gr-ais i SuSE12.3

First a **Warning! Never use gr-ais in real navigation it's neither stable or reliable. The description below is just made for ”amusement” in the purpose of getting ”real-time” updated AIS-messages to OpenCPN!**

Suggestions for suitable DVB-T USB-dongles
you can get from here: http://www.reddit.com/r/RTLSDR/comments/s6ddo
In he examples below we used two different USB-recievers: One Newsky TV28T Coax DVB-T DAB DAB+ FM Stick SDR with RTL2832U-chipset and E4000-tuner, respectively a LIFEVIEW NO DVB-T USB DELUXE LV5T with rtl2832 chipset and fc0013 tuner. -We both live in in different places in Stockholm, neither of us, living within 1000 m from the sea (or lake), 10 resp. 45 m above sealevel, have got a visible contact with a harbor. We both used the small indoor antennas that came with the dongles.

Prerequisites

The description below comes from an installation in OpenSuSE12.3 with kernel 3.7.10-1.1-desktop, in both cases x86_64. The reason we used such a new version was due to a basic misunderstanding. We thought that the cipset-modules rtl2832 and tuner-modules fc0013 or e4000 respectively were necessary for being able to install gnuradio. Kernel support for those modules were first integrated in kernel 3.6 →.

Many observations after our installation indicated that our assumption was wrong. -Finally we tried it out by shutting them -and the other dvb-t-modules- down using blacklist/rmmode.
To our surprise gnuradio worked perfectly without them!
Note: the modules rtl2832 and fc0013/e4000 are not necessary for running gnuradio, rtl-sdr, osmosdr or gr-ais. You only need them if you want to use the USB-receiver for watching digital TV (DVB-T) and listen to digital radio (DAB) with the ordinary Linux softwares. Therefore we won't go further about those modules here (f.i. different workarounds, special ways of building them a.s.o)
This means, do you want to watch DVB-T et c with ordinary software you should have kernel 3.6 or higher. If you only want to be able to run gnuradio and it's modules it should work at least from kernel 2.6 and higher. (We have seen examples of installations in f.i. Ubuntu10.04lts).

If you want to get your kernel version, simply open a terminal and execute:
```
$ uname -a
```
(Below I consequently use **$** to indicate a user-command and **#** to indicate a command as superuser if anyone should be confused. You never see $ in SuSE-Gnome-terminals though.)

Installation of the depedencies

(for the builds and softwares)

Repositories:
Here we have used quite ”standard” openSUSE repositories:
zypper repositories: Ordinary openSUSE + updates, repo-oss, repo-non-oss, Education, Factory-Packman, Packman, server:/php/applications

Adjust your PATH:
Add following to your PATH so your ”builds” finds the needed libraries when placed in the ”unordinary” /usr/local/:

That is, open the file .bashrc in your users home (~/ means your users home-library)
Open ~/.bashrc and in your favorite texteditor add the following lines and save it:
PKG_CONFIG_PATH=$PKG_CONFIG_PATH:/usr/local/lib/pkgconfig
export PKG_CONFIG_PATH
PYTHONPATH=$PYTHONPATH:/usr/local/lib64/python2.7/site-packages:/usr/lib64/python2.7/site-packages
export PYTHONPATH
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/lib:/usr/local/lib64
export LD_LIBRARY_PATH
This could possibly be done in a better way, anyway the above worked for it's purposes.

Installation of build- and other software dependencies.
(Here we include the dependencies from the build of OpenCPN because we have done that in our systems and wouldn't risk to miss any dependencies for the gnuradio-build that we might possibly have installed before.)
Possibly this zypper install is somewhat ”overkill” but, better than insufficient:
# zypper install git cmake gcc-c++ libstdc++-devel gtk2-devel rpm-build libcppunit-devel doxygen fftw3-devel gsl-devel libjack-devel libqt4-devel libSDL-devel libusb-1_0-devel orc portaudio-devel libportaudio2 python-cheetah python-devel python-lxml python-wxWidgets python-wxWidgets-devel libqwt5 qwt-devel gettext-runtime gettext-tools wxWidgets-wxcontainer-devel libbz2-devel gpsd-devel xmlto tinyxml-devel swig python-scipy python-sphinx python-docutils python-matplotlib-wx

To this comes the boost-package from http://software.opensuse.org/package/boost :
If you -like me- don't manage to install this, at least the following boost packages (that I had installed earlier for other reasons) should be sufficient (or even ”overkill”):
# zypper in boost-jam-201104 boost-license1 boost-devel libboost_locale1 libboost_iostreams1 libboost_filesystem1 libboost_serialization1 libboost_timer1 libboost_python1 libboost_test1 libboost_program_options1 libboost_chrono1 libboost_date_time1 libboost_graph1 libboost_wave1 libboost_system1 libboost_regex1 libboost_mpi1 libboost_random1 libboost_math1 libboost_thread1 libboost_signals1

To this a sourcecode installation:
Get comedilib 0.10.1 from http://www.comedi.org/download.html
Navigate your terminal to the downloaded directory
$ cd comedilib-0.10.1
Read the INSTALL notes: from there:
configure with the following -flag
./configure --with-udev-hotplug=/lib --sysconfdir=/etc
make
$ su
Password:
# make install

Note! Though these basic installations are very extensive there is still a risk that we have forgotten something? Our SuSE12.3-installs are not fresh, we have used them for many other builds and installations. We've tried to make it as complete as we could though, but if we've missed some dependencies you'll notice it from the error-reports when running ./configure or cmake below. If any you can go back and complete your dependency-installation from those reports. -That's basically how we got along to finally succeed.

There is also a possibility that some of the packages above have got dependencies that are installed later in this decription. Packages like f.i. libusb-1_0-devel python-cheetah or swig? Don't worry, I'll repeat them again when they should install with no problem. Just retry then.

Installation of gnuradio, rtl-sdr, osmosdr and gr-ais

Note! It is -as far as we have learned- important to install the software in the order we have listed them below!

1. rtl-sdr

http://sdr.osmocom.org/trac/wiki/rtl-sdr
http://inst.eecs.berkeley.edu/~ee123/fa12/rtl_sdr.html
If you didn't initially: Now install the dependency:

```
#  zypper in libusb-1_0-devel
```

[*As you may note, the repetition here of the requirements and dependencies from above get somewhat contradictory and confusing. Here you might miss cmake, git and others that comes below and you are perfectly right! These repetitions are only reminders if you didn't get all in your initial zypper installation!*]

Get rtl-sdr:

```
$ git clone git://git.osmocom.org/rtl-sdr.git
$ cd rtl-sdr/
$ mkdir build
$ cd build/
$ cmake ../ -DINSTALL_UDEV_RULES=ON  [flag from the osmocom-link above]
$ su
password:
# make install
```

The now created /etc/udev/rules.d/rtl-sdr.rules might now have to be renamed: from rtl-sdr-rules to 99-rtl-sdr.rules:

```
# cd /etc/udev/rules.d/
# mv rtl-sdr.rules 99-rtl-sdr.rules
```

Now test your USB-stick with rtl-sdr (the ”answers” and values below are only given as examples):

```
$ su
Password:
#  udevadm control --reload
# rtl_test -t
Found 1 device(s):
  0:  Lifeview LV5TDeluxe

Using device 0: Lifeview LV5TDeluxe
Detached kernel driver
Found Fitipower FC0013 tuner
Supported gain values (23): -9.9 -7.3 -6.5 -6.3 -6.0 -5.8 -5.4 5.8 6.1 6.3 6.5 6.7 6.8 7.0 7.1 17.9 18.1
18.2 18.4 18.6 18.8 19.1 19.7
No E4000 tuner found, aborting.
Reattached kernel driver
larse3:/home/larse #
```

Now test your stick's maximum "samplerate"...
To check the maximum samplerate possible on your machine,

type (change the rate down until no sample loss occurs):

$ rtl_test -s 3.2e6
lost bytes continously

$ rtl_test -s 2.8e6
lost bytes continously

$ rtl_test -s 2.6e6
just **one** lost post!

