Low Complexity LoRa Frame Synchronization for Ultra-Low Power Software-Defined Radios

Carolynn Bernier, François Dehmas, Nicolas Deparis

Low Complexity LoRa Frame Synchronization for Ultra-Low Power Software-Defined Radios

Carolynn Bernier, François Dehmas, Nicolas Deparis

Abstract

Low power wide area (LPWA) wireless networks based on the LoRa physical layer have attracted huge attention in recent years, both from industry and from academic researchers. While this rising popularity is due to this technology's demonstrated effectiveness and low cost, unfortunately, due to their complexity, the timing and frequency synchronization algorithms required to detect LoRa-modulated frames, in the context of minimum sampling rate optimum receivers, have received little attention. The aim of this paper is to fill this gap and describe how robust frame detection can be performed while focusing on minimal complexity implementations of the proposed algorithms. The ultimate goal is to propose frame detection techniques applicable to recently proposed ultra-low power software-defined receivers.

Index Terms

Chirp modulation, frequency shift chirp modulation (FSCM), Internet of Things (IoT), LoRa, low power wide area networks (LPWAN)

C. Bernier is a member of the Architectures, Circuits and Embedded Software Department and F. Dehmas and N. Deparis are with the Systems Department of CEA, LETI, Grenoble, France.

I. INTRODUCTION

Low power wide area (LPWA) wireless networks are gaining large-scale industrial acceptance and enabling new smart applications in verticals such as transportation, health, industry and agriculture. With an expected shipment of 350 million compatible nodes in 2022, and with a large number of compatible gateways already deployed on a global scale, LoRa (short for "Long Range") is an increasingly popular modulation scheme for LPWA communications [1]. This growing popularity has, in its turn, spurred a quick reaction from the research community. Indeed, a recent review of research work published from 2015 to September 2018 and concerning either LoRa or LoRaWAN, a medium access control (MAC) communication protocol based on the LoRa physical layer, shows that approximately 2000 papers have been published in this short time, clearly demonstrating the importance of this new technology to the research community [2]. A non exhaustive list of LoRa and LoRaWan-based research includes areas such as physical layer evaluation in the presence of interference, coverage tests, capacity evaluation, models for network level simulators and applications and deployments.

The fact that LoRaWAN's specifications are available in open access has clearly been beneficial to the research community. On the contrary, many details about the LoRa physical layer itself remain trade secrets. However, the importance of LoRa in the LPWA landscape prompted efforts in the IoT research community to reverse engineer the LoRa physical layer and share this information publicly [3][4][5][6]. A better understanding of the LoRa modulation format indeed enables new research activities such as the development of new localization algorithms, the development of accurate physical-layer models, the evaluation of potential security breaches, and the invention of further physical layer improvements.

Another motivation for gaining a better understanding of the LoRa physical layer is the development of LoRa-compatible demodulation software for recently proposed ultra-low power software defined radios (ULP-SDR) [7][8][9]. Indeed, compared to today's commodity IoT transceivers which are mostly implemented in hardware, software-based wireless transceivers enable the implementation of different physical layers on the same hardware. With the uncertain evolution of LPWA networks and standards, software transceivers also minimize development

cost, enable multi-standard and multi-mode applications and future-proof integrated circuit designs. Finally, software transceivers also make it possible to develop precise link quality information extraction algorithms directly within the receiver's digital baseband [10][11].

Unfortunately, while the rising popularity of LoRa-based technology is clearly due to its accessibility, i.e. low cost and simplicity of deployment, the signal processing required to recover LoRa modulated signals is, on the contrary, relatively complex. Using the terminology proposed in [12], LoRa employs a Frequency Shift Chirp Modulation (FSCM) in which the information is encoded by a frequency shift applied to a constant chirp rate symbol (a chirp is a frequency modulated signal). Before being able to demodulate the received symbols and recover the data, a LoRa receiver must also compensate for sampling, carrier frequency and symbol timing offsets that are due to unsynchronized timing references between the transmitter and the receiver. To date, little work has been published concerning the frame synchronization procedure for FSCM-modulated frames, and for LoRa frames in particular.

The purpose of this paper is to contribute to the understanding of the preamble and start-of-frame synchronization requirements of FSCM signalling schemes. In view of a potential implementation on an ULP-SDR receiver, the focus of this paper is on low complexity frame synchronization algorithms. In particular, we explain how the use of both up and down base modulated chirps within the frame's preamble is used to resolve integer symbol timing and carrier frequency offset ambiguity. From this, we deduce the maximum carrier frequency offset (CFO) that can be tolerated by the receiver. We provide simulation and measurement results showing the relationship between number of received preamble symbols and both frame detection performance and fractional CFO estimation error. Finally, we propose several ideas for lowering the complexity of certain synchronization mechanisms, such as the detection of data sample start time.

This paper is organized as follows: We start in Section II by a mathematical description of the FSCM modulation and its frame synchronization requirements assuming Nyquist rate reception. We also give two fundamental synchronization algorithms. In Section III, we discuss related work on FSCM frame synchronization. Section IV is specifically dedicated to the detection of LoRa modulated frames. Finally, Section V proposes a number of ideas for implementing new,

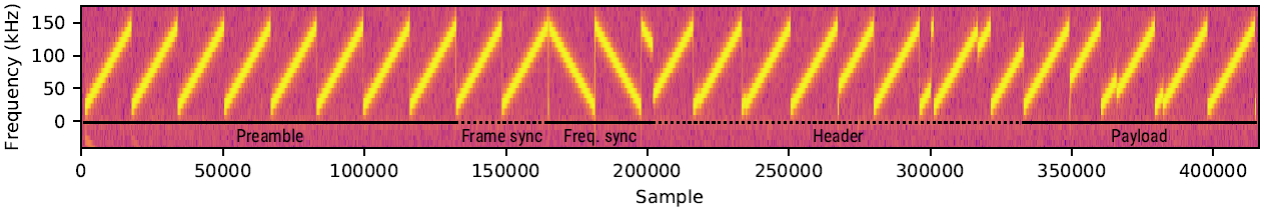

Fig. 1. Annotated spectrogram of an example LoRa signal (with settings SF=11, BW=125 kHz) Reproduced with permission from [3].

low complexity synchronization algorithms for FSCM, in particular in the light of upcoming ULP-SDR transceivers.

