Synchronisation of Low-Cost Open Source SDRs for Navigation Applications

Marco Bartolucci*, José A. del Peral-Rosado†, Roger Estatuet-Castillo†, José A. García-Molina‡§, Massimo Crisci‡, and Giovanni E. Corazza*

*DEI - Università di Bologna, Italy
Email: {marco.bartolucci4, giovanni.corazza}@unibo.it
†Universitat Autònoma de Barcelona (UAB), Bellaterra, Spain
Email: {JoseAntonio.DelPeral, Roger.Estatuet}@uab.cat
‡European Space Agency (ESA), Noordwijk, The Netherlands
Email: {Jose.Antonio.Garcia.Molina, Massimo.Crisci}@esa.int
§HE Space, The Netherlands

***Abstract*—This paper describes a new method for the synchronisation of multiple low-cost open source software-defined radios (SDR). This solution enables the use of low-cost SDRs in interesting navigation applications, such as hybrid positioning algorithms, interference localisation, and cooperative positioning among others. Time synchronisation is achieved thanks to a time pulse that can be generated either by one of the SDRs or by an external source, such as a GNSS receiver providing 1PPS signal. Experimental results show that the proposed method effectively reduces the synchronisation offset between multiple SDRs, to less than one sampling period.**

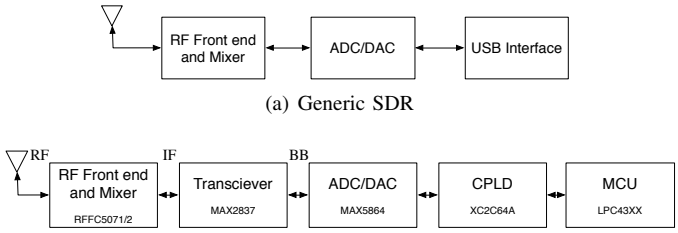
I. Introduction

In the early 2000s, *Software-Defined Radios (SDR)* started to gain popularity in the scientific community but, due to their high cost, they have been accessible only to researchers and not to ordinary users. This trend has changed in the last few years with the introduction of low-cost SDRs on the market, which has made this technology available to everyone. The cost of these devices is at least one order of magnitude lower than a comparable professional SDR. Moreover most low-cost SDRs are open-hardware and open-software, hence fully customisable. Professional SDRs offer time and frequency synchronisation of multiple SDRs, enabling a wide range of navigation applications. All of these applications require accurate time and frequency synchronisation, but low-cost SDRs generally do not support time synchronisation.

Time and frequency synchronisation of multiple SDRs enables or improves a wide range of navigation applications. It is possible to classify these navigation applications in two main categories: receiver- and network-oriented applications. Receiver-oriented applications are the ones that allow a single receiver to compute its own position, while network-oriented applications are the ones involving more than one receiver that are not usually colocated. Receiver-oriented applications may take advantage of the synchronisation of multiple SDRs to build multi-band and/or multi-system receivers. An important example of this possible use case is represented by hybrid navigation applications, i.e., receivers able to determine their own position by exploiting multiple sources or systems. This kind of receivers perform ranging measurements from different signals of opportunity (SoO) [1], such as WiFi, Bluetooth, Ultra Wideband (UWB), 3G, 4G Long Term Evolution (LTE), and prospectively 5G, as well as Global Navigation Satellite Systems (GNSS), in order to solve the positioning problem. Hence, these receivers must be able to tune their radios to different bands and demodulate different signals simultaneously. This can be done by using multiple SDRs, but synchronisation is required among them for time-based or frequency-based ranging. Another possible application falling into this category is represented by multi-band GNSS receivers, where different GNSS signals in different frequency bands are simultaneously processed, for better accuracy and robustness in harsh environments [2]. An example of network-oriented application is cooperative or peer-to-peer (P2P) positioning [3], in which multiple GNSS users cooperate to achieve a position solution in difficult environments. In this application, users need to communicate with each other either to exchange navigation data or to perform terrestrial ranging between users. The communication between users or the terrestrial ranging can be implemented using SDRs and synchronisation is mandatory for time-based and frequency-based terrestrial ranging. Another important network-oriented application is the detection and localisation of GNSS jammers or spoofers: in this case multiple sensor nodes monitor the GNSS bands and send snapshots of the signal to a server in the cloud, which detects and localise potential jammers using time difference of arrival (TDOA) and frequency difference of arrival (FDOA) [4]. Sensor nodes can be implemented using SDRs and strict synchronisation is required for TDOA/FDOA-based ranging.

Although these navigation applications are feasible with current professional SDRs, they cannot be exploited with very low-cost equipment. In this paper, we propose and validate an algorithm that enables sample-level synchronisation of multiple low-cost SDRs by using an off-the-shelf GNSS receiver. The reminder of this work is organised as follows: in Section II, we introduce SDR; in Section, III we propose a time synchronisation algorithm; in Section IV, we validate the synchronisation method experimentally and we suggest a simple statistical model for the synchronisation offset; finally, we draw the conclusions and propose future work.

978-1-5090-3885-5/16/$31.00 ©2016 IEEE



(a) Generic SDR

(b) Selected development board

Fig. 1. SDR Architecture.

