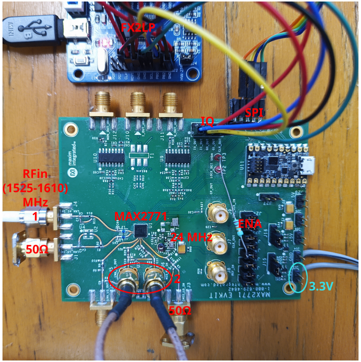
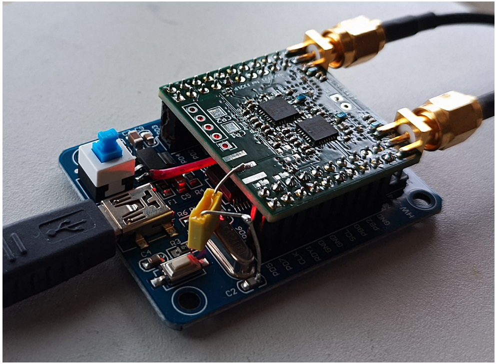
Efficient USB communication under GNU/Linux for a wideband (MAX2771-based) L-band (GNSS) SDR receiver

J.-M Friedt
FEMTO-ST Time & Frequency, Besançon, France

From 

to 

January 11, 2025

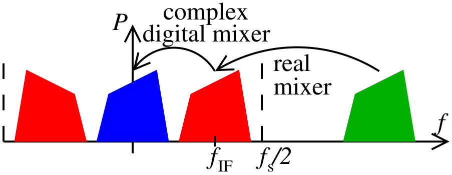
Why do we need high bandwidth data transfer?

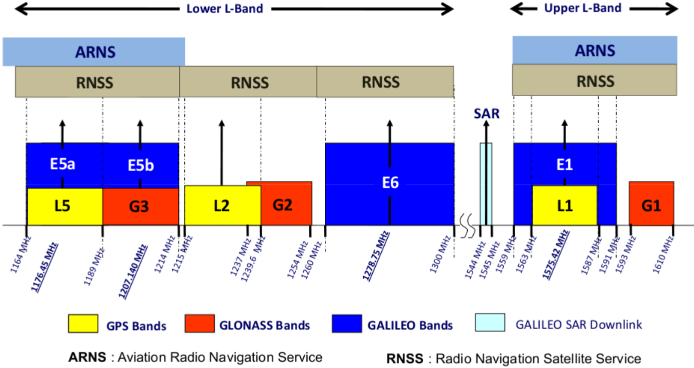
More and more GNSS constellations with increasingly complex (and accurate) coding schemes

- legacy GPS L1 C/A: 2.046 MHz bandwidth (1.023 MSps complex)
- Galileo E1: 4 MHz bandwidth (2 MSps complex)
- GPS L2C: 2.046 MHz bandwidth
- GPS L5: 24 MHz bandwidth (8 MSps complex sufficient, aeronautical band, 10 Mchips/s BPSK)
- Galileo E5a: 20.46 MHz bandwidth (5 MSps complex sufficient)

Very weak signals (below thermal noise)

⇒ might benefit from an intermediate frequency to get rid of DC noise ⇒ more bandwidth needed





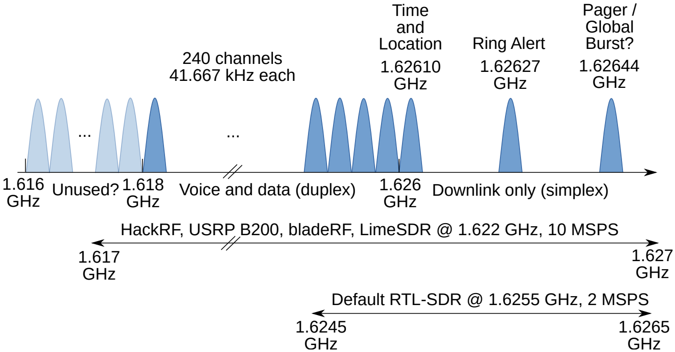
GNSS L-band (1–2 GHz) frequency allocations [a]

[a] gssc.esa.int/navipedia/index.php/GNSS_signal

Why do we need high bandwidth data transfer?

More and more GNSS constellations with increasingly complex (and accurate) coding schemes

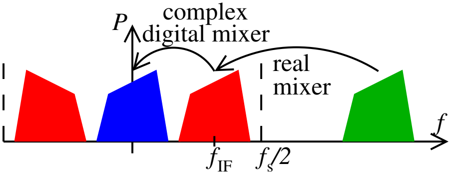
- legacy GPS L1 C/A: 2.046 MHz bandwidth (1.023 MSps complex)
- Galileo E1: 4 MHz bandwidth (2 MSps complex)
- GPS L2C: 2.046 MHz bandwidth
- GPS L5: 24 MHz bandwidth (8 MSps complex sufficient, aeronautical band, 10 Mchips/s BPSK)
- Galileo E5a: 20.46 MHz bandwidth (5 MSps complex sufficient)



Iridium LEO satellite frequency allocations [a]: 1616–1626.5 MHz

Very weak signals (below thermal noise)

⇒ might benefit from an intermediate frequency to get rid of DC noise ⇒ more bandwidth needed



(but tuning to 1420 MHz neutral hydrogen fails)

[a] https://github.com/muccc/gr-iridium

GNSS bandwidths

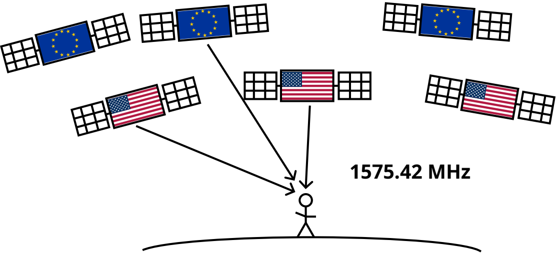
BOC($f_s/f_0, f_c/f_0$): Binary Offset Carrier (f_0 ref freq., f_s subcarrier freq., f_c chip rate)

System	Carrier [MHz]	Signal	Type	Modulation	Chipping rate	Code Length	Full length [ms]
GPS	L1 1575.420	C/A	Data	BPSK	1.023Mcps	1023	1
	L5 1176.450	I	Data	QPSK	10.23Mcps	10230	1
		Q	Pilot		10.23Mcps	10230	1
Galileo	E1 1575.42	B	Data	BOC(1,1)	1.023Mcps	4092 * 1	4
		C	Pilot		1.023Mcps	4092 * 25	100
	E5 a:1176.450	a-I	Data	AltBOC(15,10)	10.23Mcps	10230 * 20	20
		a-Q	Pilot		10.23Mcps	10230 * 100	100

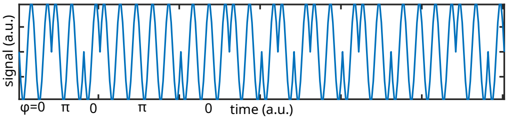
"The minimum bandwidth is generally twice the chipping rate for simple codes, while for BOC codes it is twice the sum of chipping rate and offset code rate. Thus, the minimum practical bandwidth for the Galileo E1 is 8 MHz." [1]

[1] K. Borre Lecture at SU May 27, 2009, *The E1 Galileo Signal* at http://web.stanford.edu/group/scpnt/gpslab/pubs/papers/Borre/galileo_sig.pdf

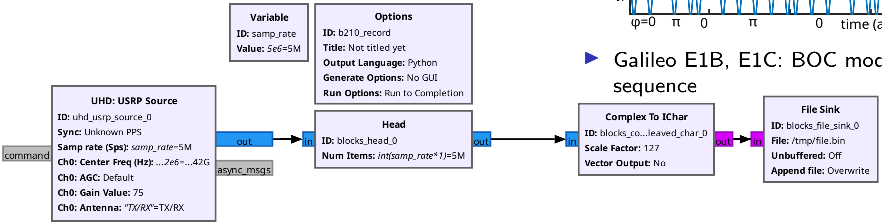
CDMA communication [2]



- CDMA: each transmitted bit is encoded as a series of pseudo-random number (PRN) sequence with 0 or 1 bits encoded as 0 or π phase
- Cross correlation $\mathrm{xcorr}(x,p)(\tau) = \int x(t) \cdot p^*(t+\tau)dt$
- Linearity: $\mathrm{xcorr}(ax+by,p) = a \cdot \mathrm{xcorr}(x,p) + b \cdot \mathrm{xcorr}(y,p)$
- In the Fourier domain: $\mathrm{FFT}(\mathrm{xcorr}(x,p)) = \mathrm{FFT}(x) \cdot \mathrm{FFT}^*(p)$
- code orthogonality: $\mathrm{xcorr}(c_i,c_j)=\delta_{i,j}$
- GPS L1 C/A: BPSK modulated Gold code sequence



- Galileo E1B, E1C: BOC modulated PRN code sequence



GNSS signals: GPS L1 C/A and Galileo E1B/E1C

- BPSK modulation of pseudo-random sequences (Gold codes [3])

```
load GNSS-matlab/prn_codes/codes_L1CA.mat;
code=interpolated(codes_L1CA(:,m),fs/1.023e6);
% 0/pi phase at baseband
```

- BOC modulation of pseudo-random sequences

```
load GNSS-matlab/prn_codes/codes_E1B.mat
Rsa=1.023e6;
Rsb=6.138e6;
m=1;
code=interpolated(codes_E1B(:,m),fs/1.023e6);
temps=[0:length(code)-1]'/fs;
sce1a=sqrt(10/11)*((sin(2*pi*temps*Rsa)>0)*2-1);
sce1b=sqrt(1/11)*((sin(2*pi*temps*Rsb)>0)*2-1);
signal=(sce1a+sce1b).*code);
```

See "Galileo open service signal-in-space interface control document (OS SIS ICD)" at https://www.gsc-europa.eu/sites/default/files/sites/all/files/Galileo_OS_SIS_ICD_v2.1.pdf (courtesy of C. Plantard, ESTEC)

E1 Signal

Figure 7 provides a generic view of the E1 CBOC signal generation.