II. FSCM MODULATION

Digital communication schemes based on linear frequency modulated (i.e. linear chirp) signals have been in use for many years in applications ranging from military communications to short-range personal area networks [13][14]. While past modulation schemes encoded information either by varying the *chirp rate*, i.e. the rate at which the RF carrier frequency is varied, between a set of possible values or by using two signals with opposite chirp rates (often named *up-chirps* and *down-chirps*), in the LoRa physical layer, the information bearing element is a frequency shift applied at the beginning of each constant rate chirp. Thus, the name frequency shift chirp modulation (FSCM) assigned to the LoRa modulation by [12].

The FSCM modulation employed in the LoRa signalling scheme is an orthogonal modulation with symbols encoded using a set of N cyclically shifted versions of a base Zadoff-Chu (ZC) sequence. The general expression for ZC sequences is defined as follows [15]:

$$u_M[k] = \begin{cases} e^{\frac{j\pi.M.k(k+1)}{N}}, & k = 0, 1...N-1 \quad \text{if N is odd} \\ e^{\frac{j\pi.M.k^2}{N}}, & k = 0, 1...N-1 \quad \text{if N is even.} \end{cases} \tag{1}$$

If M and N are relatively prime, i.e. $gcd(M, N) = 1$, the auto correlation of a ZC sequence with all $N-1$ cyclically shifted versions of itself is zero for all values of n different from zero:

$$R_{MM}[n] = \frac{1}{N}\sum_{k=0}^{N-1} u_M[k]u_M[(k+n) \bmod N]^* = \begin{cases} 1 & \text{for} \quad n = 0 \\ 0 & \text{for} \quad n \neq 0 \end{cases} \tag{2}$$

In order to simplify the extraction of the modulation value from of the set of N cyclicly-shifted versions of a *root* (also called *base*) ZC sequence, $B[k]$, it is advantageous to choose $M = 1$. Adjusting this base sequence such that its normalized frequency covers the span $[-0.5, 0.5]$, we define $B[k]$ as follows:

$$B[k] = e^{j2\pi(\frac{k^2}{2N} - \frac{k}{2})}, \quad k = 0, .., N-1 \tag{3}$$

This signal corresponds to a linear frequency modulated signal with frequency slope, or chirp rate, equal to $1/N$ and with an initial normalized frequency of -0.5. N sequences, $S_{N_0}[k]$ with $N_0 = 0, .., N-1$, are produced through N cyclic shifts of $B[k]$:

$$S_{N_0}[k] = e^{-j\pi N_0(N_0/N-1)} B[(k+N_0) \bmod N], \quad k = 0, .., N-1 \tag{4}$$

In the above equation, the exponential term is necessary to set each symbol's initial and final phase to zero, enabling a continuous phase modulation. This equation simplifies to [16]:

$$S_{N_0}[k] = e^{j2\pi(\frac{k^2}{2N} + k(\frac{N_0}{N} - \frac{1}{2}))}, \quad k = 0, .., N-1 \tag{5}$$

If a signalling bandwidth of size BW is allocated to the system, the minimum sampling frequency, $f_{s_{min}}$, is equal to BW.

A. FSCM demodulation in ideal synchronization conditions

Assuming perfect time and frequency synchronization, optimum non coherent demodulation is performed by first multiplying the received symbol $S_{N_0}[k]$ by the conjugate of the base sequence:

$$S_{N_0}[k]B^*[k] = e^{j2\pi k\frac{N_0}{N}}, \quad k = 0, .., N-1 \tag{6}$$

The symbol value N_0 is extracted from the resulting constant frequency signal using an FFT and locating the frequency index, referred to as *bin* in the following, of the peak value of

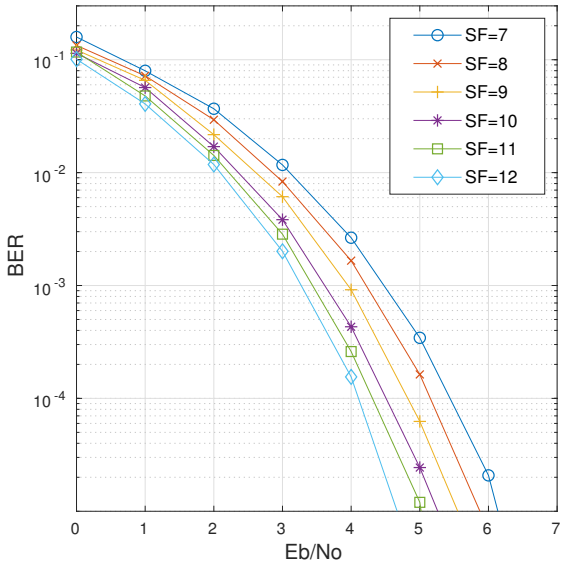

Fig. 2. Ideal non-coherent FSCM demodulation performance for $N = 2^{SF}$ and $SF = 7, .., 12$

the FFT magnitude (an operation referred to as argmax). It is shown in [12] that this is the optimal receiver. With N possible symbols, a maximum of $log_2(N)$ bits can be encoded within each symbol. Of course, it is always possible to use symbol redundancy to improve the link robustness, at the cost of information-carrying capacity. The choice of this modulation versus other modulations commonly employed in LPWA communication schemes (such as BFSK) is justified by the high energy efficiency (minimum energy per bit versus noise density, Eb/No) obtained for high modulation orders, as shown on Figure 2 [17].