$ rtl_test -s **2.5e6**
No lost posts!

Max samplerate för my LV5TD: 2.5e6

2a. comedi

If not initially done:
Get and install comedi as sourcecode from http://www.comedi.org/download.html
Unpack (tar -xvzf et c) and install comedilib 0.10.1 src
Read the file INSTALL..
$ ./configure --with-udev-hotplug=/lib --sysconfdir=/etc
$ make
$ su
Password:
# make install

2b. gr-uhd

If not initially done: First install the dependencies:
# zypper in boost-devel python-cheetah doxygen python-docutils

Get uhd from Ettus:
$ git clone git://github.com/EttusResearch/uhd.git
$ cd uhd/host
$ mkdir build
$ cmake ../
$ make [seems to work all the way]
su
#make install [seems ok]

With your favourite texteditor: Create a file /etc/ld.so.conf.d/**uhd.conf**
# cd /etc/ld.so.conf.d/
# vim uhd.conf
Just add the line: /usr/local/lib Save and Close.
Then execute
# /sbin/ldconfig
[**ldconfig** creates the necessary links and cache to the most recent shared libraries found in the directories specified on the command line, in the file */etc/ld.so.conf*, and in the trusted directories (*/lib* and */usr/lib*).]

(Here we skip gr-shd as we didn't get shd working with our installations of gnuradio. shd is

not essential to gr-ais!)

2c. gnuradio

If not done initially: Install the buildrequirements and dependencies:

```
# zypper in cmake libcppunit-devel doxygen fftw3-devel git gsl-devel libjack-devel libqt4-devel libqwt5 libSDL-devel libusb-1_0-devel orc libportaudio2 portaudio-devel python-cheetah python-devel python-lxml python-wxWidgets python-wxWidgets-devel qwt-devel wxWidgets-wxcontainer-devel xmlto
```

If you havent done initially: Also install:

```
# zypper in swig python-devel python-sphinx python-scipy
```

If not initially done: Get boost from http://software.opensuse.org/package/boost
Alternately install the boost-packages listed in the introduction.

Get gnuradio from http://gnuradio.org/redmine/news/19
Unpack it

```
$ tar xvfz gnuradio-3.6.4.1.tar.gz
$ cd gnuradio-3.6.4.1
$ mkdir build
$ cmake ../ -DQWT_INCLUDE_DIRS=/usr/include/qwt5 -DENABLE_GR_SHD=OFF
```

[-DQWT_INCLUDE_DIRS=/usr/include/qwt5 is to give the path to qwt, so it can be found shd didn't work with our gnuradio's (therefore you can skip to build and install it), hence the last flag (-DENABLE_GR_SHD=OFF)]

```
$ make
$ su
Password:
# make install
```

Installs in /usr/local.

3.gr-osmosdr

```
$ cd /home/larse/bin/git
```

Get osmosdr from git

```
$ git clone git://git.osmocom.org/gr-osmosdr
$ cd gr-osmosdr
$ mkdir build
$ cd build
$ cmake ../
-- ######################################################
-- # gr-osmosdr enabled components
-- ######################################################
--   * FunCube Dongle
--   * IQ File Source
--   * Osmocom RTLSDR
--   * RTLSDR TCP Client
--   * Ettus USRP Devices
--
-- ######################################################
-- # gr-osmosdr disabled components
-- ######################################################
--   * Osmocom IQ Imbalance Correction
--   * sysmocom OsmoSDR
--   * Osmocom MiriSDR
--   * HackRF Jawbreaker
$ make
$ su
Password:
# make install
```

With this basic installation you now have got access to among other things rtl-sdr-FM-radio. And from here you can easily add a lot of gnuradio-"modules", like gr-ais, gr-air-modes, dabreciever, fm-reciever et c. In the following we concentrate on gr-ais.

4. gr-ais

Get gr-ais from github:

```
$ cd /home/larse/bin/git [It's simply were I keep my git downloads, you can put them anywhere in your users home]
$ git clone git://github.com/chgans/gr-ais.git
$ cd gr-ais
$ mkdir build
$ cd build
$ make
$ su
Password:
# make install
```

Installs in /usr/local/

Using gnuradio-gr-ais

Now I start gr-ais help (to get the basic flags):

```
$ ais_rx.py -h
Usage: ais_rx.py [options]

Options:
  -h, --help            show this help message and exit
  -a ADDR, --addr=ADDR  UHD source address
  -s SUBDEV, --subdev=SUBDEV
                        UHD subdev spec
  -A ANTENNA, --antenna=ANTENNA
                        select Rx Antenna where appropriate
  -e ERROR, --error=ERROR
                        set offset error of USRP [default=0]
  -g dB, --gain=dB      set RF gain
  -r RATE, --rate=RATE  set fgpa decimation rate to DECIM [default=256000.0]
  -F FILENAME, --filename=FILENAME
                        read data from file instead of USRP
  -v, --viterbi         Use optional coherent demodulation and Viterbi decoder
  -t, --tcp             Start a TCP server on port 9987 instead of outputting
                        to stdout. Useful for gpsd.
  -d, --rtlsdr          Use RTL-SDR dongle
  -D ARGS, --args=ARGS  arguments to pass to UHD/RTL constructor
```

Expert:
$

Flags -d: use rtl-sdr stick/dongle, -e frequency correction, -t send as tcp-out to 127.0.0.1:9987
First check the frequency correction value (e) for your ”warm” USB-tuner, that is when it has reached it's stable working temperature:
Note! To find and set the right value for the frequency correction ”e” ppm is, beside the properties of the antenna essential to at all being able to use gr-ais, or getting anything useful out of your stick/dongle! **Note!**

