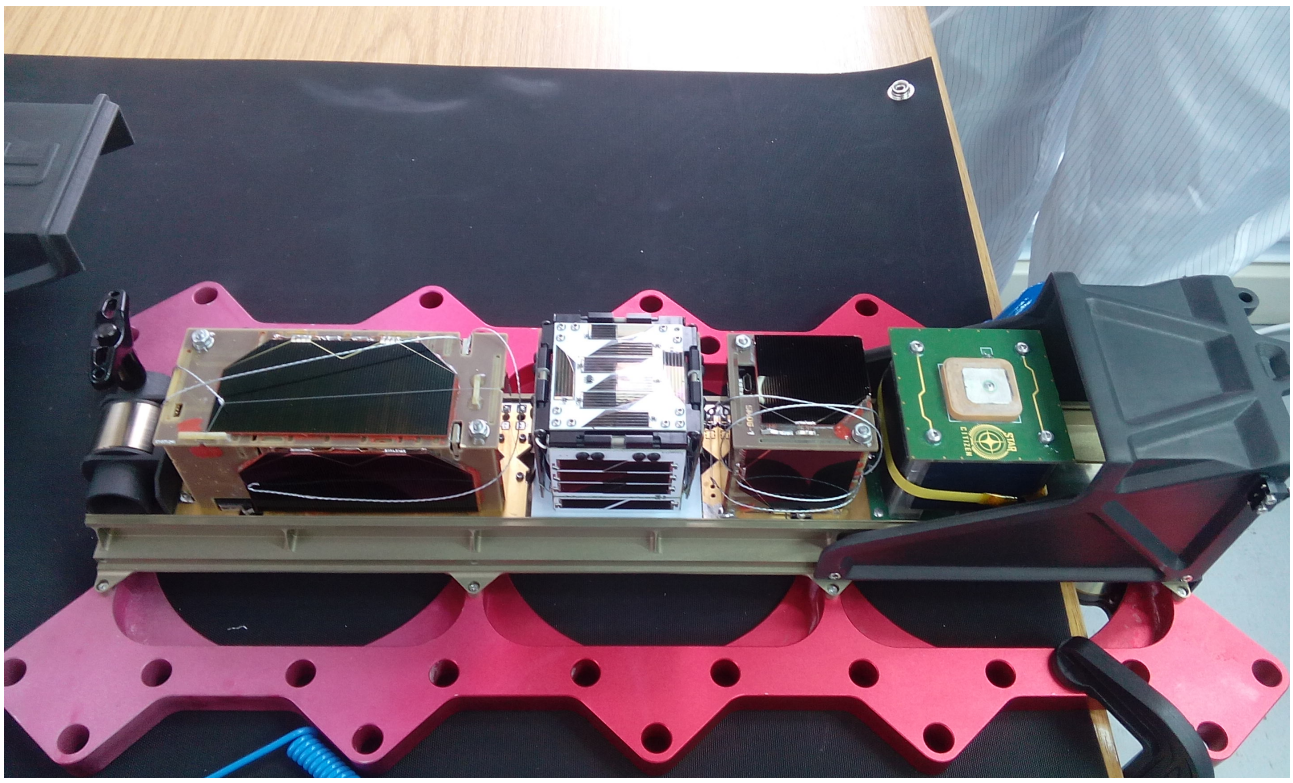

SMOG-P and ATL-1 Reception

Levente Dudás PhD

Budapest University of Technology and Economics
Dept. of Broadband Infocommunications and Electromagnetic Theory
Microwave Remote Sensing Laboratory

03.01.2020.

Contents

1 On-board COM **3**

2 Modulation **4**

3 Packet Length **4**

4 Minimal Configuration of Satellite Reception **4**
4.1 RTL-SDR R820T2 with TCXO 5
4.2 Satellite Tracking - Doppler Correction 6
4.3 Hand-Held Satellite Tracking 7
4.4 The Received Packets 10
4.4.1 SMOG-P 10
4.4.2 ATL-1 11

5 Record I-Q Signal **11**

6 WebSDR Based Satellite Reception **12**

7 Demodulate and Decode Packets with GORGON **12**

8 Appendix **19**
8.1 Doppler Corrector 19
8.2 Throttle 20
8.3 Complex Numerical Controlled Oscillator 21
8.4 Averaging as Low-Pass-Filtering 22
8.5 Decimating Filter 23
8.6 MSK Demodulator 23

1 On-board COM

The communication sub-system of SMOG-P and ATL-1 can be seen in Fig. 1.

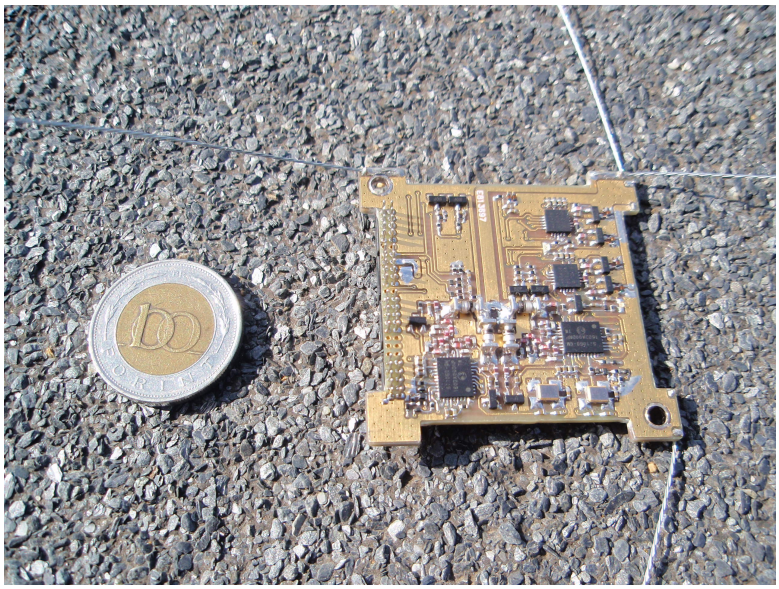

Figure 1: On-board Communication System

COM PCB contains two cold-redundant telemetry (TM) transmitters and telecommand (TC) receivers (transceivers) and two independent wide-band-used receivers (only for spectrum monitoring). The block scheme of the TC-TM transceiver can be seen in Fig. 2.