B. Nyquist-rate receivers

Differently from the reverse engineering efforts mentioned in the introduction and which employ high sampling rate USRP (Universal Software Radio Peripheral) receivers to capture LoRa-modulated frames (such as in Figure 1), the IoT context requires transceivers designed for ultra-low power consumption. Thus, in practical receivers, in order to minimize power consumption, the down-converted signal is decimated down to its minimum sampling rate $f_{s_{min}}$. In addition, both memory usage and computational complexity of the digital baseband processing algorithms must be kept to a minimum. Finally, received samples must be processed in close

to real time in order to minimize delay. The synchronization algorithms discussed in this work will focus on low complexity, minimum sampling rate receivers.

C. FSCM frame synchronization requirements

Frequency and timing synchronization are difficult to guarantee in practice due to offsets in the frequency references used by the transmitter and receiver. More precisely, these offsets will result in:

- a carrier frequency offset, CFO, which can be separated into two components: CFO_{int} and CFO_{frac}, which are, respectively, the integer and fractional parts of $N \times CFO/BW$,
- an initial symbol timing offset, STO, which can be separated into two components: STO_{int} and STO_{frac},
- an ambiguity concerning the (optional) header or payload start time, as discussed in Section V-F,
- a sampling frequency offset, SFO, which, if uncorrected, will generate an incremental symbol timing offset,
- and, potentially, since both the transmitter and receiver's quartz-based references are susceptible to drifts due, for example, to changes in temperature (drifts on the order of a few tens of Hz/s are common), a receiver may also have to compensate for changes in the CFO and SFO. If the RF synthesizer and sampling clocks are generated from the same crystal-based reference clock, CFO and SFO will suffer from correlated drifts. This effect can be particularly severe for long frames. Drift compensation methods will not be discussed in this work.

The impact of CFO and STO is illustrated in Figure 3 in which we observe the frequency component of the received modulated signal prior to sampling and after sampling at a normalized rate of $1/N$. First, we observe that CFO will shift the signal out of the receiver's reception bandwidth, defined by in the interval $[-N/2, .., N/2]$. Sampling at a normalized minimum rate of $1/N$ folds the signal back into the receiver's input bandwidth which, in Figure 3, has the effect of reconstructing complete *up-chirps*. We observe that CFO (measured in Hz) has an integer and fractional component, $CFO = CFO_{int} + CFO_{frac}$. In the example of Figure 3, the

signal is received with a CFO (vertical shift) equivalent to 3.5 frequency bins. Both CFO_{int} and CFO_{frac} must be recovered by the synchronization algorithm. Indeed, as discussed in [18], at low signal to noise ratios (SNR), CFO_{frac} will shift the FFT outputs between two integer frequency bins, resulting in demodulation errors.

Next, in the absence of synchronization, after sampling, the receiver has no way of identifying the start of the received sequence, resulting in an STO consisting of an integer number of samples, STO_{int} (with $STO_{int} < N$) plus a fraction of a sample, STO_{frac}. In the example of Figure 3, the signal is received with an STO (horizontal time shift) equal to 7.2 samples. Recovering STO_{int} is mandatory for correct alignment to the modulated symbols and compensating for STO_{frac} is necessary for concentrating the symbol energy within a single FFT bin. Most importantly, we observe that, simply by extracting the start times of the reconstructed *up-chirps*, it is impossible to resolve the timing ambiguity caused by the simultaneous impact of STO and CFO. Attempting to synchronize the receiver using the reconstructed *up-chirps*, such as in [16], will result in limited CFO performance. Nor can this ambiguity be lifted by detecting the start of the first preamble symbol using a power meter, the signal being often received at low or even sub-zero SNR conditions.

Next, assuming that both CFO and STO have been completely recovered, it is possible that an ambiguity concerning the start time of the samples corresponding to the (optional) header or payload symbols remain, leading to a false synchronization decision and a resulting packet drop. Resolving this ambiguity in a minimum complexity receiver implementation can represent a challenge which is discussed in Section V-F.

Finally, Figure 4 shows the impact of SFO on an otherwise perfectly synchronized signal. In the figure, we observe the effect of a receiver with a slightly higher sampling frequency to that of the emitter. We observe a cumulative sampling offset which has the effect of slowly moving the samples off of the desired integer frequency bins. If this offset is not corrected, the demodulator will start confusing a symbol (identified by its frequency bin) with its next neighbor. This is an important problem for high order modulations. For example, the highest order modulation in LoRa has $N = 4096$. Assuming low cost quartz crystals with a ± 20 ppm precision are used in both emitter and receiver, a potential worse case offset of 40 ppm can

