Software Defined Radio for Time and Frequency applications: example of passive monitoring of TWSTFT and other timing signals

(Invited Paper)

J.-M Friedt*
*FEMTO-ST Time & Frequency, Besançon, France. Email: jmfriedt@femto-st.fr

***Abstract*—Software Defined Radio (SDR) is a paradigm aimed at minimizing the contribution of hardware and maximizing the contribution of software when processing radiofrequency signals. While the concept is mostly used in the context of digital communication for agile systems flexible enough to adapt to unexpected conditions, accessing the raw samples right after the digitization of the radiofrequency wave provides opportunities for benefiting from information on the physical system before the signal has been processed. Not only does this access to the physical layer provide security, e.g. GNSS anti-spoofing and anti-jamming using null steering, but also the opportunity to finely recover timestamps in the context of time and frequency transfer. Free, opensource software is readily distributed for reproduction of the experiment at the Oscillator Instability Measurement Platform (OscIMP) Digital repository at `https://github.com/oscimp/gr-satre` for SATRE TWSTFT signal reception or `https://github.com/oscimp/gnss-sdr-1pps/` for GNSS-controlled 1 PPS generation, and repetition of the measurements at remote sites, a core requirement of a scientific endeavor.**

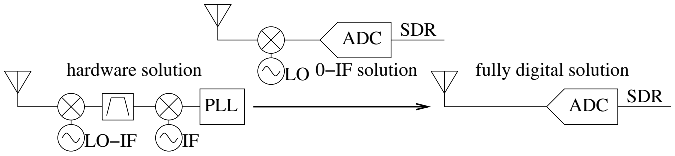


Fig. 1. Left: the hardware implementation of a super-heterodyne radiofrequency receiver, strongly application dependent. Right: the ultimate SDR, with the antenna signal sampled by the analog to digital converter (ADC) for software processing. Middle: the practical solution, with a first frequency transposition from radiofrequency band to baseband prior to ADC conversion.

The SDR paradigm aims at reducing hardware to a minimum which is mostly comprised of a (complex, real and imaginary) frequency transposition mixer and a fast analog to digital converter (Fig. 1), moving all processing to software. The software provides long term stability (a digital algorithm does not drift), reconfigurability for using the same hardware for multiple applications, and tunability [1]. Furthermore, software implementation provides logging and remote control. The availability of huge non-volatile storage capacity with large bandwidth allows for post-processing with the analysis of as many channels as needed without constraint on computational power. However, the complex SDR framework including efficient hardware access, high performance computing and relying possibly on heterogeneous processing architectures (GP-CPU, GPU, FPGA) are challenging to maintain on the long term.

I. SDR for one-way reception of Two-Way Satellite Time and Frequency Transfer (TWSTFT) signals

TWSTFT signals are broadcast by time and frequency metrology institutes every even UTC hours to compare their clocks and steer TAI and hence UTC. A geostationary relay satellite, Telstar11N, located at -37.5° over the Atlantic ocean, allows for sharing signals not only between European stations but also with North American stations at Boulder and Washington DC. The Ku band relay satellite uplink is 14 GHz transmitted by the 2.4 m parabola dish of the Very Small Aperture Terminals (VSAT) located at each metrology institute, but the 11 GHz downlink signal is readily received with any consumer grade television satellite receiver parabola reflector (Fig. 2) feeding a Low Now Block (LNB) including frequency transposition by 9.75 GHz and low noise amplifier.