II. Open Source Software-Defined Radios

Different definitions of software-defined radio are possible and we adopt the one given in [5]: a *software-defined radio* is a *radio* in which some or all of the physical layer functions are *software defined*. In the last decade the cost of SDRs has dropped, enabling a widespread diffusion of this technology. Most of the low-cost SDRs are open-source and open-hardware: schematics and PCB layouts are public; the code of the firmware and software is available free of charge and can be modified according to the users' need. Example of popular low-cost SDRs are *RTL-SDR* [6], a digital TV tuner that may be used as SDR (receive only); *HackRF One* [7], featuring a wide frequency range and a wide bandwidth (half-duplex transmit and receive), *BladeRF* [8], featuring USB 3.0 (full-duplex transmit and receive).

The aforementioned SDRs share a common architecture, shown in Figure 1(a): in receive mode, the signal coming from the antenna is filtered, amplified, and downconverted to baseband by the RF front end and mixer, then the signal is discretised and quantised by an Analogue to Digital Converter (ADC) and the samples are transmitted to a computer (host) in charge of signal processing, via an USB interface. In transmit mode, signal samples are fed to a Digital to Analogue Converter (DAC) through the USB interface, then the signal is upconverted, amplified by the RF front end, and transmitted.

A. Selected development board

Our selected development board (SDB) is HackRF One [7], a low-cost open-source and open-hardware SDR, capable of transmitting or receiving signals from 1 MHz to 6 GHz. The ADC/DAC operates at up to 20 Msps (8 bit I/Q samples). Baseband filter and transmit/receive gain are configurable by software, and pin headers on the PCB allow future expansions. The architecture of the SDB is shown in Figure 1(b) and it is composed by three stages: RF, Intermediate Frequency (IF), and baseband (BB). In receive mode, the RF signal is converted to IF by an *RFMD RFFC5071/2*, a wideband synthesiser with integrated 6 GHz mixer, and a *Maxim Integrated MAX2837* wireless broadband transceiver, is responsible for the conversion from IF to BB and the ADC/DAC (codec) functions. A *Xilinx XC2C64A* Complex Programmable Logic Device (CPLD) acts as glue logic between the codec and the *NxP LPC43XX* micro controller (MCU) which provides the USB interface to the user. The same components are responsible for the opposite functions in transmit mode. The SDB provides *clock input* and *clock output* ports for frequency synchronisation of multiple SDR. Time synchronisation is not supported by the current firmware but, thanks to the hardware and software openness, it is possible to use a time pulse for time synchronisation purposes. The next section describes in detail the synchronisation algorithm and the necessary hardware connections.

III. Synchronisation Algorithm

In the following we tackle the synchronisation of SDBs in receive mode. A similar approach may be applied to achieve the same result in transmit mode. The algorithm can be also applied with minor modification to other open source SDRs with similar architectures. The idea is to use an expansion pin header on the PCB to add new signals for time synchronisation: the start of reception should be triggered by a time pulse; the pulse may be generated by one of the SDBs or by external hardware. Figure 2 shows the hardware connections between two SDBs: the signal SYNC_IN (pin 16 of the expansion header P28, top and bottom SDBs) is the input for the synchronisation pulse. SYNC_CMD (pin 15 of the expansion header P28, top SDB) is the pulse command signal. We consider two configurations: in the first one (A), the pulse command is generated by one of the SDBs; in the second configuration (B) the pulse command is the 1PPS signal of a GNSS receiver, i.e., a time pulse with a one-second period, synchronised with GPS time. While configuration (A) does not require additional hardware, it requires the receivers to be colocated. In configuration (B), instead, SDBs may be located far away from each other. Moreover, in this case, the recordings are synchronised with GPS time. The signals SYNC_IN and SYNC_CMD are connected to the CPLD, as shown in Figure 3, along with other signals involved in the receive mode. During the initial setup, the center frequency, filters bandwidth, gains, and ADC sampling rate are configured. Then, the ADC starts streaming the samples of the received signal to the CPLD (signal ADC_DATA[7..0]). The samples are available on both rising (I) an falling edge (Q) of CODEC_CLK. The CPLD simply makes the data (HOST_DATA[7..0]) available to the MCU on the rising edge

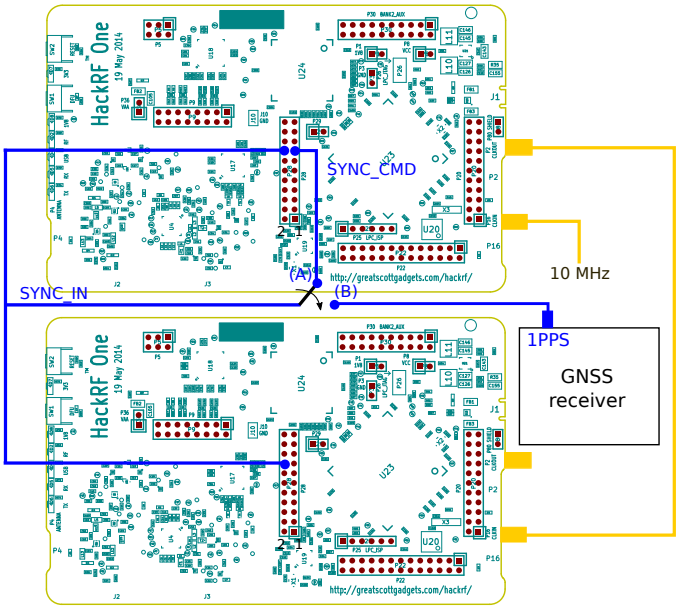


Fig. 2. Synchronisation of two SDBs: hardware connections.