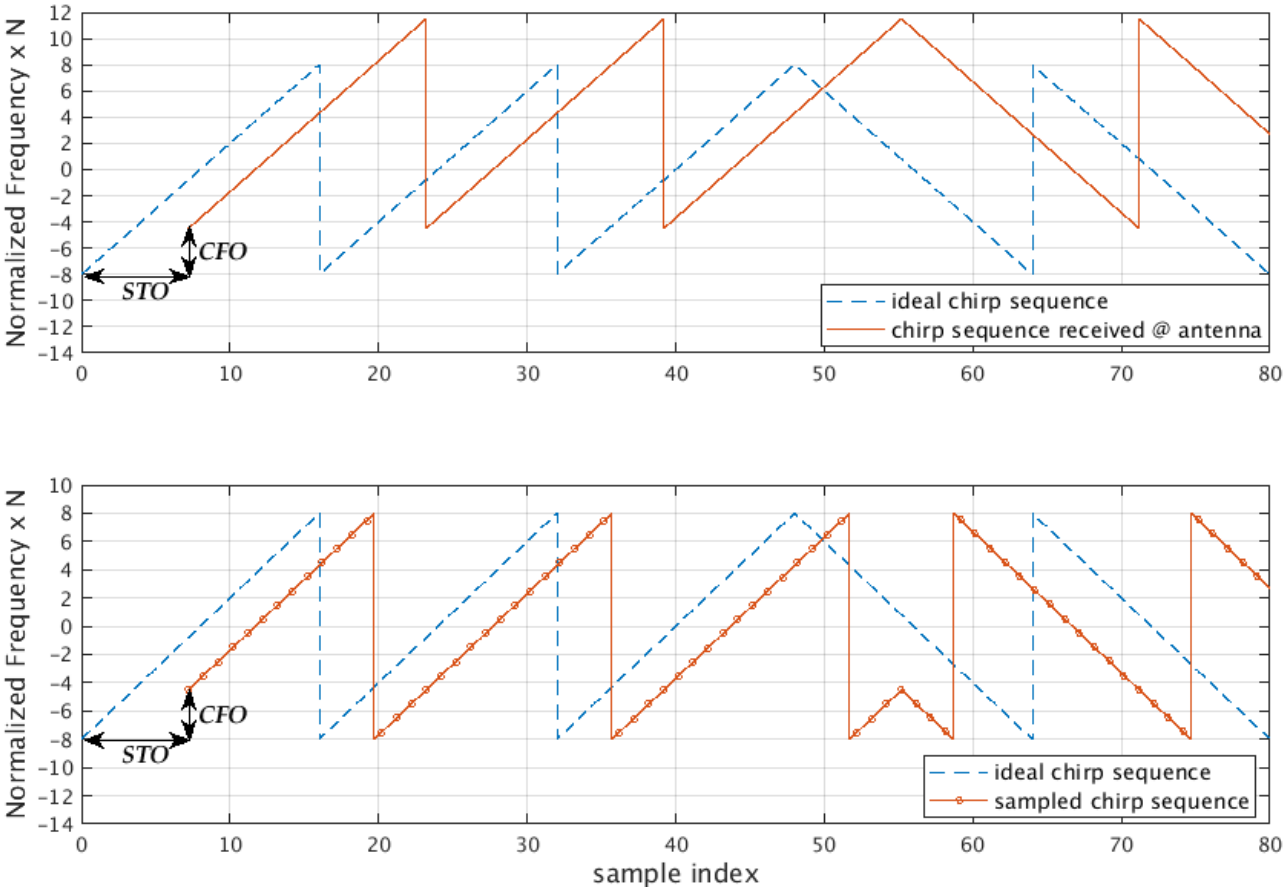

Fig. 3. Plot of $N*$ normalized frequency component of $S_{N_0}(n)$ assuming $N = 16$ and $N_0 = 0$ and for a signal composed of three *up-chirps* and two *down-chirps*. Top: signal received at the antenna suffering from an STO of 7.2 samples and a CFO of 3.5 frequency bins. Bottom: same signal after sampling at a normalized rate of $1/N$ (assuming $SFO = 0$).

lead to a drift of 0.16 sample after a single symbol. While this effect can sometimes be ignored during preamble acquisition, for example if the preamble is relatively short, this effect must imperatively be corrected during the demodulation phase of frame detection. An estimation of SFO can be extracted from CFO if the source of both offsets, the quartz crystal reference, is identical.

D. Fundamental frame synchronization algorithm

As seen previously, in minimum sampling rate receivers, it is impossible to distinguish between STO and CFO. Luckily, the presence of both *up-chirps* and *down-chirps* in a synchronization

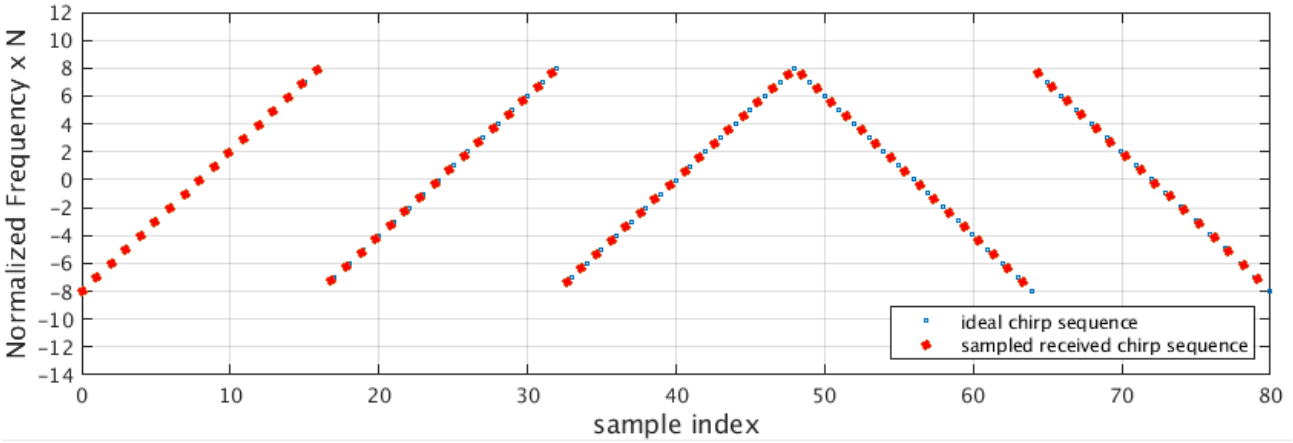

Fig. 4. Plot of $N*$ normalized frequency component of $S_{N_0}(n)$ assuming $N = 16$ and $N_0 = 0$ and for a signal composed of three *up-chirps* and two *down-chirps*. Signal after sampling at a normalized rate of $1/N$ but with a small SFO (assuming STO and $CFO = 0$).

preamble leads to an elegant solution for extracting the integer part of both values. Neglecting noise, SFO and the fractional parts of both STO and CFO, and assuming that the first part of the preamble is a base *up-chirp*, the received signal, $r[k]$, can be written as follows:

$$r[k] = A \times e^{j2\pi(\frac{k^2}{2N}+k(\frac{(N-STO_{int})}{N}-\frac{1}{2}))} \times e^{j2\pi kCFO_{int}/BW+j\phi_o} \tag{7}$$

where ϕ_o is the carrier phase offset and A is the signal amplitude. Indeed, integer STO has the same effect as symbol modulation. Multiplying by the conjugate of $B[k]$ produces:

$$r[k] \times B[k]^* = A \times e^{j2\pi k(\frac{CFO_{int}}{BW}+\frac{(N-STO_{int})}{N})} \times e^{j\phi_o} \tag{8}$$