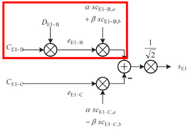


Figure 7: Modulation Scheme for the E1 CBOC Signal

The E1 CBOC signal components are generated as follows:

- $e_{E1\text{-}B}$ from the I/NAV navigation data stream $D_{E1\text{-}B}$ and the ranging code (
  then modulated with the sub-carriers $sc_{E1\text{-}B,a}$ and $sc_{E1\text{-}B,b}$

The E1-B/C composite signal is then generated according to equation Eq. 11 below, with the binary signal components $e_{E1\text{-}B}(t)$ and $e_{E1\text{-}C}(t)$. Note that as for E6, both pilot and data components are modulated onto the same carrier component, with a power sharing of 50 percent.

$$s_{E1}(t) = \frac{1}{\sqrt{2}}\Big(e_{E1-B}(t)\big(\alpha\, sc_{E1-B,a}(t) + \beta\, sc_{E1-B,b}(t)\big) - e_{E1-C}(t)\big(\alpha\, sc_{E1-C,a}(t) - \beta\, sc_{E1-C,b}(t)\big)\Big)$$

$$\text{with} \quad sc_X(t) = \operatorname{sgn}\big(\sin(2\pi\, R_{s,X}\, t)\big) \qquad \text{Eq. 11}$$

The parameters α and β are chosen such that the combined power of the $sc_{E1\text{-}B,b}$ and the $sc_{E1\text{-}C,b}$ sub carrier components equals 1/11 of the total power of $e_{E1\text{-}B}$ plus $e_{E1\text{-}C}$, before application of any bandwidth limitation. This yields:

$$e_{E1-B}(t) = \sum_{i=-\infty}^{\infty}\Big[C_{E1-B,|i|_{L_{E1-B}}}\; D_{E1-B,[i]_{DC_{E1-B}}}\; \mathrm{rect}_{T_{C,E1-B}}(t - i\, T_{C,E1-B})\Big]$$

$$e_{E1-C}(t) = \sum_{i=-\infty}^{\infty}\Big[C_{E1-C,|i|_{L_{E1-C}}}\; \mathrm{rect}_{T_{C,E1-C}}(t - i\, T_{C,E1-C})\Big] \qquad \text{Eq. 10}$$

Galileo satellites transmit ranging signals for the E1 signal with the chip rates and sub- carrier rates defined in the following Table 9.

Table 9: E1 CBOC Chip Rates and Sub-carrier Rates

Component (Parameter Y)	Sub-carrier Type	Sub-carrier Rate $R_{S,E1\text{-}Y,a}$ (MHz)	Sub-carrier Rate $R_{S,E1\text{-}Y,b}$ (MHz)	Ranging Code Chip-Rate $R_{C,E1\text{-}Y}$ (Mcps)
B	CBOC, in-phase	1.023	6.138	1.023
C	CBOC, anti-phase	1.023	6.138	1.023

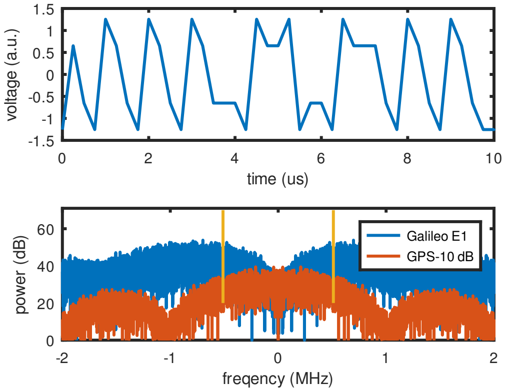
The navigation data message stream, after channel encoding, is transmitted with the symbol rate as stated in Table 10.

---

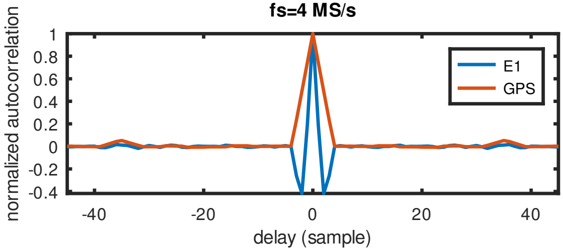
[3] All PRN sequences at https://github.com/danipascual/GNSS-matlab

Galileo bandwidth: BOC to avoid disturbing BPSK main lobe

Top: Galileo E1 waveform
Bottom: red=GPS L1 C/A spectrum (1.023 Mchip/s), blue=Galileo E1 spectrum



BPSK at X MS/s exhibits notches at $\pm X$ MHz

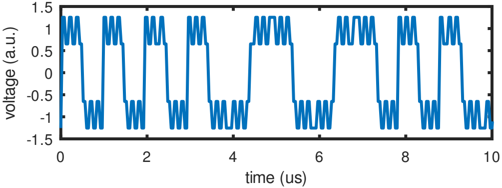


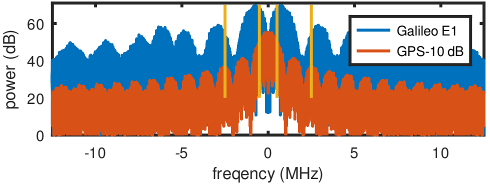
Autocorrelation function (correlation between broadcast $p(t)$ and received signal $p(t)+n(t)$), red=BPSK PRN, blue=BOC (narrower but multiple peaks)

- Byte size output to remain close to the 2-3 bit ADC of the MAX2771
- Compatible with the expectation of PocketSDR

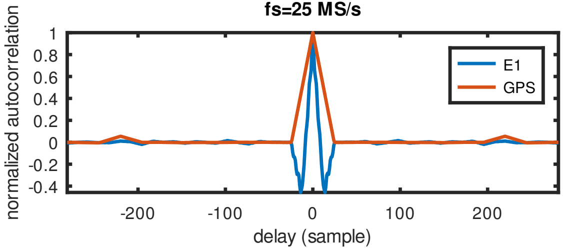
Galileo bandwidth: BOC to avoid disturbing BPSK main lobe

Top: Galileo E1 waveform
Bottom: red=GPS L1 C/A spectrum (1.023 Mchip/s), blue=Galileo E1 spectrum





BPSK at X MS/s exhibits notches at $\pm X$ MHz



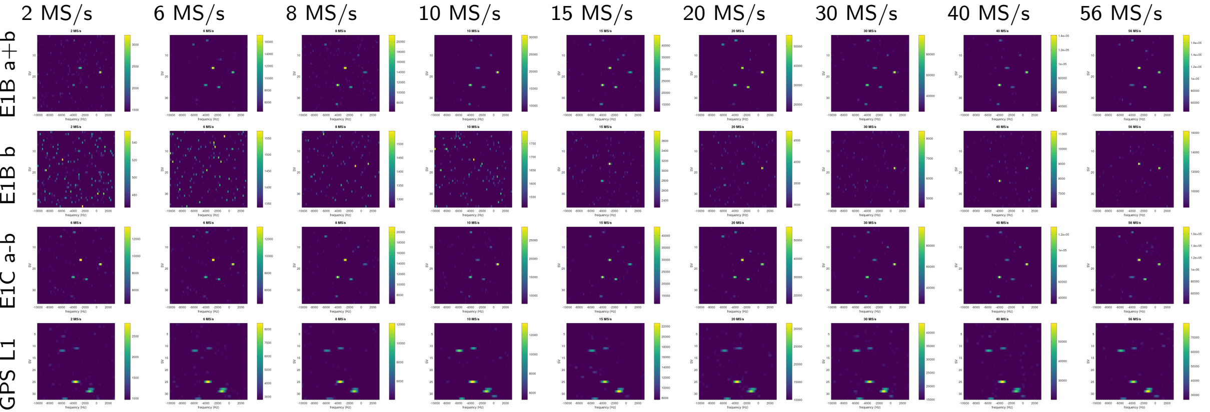
Autocorrelation function (correlation between broadcast $p(t)$ and received signal $p(t)+n(t)$), red=BPSK PRN, blue=BOC (narrower but multiple peaks)

- Byte size output to remain close to the 2-3 bit ADC of the MAX2771
- Compatible with the expectation of PocketSDR

PRN-Doppler maps for various bandwidths (B210 records)

- Replace in the GNU Radio Companion generated Python script: `self.samp_rate = samp_rate = int(XXX*1e6)` and
  `self.blocks_file_sink_0 = blocks.file_sink(gr.sizeof_char*1, f"/tmp/{XXX:02}MSps_char.bin", False)`
- execute the bash script

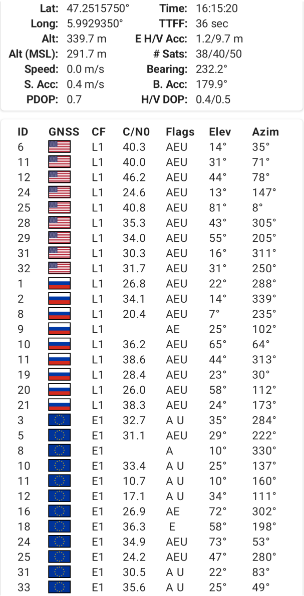
```
for  i in 2 4 5 6 8 10 12 15 20 25 30 40 48 56; do echo $i; \
cat b210_record_template.py | sed "s/XXX/$i/g"> b210_record.py; python3 ./b210_record.py;done
```



Frequency step≪code duration, 1023/1023 Mchips/s=1 ms for GPS L1, 4092/1023 Mchips/s=4 ms for E1