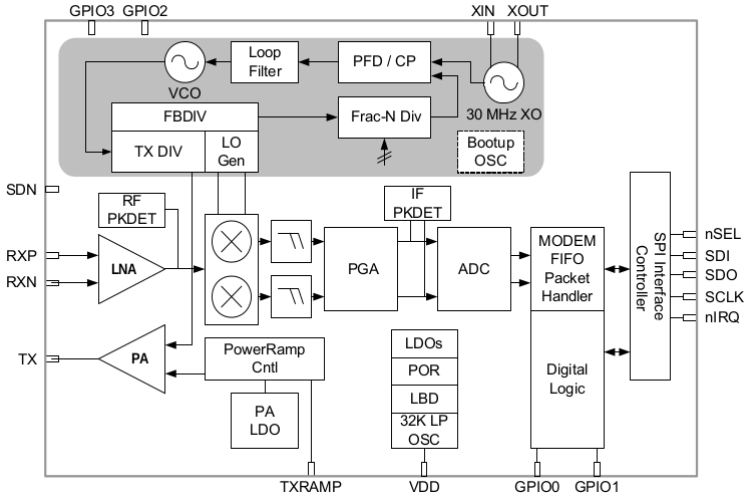

Figure 2: Block scheme of TC-TM Radio TCVR

The transceiver chip is Silabs type SI1060 [1] C8051F930 micro-controller and SI4463 UHF OOK/FSK TX-RX.

Table 1. shows the used transmit power levels versus the total power consumption of COM (include micro-controller, transmitter and RF PIN diodes current).

RF TX Power Level [dBm]	RF TX PWR [mW]	Total Power Consumption [mW]
20	100	410
17	50	297
14	25	231
10	10	165

Table 1: Transmit Power Level versus Total Power Consumption from nominal $3.3V$

2 Modulation

SMOG-P and ATL-1 use On-Off-Keying (Morse code) and Gaussian Minimal Shift Keying (GMSK) digital modulation with $BT = 0.5$ Gaussian baseband filter to be able to reach sub-optimal spectral efficiency.

The used G-MSK data rates are in Table 2.

Data Rate $[\frac{bit}{s}]$	Bandwidth [Hz]
1250	1875
2500	3750
5000	7500
12500	18750

Table 2: The used data rates and its bandwidth

According to the frequency coordination, SMOG-P is running on 437.150 MHz, ATL-1 on 437.175 MHz +/− Doppler-shift, the down-link bandwidth limit is 20 kHz.

3 Packet Length

SMOG-P and ATL-1 use AO-40 long, AO-40 short and Repeat Accumulate FEC encoders.

Table 3. contains the packet lengths and the coding schemes.

Coding Scheme	Radio Packet Length [B]	Raw Data Length [B]
AO-40 short	333	128
AO-40 long	650	256
RA	260	128
RA	514	256
TX-RX SYNC	70	6

Table 3: Coding schemes and packet lengths

4 Minimal Configuration of Satellite Reception

At least 6-element (hand-held) Yagi antenna is recommended to be able to receive SMOG-P or ATL-1 satellite - Fig. 3.

Table 4. contains the length and position of antenna elements: antenna elements are made of 13 mm wide (space-qualified - Masat-1) measuring tape and antenna boom is made of measuring rod.

RTL-SDR is connected directly to the dipole element of the Yagi antennna: shielding and center point of the SMA connector must be connected to the two half of the dipole.