Thanks to a Fourier Transform applied on this signal and the extraction of argmax, we can extract the normalized frequency $f_{up} = CFO_{int}/BW + (N - STO_{int})/N$. Now assume that in a following moment of the preamble, a *down-chirp*, necessarily suffering from the same CFO and STO as the previous signal, is received:

$$r[k] = A \times e^{-j2\pi(\frac{k^2}{2N}+k(\frac{(N-STO_{int})}{N}-\frac{1}{2}))} \times e^{j2\pi kCFO_{int}/BW+j\phi_o} \tag{9}$$

Multiplying by $B[k]$ produces:

$$r[k] \times B[k] = A \times e^{j2\pi k(\frac{CFO_{int}}{BW} - \frac{(N-STO_{int})}{N})} \times e^{j\phi_o} \tag{10}$$

Similarly, we extract the normalized frequency $f_{down} = CFO_{int}/BW - (N - STO_{int})/N$. Thanks to these two equations, the two unknowns, STO_{int} and CFO_{int}, can be estimated ($\widehat{STO}_{int}$ and $\widehat{CFO}_{int}$). Of course, this assumes that these calculations are applied in the moments when the receiver knows when to expect *up* and *down* chirps.

Interestingly, the above analysis can be used to extract the maximum CFO range that can be recovered by the receiver. Since f_{up} and f_{down} are normalized frequencies, they are defined modulo 1, i.e. any value v exceeding 1 will become $v \bmod 1$. Combining the two equations above, we find that $CFO_{int}/BW = (f_{down} + f_{up})/2$. Since $f_{down} + f_{up}$ is also a modulo 1 normalized frequency, CFO_{int}/BW can only be defined modulo 1/2. Thus, the estimation of CFO_{int} is only defined modulo $BW/2$. This means that the receiver will be able to recover a CFO limited to the range $[-BW/4, BW/4]$. For example, assuming $BW = 125$ kHz and a 868 MHz carrier, acceptable CFO must remain below 36 ppm.

E. Extracting CFO_{frac}

As stated previously, the presence of a fractional CFO will cause a loss in sensitivity at low signal to noise ratios. Thus, compensating for CFO_{frac} will recenter the FFT outputs onto integer frequency bins. The extraction of CFO_{frac} can be achieved simply during the reception of two consecutive identical symbols, e.g. two *up-chirps* or two *down-chirps*. Assuming that these two symbols are *up-chirps*, each of these symbols is processed as in (8) but this time the resulting frequency component is $(CFO_{int} + CFO_{frac})/BW + (N - STO_{int})/N$. Since this frequency is identical for both symbols, the same FFT bin, f_{up} will be selected by the argmax function. The signal present in the FFT bin corresponding to f_{up} can be expressed as:

$$e^{j2\pi k f_{up} + j2\pi k \frac{CFO_{frac}}{BW}} \tag{11}$$

This can be seen as a signal with constant frequency f_{up} but with a time-varying phase. Therefore, if the phase, respectively ϕ_1 and ϕ_2, of the signal present in this FFT bin is extracted

for these two consecutive identical symbols, we can write:

$$\phi_2 - \phi_1 = \frac{CFO_{frac}}{BW}(k+N) - \frac{CFO_{frac}}{BW}(k) \tag{12}$$

from which we find $CFO_{frac} = BW(\phi_2 - \phi_1)/N$. This technique is applicable as long as the two consecutive symbols are identical.

III. Related work

The reverse engineering efforts mentioned previously [3][4][5][6] employ wide-band software-defined radios (SDR) and over-sample captured frames emitted by LoRa-compatible RF transmitters. These samples can be stored in memory and post-processed by powerful CPU's running potentially complex synchronization and demodulation algorithms. In [3], a synchronization algorithm is proposed based on cross-correlations of the signal's instantaneous frequency. Unfortunately, this algorithm is only effective at high signal-to-noise ratios, severely limiting the sensitivity of the receiver, and has a complexity $O(N^2)$. The patent in [18] proposes an FFT-based demodulation algorithm for chirp modulated signals and, to the authors' knowledge, is the first to mention the impact of CFO_{frac} on performance. The compensation algorithm proposed is however much too complex for low cost, low power transceivers. The authors in [16] study optimal receiver algorithms, which have $O(N \log N)$ complexity, for a minimum rate receiver. However, they do not resolve the time/frequency ambiguity discussed above leading to a limited capacity for CFO compensation. In addition, the algorithm proposed for CFO_{frac} estimation is less precise and needlessly complex compared to the one presented in Section II-E, as will be discussed in Section V-D. Finally, they do not address STO_{frac} compensation. While publicly available, the algorithms described in [19] for minimal rate, optimal LoRa synchronization are difficult to understand. Providing a clear explanation of these algorithms is the focus of the following section.

IV. LoRa frame synchronization

The techniques presented in II-D and II-E are the basic functions necessary to recover STO_{int}, CFO_{int} and CFO_{frac} which are imperative to accurately achieve frame synchronization in sub-

zero SNR conditions. When and how a given receiver actually extracts this information from the received signal is implementation dependent. This section focuses on the acquisition of frames that follow the format defined in the LoRa physical layer.

A. LoRa frame format

In the FSCM modulation employed in LoRa, N is always a power of 2 since this eases FFT-based detection. For a given BW, an adaptive modulation is proposed allowing N to take on the value 2^{SF}, with $SF = \{7, 8, 9, 10, 11, 12\}$. In the context of LoRa, SF is referred to as 'spreading factor'.

Since we are interested in the synchronization phase of frame acquisition, referring back to Figure 1, we will focus on the synchronization header including the 'preamble', 'frame synchronization' (also called *sync word*) and 'frequency synchronization' symbols of LoRa frames. The preamble consists of a minimum of 2 and a maximum of 65535 un-modulated *up-chirps*. Next, while [20] mentions that the frame synchronization symbols consist of 2 symbols of value $\{x, N - x\}$, frame acquisitions made by [4] show that these symbols actually consist of two identical consecutive symbols $\{x, x\}$, the value of which is defined by the settings of the transceiver (note that they can be set to zero, making them identical to the preamble symbols). These symbols can be used to uniquely identify a network thus easing the filtering out of unwanted frames. Finally, the frequency synchronization symbols consist of 2.25 *down-chirps* symbols. Depending on the synchronization algorithm employed, the last quarter-symbol can be used by the receiver to apply the required time and frequency compensations before the start of data demodulation.

