Real-Time Low-Earth Orbit Detector Implementation for Chirp-Based Preamble Communication Systems

Matthieu Magnant, Mohamed Amine Ben Temim, Bertrand Le Gal, Guillaume Ferré, Florian Collard

Real-time low-earth orbit detector implementation for chirp-based preamble communication systems

Matthieu MAGNANT[1], Mohamed Amine BEN TEMIM[1,2], Bertrand LE GAL[1], Guillaume FERRÉ[1] and Florian COLLARD[3]
[1]Univ. Bordeaux, CNRS, Bordeaux INP, IMS, UMR 5218, F-33400, Talence, France
[2]TELNET SPACE, France
[3]EUTELSAT, France
Email: forename.name@ims-bordeaux.fr, mohamedamine.bentmime@groupe-telnet.net, fcollard@eutelsat.com

***Abstract*—Internet of Things (IoT) has been undergoing a revolution with the deployment of gateways in low earth orbit (LEO). This revolution offers what IoT has been promising since its inception, namely the connection of an object regardless of its position on Earth. Several physical layers have shown to have the sensitivity for such communications. In this paper, we propose a solution allowing an IoT network operator using LEO satellite to detect different uplink IoT communication technologies based on a same preamble. The proposed approach has been implemented on two ARM cores (Cortex-A72 and Cortex-A9) compliant with the spatial constraints. The experiments carried out demonstrate the real-time capability of the proposed approach even on low-end processor targets with approximately 10% of the CPU time.**

***Index Terms*—IoT, CSS, LPWAN, LEO satellites, preamble detection, ARM, prototyping, real-time.**

I. INTRODUCTION

With the democratization of access to space, we are now witnessing a massive deployment of satellites in low earth orbit (LEO). These satellites are mostly nanosatellites and carry all types of payloads. Whatever the payload, the satellite has digital communication means enabling it to send or receive data. The Internet of Things (IoT) has not escaped this revolution in access to space, and many projects have been or are being deployed to connect an object to the Internet regardless of its position on Earth, using LEO satellites as gateways. Connecting IoT in this way involves many technical challenges from a data communication point of view. The two major challenges are related to: 1/ the relative speed of the satellite with respect to the Earth and 2/ the field of view (FoV) of the satellite [1].

Indeed, Low Power Wide Area Network (LPWAN) technologies currently deployed on Earth, such as those from Semtech (LoRa, LR-FHSS) or NB-IoT, offer link budgets that allow LEO satellite communications. Indeed, these long-range, low-energy wireless networks support path losses of up to 150 dB for some of them. For instance, in Europe, the LoRa technology has long range modes with sensitivities close to $S = -140$dBm with Effective Isotropic Radiated Power (EIRP) up to $P_{EIRP} = 14$dBm. It seems relevant to use them as is, to communicate with LEO satellites. As they were not originally designed for such communications, it is necessary to design receivers allowing to demodulate them. However, before considering demodulating the received signals, it is necessary to detect the presence of a relevant signal. This detection is generally carried out by the intermediary of a signal called: preamble.

If the Doppler effect generated by the relative speed is independent of the communication technology used, the impact of the FoV depends on it. Indeed, the communications carried out in the free use bands (typically the industrial, scientific and medical (ISM) bands) will be more exposed to the interference phenomenon because by principle several technologies share the same frequency band without any coordination. Several strategies can be implemented to fight against collisions [2]–[5]. However, firstly, it is necessary to detect the presence of the signals before considering a treatment to deal with the interference.

In this paper, we first propose an algorithm to detect the occurrence of heterogeneous chirped preambles in the context of uplink communications to a LEO satellite. Given this application context, the performances of our detector must allow a good detection in both low and high load conditions. Moreover, it should have a low-computational complexity to respect spacial embedded system constraints. Consequently, an evaluation is then provided, focusing on the real time performances of the proposed algorithm on programmable targets compatible with a space use.

The paper is organized as follows. In section II we present the context. In section III, the proposed detection algorithm is described. This algorithm is also evaluated with synthetic data in section III and discussions on its performances are carried out. Then section IV presents implementation results (execution time, energy) of the detection algorithm on ARM programmable targets that can be used in nanosatellites. Finally, the conclusion is exposed in section V as well as the research perspectives.