PocketSDR processing scripts: process 8-byte IQ interleaved datasets

```
$ python3 PocketSDR/python/pocket_acq.py 06MSps_char.bin -f 6 -sig L1CA -prn 1-32
...
SIG= L1CA, PRN=  10, COFF=  0.71933 ms, DOP=  3546 Hz, C/N0= 33.7 dB-Hz
SIG= L1CA, PRN=  11, COFF=  0.86367 ms, DOP=  3422 Hz, C/N0= 45.0 dB-Hz
SIG= L1CA, PRN=  12, COFF=  0.98800 ms, DOP=  5000 Hz, C/N0= 34.5 dB-Hz
...
SIG= L1CA, PRN=  24, COFF=  0.61450 ms, DOP= -2152 Hz, C/N0= 34.2 dB-Hz
SIG= L1CA, PRN=  25, COFF=  0.31883 ms, DOP=  3575 Hz, C/N0= 49.4 dB-Hz
SIG= L1CA, PRN=  26, COFF=  0.42733 ms, DOP= -2947 Hz, C/N0= 33.3 dB-Hz
SIG= L1CA, PRN=  27, COFF=  0.54617 ms, DOP=  4458 Hz, C/N0= 33.7 dB-Hz
SIG= L1CA, PRN=  28, COFF=  0.08700 ms, DOP=  1011 Hz, C/N0= 46.7 dB-Hz
SIG= L1CA, PRN=  29, COFF=  0.80517 ms, DOP=  1203 Hz, C/N0= 48.3 dB-Hz
SIG= L1CA, PRN=  30, COFF=  0.58400 ms, DOP=  2521 Hz, C/N0= 34.1 dB-Hz
TIME = 5.563 s
$ python3 PocketSDR/python/pocket_acq.py 06MSps_char.bin -f 6 -sig E1B -prn 1-36
SIG= E1B , PRN=   1, COFF=  2.69133 ms, DOP=  2987 Hz, C/N0= 33.5 dB-Hz
SIG= E1B , PRN=   2, COFF=  0.81550 ms, DOP= -4747 Hz, C/N0= 33.3 dB-Hz
SIG= E1B , PRN=   3, COFF=  2.42433 ms, DOP=  3818 Hz, C/N0= 43.1 dB-Hz
SIG= E1B , PRN=   4, COFF=  3.07167 ms, DOP= -1014 Hz, C/N0= 33.5 dB-Hz
...
SIG= E1B , PRN=   9, COFF=  2.30333 ms, DOP=   874 Hz, C/N0= 33.3 dB-Hz
SIG= E1B , PRN=  10, COFF=  1.35067 ms, DOP=   499 Hz, C/N0= 41.3 dB-Hz
SIG= E1B , PRN=  11, COFF=  2.28183 ms, DOP=  2771 Hz, C/N0= 33.2 dB-Hz
SIG= E1B , PRN=  12, COFF=  3.65550 ms, DOP=  1345 Hz, C/N0= 38.8 dB-Hz
SIG= E1B , PRN=  13, COFF=  2.03700 ms, DOP=  3871 Hz, C/N0= 32.7 dB-Hz
SIG= E1B , PRN=  14, COFF=  2.72567 ms, DOP= -1271 Hz, C/N0= 32.7 dB-Hz
SIG= E1B , PRN=  15, COFF=  1.34200 ms, DOP=  2381 Hz, C/N0= 33.3 dB-Hz
SIG= E1B , PRN=  16, COFF=  3.44700 ms, DOP=  2731 Hz, C/N0= 47.6 dB-Hz
SIG= E1B , PRN=  17, COFF=  0.30967 ms, DOP=  4012 Hz, C/N0= 33.0 dB-Hz
SIG= E1B , PRN=  18, COFF=  2.34050 ms, DOP=  -642 Hz, C/N0= 47.8 dB-Hz
SIG= E1B , PRN=  19, COFF=  1.10050 ms, DOP= -4131 Hz, C/N0= 32.6 dB-Hz
...
SIG= E1B , PRN=  23, COFF=  0.87517 ms, DOP= -1763 Hz, C/N0= 32.9 dB-Hz
SIG= E1B , PRN=  24, COFF=  0.60100 ms, DOP=  3993 Hz, C/N0= 47.6 dB-Hz
SIG= E1B , PRN=  25, COFF=  3.08300 ms, DOP=  1740 Hz, C/N0= 45.2 dB-Hz
SIG= E1B , PRN=  26, COFF=  0.38833 ms, DOP=  3872 Hz, C/N0= 33.8 dB-Hz
...
SIG= E1B , PRN=  32, COFF=  0.33350 ms, DOP=  2114 Hz, C/N0= 33.4 dB-Hz
SIG= E1B , PRN=  33, COFF=  1.24983 ms, DOP=  4163 Hz, C/N0= 42.4 dB-Hz
```



Lat:	47.2515750°	Time:	16:15:20
Long:	5.9929350°	TTFF:	36 sec
Alt:	339.7 m	E H/V Acc:	1.2/9.7 m
Alt (MSL):	291.7 m	# Sats:	38/40/50
Speed:	0.0 m/s	Bearing:	232.2°
S. Acc:	0.4 m/s	B. Acc:	179.9°
PDOP:	0.7	H/V DOP:	0.4/0.5

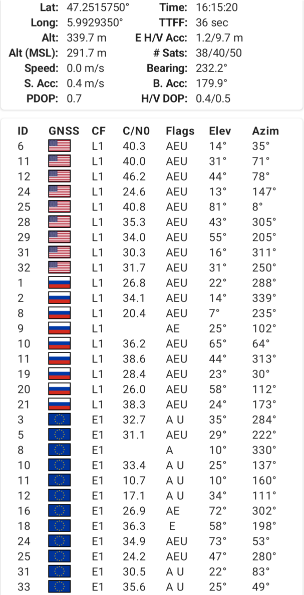
ID	GNSS	CF	C/N0	Flags	Elev	Azim
6		L1	40.3	AEU	14°	35°
11		L1	40.0	AEU	31°	71°
12		L1	46.2	AEU	44°	78°
24		L1	24.6	AEU	13°	147°
25		L1	40.8	AEU	81°	8°
28		L1	35.3	AEU	43°	305°
29		L1	34.0	AEU	55°	205°
31		L1	30.3	AEU	16°	311°
32		L1	31.7	AEU	31°	250°
1		L1	26.8	AEU	22°	288°
2		L1	34.1	AEU	14°	339°
8		L1	20.4	AEU	7°	235°
9		L1		AE	25°	102°
10		L1	36.2	AEU	65°	64°
11		L1	38.6	AEU	44°	313°
19		L1	28.4	AEU	23°	30°
20		L1	26.0	AEU	58°	112°
21		L1	38.3	AEU	24°	173°
3		E1	32.7	A U	35°	284°
5		E1	31.1	AEU	29°	222°
8		E1		A	10°	330°
10		E1	33.4	A U	25°	137°
11		E1	10.7	A U	10°	160°
12		E1	17.1	A U	34°	111°
16		E1	26.9	AE	72°	302°
18		E1	36.3	E	58°	198°
24		E1	34.9	AEU	73°	53°
25		E1	24.2	AEU	47°	280°
31		E1	30.5	A U	22°	83°
33		E1	35.6	A U	25°	49°

Android GPSTest

PocketSDR processing scripts: process 8-byte IQ interleaved datasets

```
$ python3 PocketSDR/python/pocket_acq.py 06MSps_char.bin -f 6 -sig L1CA -prn 1-32
...
SIG= L1CA, PRN=  10, COFF=  0.71933 ms, DOP=  3546 Hz, C/N0= 33.7 dB-Hz
SIG= L1CA, PRN=  11, COFF=  0.86367 ms, DOP=  3422 Hz, C/N0= 45.0 dB-Hz
SIG= L1CA, PRN=  12, COFF=  0.98800 ms, DOP=  5000 Hz, C/N0= 34.5 dB-Hz
...
SIG= L1CA, PRN=  24, COFF=  0.61450 ms, DOP= -2152 Hz, C/N0= 34.2 dB-Hz
SIG= L1CA, PRN=  25, COFF=  0.31883 ms, DOP=  3575 Hz, C/N0= 49.4 dB-Hz
SIG= L1CA, PRN=  26, COFF=  0.42733 ms, DOP= -2947 Hz, C/N0= 33.3 dB-Hz
SIG= L1CA, PRN=  27, COFF=  0.54617 ms, DOP=  4458 Hz, C/N0= 33.7 dB-Hz
SIG= L1CA, PRN=  28, COFF=  0.08700 ms, DOP=  1011 Hz, C/N0= 46.7 dB-Hz
SIG= L1CA, PRN=  29, COFF=  0.80517 ms, DOP=  1203 Hz, C/N0= 48.3 dB-Hz
SIG= L1CA, PRN=  30, COFF=  0.58400 ms, DOP=  2521 Hz, C/N0= 34.1 dB-Hz
TIME = 5.563 s
$ python3 PocketSDR/python/pocket_acq.py 06MSps_char.bin -f 6 -sig E1C -prn 1-36
SIG= E1C , PRN=   1, COFF=  1.73517 ms, DOP=  4015 Hz, C/N0= 33.2 dB-Hz
SIG= E1C , PRN=   2, COFF=  2.03017 ms, DOP= -1003 Hz, C/N0= 33.5 dB-Hz
SIG= E1C , PRN=   3, COFF=  2.42433 ms, DOP=  3800 Hz, C/N0= 42.6 dB-Hz
SIG= E1C , PRN=   4, COFF=  1.02800 ms, DOP=  3123 Hz, C/N0= 32.9 dB-Hz
SIG= E1C , PRN=   5, COFF=  3.79883 ms, DOP=  4758 Hz, C/N0= 32.9 dB-Hz
...
SIG= E1C , PRN=  10, COFF=  1.35067 ms, DOP=   501 Hz, C/N0= 41.0 dB-Hz
SIG= E1C , PRN=  11, COFF=  1.90217 ms, DOP=  3235 Hz, C/N0= 34.1 dB-Hz
SIG= E1C , PRN=  12, COFF=  3.65550 ms, DOP=  1361 Hz, C/N0= 39.8 dB-Hz
SIG= E1C , PRN=  13, COFF=  0.72617 ms, DOP=  5000 Hz, C/N0= 32.8 dB-Hz
...
SIG= E1C , PRN=  25, COFF=  3.08300 ms, DOP=  1743 Hz, C/N0= 46.1 dB-Hz
SIG= E1C , PRN=  26, COFF=  2.97683 ms, DOP=  4707 Hz, C/N0= 33.1 dB-Hz
SIG= E1C , PRN=  27, COFF=  3.58350 ms, DOP= -1902 Hz, C/N0= 33.2 dB-Hz
SIG= E1C , PRN=  28, COFF=  1.32583 ms, DOP= -1372 Hz, C/N0= 33.3 dB-Hz
SIG= E1C , PRN=  29, COFF=  3.51250 ms, DOP=   498 Hz, C/N0= 32.8 dB-Hz
SIG= E1C , PRN=  30, COFF=  0.58083 ms, DOP= -1398 Hz, C/N0= 33.6 dB-Hz
SIG= E1C , PRN=  31, COFF=  1.94067 ms, DOP=  1885 Hz, C/N0= 32.8 dB-Hz
SIG= E1C , PRN=  32, COFF=  3.14367 ms, DOP=  4896 Hz, C/N0= 32.8 dB-Hz
SIG= E1C , PRN=  33, COFF=  1.24983 ms, DOP=  4156 Hz, C/N0= 41.7 dB-Hz
SIG= E1C , PRN=  34, COFF=  1.15417 ms, DOP= -4191 Hz, C/N0= 33.2 dB-Hz
SIG= E1C , PRN=  35, COFF=  3.24267 ms, DOP=   656 Hz, C/N0= 33.0 dB-Hz
SIG= E1C , PRN=  36, COFF=  2.69433 ms, DOP=  4251 Hz, C/N0= 33.4 dB-Hz
TIME = 19.945 s
```