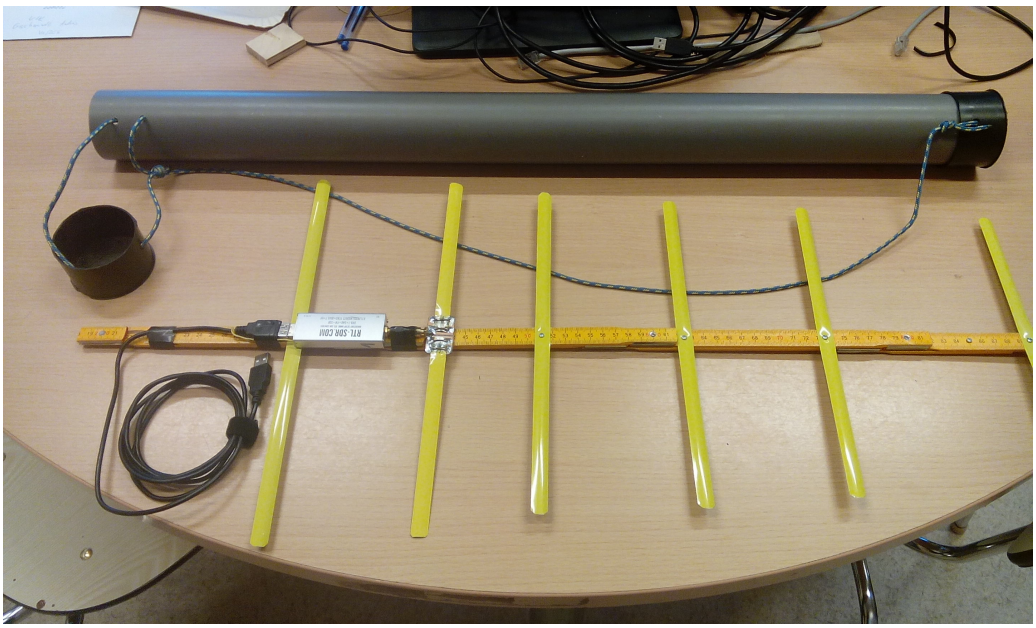

Figure 3: Block scheme of TC-TM Radio TCVR

Type	Length [mm]	Position [mm]
Director 4	260	25
Director 3	275	185
Director 2	290	297
Director 1	300	410
Dipole	323	490
Reflector	342	606

Table 4: 6-element Yagi antenna

4.1 RTL-SDR R820T2 with TCXO

You can use RTL-SDR [2] to receive SMOG-P and ATL-1 - Fig. 4.

Figure 4: RTL-SDR 4 / 5 / 6

4.2 Satellite Tracking - Doppler Correction

The satellite tracker and WEB based receiver application can be found here (Linux):

`http://152.66.80.46/smog1/receivesmogpatl1/`

Satellite tracker is modified *predict-2.2.3* for satellite tracking - thanks to KD2DB John.

The receiver program is *OpenWebRx* [3, 4] - thanks to HA7ILM András.

After download, unzip the *openwebrx.zip*.

Go into folder *openwebrx*, you can find *install.sh* bash script: **bash install.sh**.

```
# install RTL-SDR
sudo apt install rtl-sdr

# Simonyiszk CSDR compile: HA7ILM
# git clone https://github.com/simonyiszk/openwebrx.git
# git clone https://github.com/simonyiszk/csdr.git
cd csdr
make
sudo make install
sudo ldconfig
cd ..

# Automatic Doppler correction: HA7WEN
bash build.sh

# Predict-2.2.3: TNX John, KD2BD, kd2bd@amsat.org, May 2006
# This Predict was modified anno for Masat-1 tracking, now for SMOG-P and ATL-1
sudo apt install libncurses5-dev
cd predict-2.2.3
bash build2

# QTH file !!!
# edit ha7wen.qth (longitude is negative!!!)

# download the newest TLE-s
bash update
```

../openwebrx/install.sh

The *install.sh* download rtl-sdr library, install *csdr*, compile automatic Doppler corrector program, build *predict-2.2.3* and update Keplerian data of the satellites from the Internet.

Once a day you should run **bash predict-2.2.3/update**.

Plug RTL-SDR to computer USB port, and run the program in terminal (in openwebrx folder) use: **bash run.sh true** to receive ATL-1 and **bash run.sh false** to receive SMOG-P satellite.

```
# HA7WEN: SMOG-P and ATL-1 satellite tracking
# TNX: HA7ILM OpenWebRX, KD2BD Predict

# Predict: select satellite!
cd predict-2.2.3
bash run $1
# true: ATL-1 tracking
# false: SMOG-P tracking

# path.txt is the next path
# masat1file.txt contains: azimuth, elevation, downlink and uplink freq and
    enable signal (over -10 deg elev)

cd ..
```

```
# modify config_webrx.py:
# - waterfall center and RTL-SDR center freq: center_freq
# - only RTL-SDR RX or
# - RTL-SDR RX + IQ recording or
# - replay IQ data

# angle and freq information
bash wat.sh &

# run openwebrx
python openwebrx.py

# Do not forget to open in web browser: "localhost:8073"
```

../openwebrx/run.sh

Do not forget to open `localhost:8073` in WEB browser after *run.sh* - Fig. 5.

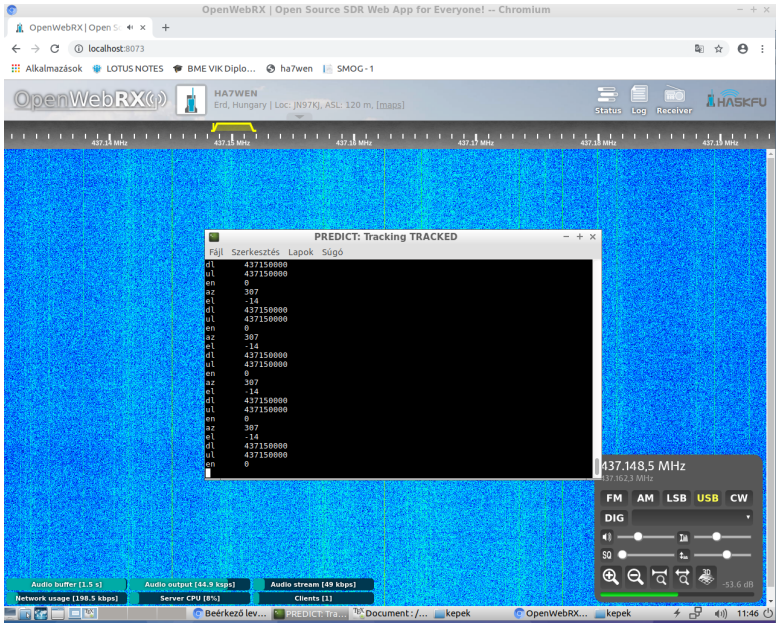

Figure 5: Waterfall diagram with tracking data

In Fig. 5, you can see the spectrum in time of the received signal and the satellite tracking data: downlink and uplink frequencies, azimuth and elevation angles.