B. LoRa frame synchronization algorithm

The information present in [19] can be used to reconstruct the synchronization algorithm present in state of the art receivers. A schematic overview of this algorithm is presented in Figure 5 in which the samples are processed from left to right while the different steps of the algorithm, corresponding to the different phases of frame synchronization, flow from top to bottom, with some processes occurring simultaneously as will be described below. While

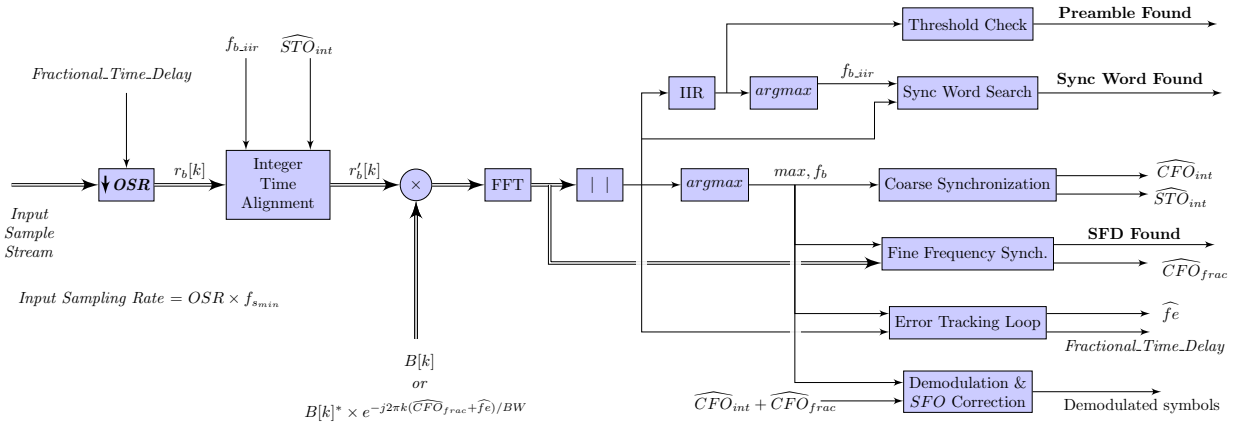

Fig. 5. Conceptual view of LoRa start-of-frame synchronization algorithm. Double arrows indicate complex signals.

existing commercially available LoRa transceivers can be programmed to emit and receive frames containing only two preamble symbols[1], measurements show that in order to synchronize correctly, in practice the receiver needs a minimum of 4 preamble symbols (Figure 8). In addition, measurements show that if the received frame contains more preamble symbols than expected, the frame is rejected. Figure 6 shows an example as seen by the emitter (top) and by the receiver (middle and bottom) of a worse-case synchronization scenario, i.e. the emitted frame contains only two preamble symbols and therefore only a single block of received samples, $r_1[k]$, contains a complete preamble symbol (for simplicity, $N = 16$). In this example, we choose frame synchronization symbols with modulation value of 2, meaning that the chirp's initial frequency is offset by 2 integer frequency bins with respect to the base chirp. The signal as seen by the receiver will suffer from STO and CFO thus resulting in a random circular frequency shift. In the example presented in Figure 6, a shift of +6 bins is applied to the signal in the top graph to obtain the signal in the middle and bottom graphs. These two graphs represent different synchronization algorithms, as will be discussed below.

1) Preamble synchronization: When the receiver is activated, it starts receiving samples in successive non-overlapping windows (referred to as 'blocks') of size N, denoted $r_b[k]$, with b the block index. These are processed as in (13) in order to extract consecutive FFT bin indices:

[1]Here we use the same terminology as in Figure 1.

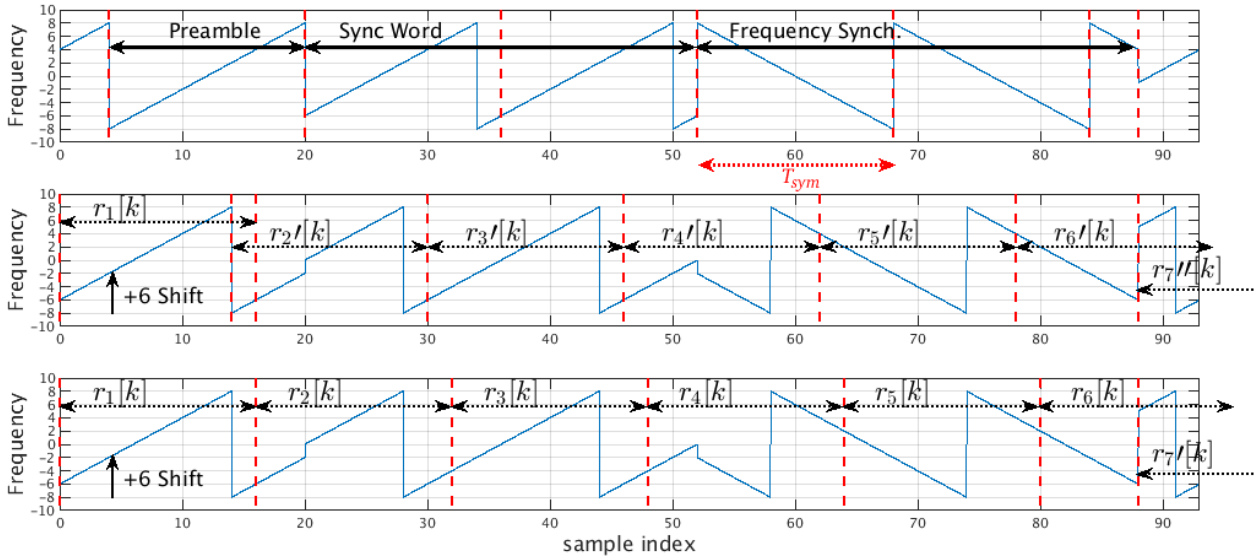

Fig. 6. Plot of $N*$ normalized frequency component of the baseband signal, assuming $N = 16$. Top: synchronization header as seen by the transmitter and assuming frame synchronization symbols with modulation value of +2. Middle and bottom: the same signal as seen by the receiver after a shift of +6 bins (assuming $SFO = 0$ and $CFO_{frac} = 0$).