Android GPSTest



Lat:	47.2515750°	Time:	16:15:20
Long:	5.9929350°	TTFF:	36 sec
Alt:	339.7 m	E H/V Acc:	1.2/9.7 m
Alt (MSL):	291.7 m	# Sats:	38/40/50
Speed:	0.0 m/s	Bearing:	232.2°
S. Acc:	0.4 m/s	B. Acc:	179.9°
PDOP:	0.7	H/V DOP:	0.4/0.5

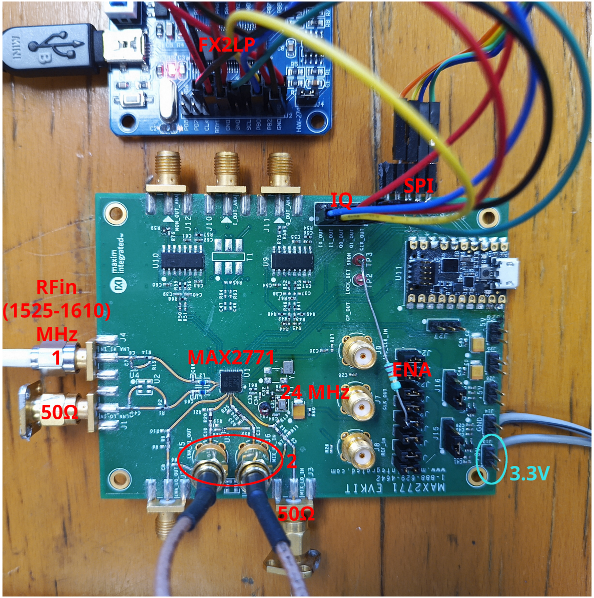
ID	GNSS	CF	C/N0	Flags	Elev	Azim
6		L1	40.3	AEU	14°	35°
11		L1	40.0	AEU	31°	71°
12		L1	46.2	AEU	44°	78°
24		L1	24.6	AEU	13°	147°
25		L1	40.8	AEU	81°	8°
28		L1	35.3	AEU	43°	305°
29		L1	34.0	AEU	55°	205°
31		L1	30.3	AEU	16°	311°
32		L1	31.7	AEU	31°	250°
1		L1	26.8	AEU	22°	288°
2		L1	34.1	AEU	14°	339°
8		L1	20.4	AEU	7°	235°
9		L1		AE	25°	102°
10		L1	36.2	AEU	65°	64°
11		L1	38.6	AEU	44°	313°
19		L1	28.4	AEU	23°	30°
20		L1	26.0	AEU	58°	112°
21		L1	38.3	AEU	24°	173°
3		E1	32.7	A U	35°	284°
5		E1	31.1	AEU	29°	222°
8		E1		A	10°	330°
10		E1	33.4	A U	25°	137°
11		E1	10.7	A U	10°	160°
12		E1	17.1	A U	34°	111°
16		E1	26.9	AE	72°	302°
18		E1	36.3	E	58°	198°
24		E1	34.9	AEU	73°	53°
25		E1	24.2	AEU	47°	280°
31		E1	30.5	A U	22°	83°
33		E1	35.6	A U	25°	49°

MAX2771 and its evaluation board[4]

- MAX2771: general purpose L-band (1–2 GHz) receiver …
- … with 2 or 3-bit ADC, up to 44 MS/s parallel output
- dedicated to "upper" (L1, 1575.42 MHz) and "lower" (L2, L5… 1200 MHz) GNSS bands.

fitted on an evaluation by Maxim IC (now ADi)

- control software only running with MS-Windows (fails with Wine)
- after reverse engineering the USB control interface (VirtualBox guest on Linux host and usbmon debugging interface [a])… no IQ streaming, only converts USB Vendor Request commands to SPI sentences, but IQ pins are left **unconnected**



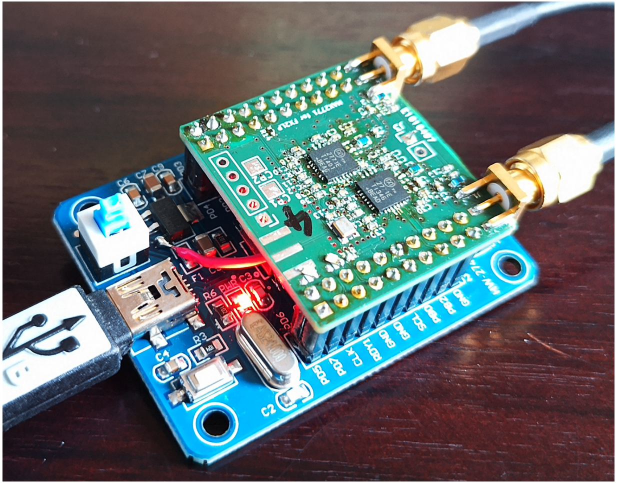
[a] `https://www.baeldung.com/linux/usb-sniffing`: `sudo mount -t debugfs device_debug /sys/kernel/debug` and `sudo modprobe usbmon` to gain access to `/sys/kernel/debug/usb/usbmon/Xu` for sniffing USB communication

[4] Farnell MAX2771EVKIT, 532 euros in Dec. 2024

PocketSDR [6]: dual MAX2771 GNSS SDR receiver [7]

- Two or four MAX2771 RF frontends [5]
- either two bands decoded simultaneously, or same band monitored with two antennas (CRPA [a])
- USB communication interface: Cypress EZ-USB FX2LP ≤48 MS/s[b]
- IQ stream either decoded using Python scripts or fed to `gnss-sdr` (FIFO for real time)

Reproduced here with a custom board



[a] Controlled Radiation Pattern Antenna for jamming/spoofing detection and mitigation

[b] compare with LUFA CDC (Communications Device Class): 100 kB/s

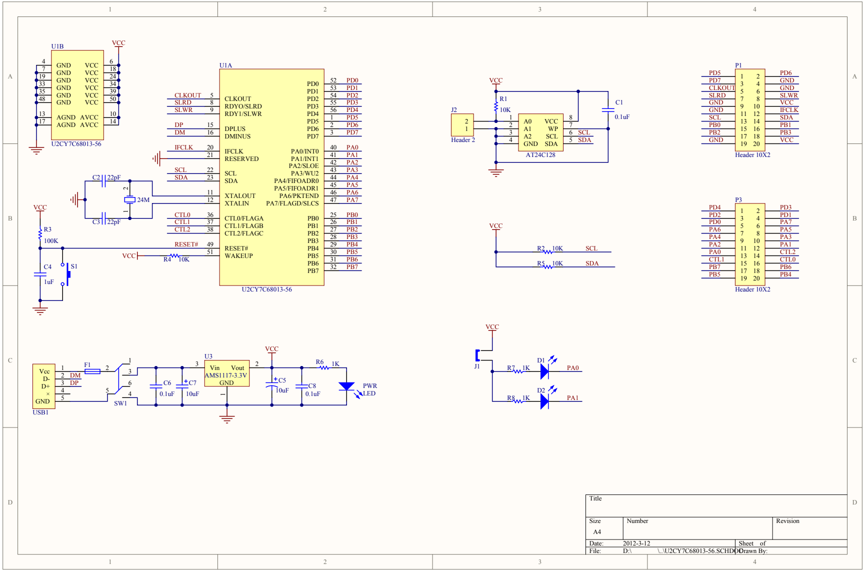
[5] See `https://navisp.esa.int/uploads/files/documents/GNSSW-MLMSC_FINAL_PRESENTATION_E12-022.pdf` "GNSS SW receiver for microlaunchers & microsatellites" (2021) for MAX2771+Zynq7030 for dual-band decoding

[6] `https://github.com/tomojitakasu/PocketSDR`

[7] `https://github.com/jmfriedt/max2771_fx2lp`

Cypress FX2LP EZ-USB [8]

- parallel input or output to USB Bulk stream
- on-board buffers with clock driving parallel data
- EEPROM storing configuration running on the 8051 embedded microcontroller
- 8051 can run custom software, e.g. software emulation of SPI communication
- possible error on PCB silkscreen (RDY1/RDY0)



[8] https://saturn.ffzg.hr/rot13/index.cgi/U2CY7C68013-56.pdf?action=attachments_download;page_name=

Getting familiar with the 8051 on the Cypress FX2LP EZ-USB

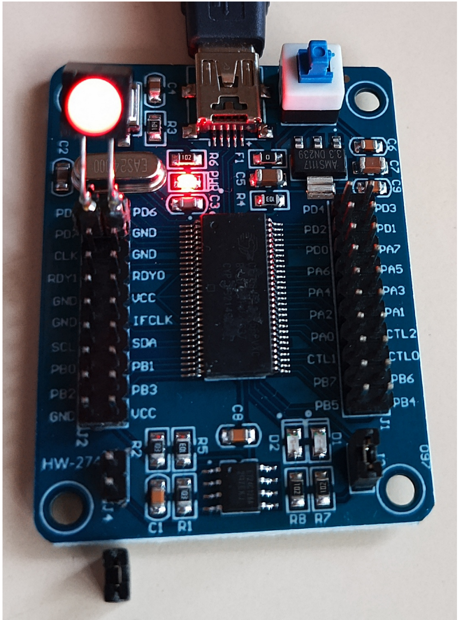
- 8-bit 8051 core is too small for gcc ⇒ use sdcc

SDCC and Keil C compilers: different endianness!