II. CONTEXT

This work concerns a project developed in collaboration with the company Eutelsat as part of the deployment of their IoT constellation by LEO satellites. The objective is to develop an algorithm to detect preamble signals that are present in the header of an information packet emitted by an

object on Earth. In addition, the waveform used to generate the preamble must be consistent with the various chips of the most widespread LPWAN technologies. Thus, this waveform must be compatible with LoRa and LR-FHSS chips such as SX1261 and SX1262, and with Sigfox chips such as S2-LPQTR and S2-LPCBQTR [6]. By analyzing the technical documentation, we have opted for chirped preambles. This choice is also motivated by the use of an ISM band in which the impact of interfering signals must be limited. In addition to the constraint of using an ISM band, the duration of the preamble and the bandwidth of the complex envelope are limited to 150 ms and 10 kHz respectively. Furthermore, a dedicated channel for preamble transmission is reserved, allowing the payload data rate not to be limited to the preamble band.

III. Chirp-based preambles detection

Let us recall the chirp spread spectrum (CSS) modulation principle. If we denote SF the number of bits per symbol, the latter consists in associating to each SF-uplet of bits a unique phase trajectory $\phi_k(t)$ of duration T among a set ω_S of $M = 2^{SF}$ different trajectories. Here T represents the symbol duration. If B denote the chirped bandwidth we obtain: $M = B \times T$. This constraint ensures phase continuity between successive modulated chirps.

The preambles of different SF are orthogonal as soon as received power difference is about 6dB. For satellite communication, such a power loss (resp. gain) corresponds to a division (resp. multiplication) by 2 of the communication range. Given that the satellite constellation orbits at an altitude of 550km, such a power difference corresponds to a FoV of at least 2.8 million km^2 i.e. 5.2 times the surface area of France. Thus, it is necessary to develop an algorithm allowing the detection of non-orthogonal preambles, even if the SF are different.

In order to maximize the number of available preambles, and considering the project constraints, the bandwidth B is fixed to 7.8kHz, and SF 7, 8 and 9 including respectively N_p 8, 4 and 2 raw up chirps (or raw down chirps) in the preamble are selected. This allows us on one hand to close the link budget and on the other hand to stay below the maximum permitted preamble duration, which is in this case $\forall SF \in \{7, 8, 9\}$, $T_p = N_p \times \frac{2^{SF}}{B} = 131$ms. If $s_p(t)$ denote the preamble complex envelope, we obtain:

$$s_p(t) = \sum_{p=1}^{N_p} e^{j\phi_0(t-(p-1)T)} \mathbb{1}_{[(p-1)T, pT)}(t) \tag{1}$$

Without loss of generality, we considered that $\forall p \in [\![1, N_p]\!]$, $\phi_0(t)$, which represents the signal instantaneous phase, is obtained by integrating the frequency $f(t) = B(\frac{t}{T} - 0.5)$. It represents the unmodulated version of the chirped signal. In this paper, the term *dechirping* will denote the operation consisting in multiplying a slice of the T-long CSS signal by the conjugate version of the unmodulated chirped signal: $e^{-j\phi_0(t)}$.

A. Proposed algorithm

In this section we present the detection algorithm we have developed. First, let us expose the complex envelope structure as received by the satellite:

$$y_r(t) = \sum_{i=1}^{N_R} \sqrt{P_i} x_i(t - t_{s,i}) e^{j(2\pi(\Delta f_i + \frac{c_{d,i} t}{2})t + \varphi_{0,i})} + w(t) \tag{2}$$

- $w(t)$ the complex additive white Gaussian noise whose variance is σ_w^2,
- N_R the number of preambles captured,
- P_i, $t_{s,i}$ Δf_i, $c_{d,i}$ and $\varphi_{0,i}$ are respectively the power, the starting time, the frequency offset, the Doppler rate and the initial phase of the i-th received signal,
- $x_i(t)$ the complex envelope of the i-th received signal:

$$x_i(t) = s_{p,i}(t)\mathbb{1}_{[0,T_p]}(t) + r_i(t - T_p)\mathbb{1}_{[T_p, T_p + T^i_{pkt})}(t) \tag{3}$$

with $r_i(t)$ (resp. $s_{p,i}(t)$) being the complex envelope of the packet (resp. preamble) of the i-th received signal and T^i_{pkt} its duration. Here $r_i(t)$ comes from an IoT object on the ground using a LPWAN technology such as LoRa, Sigfox or LR-FHSS.