4.3 Hand-Held Satellite Tracking

Manually I could rotate the antenna to the right direction of SMOG-P both in azimuth, elevation and polarization using my neurotic-network. - Fig. 6.

The received signal spectrum of SMOG-P can be seen in Fig. 7.

After SMOG-P I tracked ATL-1: Fig. 8.

The demodulator software of SMOG-P and ATL-1 is available here: `https://gnd.bme.hu:8080/` - see Fig. 9.

Figure 6: Satellite tracking with bio-mechanical antenna rotator: 03.01.2020

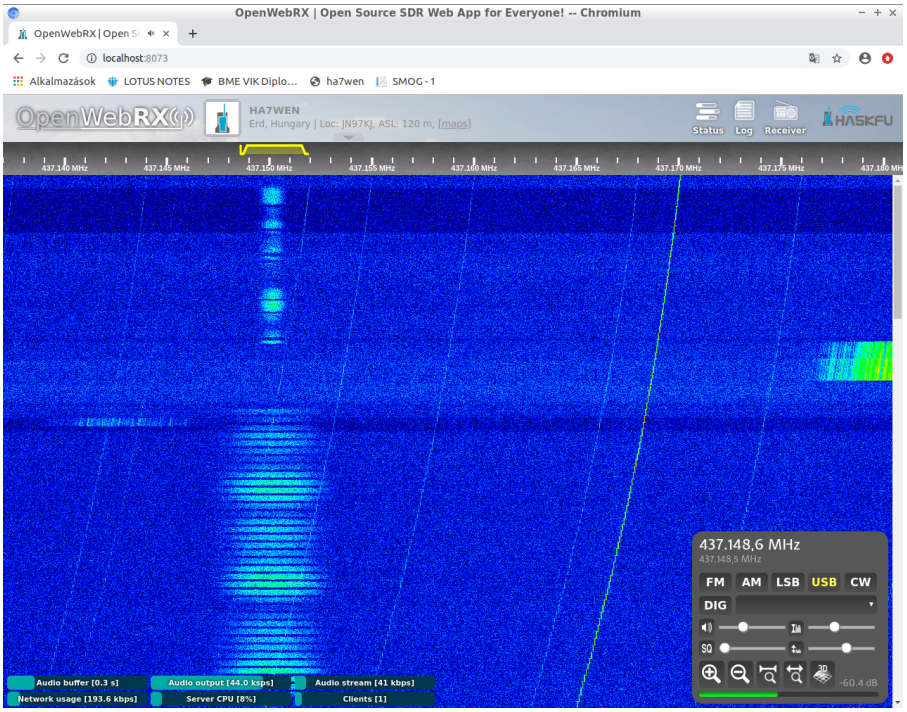

Figure 7: SMOG-P signal

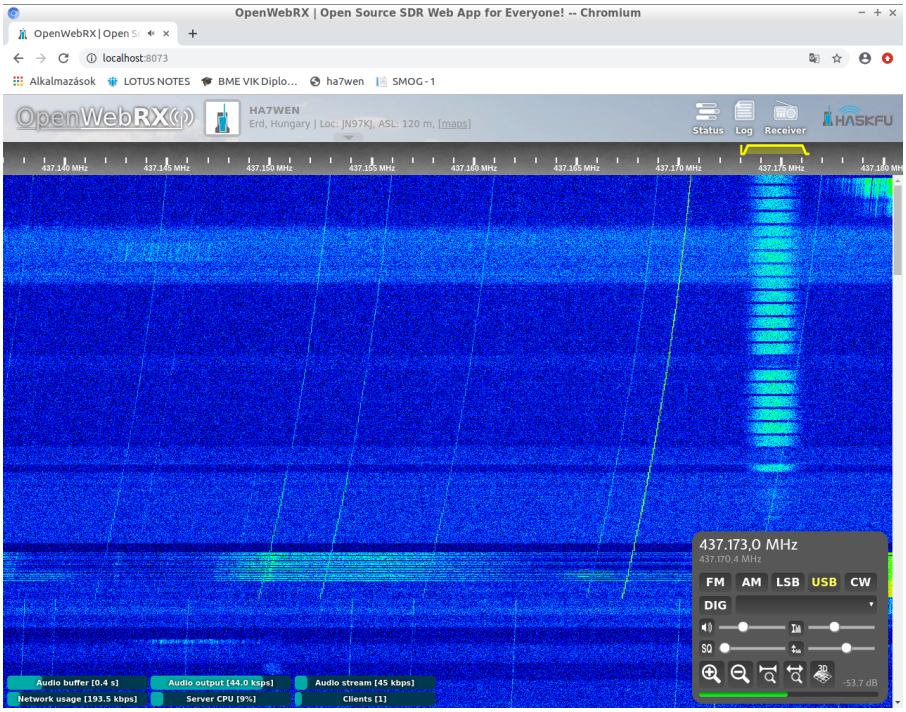

Figure 8: ATL-1 signal

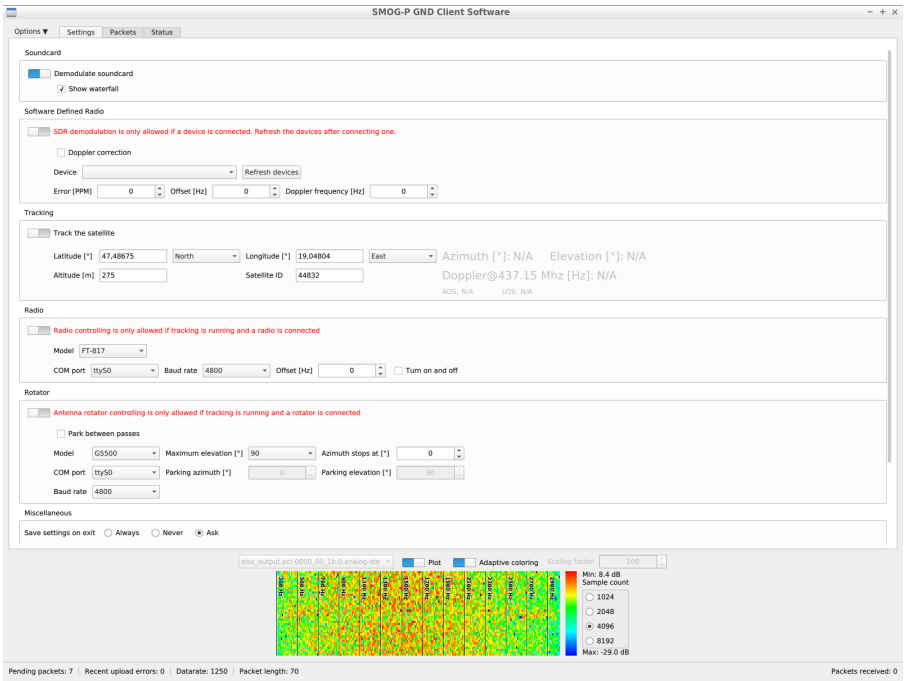

Figure 9: GND-SW GUI with sound card demodulator

4.4 The Received Packets

4.4.1 SMOG-P

```
Received 256b long packet with -132 RSSI; AUTHENTICATION: OK
  Packet type: File Fragment
  Timestamp: 2020-01-01 19:12:12
  Downloading 18/21 fragment of the file "SSA_0025" with type=5, size=4383 at
    page 384. Timestamp: 2019-12-20 08:15:31

Received 256b long packet with -132 RSSI; AUTHENTICATION: OK
  Packet type: File Fragment
  Timestamp: 2020-01-01 19:12:25
  Downloading 8/21 fragment of the file "SSA_0026" with type=5, size=4383 at
    page 512. Timestamp: 2019-12-20 08:16:04

Received 256b long packet with -132 RSSI; AUTHENTICATION: OK
  Packet type: File Fragment
  Timestamp: 2020-01-01 19:12:26
  Downloading 9/21 fragment of the file "SSA_0026" with type=5, size=4383 at
    page 512. Timestamp: 2019-12-20 08:16:04

Received 256b long packet with -132 RSSI; AUTHENTICATION: OK
  Packet type: File Fragment
  Timestamp: 2020-01-01 19:12:39
  Downloading 19/21 fragment of the file "SSA_0026" with type=5, size=4383 at
    page 512. Timestamp: 2019-12-20 08:16:04

Received 256b long packet with -132 RSSI; AUTHENTICATION: OK
  Packet type: File Fragment
  Timestamp: 2020-01-01 19:12:40
  Downloading 20/21 fragment of the file "SSA_0026" with type=5, size=4383 at
    page 512. Timestamp: 2019-12-20 08:16:04

Received 256b long packet with -132 RSSI; AUTHENTICATION: OK