$\{..., f_b, f_{b+1}, f_{b+2}, ...\}$[2]:

$$f_b = argmax(|FFT(r_b[k] \times B[k]^*)|) \tag{13}$$

Since each block of samples is not synchronized with the emitted symbols, they necessarily contain samples that belong to chirp fragments of two different symbols. In high SNR conditions, the result of (13) will reflect the frequency offset due to the larger chirp fragment. For example, if we consider blocks $r_1[k]$ and $r_2[k]$ from the bottom graph of Figure 6, these would produce $f_1 = 2$ and $f_2 = 4$. As discussed previously, the frequency offset measured in $r_1[k]$ is the result of both STO and CFO which cannot be distinguished at this point. However, f_1 can be used to realign block $r_2'[k]$ (middle graph of Figure 6) on what is seen by the receiver as the start of a base chirp.

[2]The values of f_b can be seen, as here, as FFT bin indices numbered from 1 to N, or alternatively, as normalized frequencies that take on the values k/N with $k = 1..N$.

In low SNR conditions, the use of a single preamble symbol can lead to a high error probability in the extraction of f_1. Suppose now that the receiver is turned on earlier and that the received signal contains several blocks (theoretically, up to 65535) which contain un-modulated preamble symbols. As discussed in [19], it is advantageous to average the FFT magnitudes of successive blocks before applying the $argmax$ function. Indeed, since successive symbols are identical,[3] this will average out the bins which contain only noise, easing the extraction of the correct bin. Of course, this accumulation cannot be applied to blocks containing frame synchronization samples whose modulation value is not zero. In a low complexity implementation in which the amount of sample storage space is limited, an IIR filter such as $y[n] = x[n] + \alpha y[n-1]$ can be used instead of averaging, with $\alpha < 1$ representing the proportion of the previously received blocks that is 'remembered'. The positive impact of this averaging operation can clearly be seen both in simulation and measurement on Figures 7 and 8. Figure 7 shows the simulated frame synchronization miss-detection probability versus the number of complete received preamble symbols whereas Figure 8 shows the measured sensitivity as a function of the number of transmitted un-modulated preamble symbols.

As stated in [19], robustness can be further improved by imposing that the maximum FFT magnitude value which is selected by the $argmax$ function exceed a threshold. This threshold can be designed to be proportional to the noise level present in the other frequency bins.

2) Frame (sync word) synchronization: Once a realignment by f_1 has been applied, the search for the two frame synchronization symbols (*sync word*) starts. Indeed, we expect that blocks $r'_2[k]$ and $r'_3[k]$ will produce $f'_2 = 2$ and $f'_3 = 2$ since +2 is the modulation value corresponding to the frame synchronization symbols in our example (middle graph of Figure 6). Recall however that the presence of CFO_{frac} and noise can easily cause ± 1 bin errors (or more). Thus, rather than searching for a specific sequence of bin values over successive blocks, the frame synchronization algorithm proposed in [19] monitors the FFT absolute values in the desired and undesired frequency bins (± 1) in two successive blocks.

[3]This is true only if we ignore SFO which, for very long preambles, will tend to progressively shift the bin value extracted from the accumulated magnitudes.

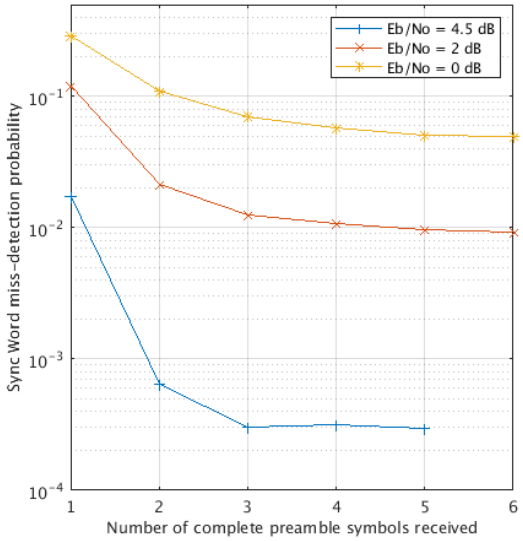

Fig. 7. Simulation (infinite precision) results showing the probability of *sync word* detection failure versus the number of complete un-modulated preamble symbols received by the receiver. Here *sync word* value is $+8$, $SF = 7$, $BW = 125$ kHz, $\alpha = 0.5$. STO_{int} and CFO are chosen randomly between [0, 1,.., $2^{SF} - 1$] and ± 34 ppm, respectively. SFO and STO_{frac} are set to zero.

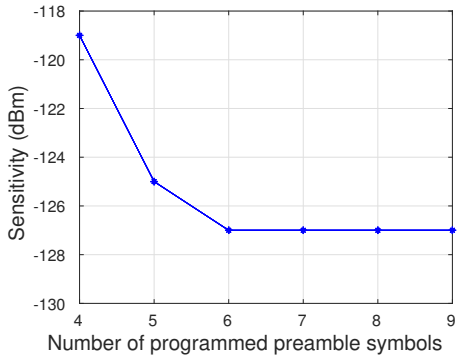

Fig. 8. Sensitivity, defined as 5% PER for 66 byte packets, measured on SX1276 chip versus the number of TX and RX programmed preamble symbols, $SF = 7$, coding rate = 4/5, non-zero frame synchronization symbols, $BW = 125$ kHz, carrier frequency = 867.1 MHz.