- custom library for managing FX2LP peripherals: FX2LIB at `https://github.com/djmuhlestein/fx2lib`
- programming using `cycfx2prog` or `fxload` (`https://github.com/mbed-ce/fxload`)
- concept of *Vendor Requests* for sharing commands through USB
- use of Python wrapper of `libusb`

```python
import usb
import binascii
VID = 0x04B4
PID = 0x8613
dev = usb.core.find(idVendor = VID, idProduct = PID)
ret=dev.ctrl_transfer(0xC0,0xa9, 0, 0, 32)
    # read EEPROM content
print(binascii.hexlify(ret))
```

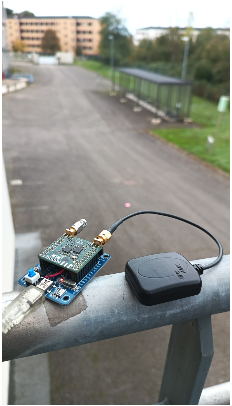
- software implementation of SPI communication protocol
- parallel↔ USB bulk communication (≤48 MB/s) matching MAX2771 maximum frequency of 44 MHz



Objective: better understanding of GNSS with access to raw samples

GNSS receiver with the ability to identify which satellite is broadcasting and solve position

- PocketSDR `python/` scripts
- GNU Radio (pre-processing) and `gnss-sdr` (post and real-time processing)



```
jmfriedt@vivado:~/max2771/PocketSDR$ python3 ./python/pocket_acq.py /tmp/1.bin -f 8 -fi 2 -sig L1CA -prn 1-32
SIG= L1CA, PRN=   1, COFF=  0.08975 ms, DOP=  3028 Hz, C/N0= 33.1 dB-Hz
SIG= L1CA, PRN=   2, COFF=  0.75187 ms, DOP= -2448 Hz, C/N0= 34.4 dB-Hz
SIG= L1CA, PRN=   3, COFF=  0.57038 ms, DOP=  2065 Hz, C/N0= 33.8 dB-Hz
SIG= L1CA, PRN=   4, COFF=  0.25862 ms, DOP=  -465 Hz, C/N0= 33.2 dB-Hz
SIG= L1CA, PRN=   5, COFF=  0.41237 ms, DOP= -4094 Hz, C/N0= 33.4 dB-Hz
SIG= L1CA, PRN=   6, COFF=  0.95925 ms, DOP=  2497 Hz, C/N0= 46.5 dB-Hz
SIG= L1CA, PRN=   7, COFF=  0.22413 ms, DOP=  -586 Hz, C/N0= 33.4 dB-Hz
SIG= L1CA, PRN=   8, COFF=  0.56837 ms, DOP= -1098 Hz, C/N0= 32.8 dB-Hz
SIG= L1CA, PRN=   9, COFF=  0.48737 ms, DOP=  2419 Hz, C/N0= 46.8 dB-Hz
SIG= L1CA, PRN=  10, COFF=  0.67150 ms, DOP=  2892 Hz, C/N0= 33.6 dB-Hz
SIG= L1CA, PRN=  11, COFF=  0.23037 ms, DOP=  4485 Hz, C/N0= 35.6 dB-Hz
SIG= L1CA, PRN=  12, COFF=  0.04938 ms, DOP=  1976 Hz, C/N0= 33.0 dB-Hz
SIG= L1CA, PRN=  13, COFF=  0.38587 ms, DOP=  2485 Hz, C/N0= 33.7 dB-Hz
SIG= L1CA, PRN=  14, COFF=  0.94000 ms, DOP= -5000 Hz, C/N0= 32.8 dB-Hz
SIG= L1CA, PRN=  15, COFF=  0.06650 ms, DOP=  -968 Hz, C/N0= 32.9 dB-Hz
SIG= L1CA, PRN=  16, COFF=  0.33550 ms, DOP=   559 Hz, C/N0= 33.6 dB-Hz
SIG= L1CA, PRN=  17, COFF=  0.73338 ms, DOP= -1888 Hz, C/N0= 40.6 dB-Hz
SIG= L1CA, PRN=  18, COFF=  0.10738 ms, DOP=  3964 Hz, C/N0= 33.5 dB-Hz
SIG= L1CA, PRN=  19, COFF=  0.66612 ms, DOP= -1221 Hz, C/N0= 46.1 dB-Hz
SIG= L1CA, PRN=  20, COFF=  0.18750 ms, DOP=  4579 Hz, C/N0= 33.3 dB-Hz
SIG= L1CA, PRN=  21, COFF=  0.64750 ms, DOP=  4567 Hz, C/N0= 33.4 dB-Hz
SIG= L1CA, PRN=  22, COFF=  0.55613 ms, DOP=  -523 Hz, C/N0= 34.2 dB-Hz
SIG= L1CA, PRN=  23, COFF=  0.68538 ms, DOP=  1030 Hz, C/N0= 33.5 dB-Hz
SIG= L1CA, PRN=  24, COFF=  0.75125 ms, DOP=  -552 Hz, C/N0= 32.9 dB-Hz
SIG= L1CA, PRN=  25, COFF=  0.45250 ms, DOP=  1380 Hz, C/N0= 33.6 dB-Hz
SIG= L1CA, PRN=  26, COFF=  0.03888 ms, DOP= -1944 Hz, C/N0= 33.8 dB-Hz
SIG= L1CA, PRN=  27, COFF=  0.70350 ms, DOP=  2997 Hz, C/N0= 33.5 dB-Hz
SIG= L1CA, PRN=  28, COFF=  0.01250 ms, DOP=  4586 Hz, C/N0= 33.0 dB-Hz
SIG= L1CA, PRN=  29, COFF=  0.87350 ms, DOP=   379 Hz, C/N0= 32.9 dB-Hz
SIG= L1CA, PRN=  30, COFF=  0.60412 ms, DOP=  2394 Hz, C/N0= 33.8 dB-Hz
SIG= L1CA, PRN=  31, COFF=  0.30588 ms, DOP= -1452 Hz, C/N0= 41.1 dB-Hz
SIG= L1CA, PRN=  32, COFF=  0.22388 ms, DOP= -4074 Hz, C/N0= 33.9 dB-Hz
TIME = 3.468 s
```

Lat:	47.2516567°	**Time:**	10:37:35
Long:	5.9929817°	**TTFF:**	9 sec
Alt:	356.8 m	**E H/V Acc:**	1.5/14.3 m
Alt (MSL):	308.8 m	**# Sats:**	21/25/32
Speed:	0.0 m/s	**Bearing:**	103.9°
S. Acc:	0.5 m/s	**B. Acc:**	179.9°
PDOP:	0.9	**H/V DOP:**	0.6/0.6

ID	GNSS	CF	C/N0	Flags	Elev	Azim
2		L1		A	5°	153°
3		L1	32.8	AEU	42°	89°
4		L1	33.3	A U	73°	66°
6		L1	29.4	A U	57°	296°
7		L1	22.9	A U	15°	172°
9		L1	28.5	AEU	69°	226°
11		L1	37.4	AEU	15°	315°
17		L1	43.8	AEU	18°	226°
19		L1	36.5	AEU	28°	250°
26		L1		A	2°	62°
31		L1	34.5	AEU	19°	43°
4		L1	16.5	U	5°	59°
5		L1	25.5	E	50°	40°

Getting started: pulse compression SNR gain

- GNSS satellites are broadcasting 50 W (47 dBm) with 13 dBi antenna gain from 20000 km away
- Friis energy conservation: $\text{FSPL} = 20\log_{10}(d^2) + 20\log_{10}(f^2) - \underbrace{147.55}_{20\log_{10}(c/4\pi)} = 182$ dB
- receiving active antenna with 35 dB gain
- thermal noise floor$=-174$ dBm/Hz$+10\log_{10}(5\cdot 10^6) = -107$ dBm
- GNSS is $(\underbrace{47}_{TX_{pow}} + \underbrace{13}_{TX_{gain}} + \underbrace{35}_{RX_{gain}} - \underbrace{182}_{FSPL}) + 107 = 20$ dB **below thermal** noise
- need to average out ... and yet keep timing capability
- a waveform with bandwith B accumulates energy during correlation to a peak with width $1/B$
- a waveform with duration T averages out noise with sliding window of duration T
- Pulse Compression Ratio $B \times T$=SNR improvement during correlation $\int s(t)\cdot p^*(t+\tau)dt$
- N-bit long PRN sequence at B bit/s lasts $T = N/B$

$$\boxed{\Rightarrow PCR = N = 30 \text{ dB gain with GPS L1 C/A } (N = 1023)}$$

$\Rightarrow$ cross-correlation of signal with **known** PRN code sequence rises to $SNR = 10$ dB

Checking if we got a signal

Codeless decoding: received signal is Doppler shifted by δf and phase modulated by pattern $\varphi(t)$:

$$s(t) = \exp\left(j2\pi\delta f \cdot t + j\varphi(t)\right) + \underbrace{n(t)}_{noise}$$

- Autocorrelation: $\int s(t)s^*(t+\tau)dt =$ $\int \exp(\cancel{j2\pi\delta f \cdot t} + j\varphi(t)) \cdot \exp(-\cancel{j2\pi\delta f \cdot (t} + \tau) - j\varphi(t+\tau))dt$ leaving $\exp(j2\pi\delta f\tau)\int \exp(j\varphi)\exp(-j\varphi(t+\tau))dt =$ $\underbrace{\exp(j2\pi\delta f\tau)}_{|\cdot|=1} xcorr(\varphi,\varphi)$

