SMOGCLI2 TELEPÍTÉSE LINUX ALATT

A smogcli2 csomag hatékony parancssori eszközöket tartalmaz a SMOG-P és ATL-1 műholdak által sugárzott rádiójelek vételére, alapsávi adatok rögzítésére, demodulálás/dekódolás, az dekódolt csomagok feltöltésére. A programcsomag több platformon is lefordítható, de jelen írás csak a LINUX operációs rendszer alatti munkával foglalkozik

A telepítés menete terminálban:

1) sudo apt install librtlsdr-dev libcurl4-gnutls-dev cmake git libfftw3-dev python python3
2) git clone https://gitlab.com/phorvath/smogcli2.git
3) cd smogcli2
4) mkdir build
5) cd build
6) cmake .. (space után két pont)
   vagy Rpi esetén:
   cmake -DCMAKE_TOOLCHAIN_FILE=cmake/arm_cortex_a53-native.cmake ..
7) make
8) sudo make install
9) cp ../scripts/smog_upload.py ./

Ezzel a telepítés megvan!

A csomag frissítése:

1) cd ~/smogcli2 (A ~ a home direktori, ahol a smogcli2 található)
2) git pull
3) cd ./build
4) make
5) sudo make install

Beállítások:

1) Ha a vett csomagokat fel akarjuk tölteni a GND szerverre, akkor regisztrálni kell a https://gnd.bme.hu:8080/auth/register oldalon.
2) Lépjünk be a smogcli2 könyvtárba

3) A qthfile.txt fájlt értelemszerűen módosítsuk. A magasság alá kell beírni a gnd-n regisztrált felhasználónevet, alá a jelszót, majd a qthfile.txt-t átmásolni a build könyvtárba. (cp qthfile.txt ./build)
4) A vevőprogram indítása a build könyvtárból tehető meg a következő paranccsal: ./smog_rtl_rx -g 50 -k (A kapcsolók jelentéséről a -h kapcsolóval tudhatunk meg többet)
5) A műhold áthaladása után a smogcli2/build könyvtárban megjelenik egy a műhold nevével kezdődő *.cf32 kiterjesztésű fájl.
6) Dekódolás: Ugyancsak a build könyvtárban található ./smog_decode -s -2 -t -2 -d -2 -p -v fájlnév.cf32 paranccsal végezhető. Itt a fájlnév a 5. pontban említett fájlnév (ha minden cf32 fájlt dekódolni akarunk, akkor *.cf32).
7) Csomagok feltöltése: Szintén a build könyvtárban található python programmal tehető meg: python smog_upload.py fájlnév.pkts (ha minden csomagot fel akarunk tölteni, akkor *.pkts)

A *.pkts kiterjesztésű fájlok a dekódoláskor jönnek létre a *.cf32-es mellett, azonos névvel, csak más kiterjesztéssel.
Előfordulhat, hogy egy-egy csomag nem megy fel elsőre, ilyenkor meg kell ismételni a műveletet és többnyire maximum 3 próbálkozás után feltöltődnek. Ha nem, akkor az a csomag sérült, amit a „bad checksum” üzenet jelez.

Sikeres munkát mindenkinek!

Melléklet: A programcsomaghoz tartozó readme fájl

1. # Build guide

Linux
Install the `rtl-sdr` development packages, and `curl` development packages (on Ubuntu,
`apt install librtlsdr-dev libcurl4-gnutls-dev cmake git libfftw3-dev`). Use `cmake` to build:
<pre><code>git clone https://gitlab.com/phorvath/smogcli2.git
cd smogcli2
mkdir build
cd build
cmake ..
make
sudo make install
</code></pre>
Run the `smog_tests` binary to see some performance stats.

Raspberry Pi 3 or newer:
Install the `rtl-sdr` development packages, and `curl` development packages (on Raspbian,
`apt install librtlsdr-dev libcurl4-gnutls-dev cmake git libfftw3-dev`).