  Packet type: File Fragment
  Timestamp: 2020-01-01 19:12:41
  Downloading 0/21 fragment of the file "SSA_0027" with type=5, size=4383 at
    page 640. Timestamp: 2019-12-20 08:16:38

Received 256b long packet with -132 RSSI; AUTHENTICATION: OK
  Packet type: File Fragment
  Timestamp: 2020-01-01 19:12:55
  Downloading 11/21 fragment of the file "SSA_0027" with type=5, size=4383 at
    page 640. Timestamp: 2019-12-20 08:16:38

Received 256b long packet with -132 RSSI; AUTHENTICATION: OK
  Packet type: File Fragment
  Timestamp: 2020-01-01 19:12:56
  Downloading 12/21 fragment of the file "SSA_0027" with type=5, size=4383 at
    page 640. Timestamp: 2019-12-20 08:16:38


real  5m11,576s
user  1m17,888s
sys 0m13,444s
```

../rx/smogp.txt

4.4.2 ATL-1

```
Received 256b long packet with -132 RSSI; AUTHENTICATION: OK
  Packet type: File Fragment
  Timestamp: 2020-01-01 19:17:37
  Downloading 7/51 fragment of the file "FS_INFO" with type=1, size=11067 at
    page 0. Timestamp: 2020-01-01 19:17:37

Received 256b long packet with -132 RSSI; AUTHENTICATION: OK
  Packet type: File Fragment
  Timestamp: 2020-01-01 19:17:37
  Downloading 9/51 fragment of the file "FS_INFO" with type=1, size=11067 at
    page 0. Timestamp: 2020-01-01 19:17:37

Received 256b long packet with -132 RSSI; AUTHENTICATION: OK
  Packet type: File Fragment
  Timestamp: 2020-01-01 19:17:37
  Downloading 12/51 fragment of the file "FS_INFO" with type=1, size=11067 at
    page 0. Timestamp: 2020-01-01 19:17:37

Received 256b long packet with -132 RSSI; AUTHENTICATION: OK
  Packet type: File Fragment
  Timestamp: 2020-01-01 19:17:37
  Downloading 13/51 fragment of the file "FS_INFO" with type=1, size=11067 at
    page 0. Timestamp: 2020-01-01 19:17:37

Received 256b long packet with -132 RSSI; AUTHENTICATION: OK
  Packet type: File Fragment
  Timestamp: 2020-01-01 19:17:37
  Downloading 14/51 fragment of the file "FS_INFO" with type=1, size=11067 at
    page 0. Timestamp: 2020-01-01 19:17:37


real   5m4,391s
user   1m22,056s
sys 0m16,080s
```