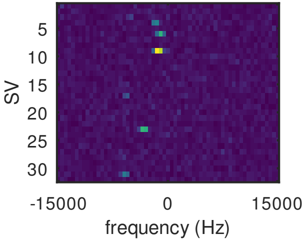
```
f=fopen(myfile); fs=1.023e6; freq0=[-1.5e4:500:1.5e4];
d=fread(f,fs*4,'int8'); d=d(1:2:end)+j*d(2:2:end);fclose(f);
plot([-length(d)+1:length(d)-1],abs(xcorr(d,d)))
```

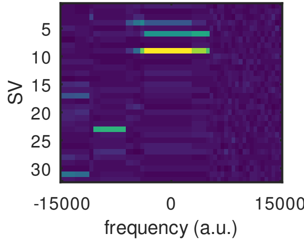
- Squaring the I+jQ signal: $s(t) = \exp(j2\pi\delta f \cdot t + j\underbrace{\varphi}_{\in[0;\pi]})$

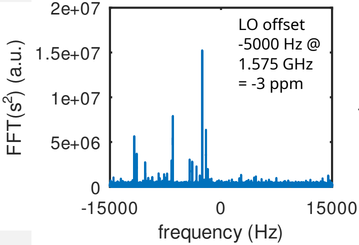
  $\Rightarrow s^2(t) = \exp(j2\pi \cdot 2\delta f \cdot t + j\underbrace{\cancel{2\varphi}}_{0[2\pi]})$

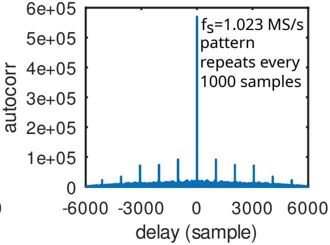
  $s^2(t) = \exp(j2\pi 2\delta f \cdot t)$ clean carrier at twice the frequency offset **but** squared noise

```
f=linspace(-fs/2,fs/2-fs/length(d),length(d));
p=find((f>min(freq0))&(f<max(freq0))); f=f(p);
df=abs(fftshift(fft(d.^2))); df=df(p); plot(f,df)
```

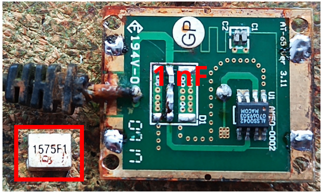








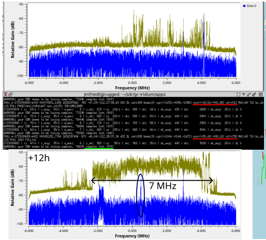
Sidenote exploration: Iridium reception

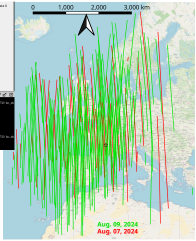


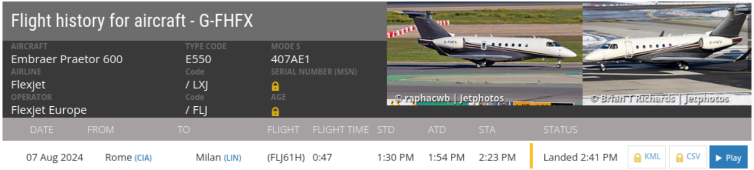
- LEO (780 km) satellite constellation broadcasting in the upper L-band,
  J. Bloom, *Eccentric Orbits: The Iridium Story – How a Single Man Saved the World's Largest Satellite Constellation From Fiery Destruction*, Grove Press (1998)
- signal well above thermal noise …
- … but (BPSK/QPSK) does not benefit from correlation to increase number of bits
- GPS L1 active patch antenna whose bandpass filter was replaced with a capacitor
- Fractional PLL of MAX2771: $RDIV \in [0:1023]$, $NDIV \leq 546$ ($f_{LO} = 1638 > 1622$ MHz) and $FDIV \in [36-32767]$,

$$\text{for } f_{LO} = \frac{f_{Xtal}}{RDIV} \times \left(NDIV + \frac{FDIV}{2^{20}}\right) \text{ with } f_{Xtal} = 24 \text{ MHz}$$

Settings: $f_{IF} = 6.5$ MHz, $f_s = 24$ MS/s ; MAX2771 spectrum around 1622 MHz ⟶







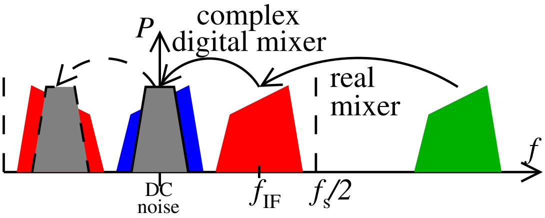

```
2024-08-07T14:02:03 [hdr: 033901010000001] Dir:DL Mode:2 REG:F-GXLI ACK:7 Label:_? (Demand mode) bID:F
2024-08-07T14:42:12                        Dir:DL Mode:2 REG:GFHFX  ACK:8 Label:_? (Demand mode) bID:Z
```

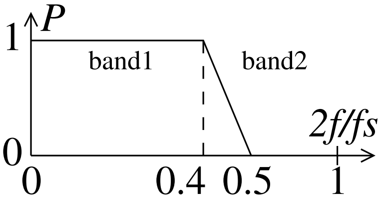
Result: ACARS message [9] from a plane between Rome and Milan (Italy), beyond the horizon from Besançon (France)

[9] https://thebaldgeek.github.io/Iridium.html

Real time GNSS reception using GNU Radio and gnss-sdr

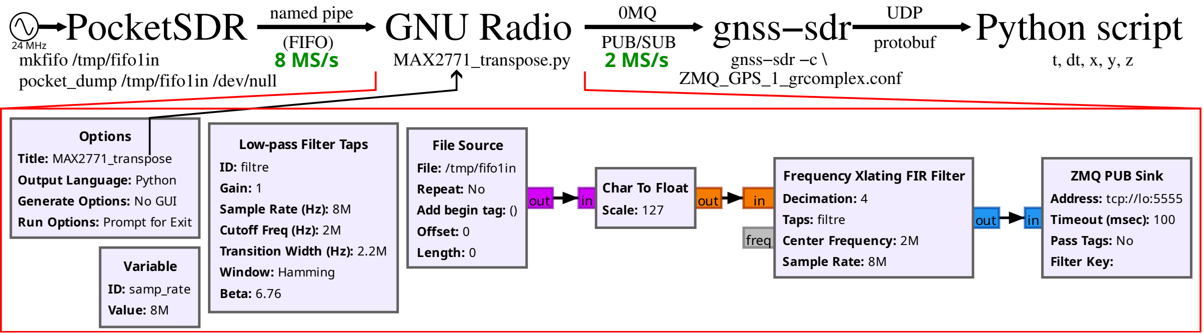
- Excessive noise close to DC ⇒ use an IF-band and software transposition
- Frequency transposition using GNU Radio Xlating FIR Filter and ZeroMQ publish ...
- or gnss-sdr Xlating FIR Filter capability.






```
SignalConditioner.implementation=Signal_Conditioner
DataTypeAdapter.implementation=Byte_To_Short
InputFilter.implementation=Freq_Xlating_Fir_Filter
InputFilter.input_item_type=short
InputFilter.output_item_type=gr_complex
InputFilter.taps_item_type=float
InputFilter.number_of_taps=5
InputFilter.number_of_bands=2
InputFilter.band1_begin=0.0
InputFilter.band1_end=0.40
InputFilter.band2_begin=0.50
InputFilter.band2_end=1.0
InputFilter.ampl1_begin=1.0
InputFilter.ampl1_end=1.0
InputFilter.ampl2_begin=0.0
InputFilter.ampl2_end=0.0
InputFilter.band1_error=1.0
InputFilter.band2_error=1.0
InputFilter.filter_type=bandpass
InputFilter.grid_density=16
InputFilter.sampling_frequency=8000000
InputFilter.IF=2000000
```

IF compensation using GNU Radio and ZMQ Pub




```
GNSS-SDR.internal_fs_sps=2000000
SignalSource.implementation=ZMQ_Signal_Source
SignalSource.endpoint=tcp://127.0.0.1:5555
SignalSource.sample_type=gr_complex
SignalSource.sampling_frequency=2000000
SignalConditioner.implementation=Pass_Through
```

- PocketSDR → GNU Radio using a named pipe and File Source
- GNU Radio → GNSS-SDR using ZMQ Publish
- GNSS-SDR Signal Source = `ZMQ_Signal_Source` receiving complex float.

Real time GPS L1 reception using PocketSDR and gnss-sdr [10]

```
Tracking of GPS L1 C/A signal started on channel 0 for satellite GPS PRN 01 (Block IIF)
Current receiver time: 1 min 49 s
New GPS NAV message received in channel 9: subframe 1 from satellite GPS PRN 21 (Block IIR) with CN0=42 dB-Hz
New GPS NAV message received in channel 5: subframe 1 from satellite GPS PRN 02 (Block IIR) with CN0=43 dB-Hz
New GPS NAV message received in channel 6: subframe 1 from satellite GPS PRN 08 (Block IIF) with CN0=44 dB-Hz
New GPS NAV message received in channel 4: subframe 1 from satellite GPS PRN 32 (Block IIF) with CN0=43 dB-Hz
First position fix at 2024-Jul-26 09:31:48.120000 UTC is Lat = 47 [deg], Long = 6 [deg], Height= 3.8e+02 [m]
Current receiver time: 1 min 50 s
The RINEX Navigation file header has been updated with UTC and IONO info.
Position at 2024-Jul-26 09:31:49.000000 UTC using 4 observations is Lat = 47.251620 [deg], Long = 5.993221 [deg],
Height = 366.06 [m]
Velocity: East: 0.91 [m/s], North: 0.65 [m/s], Up = 3.82 [m/s]
Current receiver time: 1 min 51 s
Loss of lock in channel 11!
Tracking of GPS L1 C/A signal started on channel 11 for satellite GPS PRN 19 (Block IIR)
Position at 2024-Jul-26 09:31:49.989988 UTC using 4 observations is Lat = 47.251560 [deg], Long = 5.993090 [deg],
Height = 311.77 [m]
Velocity: East: -0.83 [m/s], North: -1.32 [m/s], Up = -2.92 [m/s]
```