978-1-5090-3885-5/16/$31.00 ©2016 IEEE

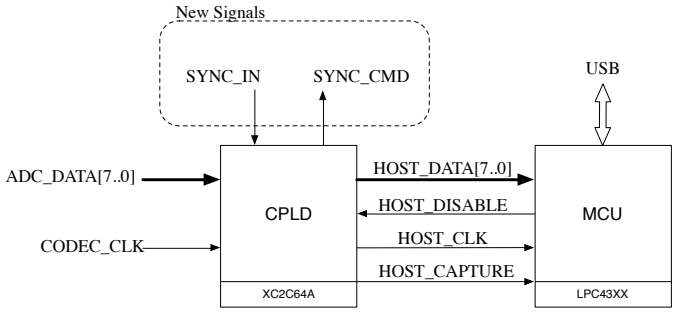


Fig. 3. CPLD and MCU signals.

of HOST_CLK, whose frequency is doubled with respect to CODEC_CLK. The MCU communicates to the CPLD the wish to start the recording, setting HOST_DISABLE to '0'. At this stage the MCU is ignoring the data samples and it will continue until the CPLD sets HOST_CAPTURE to '1'. The CPLD controls HOST_CAPTURE in order to ensure the correct alignment of interleaved I/Q samples. Once the I/Q data alignment is correct, the MCU may finally start streaming the samples via USB. When the recording phase ends, the MCU pulls HOST_DISABLE high, to signal the CPLD the end of the recording. A snippet of the original VHDL controlling the signal HOST_CAPTURE is available in Listing 1.

Listing 1. ORIGINAL VHDL (SIMPLIFIED)

```
-- MCU ignores data samples when HOST_CAPTURE = '0'. The
     following code guarantees that the MCU receives I/Q
     interleaved samples with I samples in the odd positions
      and Q samples in the even ones.
process(HOST_CLK)
begin
  if rising_edge(HOST_CLK) then
    if CODEC_CLOCK = '0' then
      HOST_CAPTURE <= not HOST_DISABLE
    end if;
  end if;
end process;
```

This architecture leads to the conclusion that signal HOST_CAPTURE is a good candidate for time synchronisation. In principle, it is sufficient to hold HOST_CAPTURE low when SYNC_IN is low or HOST_DISABLE is high. This should be enough to ensure that the synchronisation error is below one sampling period. However, the time pulse must stay high during the whole receiving phase, otherwise the stream of samples to the user is interrupted. Therefore, this method does not work with the 1PPS signal, which is typically a low duty-cycle square wave. This approach may be improved by using a latched version of the signal SYNC_IN, as shown in Listing 2. In this case, it is not required that the pulse command stays high during the whole receiving phase and this approach works with 1PPS signals.

Listing 2. MODIFIED VHDL (SIMPLIFIED)

```
-- The MCU ignores data when HOST_DISABLE='1' or
     sync_in_latched='0'. The following code allows multiple
      SDBs to start receiving samples synchronously.
process(HOST_CLK)
begin
  if rising_edge(HOST_CLK) then
    if CODEC_CLOCK = '0' then
      HOST_CAPTURE <= not HOST_DISABLE and sync_in_latched;
    end if;
  end if;
end process;

-- When the MCU pulls HOST_DISABLE low, HOST_SYNC_CMD
     becomes high (the synchronisation pulse is sent to the
     other SDBs). sync_in_latched is high when  HOST_DISABLE
      is low and there is a rising edge of SYNC_IN; it is
     low when HOST_DISABLE is high.
process(HOST_DISABLE, SYNC_IN)
begin
  SYNC_CMD <= not HOST_DISABLE;
  if HOST_DISABLE = '0' then
    if rising_edge(SYNC_IN) then
      sync_in_latched <= SYNC_IN;
    end if;
  else
    sync_in_latched = '0';
  end if;
end process;
```

In order to validate the synchronisation algorithm, we are using two different approaches in the next section, based on GNSS and LTE signal processing.

IV. EXPERIMENTAL RESULTS

This section discusses the laboratory setup, the GNSS and LTE validation approaches, and the experimental results on the synchronisation offset. A statistical model is then proposed to characterise the resulting synchronisation offset.

A. Experimental setup

The experimental testbed is located in the European Navigation Laboratory (ENL) at the European Space Agency (ESTEC, The Netherlands). A diagram of the test setup is shown in Figure 4, where the signal source is either a GNSS antenna (GNSS) or a LTE network emulator (LTE). A high-end active GNSS antenna is located at the roof of the building in open-sky conditions. The Spirent E2010S network emulator generates the LTE signal from one base station (BS) at a system bandwidth of 1.4 MHz in AWGN conditions, with a signal-to-noise ratio (SNR) around 30 dB.