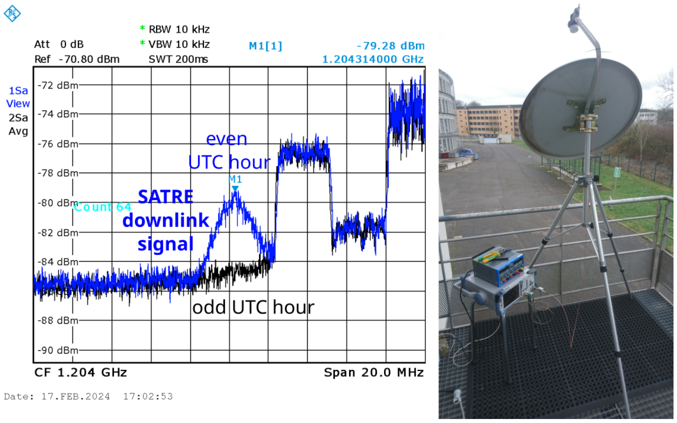


Fig. 2. Right: experimental setup, with a consumer grade satellite TV reception parabola reflector and a Low Noise Block at the focal point feeding a B210 SDR receiver. Left: recorded signal during even UTC hour (blue) when SATRE modems from metrology institutes are broadcasting, and black during odd UTC hours and no communication is occurring.

Hence, a SDR receiver connected to the LNB and tuned to the 11 GHz downlink frequency offset by 9.75 GHz is able to record the signal as documented at `webtai.bipm.org/ftp/pub/tai/data/2024/` in the `time_transfer/twstft` subdirectory. The header of each exchange file published by BIPM includes the downlink frequency, namely 10953.9500 MHz for the European link and 11497.0600 for the North American link. The SDR receiver is hence tuned to 1204 MHz, with a bandwidth of 5 MHz matching the Telstar11N channel width, and the complex IQ datastream recorded to file for post-processing. A major benefit of the SDR approach over the hardware decoding by SATRE modem is the ability to

- record all broadcasting stations at any given time without being limited by a finite number of receiver channels
- run as many computation steps on the records with no limitation on computational capability of the processing unit.

The first item impacts the passive reception capability which requires compensating for satellite motion in space which is not cancelled as would be in a TWSTFT processing. In order to identify the satellite position in space, trilateration is used considering that the broadcast station locations are known, and time of flight differences are computed to identify the satellite position and subtract its contribution from the one-way time of flight measurement. However, when using SATRE records as published by BIPM, stations only exchange two-by-two messages and trilateration requires interpolating the measurements to increase the number of observations during the even hour. In the SDR approach, not such interpolation is needed since all broadcast signals are recorded continuously.

The second item allows processing all downlink signals sequentially without requiring dedicated computational infrastructure such as Graphical Processing Units (GPU) or Field Programmable Gate Arrays (FPGA) co-processors, and allows for prototyping time of flight estimation algorithms and satellite positioning on the recorded data.

As was discussed at [2], publicly available information allows for reverse engineering the SATRE pseudo-random sequences and decode the Binary Phase Shift Keying modulated messages. The satellite position in space is computed following the classical approach used in Global Navigation Satellite System positioning, namely linearizing the range equation from $ct_i = \sqrt{(x_i - x_0)^2 + (y_i - y_0)^2 + (z_i - z_0)^2}$ with (x, y, z) the cartesian coordinates indexed with $_0$ for the satellite and $_i$ for station i, with t_i the time of flight observed for message broadcast by station i and $c = 300$ m/μs the speed of light. Due to the short code length used by SATRE modems, the time of flight difference between broadcasting stations is plagued by a 4 ms uncertainty which is manually inserted considering the known locations of the emitters and the known nominal parking position of the satellite. The Taylor series linearization of the above equation when the satellite position is offset by (dx, dy, dz) from its parking position becomes $ct_i = \frac{x_i - x_0}{R_i}dx + \frac{y_i - y_0}{R_i}dy + \frac{z_i - z_0}{R_i}dz$ with (dx, dy, dz) the unknowns solved by a pseudo-inverse matrix computation since R_i the nominal range from station i to the satellite are known and ct_i are the observations.