Checking frequency correction ”e”: three and a half different ways are described

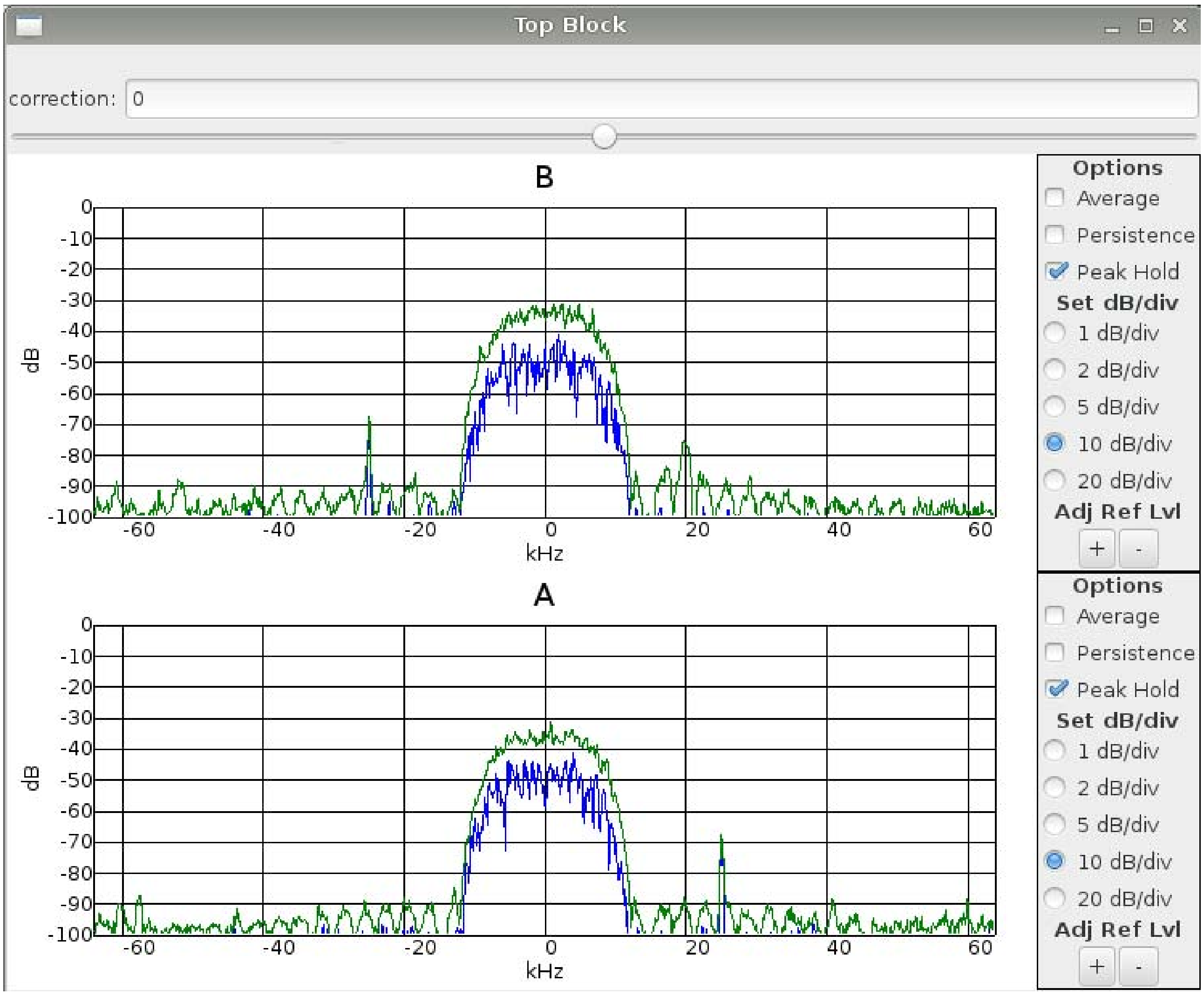
Checking the frequency correction ”e” using Gnu Radio Companion

(earlier
$ grc)
now
$ gnuradio-companion

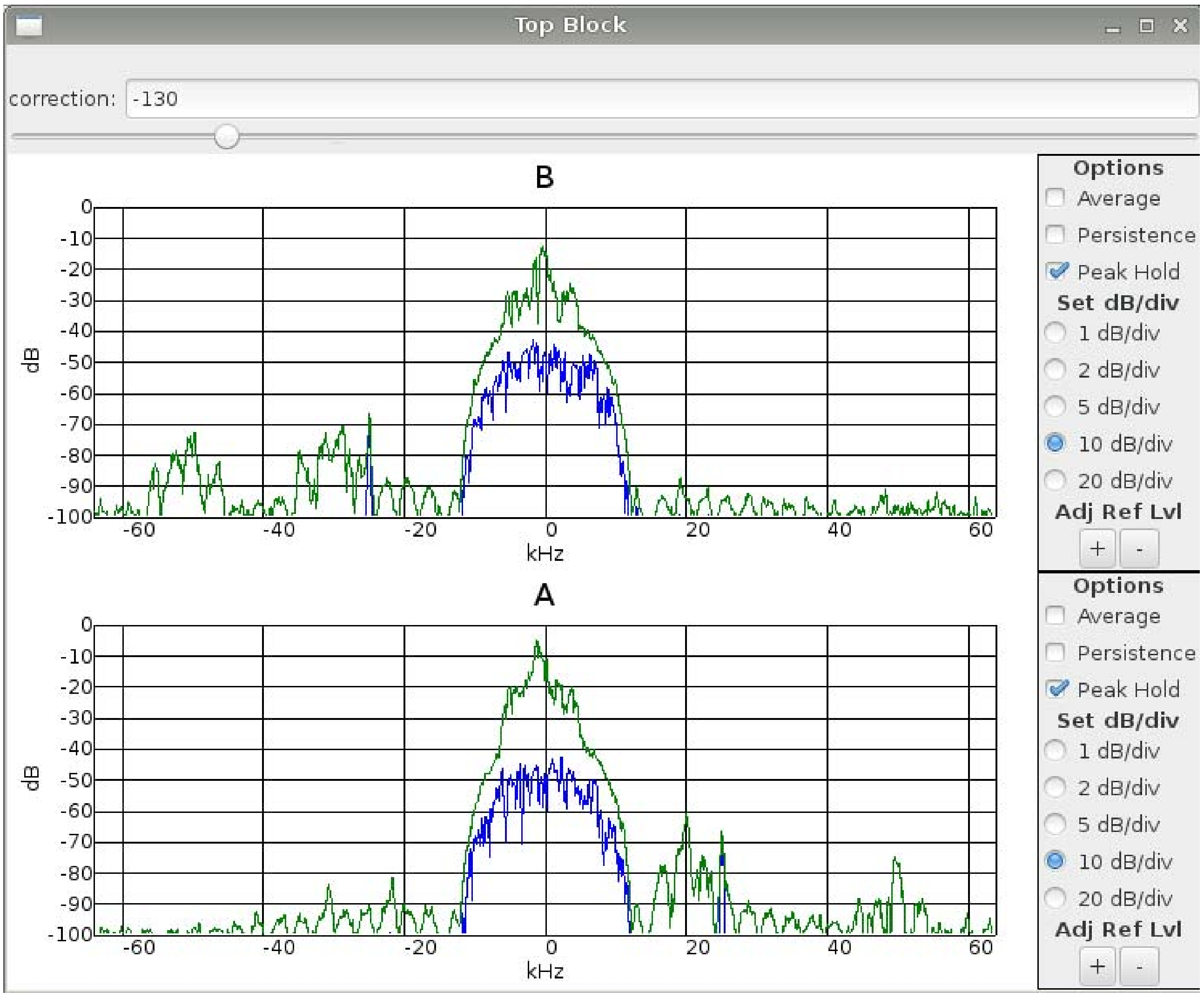
Start GRC, Open and run http://steve-m.de/files/rtl2832/gr-ais.grc from http://www.reddit.com/r/RTLSDR/comments/100lrn/need_help_on_how_to_use_a_realtek_dvbt_dongle_to/
Use the picture the author refers to as a reference (http://rof.li/pic/ais.png).
It is really not easy to find the correct frequency correction this way, but a suggestion might be:
Adjust the Reference Level upwards (+) so that the sidefrequencies get close to the base level 0.
Now vary the frequency correction with the slider in the upper field (or writing the values) step by step by f.i. values of 10 or 20.
For every new value of frequency correction you set: un- and re-check Peak hold on the right side of the interface ”Options” (This zeroes/resets the up to now achieved accumulated peak value that is not adequate for your new set value of frequency correction.).
When you in this way get a green peak-curve that is as narrow and sharp as possible it is probable that you've got a fairly good correction value for your tuner and a value for the flag ”e” in ais_rx.py

Below two pictures where I try 0 and -130 as frequency error correction. -130 I got from stepping the way described above.

Frequency correction e=0



Frequency correction e=-130



correction: -130

Options: Average [ ] Persistence [ ] Peak Hold [✓]
Set dB/div: 1 dB/div, 2 dB/div, 5 dB/div, [✓ 10 dB/div], 20 dB/div

Checking the frequency correction ”e” using SDR# (what I myself found to be simplest)
Note! SDR# requires the Mono Integrated Development Environment (http://rtlsdr.org/softwarelinux http://monodevelop.com/) to be installed, but this cannot be covered here.