The input signal goes through a RF power divider and then to the two SDBs. A reference signal of 10 MHz generated by an active hydrogen maser is used to synchronise, in frequency, the clocks of the LTE network emulator and the two SDBs. The time synchronisation is achieved with square pulse of 1 Hz obtained from either a GNSS receiver (1PPS) or an Arbitrary Function Generator (AFG). Hardware connections between the synchronisation pulse and the two SDBs, described in Section III, are implemented in a prototyping PCB, as shown in Figure 6. The signals received by the SDBs are recorded at a sampling rate F_s by a laptop PC (host) and processed using either the GNSS- or LTE-based approaches. In order to obtain statistically stable results, experiments are repeated 1000 times. SDBs are reset after each recording, to ensure that the experiments are performed in the same conditions. This has required the modification of the MCU firmware and the SDB Linux tools: a new function has been added to allow the host to reset SDBs using the USB interface. A few recordings experienced losses of samples at high sampling rates, because the slow hard drive installed in the laptop could not cope with the high data rate sample stream. To avoid this problem signals are recorded in a tmpfs [9] virtual memory disk, a disk residing in RAM memory. Another possible issue that may occur is due to the drift of the host clock with respect to GNSS time: if the host starts a recording when the 1PPS synchronisation pulse is

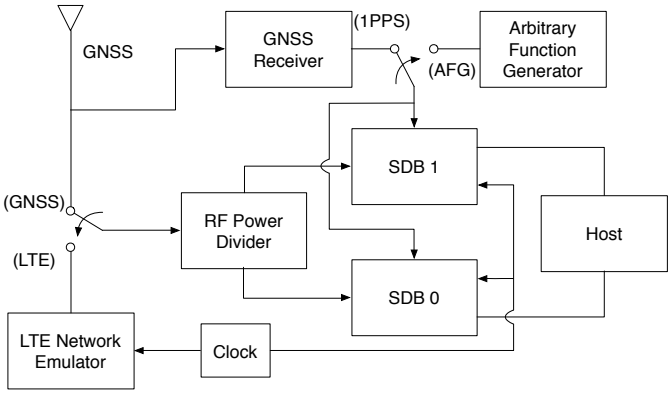
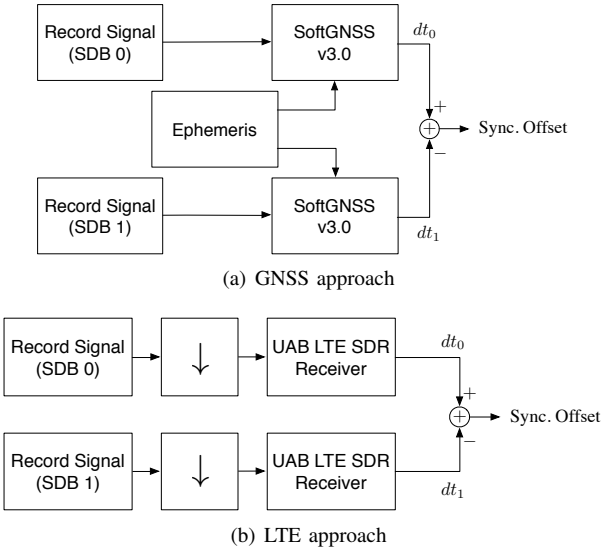


Fig. 4. Experimental setup of the laboratory.



(a) GNSS approach

(b) LTE approach

Fig. 5. Diagram of validation approaches.

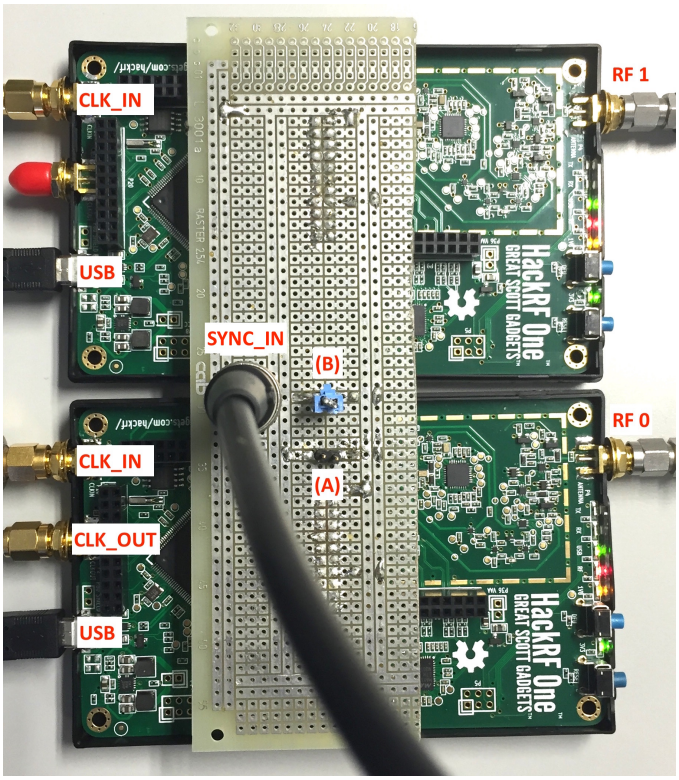


Fig. 6. SDB connection board.

high, the CPLD will not be able to detect a rising edge and the signal will not be recorded. To avoid this problem is possible to use a GNSS receiver to synchronise the clock of the host, using *GPSD* in combination with *NTP* [10].