a) First approach: As a first step, we propose an approach in which there is no preamble collision. The first approach will be extended later in this paper. Thus, the received signal sampled at $T'_s = \frac{1}{\alpha \times B_p}$, with $B_p = B$ the dedicated canal bandwidth and α the oversampling factor, is expressed as:

$$y_r(n) = \sum_{i=1}^{N_R} \sqrt{P_i} s_{p,i}(n - n_{s,i}) e^{j(2\pi(\Delta f_i + \frac{c_{d,i} \cdot nT'_s}{2})nT'_s + \varphi_{0,i})} + w(n) \tag{4}$$

with $n_{s,i} = \lfloor \frac{t_{s,i}}{T'_s} \rfloor = K_{s,i}\alpha M + \tau_{s,i}$, given that $\tau_{s,i}$ (resp. $K_{s,i}$) represents the time offset (resp. the index of the T-long initial preamble sequence) of the i-th received signal. In order to detect the presence of preambles, the receptor performs the following processes for each SF value:

1) Dechirping of $y_r(n)$ for each SF value,
2) FFT on the dechirping over $\frac{T}{T'_s}$ samples,
3) Computing of

$$T(k, p) = \sum_{j=p}^{p+N_p-1} \left| \frac{Y(k, j)}{\sigma_w} \right|^2 \tag{5}$$

with $k \in [\![0, M-1]\!]$, $p \in \{1, \ldots, N_B\}$ and $N_B \times T$ the buffer duration. $Y(k, j)$ corresponds to the k-th FFT bin of the dechirping of the j-th slice of $y_r(n)$ of duration T.

Given the significant values the Dopplers can take in the context of these communications, processes are performed in an oversampled mode compared to Nyquist, $\alpha > 1$. In

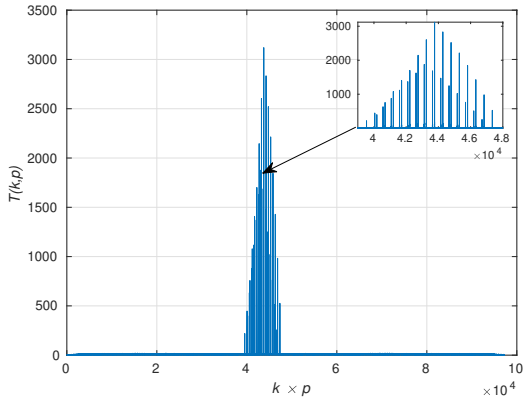

Fig. 1. $T(k,p)$ without interference, $SF = 7$, $B_p = 7.8$ kHz and $N_p = 8$

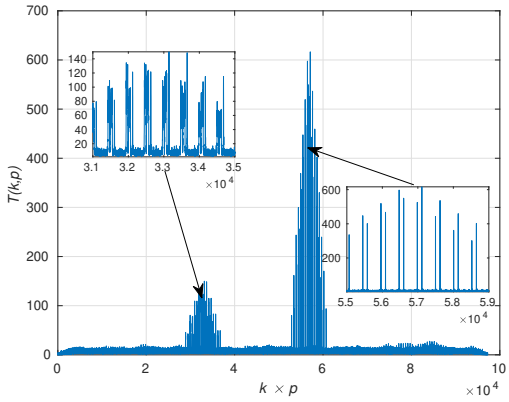

Fig. 2. Shape of $T(k,p)$ for $SF = 7$, $B_p = 7.8$ kHz and $N_p = 8$.

figure 1, we present the results obtained for the function $T(k,p)$ for the $SF = 7$ and $N_p = 8$ detection.

We observe in figure 1 that the detection of a given SF is associated with a specific pattern. As a matter of fact, at Nyquist (i.e. $\alpha = 1$) it is expected to find $2N_p - 1$ peaks in the pattern, considering that the interval between the peaks is M. Furthermore, an energy maximum is obtain for $p = K_{s,i}$, because all the FFTs contain the preamble. However, in an oversampling mode each FFT reveals 2 energy peaks (distant from M) related to the no signal aliasing and this even in the case of a perfect synchronization [7]. Thus, in order to identify the presence of a valid preamble, we are looking for this pattern.