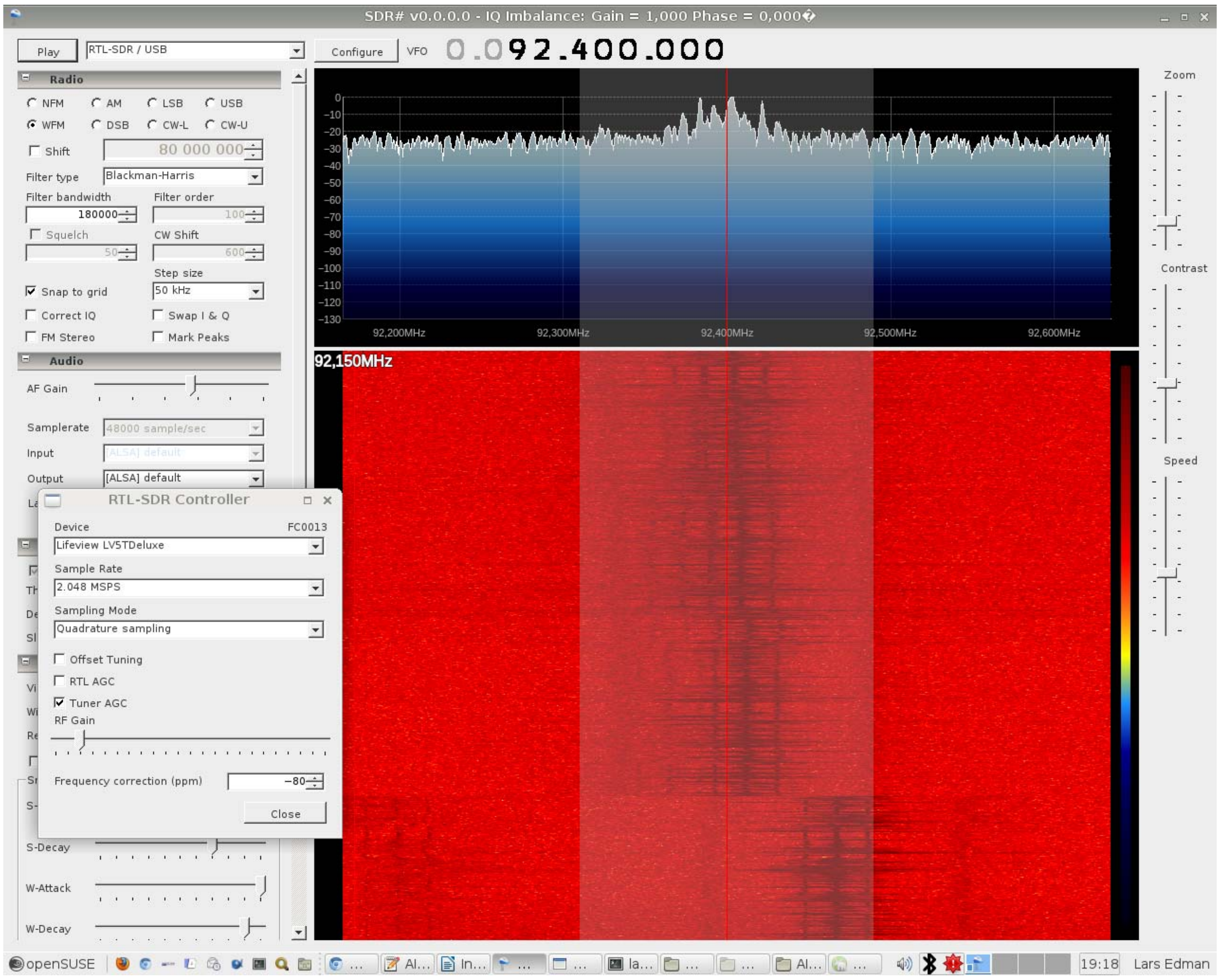
See http://www.atouk.com/SDRSharpQuickStart.html
SDR# ”e” is as mentioned frequecy correction in PPM
Use http://www.atouk.com/SDRSharpQuickStart.html#adjusting as a guide
First let the tuner reach working temperature, or it will slide in frequencies
Start SDR#

```
$ mono /directory/for/your/Release/SDRSharp.exe
```

Try set a frequency you know that contains a stable signal. (Me using the ordinary radio FM broadcasts in Sweden)
Increase Zoom and possibly Contrast till you can see if the middle of the signal (lower field) is centered over your ”scan-line” (upper field)
If the middle of the signal is to the left of the scan-line: Open Configure (Button in the tool bar), (possibly check Tuner AGC)
Now increase Frequency correction (ppm) till the signal is centered under the scan-line.
If the middle if the signal is to the right of the scan-line: Open Configure (Button in the tool bar), Decrease the Frequency correction until the signal is centered under the scan-line.
The frequency correction that centers your signal corresponds to the e-value for your tuner.

Below two pictures where I step down from the frequency correction e=-80 to e=-130

e=-80


SDR# v0.0.0.0 - IQ Imbalance: Gain = 1,000 Phase = 0,000�

Play | RTL-SDR / USB | Configure | VFO 0.092.400.000

Radio: NFM, AM, LSB, USB, [✓ WFM], DSB, CW-L, CW-U

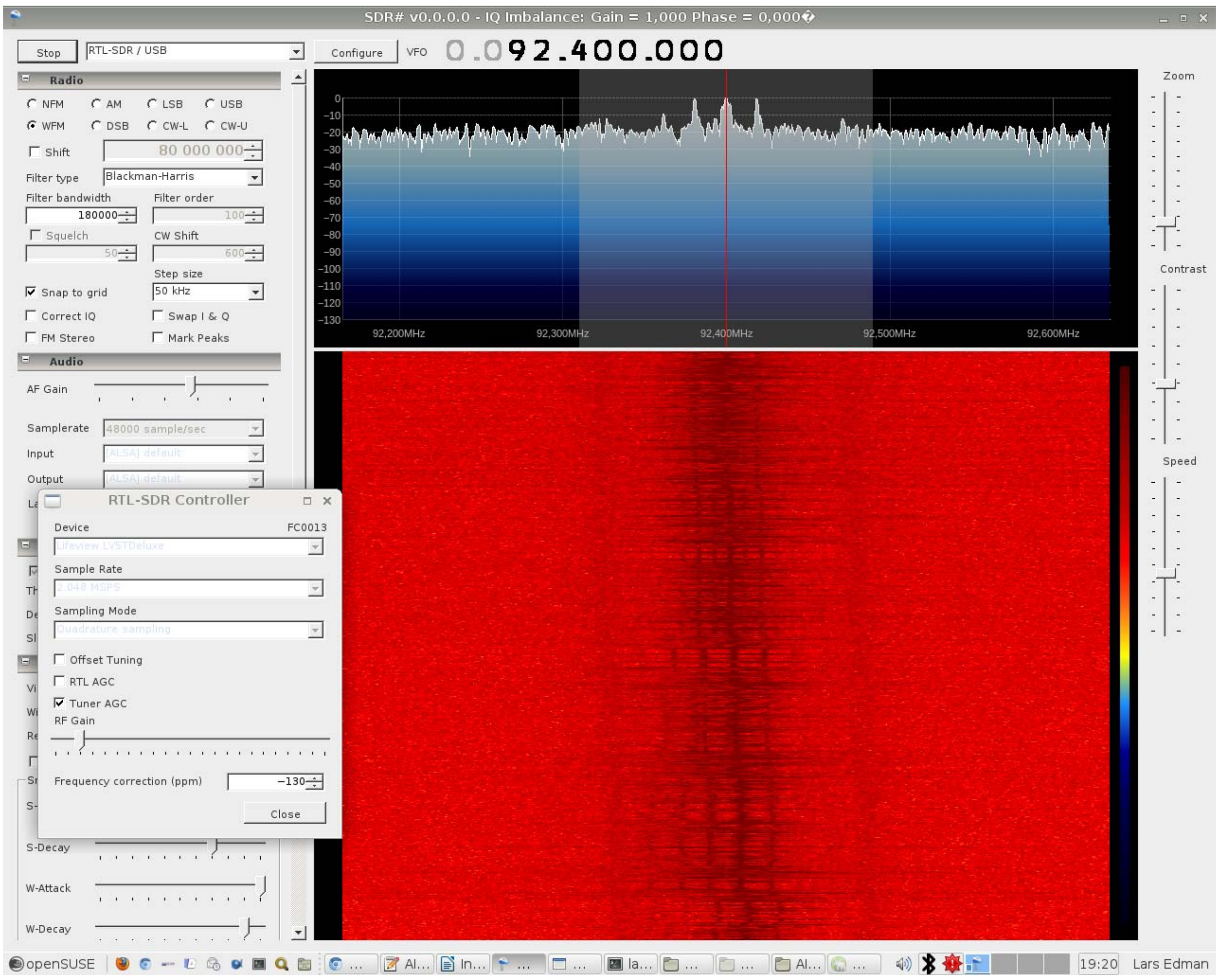
Shift: 80 000 000
Filter type: Blackman-Harris
Filter bandwidth: 180000
Filter order: 100
Squelch: 50
CW Shift: 600
Step size: 50 kHz
[✓] Snap to grid
[ ] Correct IQ
[ ] Swap I & Q
[ ] FM Stereo
[ ] Mark Peaks