B. GNSS validation approach

The GNSS validation approach is based on the estimation of the position, velocity and time (PVT) solution using the *SoftGNSS v3.0* software GPS receiver [11]. This software was originally meant to work at IF with real samples, but the recordings captured by the SDBs are at BB and the samples are complex. Therefore, we modified the acquisition and tracking stages of SoftGNSS, in order to be able to work with this kind of signals. A further modification to the acquisition and the tracking phases has been done in order to use the same satellites: only the satellites that are visible to both SDBs are taken into account for acquisition and tracking, in order to have comparable solution accuracies for both SDBs. Moreover, the software requires at least 36 seconds of recorded signals, in order to compute a PVT solution. This is due to the fact that the receiver needs to demodulate the navigation data, in order to extract the ephemeris and determine the orbit of the visible satellites: the receiver needs to demodulate 5 subframes, each composed by 300 bits (6 s, since the bit time is 20 ms). The additional 6 seconds are needed because the tracking phase may start in the middle of a subframe. Since processing 36-seconds recordings is cumbersome and time consuming, we propose an assisted GNSS (A-GNSS) method: instead of retrieving the ephemeris data from the signals, we download *Receiver Independent Exchange Format (RINEX)* files coming from the *International GNSS Service (IGS)* station located in Delft. IGS provides open access, high quality GNSS data, products, and services in support of research. The data downloaded from the IGS station is used to assist the SoftGNSS receiver and allows to obtain a PVT solution in ten seconds. The knowledge of the time of week (TOW), which is the number of 1.5 seconds elapsed from the beginning of the GPS week, is still required to compute the pseudorange measurements and hence, to obtain the navigation solution. The TOW can be read at the beginning of each subframe, therefore requires to demodulate at least 6 seconds of GPS signal. In order to further reduce the processing time, we applied the so-called coarse time positioning method [12], which allows to obtain a navigation solution without the need of reading the TOW. In this method, the time of week is treated as an additional unknown by the least squares algorithm, therefore the unknown navigation solution vector is $(X, Y, Z, c\ dt, t_W)^T$, where (X, Y, Z) is the position of the receiver, dt is the clock offset between the GPS time and the time of the receiver, c is the light velocity, and t_W is the TOW. The observation model relating pseudoranges to the navigation

978-1-5090-3885-5/16/$31.00 ©2016 IEEE

solution vector is given by

$$P_i = \sqrt{(X_i - X)^2 + (Y_i - Y)^2 + (Z_i - Z)^2} + \quad (1)$$

$$+ c\,dt + c\frac{f_{d,i}}{f_{L1}} t_W + e_i, \quad (2)$$

where (X_i, Y_i, Z_i) is the position of i-th satellite, $f_{d,i}$ is the Doppler frequency, f_{L1} is L1 band center frequency, i.e., 1575.42 MHz, and e_i is a measurement error term. Using this model, the least squares algorithm requires at least five visible satellites in order to find a navigation solution. If the total number of visible satellite is N, the geometry matrix, i.e., the Jacobian matrix of the observation model, $\mathbf{A} \in \mathcal{M}_{N\times 5}(\mathbb{R})$ is given by

$$\mathbf{A} = \begin{bmatrix} \Delta x_1 & \Delta y_1 & \Delta z_1 & 1 & c f_{d,1}/f_{L1} \\ \Delta x_2 & \Delta y_2 & \Delta z_2 & 1 & c f_{d,2}/f_{L1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \Delta x_N & \Delta y_N & \Delta z_N & 1 & c f_{d,N}/f_{L1} \end{bmatrix}, \quad (3)$$

where the first four columns are unchanged with respect to the traditional least squares navigation algorithm [11], and the last column is added for the TOW estimation. The synchronisation offset between the two SDBs is computed after the PVT solution, as the difference between the clock offsets of the two SDRs, i.e., $\theta = dt_0 - dt_1$. The GNSS approach is summarised in Figure 5(a).

Since GNSS signals have been specifically designed for precise timing, the use of GNSS signals can be a natural choice to validate the proposed synchronisation method. However, a number of possible problems may arise using this approach. The most important drawback of this validation method is that the accuracy of the clock offset estimator is not constant and depends on the number of visible satellites, on the carrier-to-noise-density ratio (C/N_0) of the received signals, and on the geometry of the satellite constellation. The high variability of the aforementioned accuracy and the fact that, occasionally it is not easy to have the same five satellites visible to both SDBs, make the use of this validation method impractical and the analysis of results troublesome. An example of synchronisation offset between the two SDBs obtained with the GNSS approach is shown in Figure 7. The synchronisation offset is estimated every 1 ms in a 5 seconds recording taken at a sampling frequency $F_s = 10$ MHz. The synchronisation offset, in this case, should be limited to $\pm 1/F_s = \pm 100$ ns but, in spite of the fact that the majority of the estimates fall within this interval, many outliers are present. This phenomenon is due to the low accuracy of the clock offset estimation caused by low C/N_0. Therefore, the GNSS approach is in practice cumbersome with the original least squares algorithm because it requires 36 seconds of data and it is not accurate enough when using the coarse time method with the described modification of the SoftGNSS receiver. In the remainder of this section, we present a simpler yet effective LTE-based validation approach.