For this purpose, starting from $T(k,p)$:

- We compute the function $M(p) = \underset{k}{max}(T(k,p))$, $p \in \{1, \ldots, N_B\}$,
- We search every energy peaks in $M(p)$ which exceed the threshold Th[1] respecting a minimum distance of N_p to ensure the detection of two different *patterns*. In other words, we try to find $\hat{K}_{s,i}$ an estimation of $K_{s,i}$ with: $\hat{K}_{s,i} = \underset{p}{argmax}(M(p))$,
- We verify that the energy peaks on the sequence $K_{s,i} - 1$ and $K_{s,i} + 1$ exceed the threshold and are located at the same frequency as the energy peak of the sequence $K_{s,i}$.

In the following, we propose an evolution of the detection method to take into account the interferences.

b) Second approach: In order to understand the inter-SF interference type effects on the first proposed approach, we represent in figure 2 the shape of $T(k,p)$ when a detection for $SF = 7$ and $N_p = 8$ is performed. We observe the wanted pattern but also a different pattern corresponding in this case to the presence of a $SF = 9$ and $N_p = 2$ preamble. Observing the latter, it can be seen that this $SF = 9$ preamble may cause a false detection. In order to limit this drawback and considering the pattern structure (many more energy peaks are present), we complete the detection with a constraint of type peak to average power ratio (PAPR) such that if $\frac{\underset{k}{max}(T(k,p))}{\underset{k}{mean}(T(k,p))}$, $p \in \{K_{s,i} - 1, K_{s,i}, K_{s,i} + 1\}$ exceeds a given threshold then the preamble is valid. This is true whatever the SF concerned by the interference.

When several preambles of same SF are interfering, it is necessary to perform a specific procedure. We propose to use the reception power of preambles to perform iteratively a successive suppression of interferences. In figure 3 is represented the shape of the function $T(k,p)$ when several preambles of same SF are in collision (here $SF = 9$ and $N_p = 2$). As it stands, our approach will detect only the preamble with the strongest received power. To overcome this limitation, the contribution of each valid preamble is suppressed from the function $T(k,p)$: the detected preamble energy peaks are reset to 0. Thus in this iterative way, we can detect several preambles in collision. In the current version of the algorithm, we fixed a maximum number of iterations N_{itr}. Figure 3 represents a result example after one iteration. The detected preambles have been deleted, allowing the process of preambles in collision.

In order to show the relevance of our approach, we first propose simulation results. Then a Real-Time version of the detector on synthetic data is developed.

B. Results and discussions

The aim of this section is to evaluate the detection performances of our algorithm for different system loads. We define the system load as $\frac{N_R T_p}{T_B}$, with $T_B = N_B T$ the duration of the recording being processed. Considering the bandwidth constraint of our application (lower than 10kHz), as we mentioned before, we propose to choose the bandwidth 7.8kHz. It should be noted that this bandwidth is supported by the LoRa chips [8].

In the upcoming simulations, we consider that the number of packets received in a buffer follows the Poisson

[1]The threshold calculus is not detailed due to lack of space, however it consists in performing a binary hypothesis test and using $T(k,p)$ as a test variable under the noise hypothesis.