Audio
AF Gain
Samplerate: 48000 sample/sec
Input: [ALSA] default
Output: [ALSA] default

S-Decay
W-Attack
W-Decay

RTL-SDR Controller
Device: Lifeview LV5TDeluxe — FC0013
Sample Rate: 2.048 MSPS
Sampling Mode: Quadrature sampling
[ ] Offset Tuning
[ ] RTL AGC
[✓] Tuner AGC
RF Gain
Frequency correction (ppm): −80
Close

e=-130


SDR# v0.0.0.0 - IQ Imbalance: Gain = 1,000 Phase = 0,000�

Stop | RTL-SDR / USB | Configure | VFO 0.092.400.000

Radio: WFM selected; Shift 80 000 000; Filter type Blackman-Harris; Filter bandwidth 180000; Filter order 100; Squelch 50; CW Shift 600; Step size 50 kHz; Snap to grid ✓; Correct IQ; Swap I & Q; FM Stereo; Mark Peaks

Audio: AF Gain; Samplerate 48000 sample/sec; Input [ALSA] default; Output [ALSA] default

RTL-SDR Controller
Device FC0013: Lifeview LV5TDeluxe
Sample Rate: 2.048 MSPS
Sampling Mode: Quadrature sampling
Offset Tuning
RTL AGC
Tuner AGC ✓
RF Gain
Frequency correction (ppm) −130
Close

Checking the frequency correction ”e” using RTLSDR-Scanner
http://eartoearoak.com/software/rtlsdr-scanner
An additional way to get a value for ”e” for your tuner is using RTLSDR-Scanner, that gives a at least *virtually* more accurate value for ”e” than GRC or SDR#.
Note! RTLSDR-Scanner must be executed from your downloaded directory RTLSDR-Scanner/src **Note!**

If not done initially: First install the dependency python-matplotlib-wx
```
# zypper in python-matplotlib-wx
The following NEW packages are going to be installed:
  python-dateutil python-matplotlib python-matplotlib-wx python-six python-tz
```

Get RTLSDR-Scanner from
git://github.com/EarToEarOak/RTLSDR-Scanner.git
```
$ cd /home/larse/bin/git
$ git clone git://github.com/EarToEarOak/RTLSDR-Scanner.git
$ cd RTLSDR-Scanner
```
Note! readme.md contains very useful information. Note!
```
$ cd src
```

Check that all dependencies are at hand:
Execute:
```
$ python rtlsdr_scan_diag.py
```

```
rtlsdr_scan_diag
Tests for missing libraries
pyrtlsdr not found
        Download from 'https://github.com/roger-/pyrtlsdr'
```

Now get and install pyrtlsdr from git://github.com/roger-/pyrtlsdr.git [not done before!]

```
$ cd ../..        [in my home it's just a way to get back to my home/bin/git library]
$ git clone git://github.com/roger-/pyrtlsdr.git
$ cd pyrtlsdr
```

Read readme.md

```
$ su
Password:
# python setup.py install
# exit
```

Navigate with the terminal back to your library RTLSDR-Scanner/src

```
$ cd RTLSDR-Scanner/src
```

Check again for any possible other dependencies:
Execute:

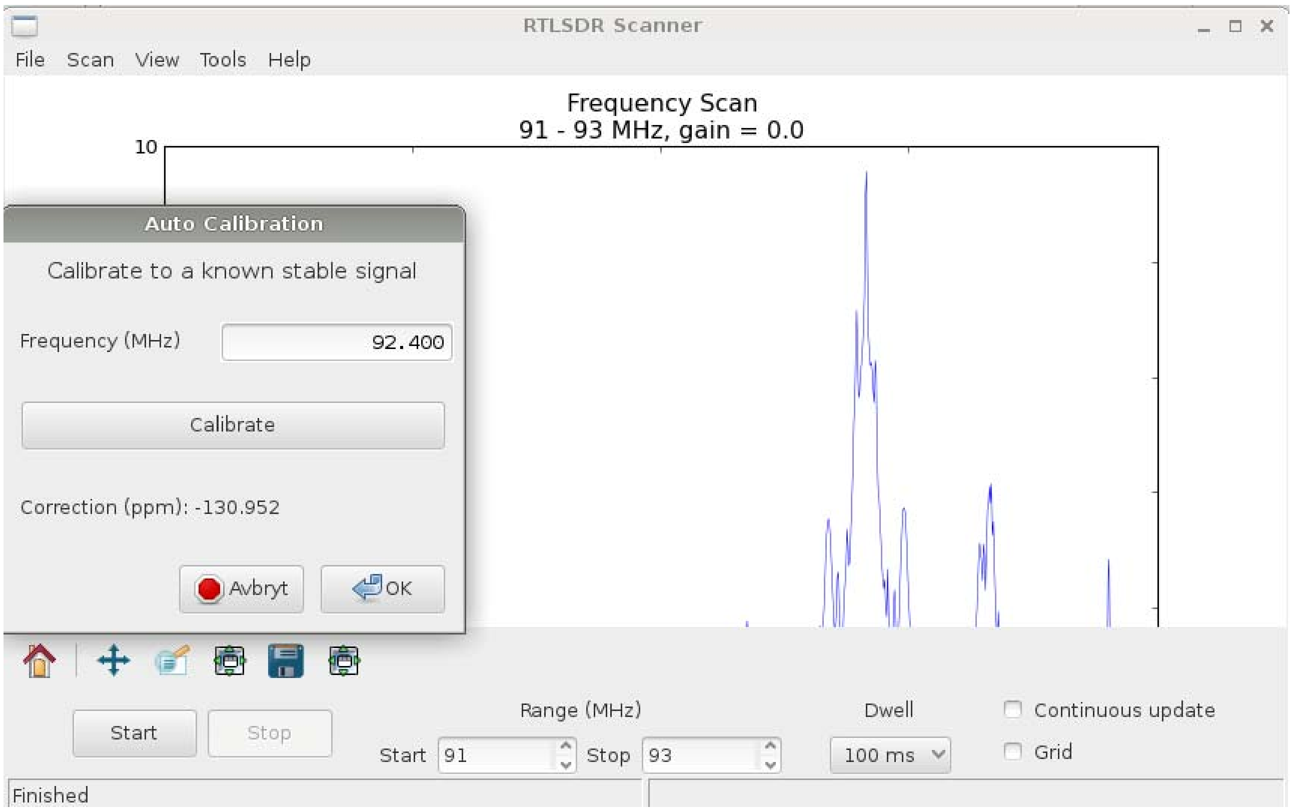
```
$ python rtlsdr_scan_diag.py
No problems found
```