C. LTE validation approach

Terrestrial signals can be used as signals of opportunity to validate or calibrate the synchronisation procedure, in case there is a lack of visible GNSS satellites. This opportunistic approach can be based on cellular systems (e.g. 2G, 3G or 4G),

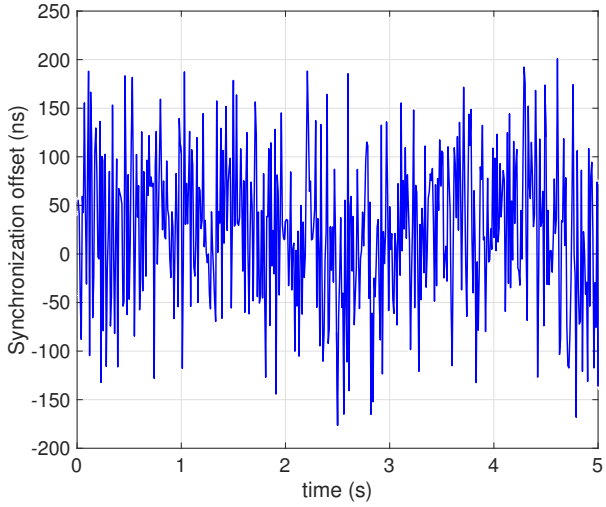


Fig. 7. Synchronisation offset using the GNSS approach.

broadcast television (e.g. DVB-T), Wi-Fi, Bluetooth or any other radio signal. The selection of the opportunistic system depends on the signal availability, and the signal bandwidth to achieve a certain time-delay estimation (TDE) accuracy. An accessible time reference within the signal can be useful to validate the synchronisation over time. Thus, this paper considers the 4G LTE system due to its wide adoption in urban environments, high signal bandwidth (up to 20 MHz), and periodical time reference, i.e. system frame number (SFN) every 10 ms. In contrast to the GNSS approach described in the previous section, the LTE validation procedure is based on the estimation of the time-delay from the signal transmitted by a single base station with unknown location.

As it is shown in Figure 4 and 5(b), the LTE signal is first split and fed to the two SDBs that record a snapshot of 200 ms. The signal is then resampled to 2 MHz. The UAB LTE SDR software receiver, which is described in [13] and [14], is used to acquire and track the 1.4-MHz LTE signal captured by each SDB. The noise bandwidth of the delay-locked loop and frequency-locked loop is set to 30 Hz and 50 Hz, respectively, and the resolution of the TDE is defined to 0.25 ns. The SFN is calculated by using the LTE Cell Scanner software developed by [15]. Only ten LTE radio frames are tracked, and the last time-delay estimate of each processed capture is used to calculate the synchronisation offset between the two SDBs. The synchronisation is validated between multiple captures by using the time between recordings and the SFN.

D. Synchronisation offset

The probability density function (PDF) of the synchronisation offset is computed based on the LTE approach, by considering the 1PPS and AFG pulses. As it is shown in Figure 8, the resulting synchronisation offset is within $\pm 1/F_s$, i.e., ± 200, ± 100 and ± 50 ns for F_s equal to 5, 10 and 20 MHz, respectively. This test confirms the achievable accuracy of the proposed synchronisation procedure, which is bounded by the sampling period of the SDB. In addition, the same results are obtained with both synchronisation pulses, demonstrating the flexibility and reproducibility of the proposed algorithm.

978-1-5090-3885-5/16/$31.00 ©2016 IEEE

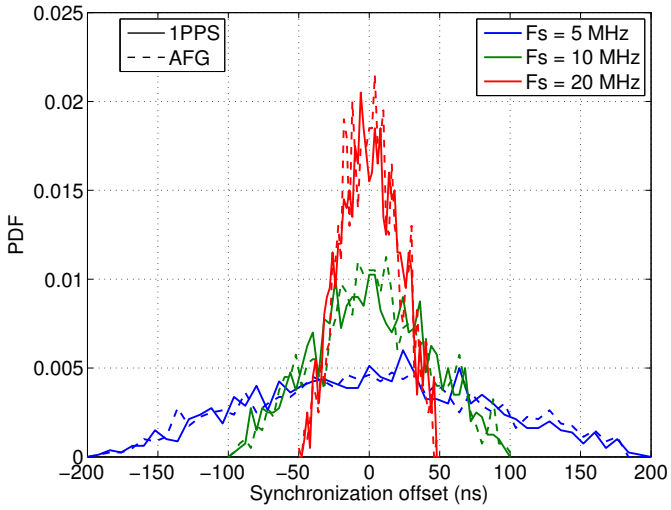


Fig. 8. PDF of the synchronisation offset using the LTE approach.

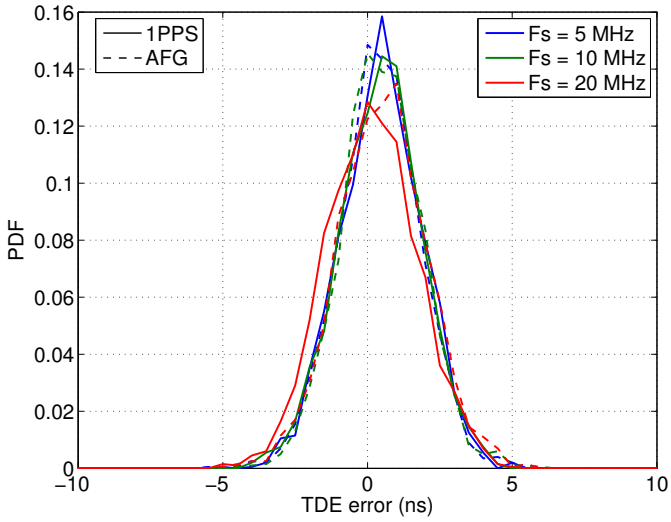


Fig. 9. PDF of TDE error with the LTE approach.