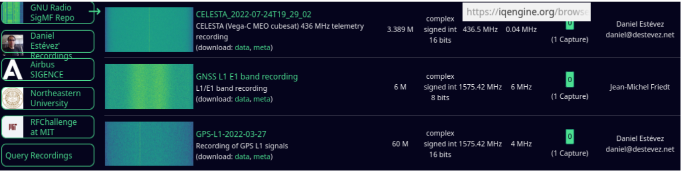
GNSS-SDR state machine

1. **Tracking**: a SV has been associated with a channel and signal is searched
2. **GPS L1 C/A tracking bit synchronization locked**: signal found!
3. **New GPS NAV message received**: navigation message, towards PVT solution

[10] movie of the reception sequence at `https://www.youtube.com/watch?v=B5UcFnkbXIk`

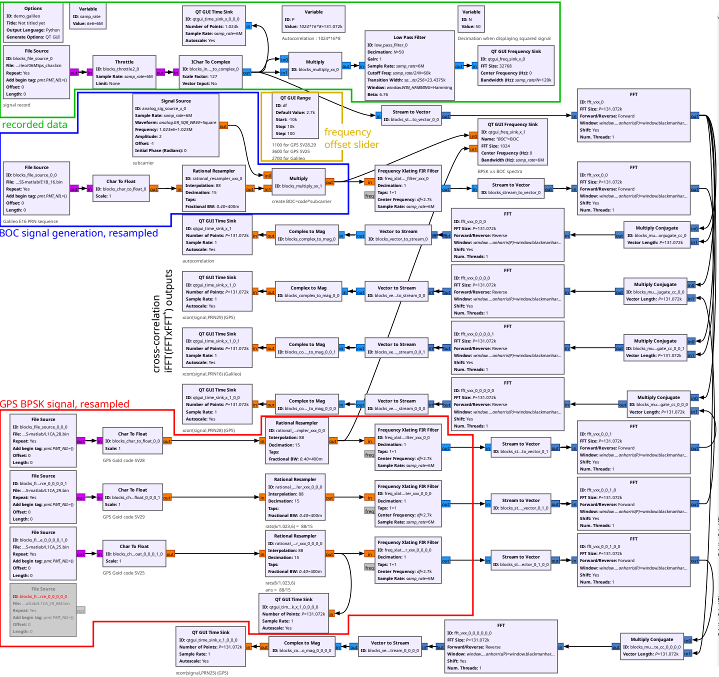
Remembering how datasets were recorded: sigMF

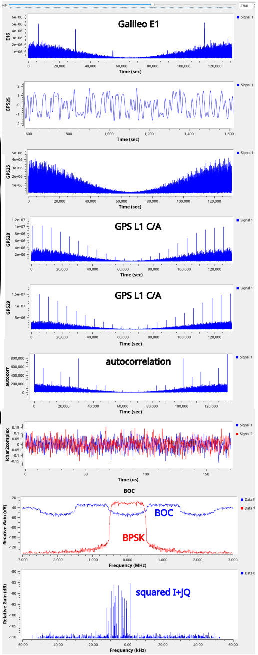
- Recording **binary** files (efficient storage) but how were samples stored? Integer v.s float, real or complex interleaved? sampling rate? Endianness?
- SigMF meta format: `https://github.com/sigmf/SigMF`
- Detailed description at `https://sigmf.org/index.html`
- Each dataset `sigmf-data` is associated with a human-readable format description `sigmf-meta`



- See datasets at `iqengine.org`

```
{
 "global": {
  "antenna:gain": 35,
  "antenna:type": "none",
  "core:version": "1.0.0",
  "core:datatype": "ci8",
  "core:description": "L1/E1 band recording",
  "core:sample_rate": 6E6,
  "core:author": "Jean-Michel Friedt",
  "core:recorder": "GNU Radio",
  "core:hw": "Ettus Research B210",
  "core:license": "CC BY-SA"
  },
  "captures": [
    {
     "core:sample_start": 0,
     "core:frequency": 1575.42E6
    }
  ],
  "annotations": []
}
```





Top to bottom: byte-wide recorded signal (B210)

x-correlation Galileo PRN

interpolated BPSK

cross-correlation GPS PRN (incorrect frequency shift)

cross-correlation GPS PRN

cross-correlation GPS PRN

auto-correlation

signal (± 8 or 3 bits)

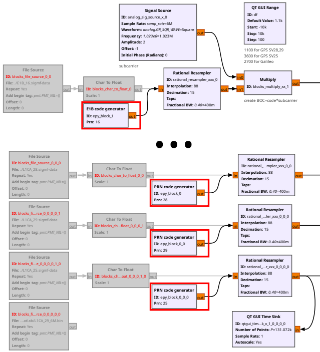
BPSK v.s BOC spectra

squared signal (pure carriers)

Custom Python block for GPS/Galileo PRN generation

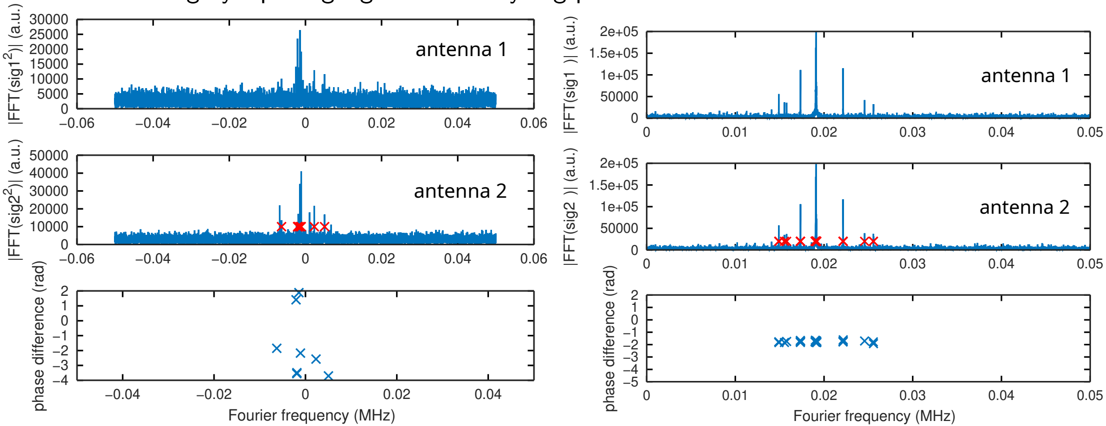
- Marc Lichtman (IQEngine): rather than storing PRN sequence, generate on the fly.
- Python PRN generators found at `https://github.com/pmonta/GNSS-DSP-tools.git` in `gnsstools/gps` and `gnsstools/galileo`
- Copy-paste in GNU Radio `Python Block` with `in_sig=[], out_sig=[np.float32]`
- The scheduler requested number of items is unknown ⇒ fill with `intval` integer copies of the PRN and then circular buffer with the PRN sequence: `self.x=self.x[fracval:]+self.x[:fracval]` the fractional part of the length of the code returned in the output buffer

```
...
def e1b_code(prn):
  if prn not in codes:
    codes[prn] = e1b_parse_hex(prn)
  return codes[prn]

boc11 = np.array([1.0,-1.0])
...
class blk(gr.sync_block):  # other base classes are basic_block, decim_block, interp_block
    def __init__(self, PRN=16):  # only default arguments here
        gr.sync_block.__init__(
            self, name='E1B code generator',
            in_sig=[], out_sig=[np.float32] # no input, only output
        )
        self.PRN = PRN
        self.x=list(1.-2.*e1b_code(self.PRN))
        self.xpos=0

    def work(self, input_items, output_items):
        intval=len(self.x)*(len(output_items[0])//len(self.x))  # integer number of copies of PRN code
        fracval=len(output_items[0])-intval                      # fractional length of the PRN code
        output_items[0][0:intval] = self.x*(len(output_items[0])//len(self.x)) # *list = copy
        output_items[0][intval:intval+fracval] = self.x[0:fracval]
        self.x=self.x[fracval:]+self.x[:fracval]                 # rotate x
        return len(output_items[0])
```



GNSS spoofing detection

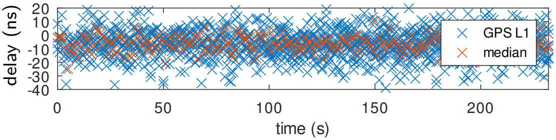
- Use direction of arrival to identify inconsistent beam pattern
- Satellite distributed over many azimuth/elevation while unique spoofer will exhibit unique phase difference between antennas for all satellites
- Avoid decoding by squaring signal and analyzing phase of visible tones

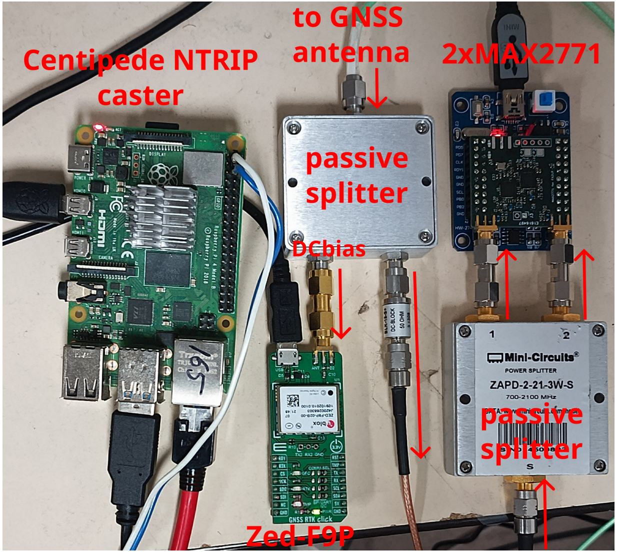


- P receivers can cancel $P-1$ jamming/spoofing sources by null steering, but dynamic range limited here to few bits (6.02 dB/bit)