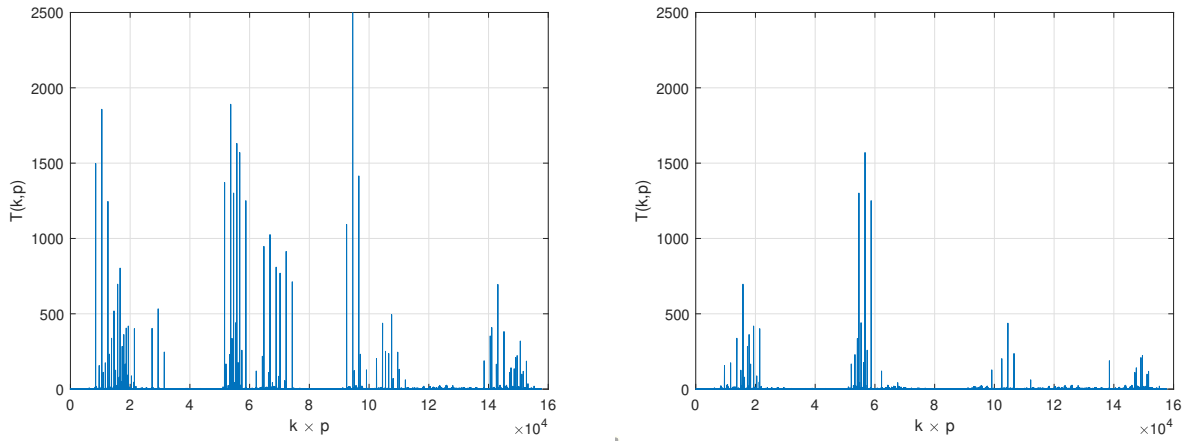

Fig. 3. Shape of $T(k,p)$, iterative process $SF = 9$, $B = 7.8$ kHz and $N_p = 2$ - (left) Initial state - (right) After 1 iteration

TABLE I
SIMULATION PARAMETERS

Carrier f_c (MHz)	868
Preamble bandwidth B_p (kHz)	7.8
CFO max Δf_{max} (kHz)	20
DR max DR_{max} (Hz/s)	300
Power spread PS (dB)	5
Mean SNR (dB)	0
Oversampling factor α	$\{2, 4\}$
Probability of false alarm P_{fa}	10^{-3}
Buffer duration T_B (s)	5
Number of iterations N_{itr}	$\{2, 3, 4\}$
Monte Carlo number	10000

distribution. The preamble power is uniformly distributed in a set coherent with the link budgets of our application. Considering the satellite FoV, the received power will be spread in a range of about 5dB (LEO satellite at 550km). This power spread will be noted PS. Simulation parameters are given in the table I.

The carrier frequency offset (CFO) and the Doppler rate (DR) are respectively uniformly distributed in $[-\Delta f_{max}, \Delta f_{max}]$ and $[-DR_{max}, DR_{max}]$.

We propose to evaluate the impact of the number of iterations and the oversampling factor on the preamble detection. For this purpose, figure 4 represents the probabilities of good and false detection as a function of the system load.

For $\alpha = 4$, we observe that the increase of N_{itr} improves the good detection rate while deteriorating slightly the false detection probability, in particular for low system loads. We also notice that for low system loads, the algorithm operates perfectly and that for high system loads, for example 2.6 (i.e. on average 100 preambles received in $T_B = 5$s), the good detection probability is 0.92 (resp. 0.9) for $N_{itr} = 4$ (resp. $N_{itr} = 3$).

Finally, we observe from figure 5 curves that the detection performances are inversely proportional to the increase of SF. This result is explained by the number of chirp in the preamble N_p, which is divided by 2 when increasing the SF by one in order to maintain the same preamble duration.

IV. REALTIME EVALUATION

To demonstrate that the proposed approach is workable in the nanosatellite context, a windowed version of the previously described detection algorithm was written in C++. To fulfill real time constraints, the SIMD features [9] of current ARM processors that are compliant with space usage were used. They improve the temporal characteristics of the detector implementation by parallelizing data processing to minimize the latency and thus the energy consumed. The use of intrinsic NEON [9] enabled in part the implementation thanks to the auto-vectorization techniques [10], [11] provided by GCC and LLVM tool chains. However, some critical parts of the detector algorithm had to be manually optimized, such as for instance the dechirping step, the argmax search step, as well as the FFT computation. For this last one, following a study of the performances of the numerous open-source solutions such as [12], we selected the *pfft* library [13].

The detectors thus developed have been deployed on two test platforms based on the use of ARM cores:

- $\mathcal{P}_1$ platform - the platform is composed of a 64-bit ARM Cortex-A72 processor (Raspberry Pi 4), whose clock frequency is set to 1.5 GHz. Its characteristics are similar to those of the hardened and radiation tolerant core (LS1046) and therefore spatializable in nanosatellites developed by Teledyne e2v (freq. 1.8 GHz) [14]. The ARM core has a L1 instruction cache of 48 KB and a L1 data cache of 32 KB Data while it has 1024 KB of L2 cache.
- $\mathcal{P}_2$ platform - the second platform is composed of a 32-bit ARM Cortex-A9 processor operating at a frequency of 650 MHz. The latter is from a Xilinx PYNQ Z2 board. This type of FPGA including an ARM core is also spatializable [15]–[17]. The ARM core has an