These results are validated by measuring the error of the TDE. As it is shown in Figure 9, the PDF of this error is confined within ± 5 ns. Thus, the estimation error of the test practically has no effect on the results. The mean μ_θ and standard deviation σ_θ of the synchronisation offset and the mean μ_τ and standard deviation σ_τ of the TDE error can be seen in Table I(a).

The main difference between the use of a synchronisation pulse generated by a GNSS receiver and a signal generator is the clock drift. Since the GNSS receiver is synchronised to the accurate atomic clocks of the satellites, the resulting square pulse is very stable. In contrast, the synchronisation pulse of the AFG has a clock drift due to its local oscillator. This effect can be observed in Figure 10, where the TDE of a single SDB is plotted over time. The TDE obtained with the AFG for different signal captures has a noticeable drift, which is wrapped to 10 ms (i.e., length of a radio frame) in the case of LTE, while there is no TDE drift with 1PPS.

TABLE I. MEAN μ_θ AND STANDARD DEVIATION σ_θ OF THE SYNCHRONISATION OFFSET AND TDE ERROR FOR THE LTE APPROACH.

(a)

Sampling frequency	Signal source	Sync. pulse	Sync. offset μ_θ (ns)	Sync. offset σ_θ (ns)	TDE error μ_τ (ns)	TDE error σ_τ (ns)
5 MHz	LTE	AFG	1.45	82.56	0.40	1.34
		1PPS	-0.55	80.10	0.44	1.34
10 MHz	LTE	AFG	2.12	40.19	0.47	1.32
		1PPS	0.91	40.69	0.42	1.36
20 MHz	LTE	AFG	0.60	19.73	0.49	1.46
		1PPS	0.81	20.62	0.13	1.49

(b)

Signal source	Normalised Sync. offset μ_θ (samples)	Normalised Sync. offset σ_θ (samples)
LTE	0.01	0.40

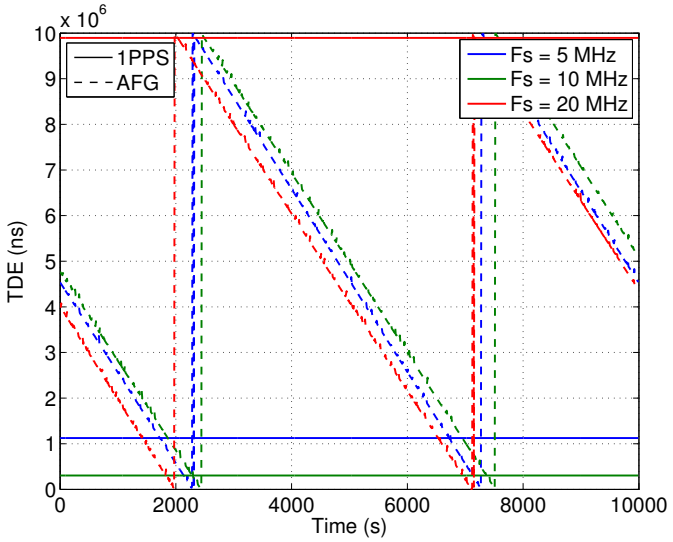


Fig. 10. Synchronisation offset over time with the LTE approach.

E. Statistical model of the synchronisation offset

Due to the familiarity of the previous test results, a statistical model is here defined to characterise the synchronisation offset of the proposed algorithm. For this purpose, the synchronisation offset is normalised by $1/F_s$ for each test, resulting in the bar plot of Figure 11. These results lead to the conclusion that a truncated normal distribution [16] can fit the synchronisation offset to the algorithm. Thus, the resulting synchronisation offset can be modelled as

$$\theta \sim \begin{cases} \mathcal{N}(\mu_\theta, \sigma_\theta), & \text{if } \theta \in (-1, 1), \\ 0, & \text{otherwise}, \end{cases} \tag{4}$$

where the PDF of the truncated Gaussian distribution is

$$f(\theta, \mu_\theta, \sigma_\theta) = \frac{\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{(\theta-\mu_\theta)^2}{2\sigma^2}\right\}}{\frac{1}{2}\mathrm{erf}\left\{\frac{1-\mu_\theta}{\sigma_\theta\sqrt{2}}\right\} - \frac{1}{2}\mathrm{erf}\left\{\frac{-1-\mu_\theta}{\sigma_\theta\sqrt{2}}\right\}}, \tag{5}$$

and $\mathrm{erf}\{\cdot\}$ is the error function. The truncated normal distribution is shown in Figure 11 by using the fitting parameters of Table I(b), obtained with maximum likelihood estimators [16].

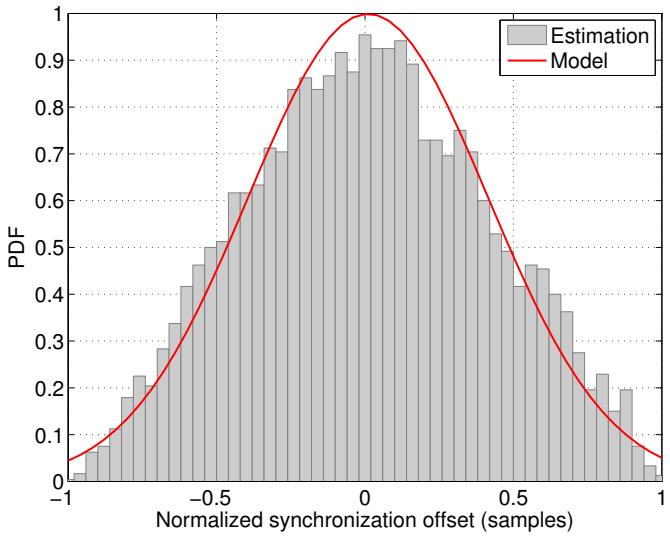


Fig. 11. Statistical model of the synchronisation offset

The observed resemblance suggests the validity of the model and confirms the performance of the proposed synchronisation method, whose accuracy is confined within a sampling period.