Now execute RTLSDR-Scanner [*from it's library RTLSDR-Scanner/src]*

```
$ python rtlsdr_scan.py
```

The scanner now runs a basic scan of the frequency band 87-108 MHz
The band we are interested in are the AIS-frequencies (vessels)
Channel 87: 161.975 MHz (AIS, digital)
Channel 88: 162.025 MHz (AIS, digital)
and that the tuner is reasonably right set for those, or actually the mid-frequency 162.000 MHz. (Earlier with GRC and SDR# I haven't managed to get any closer than -e~-130)
In the tool bar of RTLSDR-Scanner->Tools: Auto calibration: ”calibrate against a known stable signal”. It's hard to set the AIS-frequencies as ”known stable signals”, there is a carrier but not much more when no one is using them.

If I for instance try 162.025 the result gets very confusing compared to what I have achieved with GRC and SDR# (e=1046.382 ppm and several peaks)?
-Therefore as with SDR# I choosed one of the frequencies in the Swedish FM-band: 92.400 MHz and executed from toolbar->Tools->Auto Calibration: Calibrate:



that gives Correction (ppm) e=-130.952 ppm
By repeated tests within the Swedish FM-band on ”known stable signal transmitters” it turns out that the e-values from RTLSDR-Scanner are not very stable.
This could indicate that my tuner is unstable, but the use of it with gr-ais and an error correction of -130 contradicts that. With as little as a deviation of 20-30 from correct e it shouldn't receive any signals, but it does in a stable way.
To check for a correct value for the frequency correction ”e” is as seen no simple task, nevertheless essential. If you're as little as 20-40 ppm wrong, the reception of AIS-messages will fail!

Summary of the frequency correction (and a manual way to fine tune it)
For me with the FC0013 tuner the baseline value, given in SDR#, the frequency correction is -130, that is the same as for my friend Håkan with his E4000-tuner. This is the baseline value for e I'll use in the gr-ais command:

```
$ ais_rx.py -d -e -130
linux; GNU C++ version 4.7.2 20130108 [gcc-4_7-branch revision 195012]; Boost_104900; UHD_003.005.002-56-g34052015

gr-osmosdr 871f0cc2 (0.0.1git) gnuradio 3.6.4.1
built-in device types: file fcd rtl rtl_tcp uhd
Using device #0 Realtek RTL2838UHIDIR SN: 00000001
Detached kernel driver
Found Fitipower FC0013 tuner
A: Sampling @ 256000.000000, decim @ 4
>>> gr_fir_ccc: using 3DNow!Ext
Using Volk machine: sse2_64_mmx_orc
>>> gr_fir_fff: using 3DNow!
>>> gr_fir_ccf: using 3DNow!
B: Sampling @ 256000.000000, decim @ 4
/home/larse/.gnuradio/prefs/gr_vmcircbuf_default_factory: No such file or directory
```

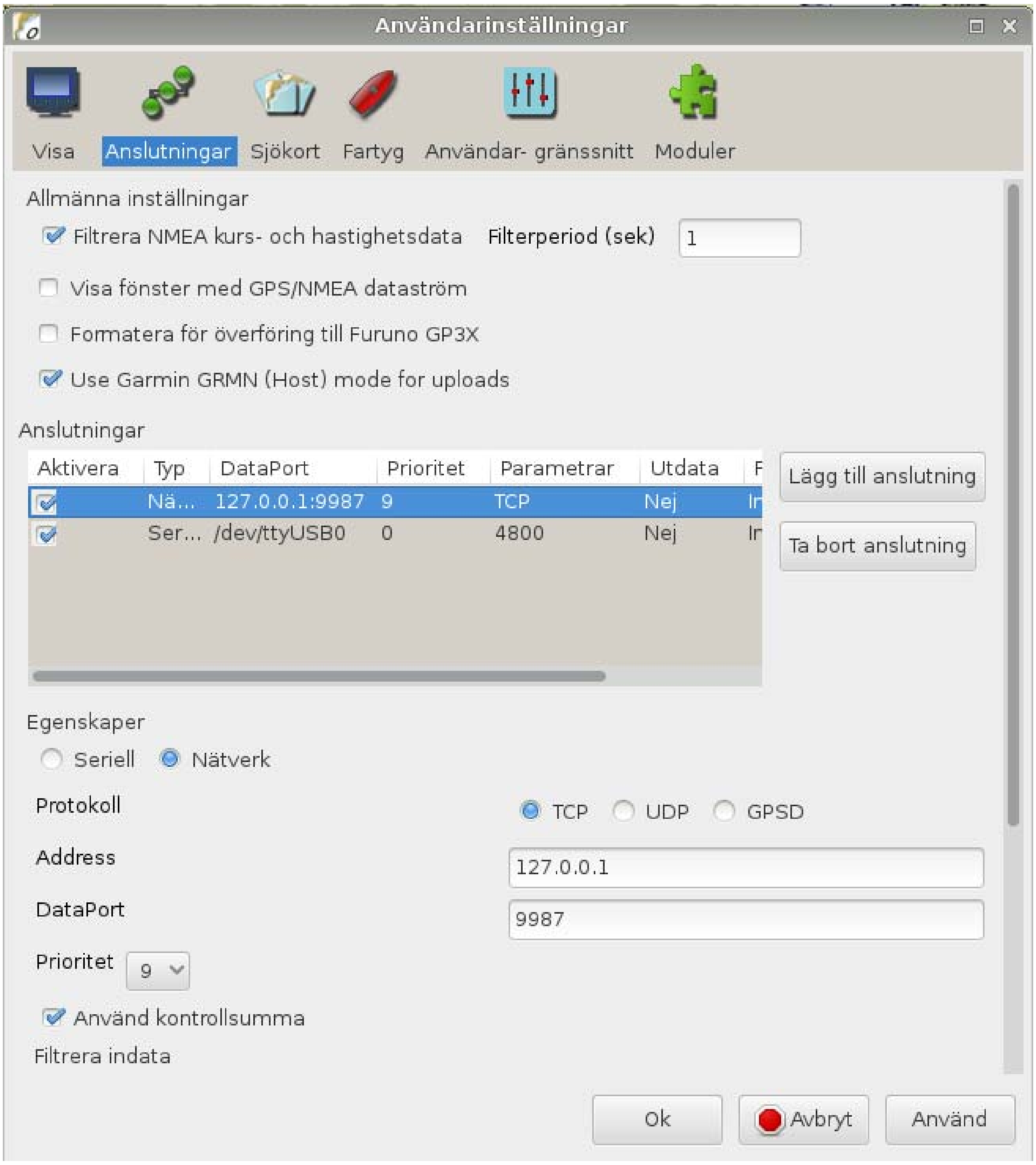
```
gr_vmcircbuf_createfilemapping: createfilemapping is not available
!AIVDM,1,1,,B,33u2AV5P001BUgnQtHqeN?wb0000,0*22
!AIVDM,1,1,,B,B3uDniP000Da;l8O6VL03wp5oP06,0*23
!AIVDM,1,1,,B,H3uDniTlDBE5847C:970000P;220,0*31
!AIVDM,1,1,,A,53u2AV01VPKu<TDb2210thHu9>2222222222221?5H;466se074sS@C0DTm4PC
Q8888888,0*59
```

You now can use your baseline value to start a fine tuning of your tuners error correction. It can though be somewhat time consuming but is worthwhile.
You simply start gr-ais and note the number of AIS-messages you get in a certain amount of time. This of course requires that you know there are any reasonably continous AIS-transmitters in your surroundings!
You take your baseline value, note the output, then from that value step upwards and downwards in steps of say 5 units and for each value of e you note the output.
Note! It can take up to a minute before the first message comes in the form of ”!AIVDM et c” and up to 5 minutes before you receive more. It takes lot of patience, but is as I said worthwhile!