Use `cmake` to build:
<pre><code>git clone https://gitlab.com/phorvath/smogcli2.git
cd smogcli2
mkdir build
cd build
cmake -DCMAKE_TOOLCHAIN_FILE=cmake/arm_cortex_a53-native.cmake ..
make
sudo make install
</code></pre>

Run the `smog_tests` binary to see some performance stats.
<pre><code>pi@raspberrypi:~/smogcli2/build $ ./smog_tests
buffer_t: test passed
convert_u8_f32: test passed
convert_u8_f32: 116.536 msps

filter_fir_f32: test passed
filter_fir_f32: taps16 dec2 vec2 speed 2.78684 msps
filter_fir_vec2_f32: test passed
filter_fir_vec2_f32: taps16 dec2 neon 24.0684 msps
filter_fir_tap16_dec2_vec2_f32: test passed
filter_fir_tap16_dec2_vec2_f32: neon 70.3193 msps
sinusoid_source_cf32: test passed
sinusoid_source_cf32: 106.66 msps
fft_cf32: test passed
fft_cf32: win2048 str512 speed 5.63611 msps
norm_cf32: 168.986 msps
</code></pre>

On a Pi3+
<pre><code>pi@raspberrypi:~/smogcli2/build $ ./smog_tests
buffer_t: test passed
convert_u8_f32: test passed
convert_u8_f32: 104.792 msps
filter_fir_f32: test passed
filter_fir_f32: taps16 dec2 vec2 speed 3.35106 msps
filter_fir_vec2_f32: test passed
filter_fir_vec2_f32: taps16 dec2 neon 27.5301 msps
filter_fir_interp_vec2_f32: test passed
filter_fir_interp_vec2_f32: taps16 interp2 4.71385 msps
filter_fir_tap16_dec2_vec2_f32: test passed
filter_fir_tap16_dec2_vec2_f32: neon 73.7292 msps
sinusoid_source_cf32: test passed
sinusoid_source_cf32: 107.59 msps
fft_cf32: test passed
fft_cf32: win2048 str512 speed 6.40181 msps
norm_cf32: 146.421 msps
</code></pre>

WARNING: Older Pi hardware is probably not powerful enough to record two channels simultaneously.

Usage

Specify your location

Place qthfile.txt with your geographical location into the same directory. A sample qthfile.txt is included for reference. qthfile.txt should contain
- An identifier (e.g., your callsign) - this will appear verbatim at the end of the file name in every recording
- Longitude in decimal format
- Latitude in decimal format (E: negative value, W: positive value)
- Altitude in meters

NOTE: If you would like to use the packet upload scripts, add two more lines at the end, containing your username and password obtained from [link] (https://gnd.bme.hu:8080/index)

Provide TLE data

The program tries to read `tle.txt`, containing 2-line Kepler elements, in the same directory. With the `-k` switch, the program will try to download the most recent TLE file `cubesat.txt` and store it as `tle.txt` in the dirrent directory. If the switch `-a` is also given, the complete TLE database `active.txt` is downloaded instead. (Beware, it is over 400KB.)

Record

<pre><code>pi@raspberrypi:~/smogcli2/build $ ./smog_rtl_rx -k -g 45 -p
INFO: trying to download TLE from celestrak.com...
INFO: TLE (active.txt) downloaded successfully
INFO: TLE and QTH files loaded
INFO: found TLE data for SMOG-P (44832)
INFO: found TLE data for ATL-1 (44830)
INFO: TLE age is 0 days
INFO: found 1 device(s):
0: Realtek RTL2838UHIDIR, SN: 00000001
INFO: using device 0: Generic RTL2832U OEM
Found Rafael Micro R820T tuner
Exact sample rate is: 2000000.052982 Hz
[R82XX] PLL not locked!
INFO: setting dongle sample rate to 2000000

INFO: setting fixed gain 445
INFO: disabled bias-T on GPIO PIN 0
INFO: setting dongle center frequency to 437.095 MHz
INFO: setting dongle ppm correction to -1
INFO: real center frequency is 437095000.0 Hz
INFO: starting dongle thread with 1048576 bytes of buffer
Allocating 15 zero-copy buffers
INFO: SMOG-P azimuth: 2.17108, elevation: -40.1313, doppler: 8058.68 Hz
INFO: ATL-1 azimuth: 359.268, elevation: -50.4449, doppler: 6680.11 Hz
INFO: maximum signal amplitude: 0.25
</code></pre>

The program will start writing a binary IQ file whenever one of the satellites is above -2 degrees elevation. The file name is automagically generated from the name of the satellite, the absolute time and the QTH name. When the satellite passes, the file will be closed and the program waits for a new pass. The default mode of operation assumes a sample rate of 1.6 Msps, which will be decimated to yield 50 ksps at the output. The received passband is flat up to 18 kHz and is steeply cut above that frequency. If the switch `-S` is also given, the sample rate will be increased to 2 Msps to obtain an output sample rate of 62.5 ksps to be compatible with the `gorgon` demodulator. The output format is FC32 (i.e., interleaved 32-bit float IQ values). [Inspectrum](https://github.com/miek/inspectrum) is highly recommended to inspect the contents of the output files.