Time transfer using MAX2771s: pseudorange differences to time

- Collect data from dual MAX2771, either both tuned to E1 or E1/E5
```
pocket_conf conf/pocket_L5L1_8MHz_4MHz.conf
pocket_dump -t 300 1.bin 2.bin
```
- Record NTRIP observation from Zed-F9P using RTKLib:
```
str2str -in ntrip://caster.centipede.fr:2101/ENSMM \
  -out file://file.rtcm3
```
- Generate RINEX files with observation pseudo-ranges (gnss-sdr): `gnss-sdr -c File_Galileo_E1_char.conf`
- Generate RINEX from recorded NTRIP records (RTKLib)
```
convbin file.rtcm3 # extension is important!
```
- Subtract RINEX pseudoranges using rinex-cli[11]
```
rinex-cli --fp E1.250 diff ublox.250
```
- Convert RINEX subtraction to CSV (median value):
```
rinex-cli --fp WORKSPACE/E1/DIFFERENCED.250 \
  filegen --csv
```





[11] rinex-cli is part of GeoRust, examples at https://github.com/georust/rinex/tree/main/tutorials/DIFF

Time transfer using MAX2771s: pseudorange differences to time

- Collect data from dual MAX2771, either both tuned to E1 or E1/E5

```
pocket_conf conf/pocket_L5L1_8MHz_4MHz.conf
pocket_dump -t 300 1.bin 2.bin
```

- Record NTRIP observation from Zed-F9P using RTKLib:

```
str2str -in ntrip://caster.centipede.fr:2101/ENSMM \
  -out file://file.rtcm3
```

- Generate RINEX files with observation pseudo-ranges (gnss-sdr): `gnss-sdr -c File_Galileo_E1_char.conf`
- Generate RINEX from recorded NTRIP records (RTKLib)

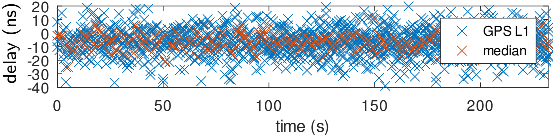
```
convbin file.rtcm3 # extension is important!
```

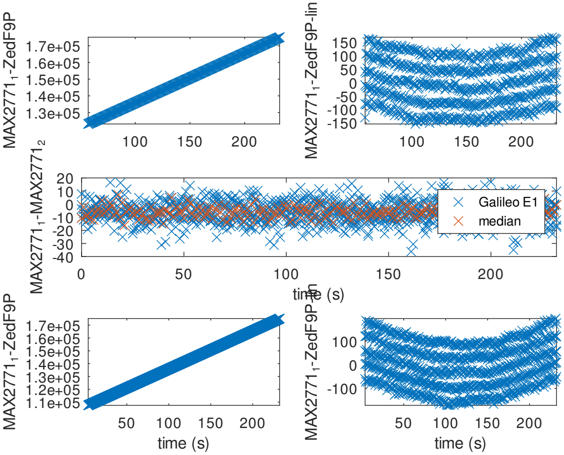
- Subtract RINEX pseudoranges using rinex-cli[11]

```
rinex-cli --fp E1.250 diff ublox.250
```

- Convert RINEX subtraction to CSV (median value):

```
rinex-cli --fp WORKSPACE/E1/DIFFERENCED.250 \
  filegen --csv
```





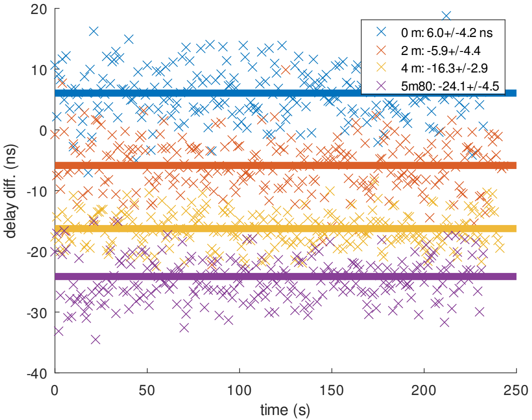
Results (8MS/s, 4 MHz IF): (6 ns $\ll$ 1/8 MHz)

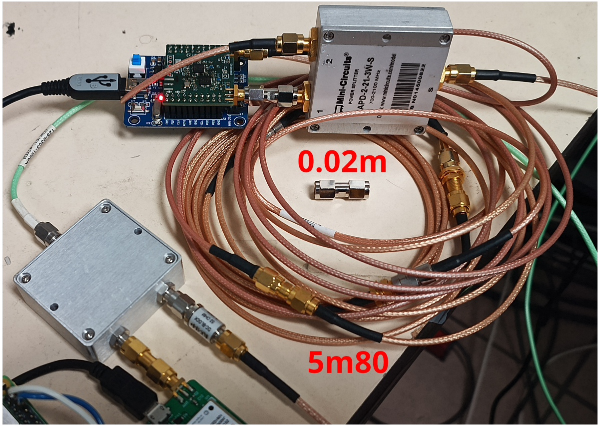
Constellation	$\langle\cdot\rangle$ (ns)	σ (ns)
Galileo E1	6.0	4.2
GPS L1	6.2	5.6

[11] rinex-cli is part of GeoRust, examples at https://github.com/georust/rinex/tree/main/tutorials/DIFF

Time transfer over GNSS

Change cable length of one receiver with respect to the other with 2, 4 and 5.8 m





20 cm/ns in a coaxial cable $\Rightarrow$ 2 m in 10 ns, 4 m in 20 ns and 5.8 m in 29 ns

Crosses = measurements, thick solid line = mean value

Conclusion (see `https://github.com/jmfriedt/max2771_fx2lp`) [12]

- PocketSDR update using exclusively opensource tools (`sdcc`)
- Better understanding of time transfer over satellite links [a]
- Opportuiniy to grasp low SNR signal acquisition/low bit count ADC ⟶
- **Perspective**: add clock steering over SPI (e.g. AD9851) for SDR-GPSDO – has demonstrated PocketSDR→GNU Radio →gnss-sdr running for >1 week using `gnss-sdr`'s UDP PVT monitor for steering the clock driving the MAX2771 [b]

Project cost estimate:

Item	Supplier/Volume	Cost
MAX2771	Mouser/6	9.96/p
FX2LP board	Aliexpress/1	5/p
passive/connectors	Mouser	<5
30.1 mm×34.6 mm, 4-layer FR4	Eurocircuits/25	8/p
AS-ANT2B-OEM-L1L2-02SMA-00	Mouser/1	75
Total		100 €

[a] K. Borre & *al.*, *GNSS Software Receivers*, Cambridge University Press (2023)

[b] github.com/acebrianjuan/gnss-sdr-pvt-monitoring-client

A basic parameter of a recording system is its data rate, ν_b (bits s^{-1}). This parameter limits the number of bits that can be recorded in a given time and, thus, also the sensitivity of continuum observations in which the potential IF bandwidth is larger than $\nu_b/2N_b$, where N_b is the number of bits per sample. The signal is represented by samples having Q quantization levels taken at β times the Nyquist rate. For N samples, there are Q^N possible data configurations, which require a minimum of $N \log_2 Q$ bits. Therefore, as noted in Sect. 8.4.3, the maximum RF bandwidth is

$$\Delta\nu = \frac{\nu_b}{2\beta N_b} = \frac{\nu_b}{2\beta \log_2 Q} . \qquad (9.164)$$

The signal-to-noise ratio obtained in time τ is proportional to $\eta_Q\sqrt{\Delta\nu\tau}$, where η_Q is the quantization efficiency (see Table 8.3). From Eq. (9.164),

$$\eta_Q\sqrt{\Delta\nu\tau} = \eta_Q\sqrt{\frac{\nu_b\tau}{2\beta N_b}} . \qquad (9.165)$$

If τ is the recording time, $\nu_b\tau$ is equal to the number of recorded bits. The quantity $\eta_Q/\sqrt{\beta N_b}$ thus provides an indication of the performance per bit, which it is desirable to maximize. For two- and four-level sampling, the obvious encoding

A.R. Thompson *& al.*, *Interferometry and Synthesis in Radio Astronomy, Third Edition*, Springer Open (2017)

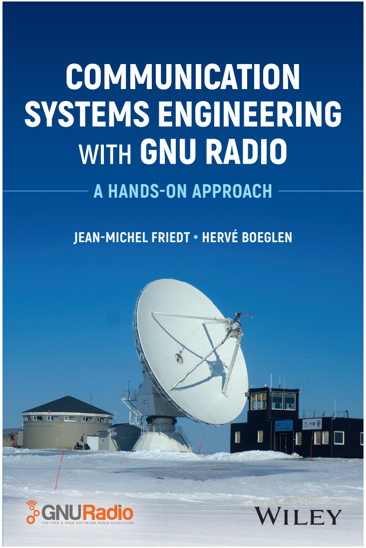
[12] J.-M Friedt, *Programmation USB sous GNU/Linux : application du FX2LP pour un récepteur de radio logicielle dédié aux* signaux de navigation par satellite (1 & 2/2), Hackable (2024/2025) translated to English on FOSDEM web page

Conclusion (see https://github.com/jmfriedt/max2771_fx2lp) [12]

- PocketSDR update using exclusively opensource tools (`sdcc`)
- Better understanding of time transfer over satellite links [a]
- Opportuiniy to grasp low SNR signal acquisition/low bit count ADC ⟶
- **Perspective**: add clock steering over SPI (e.g. AD9851) for SDR-GPSDO – has demonstrated PocketSDR→GNU Radio →gnss-sdr running for >1 week using `gnss-sdr`'s UDP PVT monitor for steering the clock driving the MAX2771 [b]

Project cost estimate:

Item	Supplier/Volume	Cost
MAX2771	Mouser/6	9.96/p
FX2LP board	Aliexpress/1	5/p
passive/connectors	Mouser	<5
30.1 mm×34.6 mm, 4-layer FR4	Eurocircuits/25	8/p
AS-ANT2B-OEM-L1L2-02SMA-00	Mouser/1	75
Total		100 €



[a] K. Borre & *al.*, *GNSS Software Receivers*, Cambridge University Press (2023)

[b] github.com/acebrianjuan/gnss-sdr-pvt-monitoring-client

[12] J.-M Friedt, *Programmation USB sous GNU/Linux : application du FX2LP pour un récepteur de radio logicielle dédié aux* signaux de navigation par satellite (1 & 2/2), Hackable (2024/2025) translated to English on FOSDEM web page