Settings OpenCPN

A note concerning gpsd: During the installation of dependencies we included gpsd-devel:
Myself earlier in OpenCPN never succeeded to get my Garmin-GPS connect to OPENCPN in a stable way using gpsd. Therefore I removed gpsd and gave my GPS a fixed serial connection in OpenCPN (/dev/ttyUSB0). In a similar way I here gave the gr-ais' AIS-TCP-server a fixed network connection. Hopefully you've got better luck with gpsd!

Settings OpenCPN (without gpsd):
Options: Connections:
Add connection:
Properties: Network
Protocol: TCP
Address: 127.0.0.1
DataPort: 9987
Or as in the picture below (though in swedish):


Användarinställningar

Visa | Anslutningar | Sjökort | Fartyg | Användar- gränssnitt | Moduler

Allmänna inställningar

- [✓] Filtrera NMEA kurs- och hastighetsdata — Filterperiod (sek) 1
- [ ] Visa fönster med GPS/NMEA dataström
- [ ] Formatera för överföring till Furuno GP3X
- [✓] Use Garmin GRMN (Host) mode for uploads

Anslutningar

Aktivera	Typ	DataPort	Prioritet	Parametrar	Utdata	F
✓	Nä...	127.0.0.1:9987	9	TCP	Nej	Ir
✓	Ser...	/dev/ttyUSB0	0	4800	Nej	Ir

Lägg till anslutning | Ta bort anslutning

Egenskaper

Seriell [✓ Nätverk]

Protokoll: [✓ TCP] UDP GPSD

Address: 127.0.0.1

DataPort: 9987

Prioritet: 9

- [✓] Använd kontrollsumma

Filtrera indata

Ok | Avbryt | Använd

Under Ships Tab AIS Targets: Set the Display/Alerts et c you want.

With gr-ais I now send the AIS-NMEA-outdata over tcp (flag -t) to OpenCPN

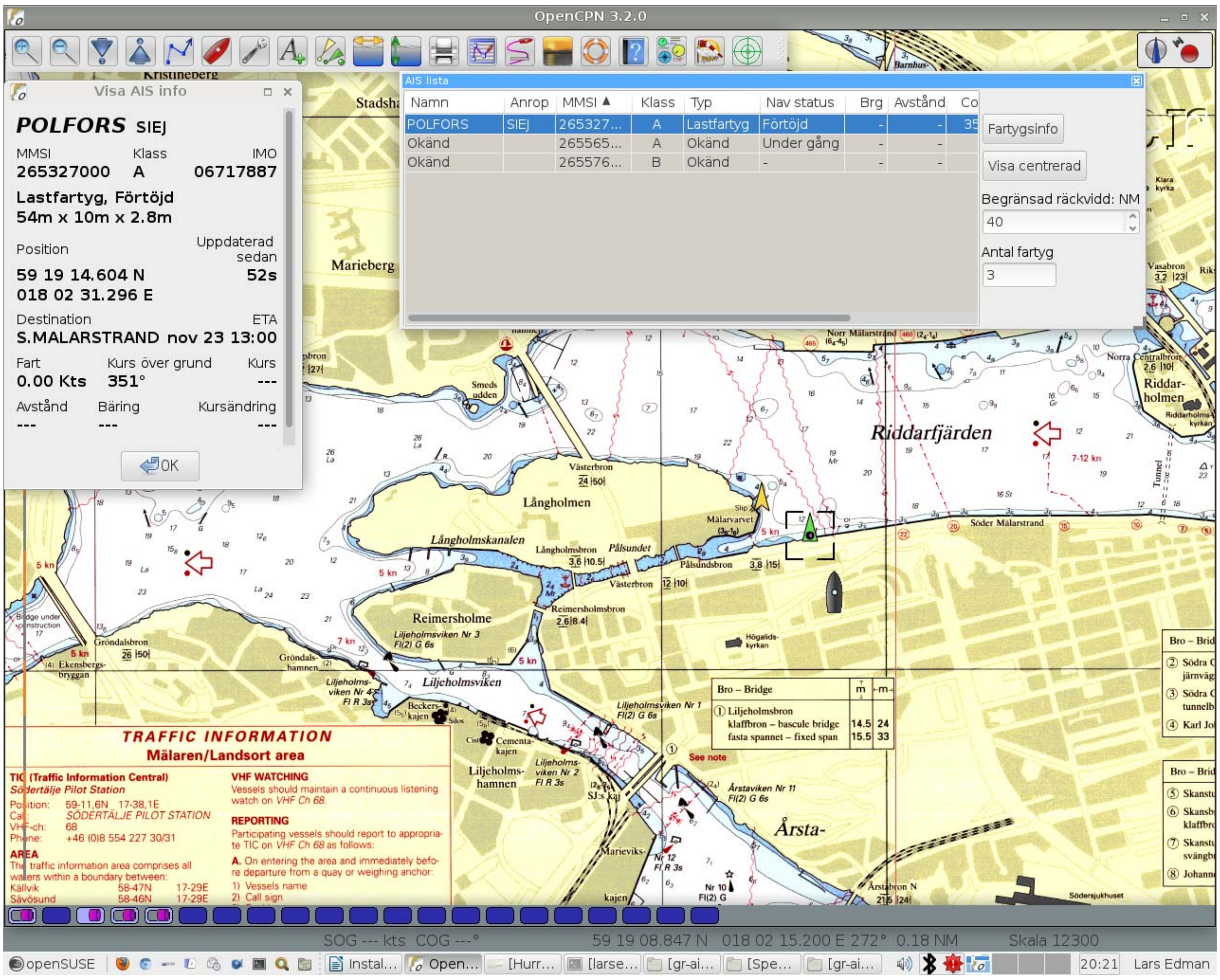
```
$ ais_rx.py -d -e -130 -t
linux; GNU C++ version 4.7.2 20130108 [gcc-4_7-branch revision 195012]; Boost_104900;
UHD_003.005.002-56-g34052015

gr-osmosdr 871f0cc2 (0.0.1git) gnuradio 3.6.4.1
built-in device types: file fcd rtl rtl_tcp uhd
Using device #0 Realtek RTL2838UHIDIR SN: 00000001
Detached kernel driver
Found Fitipower FC0013 tuner
```

```
A: Sampling @ 256000.000000, decim @ 4
>>> gr_fir_ccc: using 3DNow!Ext
Using Volk machine: sse2_64_mmx_orc
>>> gr_fir_fff: using 3DNow!
>>> gr_fir_ccf: using 3DNow!
B: Sampling @ 256000.000000, decim @ 4
Connections:  1
Connections:  2
Connections:  3
Connections:  4
Connections:  5
Connections:  6
Connections:  5
Connections:  4
Connections:  3
Connections:  2
Connections:  1
```

Ctrl(Right)Click in the the chart display in OpenCPN: AIS Target List
In the list that in due time comes up you can mark a line for a ship your interested in and press Ship/Vessel info.