When using BIPM published measurements, the result of the satellite position allows for reducing the timing uncertainty from ±30 μs to ±30 ns, from the freely moving satellite to its compensation by identifying the location and subtracting its contribution. This result is achieved by considering that the CH station is not used in computing the satellite position and all other broadcasting station measurements are used for estimating (dx, dy, dz), and this contribution is subtracted from the CH measurement which is also published by BIPM (Fig. 3).

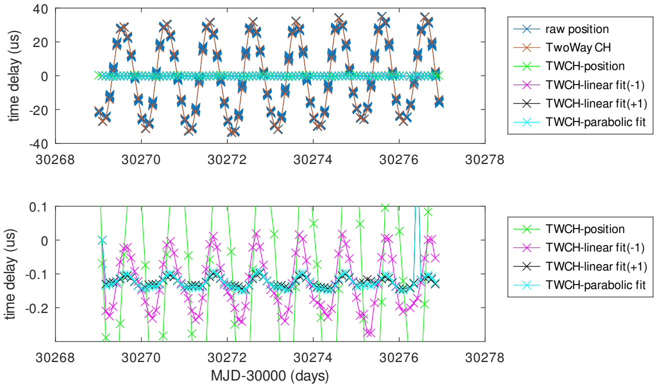


Fig. 3. Impact of various interpolation schemes of the BIPM published datasets. Top the time of flight measurements used for estimating the satellite position, excluding CH. Bottom: time of flight difference between CH observations published by BIPM after subtracting the contribution of the satellite motion in space around its parking position.

In a practical passive measurement setup (Fig. 2), an Ettus Research B210 receiver is connected to the LNB output of a consumer grade satellite TV reception antenna and 5-second long records are collected during even UTC hours and the broadcasting stations are exhibiting best geographical distribution for satellite position computation, including SP in Sweden and ROA in southern Spain. Due to the poorer signal to noise ratio from the smaller parabola reflector, the result is much worse than the one obtained when using BIPM published datasets (Fig. 4), but still demonstrates the feasibility. Most limiting in the current approach, the pseudo-inverse calculation does not include any filtering and insight in the physics of satellite motion (Kalman filtering) nor is any celestial mechanics involved in estimating the satellite position. These developments are in progress.

In order to improve modelling of the satellite kinematic behavior, velocity measurements are desirable. Since neither the satellite bent-pipe transponder transposing the 14-GHz uplink signal to the 11-GHz downlink carrier, nor the LNB local oscillator, are metrological, the carrier frequency is not used for estimating the satellite velocity. Instead, the code drift rate at the output of the correlator is a fine estimate of the satellite motion in the ±5 ns/s range. However, this fine time of flight measurement is dependent on the sampling rate defined by the SDR local oscillator. When collecting data from an analog to digital converter (ADC) clocked by the on-board temperature compensated quart crystal oscillator (TCXO), the frequency offset (2 ppm) and fluctuation over time is clearly visible, preventing the estimate of the satellite velocity. Controlling the ADC with an external hydrogen

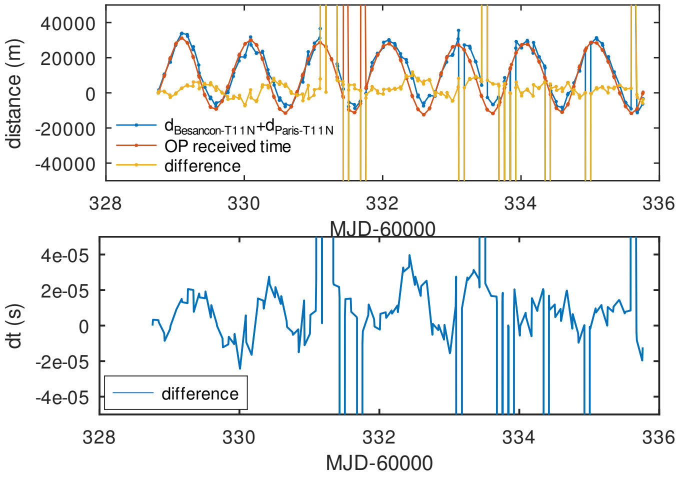


Fig. 4. Passive recording of the SATRE signals using the setup shown in Fig. 2, and resulting time of flight difference between the downlink signals and the estimated satellite motion contribution.

maser generated frequency source allows for recovering the satellite velocity (Fig. 5). On the other hand, this measurement means that the TXCO can be controlled to the ±5 ppb level of the satellite speed in an approach similar to the GNSS disciplined oscillator without compensating for satellite orbital behavior.