Other options are self-explanatory. The dongle clock correction can be **fractional**. You can use [kalibrate-rtl](https://github.com/steve-m/kalibrate-rtl) to obtain a calibration value for your dongle. (And better buy a TCXO model. TCXO dongles will probably not need any calibration or correction.)

You can specify any other satellite with its corresponding downlink frequency (useful for testing purposes). The program is also able to write WAV files if you want to replay them i.e., in SDRSharp (or whatever its current name is).

By default, the dongle will operate in automatic gain control mode. The AGC of the rtl-sdr is meant to handle wideband signals, therefore it might act weirdly on weak narrowband signals. Experience shows that, in the absence of signals, auto gain will yield a somewhat higher gain setting than the maximum gain one can set manually. You can specify a fixed manual gain and probably you should do so.

You might leave the program running ad infinitum. I'd suggest running it from `tmux` or `screen` because the program keeps spitting out stuff to the stdout. Ctrl-C terminates the program. The decoder expects 50000 sps recordings.

<pre><code>
SMOG-P/ATL-1/SMOG-1 recorder for RTL2832 based DVB-T receivers

Usage: smog_rtl_rx [-options]
	-d device_index (default: 0)
	-T enable bias-T on GPIO PIN 0 (works for rtl-sdr.com v3 dongles)
	-g tuner gain (default: automatic, NOT RECOMMENDED)
	-p kalibrate-sdr reported fractional ppm error (default: 0.0)
	-i track the given primary satellite ID (default: 44832)
	-F downlink frequency for primary sat (default: 437150000.0 Hz)
	-j track the give satellite ID as secondary sat (default: 44830)
	-G downlink frequency for secondary sat (default: 437174500.0 Hz)
	-k download TLE data from celestrak.com
	-S use 2 Msps/62.5 ksps mode (default: 1.6 Msps/50 ksps)
	-O dump 8x downconverted samples to STDOUT in binary
	-b disable the 1/128 rescaling of raw samples
	-a download active.txt instead of cubesat.txt from celestrak.com
	-h prints this help message
</code></pre>

The option -O enables an independent downconverter chain, and dumps the downconverted samples to the standard output. This feature might be used to pipe received samples to OpenWebRX for visualization. The center frequency is in the middle between the nominal frequencies of SMOG-P and ATL-1 (437.1625 MHz), the sample rate is 200 ksps, and the pass pand is flat to +/- 75 kHz. It is therefore recommended to clip the frequency plots at +/- 75 kHz from the center frequency.

Decode

`smog_decode` can be used to search for telemetry packets in the recording and decode them.

<pre><code>
pi@raspberrypi:~/smogcli2/build $ smog_decode -h
SMOG-P/ATL-1/SMOG-1 demodulator and decoder

Usage: smog_decode [-options] [filenames]
-T runs internal test
-b bits per second (default: 1250, 2500, 5000 and 12500 bps)
-r sampling rate of input (default: 50000 sps)
-t tone detection sensitivity in dB (default: 2.0 dB)
-s sync detection sensitivity in dB (default: 1.0 dB)
-d data detection sensitivity in dB (default: 0.0 dB)
-v enable viterbi demodulator (default: false)
-g print raw 650 byte packets for gorgon (default: false)
-p write packet files instead of stdout (default: false)
</code></pre>

Usage is straightforward. Supply the recordings as arguments (wildcards are accepted so that multiple files can be handled in a single pass). It will try to recognize the file format based on its extension (cf32 or cs16). By default, the program will search for every legitimate bit rate, but one can selectively enable just a selected one by the switch -b. The argument -v will use a more sophisticated demodulation algorithm. Normally, the program will print out the decoded packets to the screen. The switch -g will change the output format to correspond to that of the "gorgon" demodulator, i.e., the output of smog_decode can be directly piped to a single running gorgon instance (smogpgnd or atlgnd). The switch -p will write the packets to a file with identical name and extension .pkts. These files can be uploaded to the server using the Python3 script smog_upload.py.

Most users will want to do something like
<pre><code>
pi@raspberrypi:~/smogcli2/build $ smog_decode -p -v SMOG-P_16668*.cf32
pi@raspberrypi:~/smogcli2/build $ python3 ../scripts/smog_upload.py SMOG-P_16668*.pkts
</code></pre>

Other tools

`smog_fc2wav`

This tool converts .cf32-format recordings to 16-bit WAV files, suitable e.g. for SDRSharp.