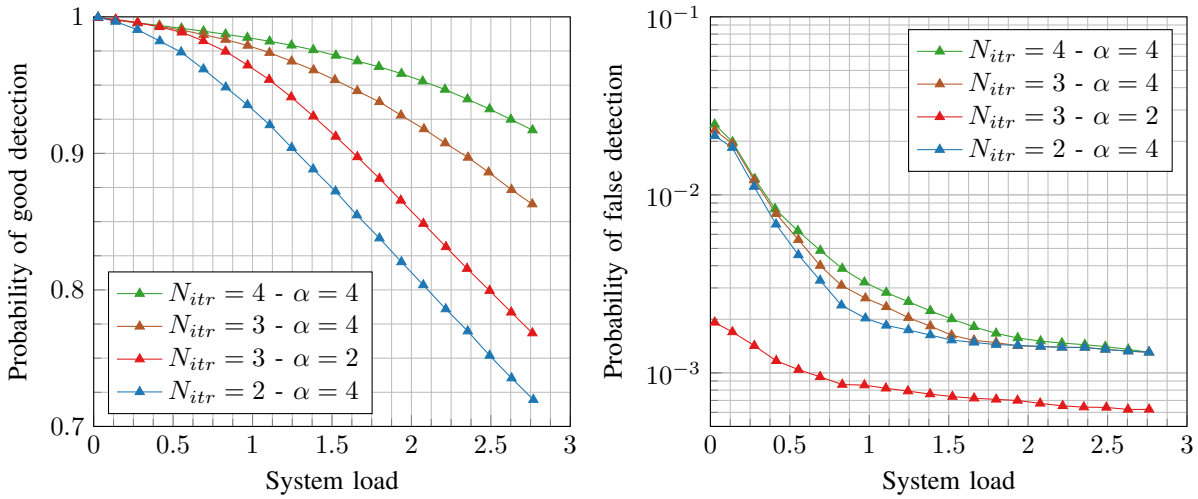

Fig. 4. Good and false detection probabilities for the proposed algorithm depending on system load, α and N_{itr} values ($SNR = 0$dB, $PS = \pm 5$dB).

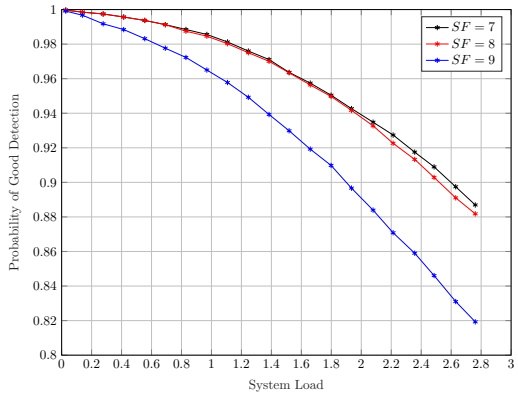

Fig. 5. Probabilities of good detection for the proposed algorithm depending on system load and SF value when $\alpha = 4$ and $N_{itr} = 3$ ($SNR = 0$dB, $PS = \pm 5$dB).

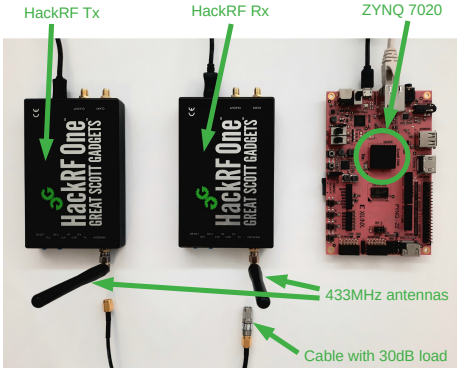

Fig. 6. Setup used: HackRF Tx sends IQ samples - HackRF Rx receives a signal - Detection algorithm runs on Zynq target hosted in Pynq Z2 board and detection is performed on HackRF Rx signal.

icache of 32 KB and a dcache of 32 KB while it has 512 KB of level 2 cache.

The behavior of the detector developed in C++ was first validated using the $\mathcal{P}_2$ platform and commonly used software defined radio modules such as ETTUS b205 and HackRF as shown in figure 6. Experiments have been carried out using optimized antennas for the 433 MHz ISM band, but also with 30 dB attenuators to simulate the communication channel. In Matlab, a script was written to generate the sequences of preambles sent using a first SDR module. Then, our program processes the samples received from the second SDR module using the algorithm proposed in this paper (section III-A).