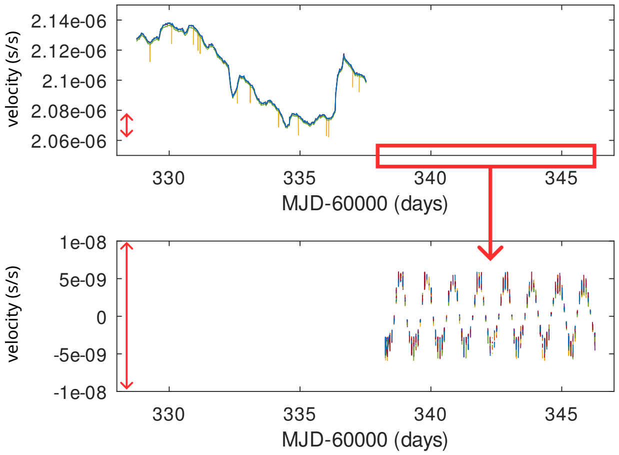


Fig. 5. Impact of the local oscillator on the velocity measurement as estimated with the code drift rate. At first the crystal oscillator (TCXO) on-board the B210 receiver is used, exhibiting a 2.1 ppm frequency offset and long term fluctuations and drift, and during the second half of the measurement the SDR is clocked by a 10 MHz from a hydrogen maser.

Despite the multistatic emission of the broadcast signal allowing for trilateration of the satellite position, estimating the velocity vector (rather than only the projection to the receiver) would require fine velocity estimate since the geometry is unfavorable with a baseline extending 2700 km over the European continent and the ground to satellite distance of 39000 km, leading to a configuration close to a monostatic measurement which would require sub-picosecond/s velocity measurement for estimating the full velocity vector components.

II. SDR FOR SECURE GNSS RECEPTION WITH 1-PPS GENERATION

The ability to record SDR signals for post-processing also means that SDR is perfectly suited for replay attacks in which the recorded signal is broadcast in order to make the receiver believe the erroneous signal is indicative of the current time or frequency information. These replay attacks, trivially implemented at `github.com/oscimp/usrp_recordAndReplay` using SDR, similar to signal spoofing attacks, are more subtle than jamming attacks which are readily detected with a loss of service: the receiver believes that a signal is received and is unable to identify whether the genuine information or a spoofed information is being decoded.

GNSS and especially the legacy GPS L1 civilian signal is especially sensitive to such attacks, lacking any means of authenticating the downlink messages. SDR is perfectly suited for generating such spoofing attacks, assuming as was introduced above that a local oscillator representative of the GNSS oscillator stability is used to clock the SDR. On the other hand, since SDR is able to receive the raw IQ stream from multiple antennas, a direction of arrival analysis can be performed prior to decode the GNSS payload and estimate the receiver position: inconsistent direction of arrivals with a single spoofing source generating the signals expected to be broadcast by a satellite constellation distributed in space is an indication of spoofing attack. Once detected, null steering can cancel this incoming spoofing signal, allowing to recover the genuine information [3].

However, once SDR has allowed to cancel the jamming or spoofing source using null-steering, the payload is decoded using e.g. the free and opensource `gnss-sdr` [4], leading to the positioning of the receiver and the estimate of the time offset between the local clock and the GNSS clock. From this information, the local oscillator is steered to match GNSS frequency, and time delays are introduced to match phases. However, the only timing information available to the SDR receiver is the time at which samples were digitized by the ADC, since all subsequent operations are asynchronous or could even occur as batch processes if First In First Out (FIFO) interfaces are used. Hence, the clock to be steered is the *one driving* the ADC, *not* the processing unit running `gnss-sdr`.