../rx/atl1.txt

5 Record I-Q Signal

config_webrx.py contains the following line:

```
start_rtl_command="rtl_sdr -s 250000 -f {center_freq} -p {ppm} -g {rf_gain} - |
    doppler.out 250000 predict-2.2.3/masat1file.txt 4 1 | tee -a gnd.iq ".format(
    rf_gain=rf_gain, center_freq=center_freq, samp_rate=samp_rate, ppm=ppm)
```

During the on-line reception, *tee -a gnd.iq* means to record the received data to *gnd.iq* file.

If you want to re-play the recorded I-Q data comment the previous and uncomment the following:

```
start_rtl_command="cat gnd.iq | throttleu8iq.out 62500  ".format(rf_gain=rf_gain
    , center_freq=center_freq, samp_rate=samp_rate, ppm=ppm)
```

6 WebSDR Based Satellite Reception

If you have no receiver antenna and/or SDR, you can track the satellites with our Doppler-corrected WebSDRs: `http://152.66.80.46/`.

Use USB (upper side band, 1500 Hz center) demodulation and GND-SW GUI with sound signal from the OpenWebRx.

1. HA7WEN Levi in Érd - `http://152.66.80.46:8073/`
2. HA7JOD Dani in Érd - `http://152.66.80.46:8074/`
3. HA4TI Tibi in Budapest - `http://152.66.80.46:8077/`
4. BME Etető in Budapest - `http://152.66.80.46:8075/`

7 Demodulate and Decode Packets with GORGON

Download *gorgon.zip* from `http://152.66.80.46/smog1/receivesmogpatl1` and unzip to your home folder (next to *openwebrx*).

Read *readme.txt*.

cd gorgon and install it with **bash build.sh**.

During the satellite reception, you can read the exact center frequency of the received signal of the satellites, e.g. SMOG-P 437150500Hz and ATL-1 437174900Hz because of the frequency error of your RTL-SDR.

Before SMOG-P demodulation, you must set the up-converting frequency of the GMSK demodulator in script *smoggorgon.sh* in line *cnco.out $fs 12000*: 1.

$$437612500 - 437150500 = 12000Hz \tag{1}$$

```
#!/bin/bash
fs=250000

dr12=12500
let dec12=$fs/$dr12/2
avg12=13

dr5=5000
let dec5=$fs/$dr5/2
avg5=30

cd smogpgnd
cmd="./smogpgnd"

f12="smogfifo12"
f5="smogfifo5"
f2="smogfifo2"
f1="smogfifo1"

rm $f12
mkfifo $f12
cat $f12 | \
avgdec.out $dec12 | \
mskdem.out | \
./packet_looter | \
$cmd &

```

```
rm $f5
mkfifo $f5
cat $f5 | \
mskdem.out | \
./packet_looter | \
$cmd &

rm $f2
mkfifo $f2
cat $f2 | \
mskdem.out | \
./packet_looter | \
$cmd &

rm $f1
mkfifo $f1
cat $f1 | \
mskdem.out | \
./packet_looter | \
$cmd &

time cat $1 | \
rtl2cf.out | \
resample.out 4 | \
cnco.out $fs 12500 | \
avg.out $avg12 | \
tee $f12 | \
avg.out $avg5 | \
avgdec.out $dec5 | \
tee $f5 | \
avg.out 2 | \
avgdec.out 2 | \
tee $f2 | \
avg.out 2 | \
avgdec.out 2 | \
tee $f1 > /dev/null
```

../gorgon/smoggorgon.sh

Before ATL-1 demodulation, you must set the down-converting frequency of the GMSK demodulator in script *atlgorgon.sh* in line *cnco.out $fs -12400*: 2.

$$437162500 - 437174900 = -12400Hz \tag{2}$$

```
#!/bin/bash
fs=250000

dr12=12500
let dec12=$fs/$dr12/2
avg12=13

dr5=5000
let dec5=$fs/$dr5/2
avg5=30

cd atlgnd
cmd="./atlgnd"

f12="atlfifo12"
f5="atlfifo5"
```

```
f2="atlfifo2"
f1="atlfifo1"

rm $f12
mkfifo $f12
cat $f12 | \
avgdec.out $dec12 | \
mskdem.out | \
./packet_looter | \
$cmd &

rm $f5
mkfifo $f5
cat $f5 | \
mskdem.out | \
./packet_looter | \
$cmd &

rm $f2
mkfifo $f2
cat $f2 | \
mskdem.out | \
./packet_looter | \
$cmd &

rm $f1
mkfifo $f1
cat $f1 | \
mskdem.out | \
./packet_looter | \
$cmd &

time cat $1 | \
rtl2cf.out | \
resample.out 4 | \
cnco.out $fs -12000 | \
avg.out $avg12 | \
tee $f12 | \
avg.out $avg5 | \
avgdec.out $dec5 | \
tee $f5 | \
avg.out 2 | \
avgdec.out 2 | \
tee $f2 | \
avg.out 2 | \
avgdec.out 2 | \
tee $f1 > /dev/null
```

../gorgon/atlgorgon.sh

The *smoggorgon.sh* and *atlgorgon.sh* scripts can run GMSK demodulators for all used datarates (12500, 5000, 2500, 1250 $\frac{bit}{s}$) and decoders (AO-40 long, AO-40 short, RA) parallel without SYNC packet reception (and synchronization to the actual data-rate and coding scheme).