Provided that there are som vessels transmitting ais in your surroundings you should now be able to see something in OpenCPN. Below two examples:
The ”black vessel” on shore is the GPS-position of my indoor antenna ;-)



Visa AIS info

POLFORS **SIEJ**

MMSI	Klass	IMO
265327000	**A**	**06717887**

Lastfartyg, Förtöjd
54m x 10m x 2.8m

Position	Uppdaterad sedan
59 19 14.604 N	**52s**
018 02 31.296 E	

Destination	ETA
S.MALARSTRAND	**nov 23 13:00**

Fart	Kurs över grund	Kurs
0.00 Kts	**351°**	**---**

Avstånd	Bäring	Kursändring
---	**---**	**---**

OK

AIS lista

Namn	Anrop	MMSI ▲	Klass	Typ	Nav status	Brg	Avstånd	Co
POLFORS	SIEJ	265327...	A	Lastfartyg	Förtöjd	-	-	35
Okänd		265565...	A	Okänd	Under gång	-	-	
Okänd		265576...	B	Okänd	-	-	-	

Fartygsinfo

Visa centrerad

Begränsad räckvidd: NM
40

Antal fartyg
3

TRAFFIC INFORMATION
Mälaren/Landsort area

TIC (Traffic Information Central)
Södertälje Pilot Station
Position: 59-11,6N 17-38,1E
Call: SÖDERTÄLJE PILOT STATION
VHF-ch: 68
Phone: +46 (0)8 554 227 30/31

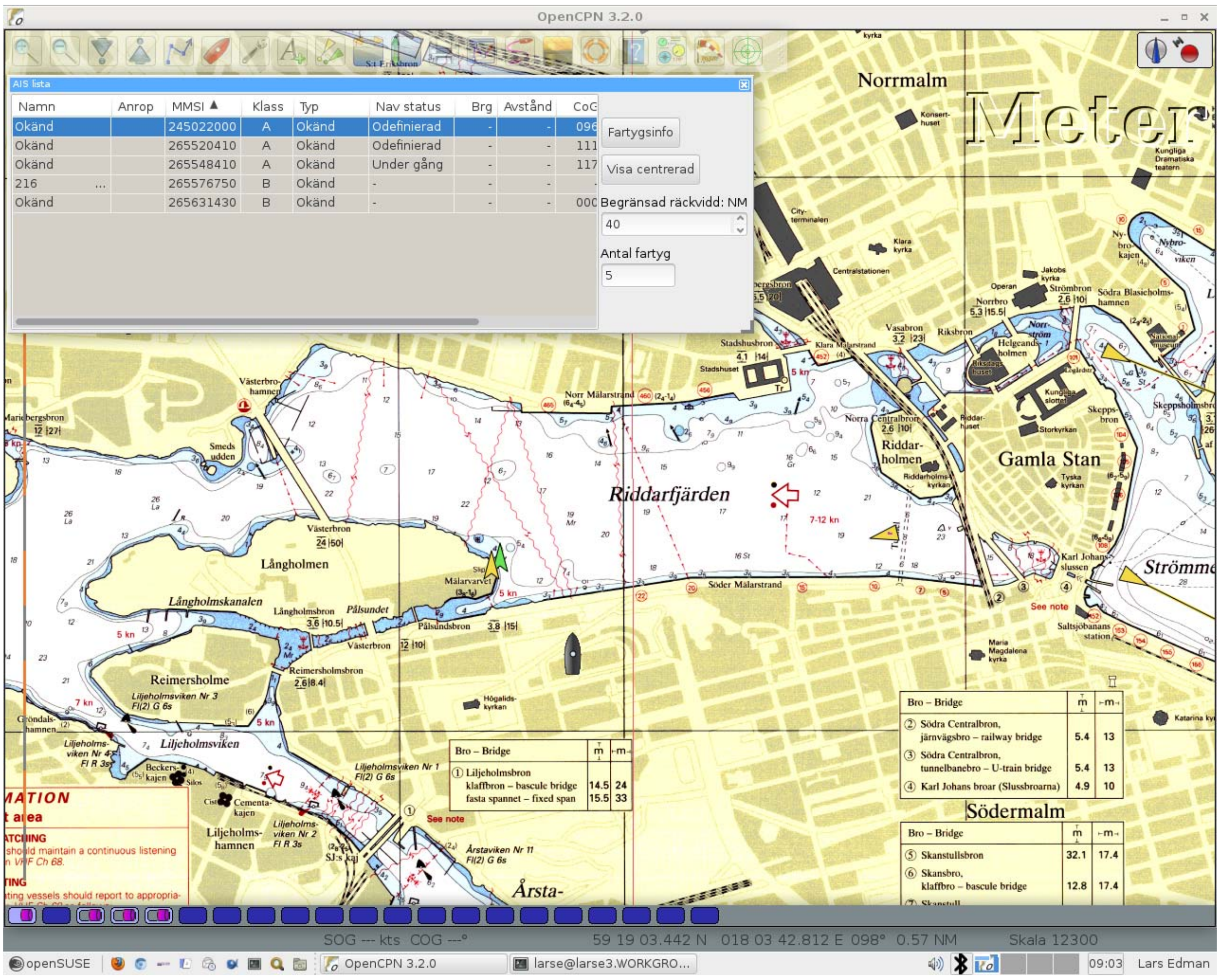
AREA
The traffic information area comprises all waters within a boundary between:
Källvik 58-47N 17-29E
Sävösund 58-46N 17-29E

VHF WATCHING
Vessels should maintain a continuous listening watch on VHF Ch 68.

REPORTING
Participating vessels should report to appropriate TIC on VHF Ch 68 as follows:
A. On entering the area and immediately before departure from a quay or weighing anchor:
1) Vessels name
2) Call sign

SOG --- kts COG ---° 59 19 08.847 N 018 02 15.200 E 272° 0.18 NM Skala 12300

openSUSE | Instal... | Open... | [Hurr... | [larse... | [gr-ai... | [Spe... | [gr-ai... | 20:21 | Lars Edman



Namn	Anrop	MMSI ▲	Klass	Typ	Nav status	Brg	Avstånd	CoG
Okänd		245022000	A	Okänd	Odefinierad	-	-	096
Okänd		265520410	A	Okänd	Odefinierad	-	-	111
Okänd		265548410	A	Okänd	Under gång	-	-	117
216 ...		265576750	B	Okänd	-	-	-	-
Okänd		265631430	B	Okänd	-	-	-	000

Begränsad räckvidd: NM 40

Antal fartyg 5

SOG --- kts COG ---° 59 19 03.442 N 018 03 42.812 E 098° 0.57 NM Skala 12300

As you can see my little indoor antenna can receive signals from transmitters quite far away.

Systemdemands
gr-ais demands quite a lot of your system. In the example below I installed it on a reasonably old AMD64 machine with a single core AMD3000+ processor with nVidia bridges:

```
$ ps aux
USER   PID   CPU  MEM    VSZ      RSS    TTY    STAT START   TIME   COMMAND
myself  10389 41.2  4.5     1274176 92852 pts/0   Sl+     19:49      11:08   /usr/bin/pytho
```

The work that at last led to a working installation:
A ”joint venture” by Håkan Nilsson, Lidingö and Lars Edman, Stockholm.

Responsible for this description (including all possible errors and lacking information):
2013-05-03
Lars Edman
Stockholm
Sweden