In past investigations [5], a dedicated GNU Radio signal processing block was introduced between the signal source collecting the IQ streams from hardware and the subsequent processing steps, `gnss-sdr` being based on the free, opensource implementation of SDR named GNU Radio. Doing so, the `gnss-sdr` source code must be tuned for each new release, making the proposed solution difficult to maintain on the long term. Since SDR relies on software processing of datastreams, distributed computing is perfectly suited: `gnss-sdr` provides a remote monitoring framework in which all internal states of the position, velocity and timing (PVT) solver are shared from a networking socket. Hence, steering the clock driving the ADC is handled by a process independent of `gnss-sdr`, with the added benefit of accessing the Time of Week information broadcast by the monitoring server with the bit duration of 20 ms resolution, removing this uncertainty that remained in the past implementation.

III. Conclusion

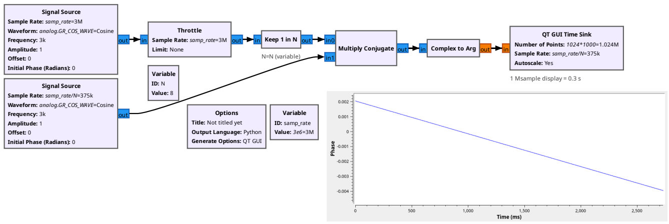
Software defined radio provides an innovative framework for radiofrequency signal processing reducing to a minimum the hardware contribution – source of drift and unreliability – and maximizing the software component. SDR setups are readily expanded and reproduced by copying software processing blocks, only limited by data transfer bandwidth and computational power.

However, two major hindrances are identified for the wide deployment of SDR in time and frequency metrology:

- software maintenance and long term consistency
- digital computation inaccuracies.

The first issue is a general challenge of increasingly abstract software relying on more and more external libraries and tools. At the time of this writing, installing GNU Radio on a Debian GNU/Linux binary distribution requires installing 740 packages (output of `debtree gnuradio | cut -d\- -f1 | grep \"` piped to `| sort | uniq | wc -l`), all depending on each other for the proper operation of the complete system. This reliance on external libraries induces that any Application Programming Interface (API) change in any of these dependencies breaks the complete system. Long term software maintenance is becoming increasingly challenging as libraries are constantly evolving, possibly in directions diverging from their original or intended uses. Major software framework restructuration – e.g. C++ to Python communication shifting from SWIG to `pybind` in the case of GNU Radio – breaks all backward compatibility and involves all Out of Tree contributors to correct their processing block implementation. Unless staff is dedicated to software maintenance, this hassle is becoming an increasingly heavy burden on engineers and researchers.

The second point is a core issue breaking the "stability" promise initially stated at the introduction of this article [1]. While a given algorithm will always lead to the same result when run on the same digital computer, various implementations of high level mathematical functions might lead to different results even though mathematically they are expected to reach the same result. Not considering the issue of single precision or double precision floating point number representation, even as simple a relation as $\exp(j(\omega/2)t)\cdot\exp(-j\omega(t/2)) = 1$ is not verified in practice. Indeed, the implementation of the trigonometric functions and the associated numerically controlled oscillator (NCO) involves incrementing a phase φ at each discrete time step with $\varphi \leftarrow \varphi + 2\pi \cdot f/f_s$ when synthesizing an oscillator at frequency f sampled at f_s. Since $\varphi \in [0 : 2\pi)$, the NCO implementation checks whether φ lies within these bounds, and adds or subtracts integer numbers of 2π to remain in this interval. However 2π being irrational, it cannot be represented accurately, neither in integer nor in floating point representations. In the above expression, the first term with $\omega/2 \cdot t$ represents a signal generated at sampling rate f_s at frequency $f/2$, while the second expression with $\omega \cdot t/2$ represents a signal at frequency f generated at sampling rate $f_s/2$, also resulting from generating a signal at frequency f and sampling rate f_s but decimated by a factor of 2. The GNU Radio implementation of this expression, using two signal sources, a decimating block (Keep 1 in N) and a complex conjugate multiplication, expected to lead to a constant phase output induced by possible different delays in the two branches, actually exhibit a slow drift induced by the inaccuracy of the trigonometric function calculation (Fig. 6).