After satellite tracking, you can find recorded *gnd.iq* file in folder *openwebrx*.

In case of SMOG-P reception, run: **bash smoggorgon.sh /home/levi/openwebrx/gnd.iq**; in case of ATL-1: **bash atlgorgon.sh /home/levi/openwebrx/gnd.iq**.

After demodulation and decoding, you can upload received packets to BME GND server by running: **bash packet_sender.sh** in *smogpgnd* or *atlgnd* folder. Do not forget to set *username*

or *password* before trying to upload.

In folder *openwebrx*, *gnd.iq* file will be appended, so its size will be increased if you do not archive it with e.g.: **mv gnd.iq 20200106.iq** in folder *openwebrx*.

List of Figures

1 On-board Communication System 3
2 Block scheme of TC-TM Radio TCVR 3
3 Block scheme of TC-TM Radio TCVR 5
4 RTL-SDR 4 / 5 / 6 5
5 Waterfall diagram with tracking data 7
6 Satellite tracking with bio-mechanical antenna rotator: 03.01.2020 8
7 SMOG-P signal 8
8 ATL-1 signal 9
9 GND-SW GUI with sound card demodulator 9

List of Tables

1 Transmit Power Level versus Total Power Consumption from nominal $3.3V$ . . . 4
2 The used data rates and its bandwidth 4
3 Coding schemes and packet lengths 4
4 6-element Yagi antenna 5

References

[1] https://www.silabs.com/wireless/proprietary/si10xx-sub-ghz-mcps/device.si1060

[2] https://www.rtl-sdr.com/rtlsdr4everyone-sdruno-1-04-guide-updated-and-overview-of-rt

[3] https://github.com/simonyiszk/openwebrx.git

[4] https://github.com/simonyiszk/csdr.git

8 Appendix

8.1 Doppler Corrector

The Doppler corrector is mixing the incoming baseband I-Q signal according to the Predict calculated Doppler shift to the 0-Doppler baseband. After Doppler correction, on the waterfall diagram you can see constant carrier signal (vertical).

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <complex.h>
#include <malloc.h>
#include <unistd.h>

#define INBUFLEN 16384
#define DC 128
void readDopplerFile(char *fn,int *fd){
  FILE *fp;
  fp=fopen(fn,"rt");
  int az,el,en;
  long dl,ul;
  if(fp!=NULL){
    fscanf(fp,"az\t%d\n",&az);
    fscanf(fp,"el\t%d\n",&el);
    fscanf(fp,"dl\t%ld\n",&dl);
    fscanf(fp,"ul\t%ld\n",&ul);
    fscanf(fp,"en\t%d\n",&en);
    fclose(fp);
    *fd=dl-(dl+ul)/2;
  }
}
int main(int argc,char **argv)
{
  if(argc==1){
    printf("%s <sampling freq> <doppler file> <decimating factor> <0 real, 1 iq>
    ",argv[0]);
    return 0;
  }
  long fs=256000;
  if(argc>1) fs=atol(argv[1]);
  int fd=0;
  readDopplerFile(argv[2],&fd);
  int dec=10;
  if(argc>3) dec=atoi(argv[3]);
  int deccnt=0;
  complex double integrate=0+0*I;
  complex double diff=0+0*I;
  complex double mem=0+0*I;
  complex double inputsample=0+0*I;
  //complex double LO[fs];
  complex double *LO;
  LO=malloc(fs*sizeof(double complex));
  long i;
  for(i=0;i<fs;i++) LO[i]=cos(2.0*M_PI/fs*i)-sin(2.0*M_PI/fs*i)*I;
  long locnt=0;
  long incnt=1;
  unsigned char inputbuffer[INBUFLEN];
  int outbuflen=INBUFLEN/dec*2;
  unsigned char outputbuffer[outbuflen];
```

```
  long outcnt=0;
  double norm=1.0/(dec*log((double)dec)/log(2.0));
  norm=1.0/dec;
  int iq=1;
  if(argc>4) iq=atoi(argv[4]);
  while(incnt!=0){
    readDopplerFile(argv[2],&fd);
    incnt=fread(inputbuffer,1,INBUFLEN,(FILE*)stdin);
    if(incnt>0) for(i=0;i<incnt;i+=2){
      inputsample=(inputbuffer[i]-DC)+(inputbuffer[i+1]-DC)*I;
      if(fd>=0) integrate+=(inputsample*LO[locnt]);
      else integrate+=(inputsample*conj(LO[locnt]));
      locnt=(locnt+abs(fd))%fs;
      deccnt=(deccnt+1)%dec;
      if(deccnt==0){
        diff=integrate-mem;
        mem=integrate;
        outputbuffer[outcnt]=creal(diff)*norm+DC;
        outcnt=(outcnt+1)%outbuflen;
        if(iq){
          outputbuffer[outcnt]=cimag(diff)*norm+DC;
          outcnt=(outcnt+1)%outbuflen;
        }
        if(outcnt==0){
          fwrite(outputbuffer,1,outbuflen,(FILE*)stdout);
        }
      }
    }
    usleep(1000);
  }
  free(LO);
  return 0;
}
```

../openwebrx/doppler.c

8.2 Throttle

Throttle module feeds the *openwebrx* input with quasi-real-time I-Q samples while re-playing.

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <malloc.h>
int main(int argc, char **argv)
{
  unsigned long fs=64000;
  if(argc>1) fs=atol(argv[1]);
  unsigned char *iq;
  iq=malloc(fs*2);
  if(iq==NULL) return -1;
  int k;
  unsigned long t=0;
  while(1){
    k=fread(iq,2,fs,stdin);
    if(feof(stdin)) break;
    if(k>=fs){
      fwrite(iq,2,fs,stdout);
      fflush(stdout);
      fprintf(stderr,"%ld s read\n",++t);
```

```
      fflush(stderr);
      sleep(1);
    }
    else{
      usleep(10);
    }
  }
  free(iq);
  return 0;
}
```