V. Conclusion

This paper proposes a sample-level time synchronisation method for multiple low-cost software-defined radio (SDR), along with a statistical model for the synchronisation offset. The synchronisation algorithm has been implemented on a low-cost SDR platform, by using an off-the-shelf GNSS receiver to provide a time reference. Experimental tests has been conducted to validate the capability of this approach to reduce the synchronisation offset to less than one sampling period. Time synchronisation of multiple radios enables a wide range of navigation-oriented applications, for a fraction of the cost of professional SDR equipment. Future work is aimed to demonstrate promising navigation solutions, such as hybrid localisation, interference detection and localisation, or cooperative positioning.

Acknowledgment

This work was partially supported by the University of Bologna - Doctoral School in Electronic, Telecommunications and Information Technology and by the ESA under NPI programme No. 4000110780/14/NL/AK.

Disclaimer

The views presented in the paper represent solely the opinion of the authors and not necessarily the view of ESA.

References

[1] D. Dardari, P. Closas, and P. M. Djurić, "Indoor tracking: Theory, methods, and technologies," *IEEE Transactions on Vehicular Technology*, vol. 64, no. 4, pp. 1263–1278, 2015.

[2] D. Chen, W. Pan, P. Jiang, J. Jin, T. Mo, and J. Zhou, "Reconfigurable dual-channel multiband RF receiver for GPS/Galileo/BD-2 systems," *IEEE Transactions on Microwave Theory and Techniques*, vol. 60, no. 11, pp. 3491–3501, Nov 2012.

[3] L. Deambrogio, C. Palestini, F. Bastia, G. Gabelli, G. E. Corazza, and J. Samson, "Impact of high-end receivers in a peer-to-peer cooperative localization system," in *Ubiquitous Positioning Indoor Navigation and Location Based Service (UPINLBS), 2010*, Oct 2010, pp. 1–7.

[4] J. A. Garcia-Molina and M. Crisci, "Cloud-based localization of GNSS jammers," in *Proceedings of the 28th International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS+ 2015)*, September 2015, pp. 3289 – 3295.

[5] SDR Forum, "SDRF cognitive radio definitions," http://www.sdrforum.org/pages/documentLibrary/documents/SDRF-06-R-0011-V1_0_0.pdf, 2007, [Online; accessed 09-June-2016].

[6] "RTL-SDR.com, RTL-SDR (RTL2832U) and software defined radio news and projects. also featuring Airspy, HackRF, FCD, SDRplay and more." http://www.rtl-sdr.com, [Online; accessed 09-June-2016].

[7] "HackRF One, an open source SDR platform," https://greatscottgadgets.com/hackrf/, [Online; accessed 09-June-2016].

[8] "bladeRF - the USB 3.0 Superspeed Software Defined Radio," http://nuand.com, [Online; accessed 09-June-2016].

[9] "tmpfs documentation," https://www.kernel.org/doc/Documentation/filesystems/tmpfs.txt, [Online; accessed 09-June-2016].

[10] "GPSD time service howto," http://www.catb.org/gpsd/gpsd-time-service-howto.html, [Online; accessed 09-June-2016].

[11] K. Borre, D. M. Akos, N. Bertelsen, P. Rinder, and S. H. Jensen, *A software-defined GPS and Galileo receiver: a single-frequency approach*. Springer Science & Business Media, 2007.

[12] F. S. T. Van Diggelen, *A-GPS: Assisted GPS, GNSS, and SBAS*. Artech House, 2009.

[13] J. A. del Peral-Rosado, J. M. Parro-Jiménez, J. A. López-Salcedo, G. Seco-Granados, P. Crosta, F. Zanier, and M. Crisci, "Comparative results analysis on positioning with real LTE signals and low-cost hardware platforms," in *2014 7th ESA Workshop on Satellite Navigation Technologies and European Workshop on GNSS Signals and Signal Processing (NAVITEC)*, Dec 2014, pp. 1–8.

[14] J. A. del Peral-Rosado, J. A. López-Salcedo, G. Seco-Granados, P. Crosta, F. Zanier, and M. Crisci, "Downlink synchronization of LTE base stations for opportunistic ToA positioning," in *2015 International Conference on Location and GNSS (ICL-GNSS)*, June 2015, pp. 1–6.

[15] "Evrytania LLC, LTE cell scanner," https://github.com/Evrytania/LTE-Cell-Scanner, [Online; accessed 09-June-2016].

[16] A. C. Cohen, "Estimating the mean and variance of normal populations from singly truncated and doubly truncated samples," *Ann. Math. Statist.*, vol. 21, no. 4, pp. 557–569, 12 1950. [Online]. Available: http://dx.doi.org/10.1214/aoms/1177729751