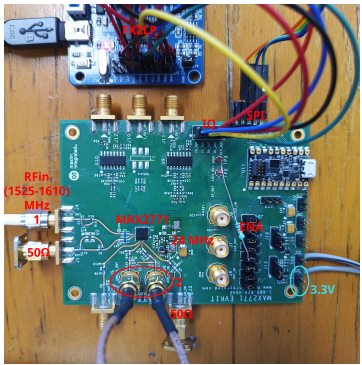
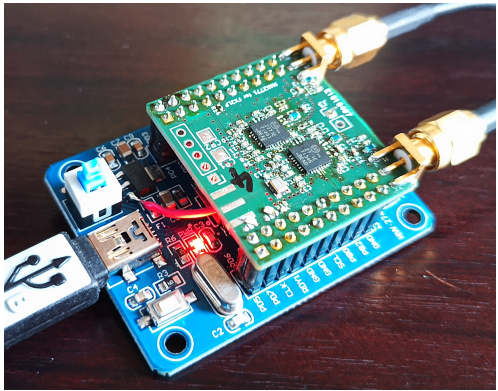
Efficient USB communication under GNU/Linux for a wideband L-band (GNSS) SDR receiver: positioning solution

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

From 

to 

April 23, 2025

From acquisition to tracking

- Cold start: need to identify which satellite is visible with which frequency offset (**acquisition**)
- Once satellites are identified, **tracking** of frequency offset and delay in closed control loops
- Phase **tracking** using `atan(Q/I)` π-insensitive...
- ... and BPSK bit identification using `atan2(Q,I)` (using the sign of I and Q to be π sensitive)
- Existing implementations: `gnss-sdr`, PocketSDR and RTKLib tools (T. Takasu and demo5),

Output: Position, Velocity and Time (PVT), possibly in standard formats (RINEX, RTCM)

```
Current receiver time: 13 s
GPS L1 C/A tracking bit synchronization locked in channel 8 for satellite GPS PRN 20 (Block IIR)
GPS L1 C/A tracking bit synchronization locked in channel 1 for satellite GPS PRN 13 (Block IIR)
...
Current receiver time: 22 s
New GPS NAV message received in channel 8: subframe 2 from satellite GPS PRN 20 (Block IIR) with CN0=45 dB-Hz
New GPS NAV message received in channel 1: subframe 2 from satellite GPS PRN 13 (Block IIR) with CN0=44 dB-Hz
...
New GPS NAV message received in channel 1: subframe 1 from satellite GPS PRN 13 (Block IIR) with CN0=44 dB-Hz
GPS L1 C/A tracking bit synchronization locked in channel 10 for satellite GPS PRN 07 (Block IIR-M)
First position fix at 2024-Jul-22 17:57:18.100000 UTC is Lat = 47.2517 [deg], Long = 5.99328 [deg], Height= 364.788 [m]
Current receiver time: 1 min 17 s
Position at 2024-Jul-22 17:57:19.000000 UTC using 4 observations is Lat = 47.251622 [deg], Long = 5.993225 [deg],
Height = 361.46 [m]
Velocity: East: 0.32 [m/s], North: -0.04 [m/s], Up = -0.05 [m/s]
...
```

gnss-sdr

- clone (`git`) and compile (`mkdir build && cd build && cmake ../ && make`) to generate `build/src/gnss-sdr` executable,
- use one of the configuration files in `250103_8MSps_4MHzIF/2MHzIF` to process a file recorded from one of the MAX2771 output
  - tune all references in the configuration file to the sampling frequency
  - tune the record file name and data storage format (`byte`, `short`, prefixed with `i` of interleaved real/imaginary[1])
  - check the `gnss-sdr` online manual and the `conf/` configuration files to decode various constellations in a given frequency band, depending on sampling rate
  - if an IF was introduced, `gnss-sdr` relies of GNU Radio's `Xlating FIR Filter` to bring the signal to baseband
- decode the current position and time
- notice the newly (≥1 April 2025) added `MAX2771_EVKIT_Signal_Source_FPGA` signal source[2] (untested) – requires SPIdev support on the (unidentified) Linux platform collecting samples.

[1] `https://gnss-sdr.org/docs/sp-blocks/signal-source/`
[2] `https://gnss-sdr.org/gnss-sdr-v0020-released/`

gnss-sdr's Monitor tool

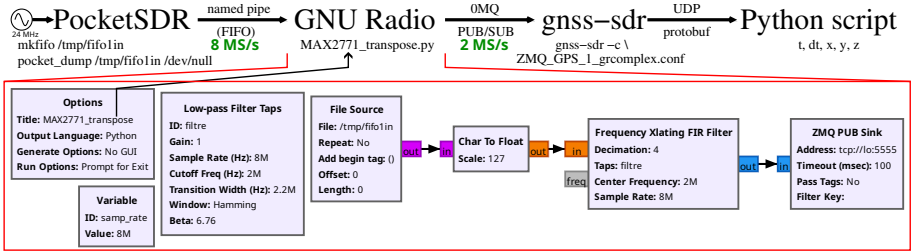
- GNSS-SDR allows for probing all Observables (pseudo-ranges) and PVT (Position, Velocity and Time) solver internal states: the Monitor[3] capability
- see `https://github.com/acebrianjuan/gnss-sdr-pvt-monitoring-client` for an example of UDP client monitoring all variables...
- or consider ProtoBuf configuration[4] files provided with `gnss-sdr` source codes for generating a custom Python client.

[3] `https://gnss-sdr.org/docs/sp-blocks/monitor/`

[4] see the content of `gnss-sdr/docs/protobuf/`

gnss-sdr real time capability

- gnss-sdr can accept signal streamed over a named pipe (FIFO) with the `Fifo_Signal_Source`
- ... or transferred through a 0-MQ socket (`ZMQ_Signal_Source`).
- An intermediate GNU Radio flowchart might handle the `pocket_dump` output to feed gnss-sdr with the right data format
- Check processing capability for real time decoding. So far, I have only been able to real-time process GPS L1 C/A.



PocketSDR tools

- In addition to Tomoji Takasu's `pocket_acq` for identifying constellations, `pocket_trk` can provide a solution
- Consider how enhancements can be brought by merging solutions (e.g. NTRIP caster broadcasting RTCM pseudo-ranges for real-time correction) from different receivers
- See ESA's *GNSS Data Processing Vol. 1* at [5], pages 140– (Eq. 6.6) on how to linearize range equations and iteratively identify the position and time offset solution from the known satellite positions and pseudo-ranges.

[5] `https://gssc.esa.int/navipedia/GNSS_Book/ESA_GNSS-Book_TM-23_Vol_I.pdf`

RTKLib

- RTK: Real Time Kinematic to merge multi-receiver solutions and correct for ionospheric and tropospheric delay
- Low-quality GNSS receiver version: `https://github.com/rtklibexplorer/RTKLIB` described in the online-articles at `https://rtklibexplorer.wordpress.com/`
- `gnss-sdr` can generate an RTCM stream processed with RTKLib (even though `gnss-sdr` is using RTKLib to generate its own PVT solution, see `gnss-sdr/src/algorithms/libs/rtklib/` used in `gnss-sdr/src/algorithms/PVT/`.