The timing performances of the detector implementation were estimated with $SF \in [7, 8, 9]$ and $\alpha = 4$ while $B = 7.8$kHz in adequacy with the waveforms to detect. The detector works on a sliding window of duration equal to $4 \times Np$ chirps. It means that at each iteration of the detector, $\alpha \times Np_7 \times 2^7$ new IQ samples are added in the sliding window. The detectors have a variable execution time that vary according to the number of frames in the sliding window due to the iterative frame cancellation process. Consequently, in addition to the average execution time of the detector, we also measured the worst-case ones. Moreover, different scenarios were evaluated in terms of system load in a manner equivalent to what was proposed in Section III-B. It should be noted that we have simulated about 60 seconds of signal reception. The experimental results obtained for the platforms $\mathcal{P}_1$ and $\mathcal{P}_2$ are summarized in the Table II.

Platform $\mathcal{P}_1$ takes at most 4.4 ms to process a window and acquire next data when the system load is low (0.2). A worst-case execution time of 5.2 ms was measured for high system load (2.2). Knowing that the maximum execution time derived from the real time constraint is set to 131 ms, this software implementation on $\mathcal{P}_1$ fulfill requirements. Moreover, it requires only $\approx 5\%$ of the ARM core resources.

TABLE II
WINDOW PROCESSING TIMES WHEN $\alpha = 4$ AND $Nitr = 3$.

Load	Target	t_{min} (ms)	t_{max} (ms)	t_{avg} (ms)	t_{median} (ms)
Low (0.2)	$\mathcal{P}_1$	2.1	4.4	2.4	2.4
	$\mathcal{P}_2$	12.3	14.2	13.6	13.9
Medium (1.2)	$\mathcal{P}_1$	2.1	6.2	2.4	2.4
	$\mathcal{P}_2$	12.3	14.1	13.3	13.4
High (2.2)	$\mathcal{P}_1$	2.1	5.2	2.4	2.4
	$\mathcal{P}_2$	12.4	14.3	13.5	13.5

The results for the second Xilinx Zynq platform evolve similarly. The detector execution time ranges from 12.3 ms to 14.3 ms depending on the system load. This performance gap between $\mathcal{P}_1$ and $\mathcal{P}_2$ is mainly explained by the $2\times$ difference in operating frequency between the platforms. However, even on the $\mathcal{P}_2$ platform, the real time constraint is respected with a worst-case time of 14.3 ms which is only $\approx 11\%$ execution time available to respect real-time constraint on the ARM Cortex-A9 core. These evaluations demonstrate the real-time capability of the presented detection algorithm in spatial context. This acknowledgment is valid even for low-end platforms, such as the $\mathcal{P}_2$ platform.

To further improve the timing performances of the presented detectors, several ways are still open. The first one would consist in transforming the software descriptions which manipulate floating-point data into fixed point to speed-up the processing times and to minimize the memory footprint. Moreover, for the platform $\mathcal{P}_2$ integrating an FPGA, it is possible to easily deport, for example, the FFT processing [18] to hardware accelerators using HLS synthesis tools [19]–[21] or existing IPs. This approach could also be applied to other time-consuming processing tasks to minimize energy consumption and power dissipation.

V. CONCLUSION

In this paper, we have presented a system for detecting IoT communications from a nanosatellite. The proposed low complexity algorithms that compose the receiver have been first theoretically evaluated. They enable to detect LoRa frames interferences with different SF values but also inter-SF interferences. Simulations demonstrated the efficiency of the approach and evaluated the impact of parameter values. In a second time, the timing performances of algorithm implementation on ARM platforms allowed to validate the approach in an experimental way thanks to ETTUS and HackRF SDR modules. Real-time performances were obtained thanks to a SIMD parallelization of the detection algorithm whose execution time represents less than 10% of capabilities of the two spatializable ARM cores (ARM Cortex-A9 & A53). Future work will focus on the development of hardware accelerators and the deployment of the detectors on our industrial partner's Zynq UltraScale+ target to carry out tests under real space conditions.

REFERENCES

[1] M. Ben Temim, G. Ferré, and R. Tajan, "A new lora-like transceiver suited for LEO satellite communications," *Sensors*, vol. 22, 2022.