../openwebrx/throttleu8iq.c

8.3 Complex Numerical Controlled Oscillator

CNCO is converting the recorded IQ signal up or down to the 0 Hz IQ baseband.

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <complex.h>
#include <malloc.h>
#include <unistd.h>

int main(int argc, char **argv)
{
  long fs=100000;
  if(argc>1) fs=atol(argv[1]);
  long f=10000;
  if(argc>2) f=atol(argv[2]);
  complex float *lo;
  lo=malloc(sizeof(complex float)*fs);
  if(lo==NULL) return 1;
  complex float *in;
  in=malloc(sizeof(complex float)*fs);
  if(in==NULL) return 2;
  complex float *out;
  out=malloc(sizeof(complex float)*fs);
  if(out==NULL) return 3;

  long i;
  for(i=0;i<fs;i++){
    float p=2.0*M_PI/fs*i;
    if(f>0){
      lo[i]=cos(p)+sin(p)*I;
    }
    else{
      lo[i]=cos(p)-sin(p)*I;
    }
  }
  if(f<0) f=-f;
  long loi=0;
  while(1){
    long k=fread(in,sizeof(complex float),fs,stdin);
    if(feof(stdin)) break;
    if(k>=fs){
      long t;
      for(t=0;t<fs;t++){
        out[t]=in[t]*lo[loi];
        loi=(loi+f)%fs;
```

```
      }
      fwrite(out,sizeof(complex float),fs,stdout);
    }
    else{
      usleep(100);
    }
  }
  free(lo);
  free(in);
  free(out);
  fflush(stdout);
  return 0;
}
```

../gorgon/sp/cnco.c

8.4 Averaging as Low-Pass-Filtering

In *gorgon/sp*, *avg.c* forms the elementary low-pass-filter.

```
#include <stdio.h>
#include <stdlib.h>
#include <complex.h>
#include <unistd.h>
#include <malloc.h>

int main(int argc, char **argv)
{
  complex float in[1];
  long n=1;
  if(argc>1) n=atol(argv[1]);
  complex float *buf;
  buf=malloc(n*sizeof(complex float));
  if(buf==NULL) return -1;
  long i;
  for(i=0;i<n;i++) buf[i]=0+0*I;
  complex float out[1]={0+0*I};
  int index=0;
  while(1){
    int k=fread(in,sizeof(complex float),1,stdin);
    if(feof(stdin)) break;
    if(k>0){
      out[0] -= buf[index];
      buf[index]=in[0];
      out[0] += buf[index];
      index=(index+1)%n;
      fwrite(out,sizeof(complex float),1,stdout);
    }
    else{
      usleep(10);
    }
  }
  free(buf);
  return 0;
}
```

../gorgon/sp/avg.c

8.5 Decimating Filter

avgdec.c is responsible for the decimation of the incoming IQ signal (decrease sampling rate and low-pass-filtering).

```
1000 #include <stdio.h>
     #include <stdlib.h>
1002 #include <complex.h>
     #include <unistd.h>
1004 #include <malloc.h>

1006 int main(int argc, char **argv)
     {
1008   unsigned long n=1024;
       if(argc>1) n=atol(argv[1]);
1010   complex float *in;
       in=malloc(sizeof(complex float)*n);
1012   if(in==NULL) return -1;
       unsigned long k;
1014   unsigned long i;
       complex float out[1];
1016   while(1){
         k=fread(in,sizeof(complex float),n,stdin);
1018     if(feof(stdin)) break;
         if(k>0){
1020       out[0]=0+0*I;
           for(i=0;i<n;i++){
1022         out[0]+=in[i];
           }
1024       out[0]/=n;
           fwrite(out,sizeof(complex float),1,stdout);
1026     }
         else{
1028       usleep(10);
         }
1030   }
       free(in);
1032   return 0;
     }
```

../gorgon/sp/avgdec.c

8.6 MSK Demodulator

mskdem.c is the demodulator of the IQ MSK or GMSK signal: two IQ samples / databit.

```
1000 #include <stdio.h>
     #include <stdlib.h>
1002 #include <complex.h>
     #include <unistd.h>
1004

     #define LL 32
1006

     int main(int argc, char **argv)
1008 {
       complex float m=0+0*I;
1010   complex float in[1];
       complex float out[1];
1012   float t=0+0*I;
       float x=0+0*I;
```

```
1014    while(1){
          int k=fread(in,sizeof(complex float),1,stdin);
1016      if(feof(stdin)) break;
          if(k>0){
1018        t=cimag(in[0]*conj(m));
            m=in[0];
1020        x=1./LL*t+(LL-1.0)/LL*x;
            out[0]=t+x/2*I;
1022        fwrite(out,sizeof(complex float),1,stdout);
          }
1024      else{
            usleep(100);
1026      }
        }
1028    return 0;
    }
```

../gorgon/sp/mskdem.c