Signal Source
Sample Rate: samp_rate=3M
Waveform: analog.GR_COS_WAVE=Cosine
Frequency: 3k
Amplitude: 1
Offset: 0
Initial Phase (Radians): 0

Throttle
Sample Rate: samp_rate=3M
Limit: None

Keep 1 in N
N=N (variable)

Multiply Conjugate

Complex to Arg

QT GUI Time Sink
Number of Points: 1024*1000=1.024M
Sample Rate: samp_rate/N=375k
Autoscale: Yes
1 Msample display = 0.3 s

Signal Source
Sample Rate: samp_rate/N=375k
Waveform: analog.GR_COS_WAVE=Cosine
Frequency: 3k
Amplitude: 1
Offset: 0
Initial Phase (Radians): 0

Variable
ID: N
Value: 8

Options
Title: Not titled yet
Output Language: Python
Generate Options: QT GUI

Variable
ID: samp_rate
Value: 3e6=3M

Fig. 6. GNU Radio Companion flowchart mixing (complex conjugate multiplication) one signal generated at $f/8$ and sampling rate f_s on one side, and at f and sampling rage $f_s/2$ on the other branch, the $f_s/8$ sampling rate being achieved by keeping one very 8 samples. Bottom right: phase of the output of the mixer, expected to be constant but observed to be drifting with time.

While digital communication and demodulation schemes in general are designed to compensate for these frequency differences between emitter and receiver local oscillators, these trigonometric function calculation issues might impact the use of digital signal processing for high stability time and frequency generation and dissemination applications.

Acknowledgment

The free and opensource SDR community is acknowledged for developing and maintaining the software frameworks allowing for this work to be completed, including authors and maintainers of GNU Radio and `gnss-sdr`. The digital electronics workpackage of the Oscillator Instability Measurement Platform (OscIMP) grant prompted these investigations and supported financially hardware acquisition.

References

[1] D. A. Mindell, *Digital Apollo: Human and machine in spaceflight*. Mit Press, 2011.

[2] J.-M. Friedt, "Passive reception of two-way satellite time and frequency transfer (TWSTFT) signals from a geostationary satellite, or GPS upside down," in *Proc. 12th GNU Radio Conference*, vol. 7, no. 1, 2022.

[3] W. Feng, J.-M. Friedt, G. Goavec-Merou, and F. Meyer, "Software defined radio implemented GPS spoofing and its computationally efficient detection and suppression," *IEEE Aerospace and Electronic Systems Magazine*, vol. 36, no. 3, pp. 36–52, 2021.

[4] C. Fernández–Prades, J. Arribas, P. Closas, C. Avilés, and L. Esteve, "GNSS-SDR: An open source tool for researchers and developers," in *Proc. 24th Intl. Tech. Meeting Sat. Div. Inst. Navig.*, Portland, Oregon, Sept. 2011, pp. 780–794.

[5] D. Rabus, G. Goavec-Merou, G. Cabodevila, F. Meyer, and J.-M. Friedt, "Generating a timing information (1-PPS) from a software defined radio decoding of GPS signals," in *2021 Joint Conference of the European Frequency and Time Forum and IEEE International Frequency Control Symposium (EFTF/IFCS)*. IEEE, 2021, pp. 1–2.