[2] J. Luo, A. Ito, A. Sasaki, M. Hasegawa, Y. Nagao, Y. Hiramatsu, K. Torii, S. Ashibe, and T. Aoki, "A study on adjacent interference of lora," in *Proceedings of CANDARW*, 2020.

[3] G. Ferre, "Collision and packet loss analysis in a lorawan network," in *Proceedings of EUSIPCO*, 2017.

[4] A. A. TESFAY, E. P. SIMON, G. FERRÉ, and L. CLAVIER, "Serial interference cancellation for improving uplink in lora-like networks," in *Proceedings of PIMRC*, 2020.

[5] A. N. de São José, N. Chopinet, E. P. Simon, A. Boé, T. Vantroys, C. Gransart, and V. Deniau, "A comparative analysis of lora and lorawan in the presence of jammers and transient interference," in *Proceedings of EMC Europe*, 2022.

[6] STMicroelectronics, "Ultra-low power, high performance, sub-1 GHz transceiver," 2021. [Online]. Available: https://www.st.com/resource/en/datasheet/s2-lp.pdf

[7] G. Colavolpe, T. Foggi, M. Ricciulli, Y. Zanettini, and J. Mediano-Alameda, "Reception of LoRa signals from LEO satellites," *IEEE TAES*, 2019.

[8] LoRa Alliance, "RP2-1.0.2 LoRaWAN Regional Parameters," 2020. [Online]. Available: https://lora-alliance.org/resource_hub/rp2-102-lorawan-regional-parameters/

[9] Arm, "Neon Overview," https://developer.arm.com/Architectures/Neon, 2023, last accessed 2022-04-14.

[10] M. Alvanos and P. Trancoso, "Video SIMDBench: Benchmarking the compiler vectorization for multimedia applications," in *Proceedings of DSD*, Limassol, Cyprus, 2016.

[11] O. V. Moldovanova, M. G. Kurnosov, and A. Mel'nikov, "Energy efficiency and performance of auto-vectorized loops on intel xeon processors," in *Proceedings of RPC*, Vladivostok, Russia, 2018.

[12] M. Frigo and S. Johnson, "The Design and Implementation of FFTW3," *Proceedings of the IEEE*, vol. 93, no. 2, 2005.

[13] J. Pommier, "Pffft, a pretty fast fourier transform," https://bitbucket.org/jpommier/pffft/src/master/, 2011, last accessed 2023-04-14.

[14] *The Future is Now: Teledyne e2v's Quad Core ARM Cortex-A72 Radiation Tolerant Microprocessor Revolutionizes Heavy Computing for Space Systems & Satellite Projects*, Teledyne e2v, 2020.

[15] B. P. Acosta, E. Angel Guzmán, J. A. Osuna Valdez, L. Rizo Domínguez, M. Mendoza Barcenas, and M. I. R. Prieto Meléndez, "Adaptable o-board computer for nanosatellites," in *Proceedings of UEMCON*, NY, USA, 2022.

[16] R. Camarero, L. O. Espluga, and A. Ressouche, "Lossless image compression on the eyesat nanosat," in *Proceedings of the On-Board Payload Data Compression Workshop*, Rome, Italy, 2016.

[17] C. M. Fuchs, N. M. Murillo, A. Plaat, E. van der Kouwe, D. Harsono, and T. P. Stefanov, "Fault-tolerant nanosatellite computing on a budget," in *Proceedings of RADECS*, Goteborg, Sweden, 2018.

[18] H. Almorin, B. Le Gal, J. Crenne, C. Jego, and V. Kissel, "High-throughput FFT architectures using HLS tools," in *Proceedings of ICECS*, 2022.

[19] Xilinx, "Vitis High-Level Synthesis User Guide," https://docs.xilinx.com/r/en-US/ug1399-vitis-hls/Introduction, 2022, last accessed 2023-04-14.

[20] Microchip, "SmartHLS™ Compiler Software," https://www.microchip.com/en-us/products/fpgas-and-plds/fpga-and-soc-design-tools/smarthls-compiler, 2022, last accessed 2023-04-14.

[21] Siemens, "Catapult High-Level Synthesis and Verification," https://eda.sw.siemens.com/en-US/ic/catapult-high-level-synthesis/hls/c-cplus/, 2022, last accessed 2023-04-14.