

S19.2E-12603-28545.conf

71
At the beginning of a line you will always see the name of the configuration file. If it is not followed by a link, tuning the channel has failed. Perhaps the channel was not running when you created the channels database (a solution for such channels will be shown below).

If the name of the configuration file is followed by a link, analyzing the channel failed for some reason. Analyzing may fail, because some channel property is not supported by rtranscode. This may be:
1. a video encoding which is not supported (only MPEG2 and H264)
2. audio encoding(s) not supported (only MPEG 1,2,3, AC3 and AAC)
3. video resolution (width x height) is not supported by any video mode

Problems 1 and 2 are not easy to resolve, but for problem 3 you may find a solution in chapter IV, C).

To find the cause of the problem, you should manually tune the channel (or transponder) in test mode as shown in part 1), using the configuration file name from the failed_channels list.

Open a second terminal and run
```
omxplayer -i URI
```
where is URI is taken from the failed_channels.txt file, e.g.
```
omxplayer -i http://127.0.0.1:4242/bysid/6392
```

You will find the relevant information in the description of the video and audio streams.

If a channel could not be added to the database, because it wasn't running, you an add it at a later time using another script:
```
./add_channels.py path-to-config-file
```
You must use the full path to the configuration file, e.g.
```
./add_channels.py /home/pi/transcoder4/mumudvb/channels/S19.2E-12603-28545.conf
```

If the channel is found and successfully analyzed, it will be added to the (end of the) database and a modified user script “rt_mumudvb.py” will also be created. The added channel will appear in the last group. You should open “mumudvb_channels.dat” and move the line into the group to which it belongs.

If everything works and you want to use MuMuDVB as your default backend, you should run the following commands:
```
sudo cp mumudvb_channels.dat /usr/local/share/rtranscode/channels.dat
sudo cp rt_mumudvb.py /usr/local/share/rtranscode/tuner.py
```
E) Organizing Your Channel Database

DVB-C and DVB-S may deliver a huge number of channels, sometimes even more than a thousand. Even if you have removed a lot from your channels.conf file, the number is often too large to be handled without sensible grouping. When creating the channel databases (both for GnuTV and MuMuDVB), automatic grouping is applied, based on provider names or transponder frequencies. But this is not a really satisfactory solution. It's much better to create groups using sensible categories.

Grouping is used by rtranscode in menu mode and by the kweb and web interfaces. They will be much more user friendly, if you put some effort into sensible grouping. Here's my recommended way:

Start with a new (empty) file using leafpad. Add a number of sensible group names like this (use your own categories):

```
[Public TV]
[Private TV]
[News]
[Sports]
...
```

It may also be useful, to create separate groups for SD and HD channels.

Now open your channel database file and copy line by line (not the group names, just the channel description) into the new file, placing it into the matching group area. You may prefer important channels to appear on top.

Save the file finally as your new channels database file.
IX Copyright, License and Source Code

http-launch:
Copyright (C) 2013 Sebastian Dröge <slomo@circular-chaos.org>
Modifications by Guenter Kreidl <gkreidl@krefelder-life.de> Copyright (C) 2015-2018

Karrigell:
Copyright (c) 2009, Pierre Quentel pierre.quentel@gmail.com
You will find the original license text in the web directory

MuMuDVB:
© Brice Dubost
Website: http://mumudvb.net/

gst-omx:
Maintainer: Sebastian Dröge
Website: https://gstreamer.freedesktop.org/modules/gst-omx.html

rtranscode, rtc4.so and all supplied Python scripts:
Copyright (C) 2018 Guenter Kreidl <gkreidl@krefelder-life.de>

Documentation:
Copyright (C) 2018 Guenter Kreidl
It may be distributed without charge in digital form, but not modified in any way.

The programs I created for this project are free software; you can redistribute them and/or modify them under the terms of the GNU Library General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

These programs are distributed in the hope that they will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Library General Public License for more details.

The source codes are part of the distribution. You will find them in the "sources" directory.

rtranscode and rtc4.so are written in Python and compiled with nuitka. You can use the compile script to compile a modified version (nuitka required).
http-launch.c contains my modifications to the original program by Sebastian Droege. You cannot compile it directly.

Clone from https://github.com/sdroege/http-launch up to commit 5e7b8a936bfcb2f86f6e78c69929d5ace617dc7. Replace the file http-launch.c with the version I supplied. Run "./configure" and then "make".

The directory gst-omx contains two source files from gstreamer-omx 1.10.5 with a few patches which I have added. These sources have been used to build gstreamer1.0-omx-rpi_1.10.5-2+rpi+patches_armhf.deb.