An effective frame filtering feature can be achieved simply by programming the receiver with the number of un-modulated preamble symbols it should expect before successfully detecting the frame synchronization symbols[4]. In this way, transmitted frames with longer preambles than expected can be automatically rejected.

3) Coarse time and frequency synchronization: Once frame synchronization is achieved, the next two blocks, $r'_4[k]$ and $r'_5[k]$ in our example, are used to find the frequency synchronization symbols. This is achieved as in (13) but this time with the un-conjugated base sequence $B[k]$. Since $r'_5[k]$ contains only *down-chirp* samples, f'_5 is more reliable than f'_4 and is probably used as f_{down}. Next, since f_{up} is f_1 calculated above, estimates of STO_{int} and CFO_{int} ($\widehat{STO}_{int}$ and $\widehat{CFO}_{int}$) can be calculated as discussed in II-D.

4) Fine frequency synchronization: As discussed in II-E, the phase of the FFT output in bin f_{down} of the two more recently processed blocks ($r'_4[k]$ and $r'_5[k]$ in our example) is used to estimate CFO_{frac}. Unfortunately, since it is calculated using only two blocks, this estimation, $\widehat{CFO}_{frac}$, will be relatively imprecise, as shown in Figure 10, leaving a residual fractional frequency error that will be corrected by the tracking loop described below. As discussed in [16], $\widehat{CFO}_{frac}$ can be compensated by multiplying the received signal, or the reference base chirp, by:

$$e^{-j2\pi k\widehat{CFO}_{frac}/BW}, \quad k = 0, .., N-1 \tag{14}$$

5) Fine error tracking loop: At this point, we have successfully estimated $\widehat{CFO}_{int}$, $\widehat{STO}_{int}$ and $\widehat{CFO}_{frac}$. The presence of STO_{frac}, which generates inter-symbol interference, and a small residual fractional frequency error will have as consequence to spread the symbol energy onto more than one FFT bin. (Note that this is true even in steps 1 and 2 presented above.) For every block $r_b[k]$ processed according to (13) and producing f_b, the extent to which the symbol energy is shifted to the next nearest FFT bins, which is indicative of the fractional timing and frequency errors, is estimated in [19] by subtracting FFT magnitude of the next higher FFT bin (modulo N) from the FFT magnitude of the next lower FFT bin (modulo N) and dividing

[4]Measurements confirm that this feature is implemented on commercially available hardware.

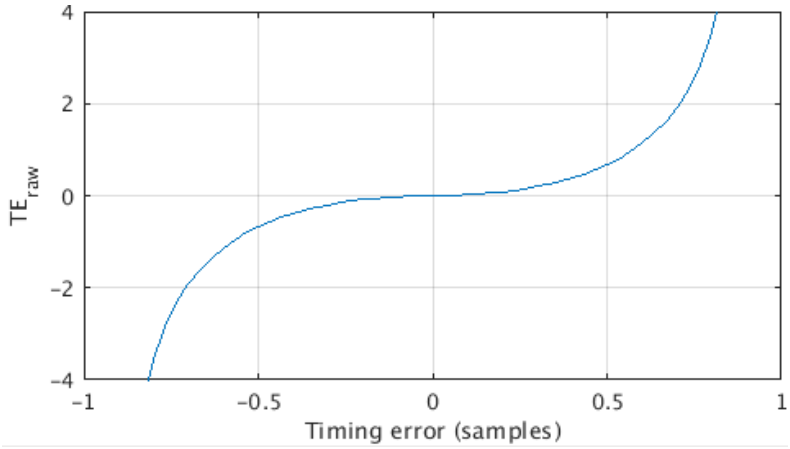

Fig. 9. Plot of TE_{raw} versus the *'timing error'*. This curve is obtained in simulation by oversampling a base chirp sequence, down-sampling with a fractional timing error, and comparing the FFT magnitudes in the adjacent bins versus the primary frequency bin.

the result by the FFT magnitude in bin f_b. This produces a "raw timing error", TE_{raw}, that can be converted to a fractional *'timing error'* by inverting the conversion function plotted on Figure 9. The *'timing error'* corresponds to a fraction of the sample duration $1/f_{s_{min}}$ and is an approximation of STO_{frac} if the residual fractional frequency error is ignored. Figure 9 is obtained in simulation by measuring the magnitude of the FFT output in adjacent versus desired bins when the received symbol is sampled with a fractional timing offset.

Recall however that this *'timing error'* is produced by simultaneous fractional time and frequency offsets which cannot be distinguished. As explained in [19], thanks to the time/frequency equivalency of chirps, small time misalignments can be compensated by a proportionally small frequency offsets. Two compensation methods are therefore used simultaneously to correct this *'timing error'*: Part of this error can be compensated in the decimation chain of the receiver's digital front-end (DFE). For example, suppose that a factor of 10 decimation is applied before the samples are produced at the minimum sampling rate $f_{s_{min}}$. Fractional timing offsets that are multiples of $1/10$ can easily be created by shifting the decimation operator's input by a corresponding number of undecimated samples. The remaining part of the *'timing error'*, $\widehat{te}$, can be converted to a frequency error by applying the time to frequency conversion allowed

by the time/frequency equivalency of chirps: $\widehat{fe} = (BW \times \widehat{te})/N$. Frequency compensation can then be applied by applying a constant frequency offset to the base chirp used in (13) as follows:

$$B[k]' = B[k] \times e^{-j2\pi k\widehat{fe}/BW}, \quad k = 0, .., N-1 \tag{15}$$

In [19], it is proposed that this error tracking loop can be activated starting from the very first block of samples received. The idea is that, since the effect of this compensation is to recenter symbol energy onto a single bin, the loop should ease the detection of the preamble and frame synchronization symbols.[5] As stated in [19], since block b is processed at the same time as block $b+1$ is being sampled in the DFE, the fractional timing compensation can be updated in the decimation chain only for block $b+2$.

6) Data demodulation: Thanks to the above tracking loop, at this point in the algorithm, we can successfully compensate for $\widehat{CFO}_{int}$, $\widehat{CFO}_{frac}$, $\widehat{STO}_{int}$ and $\widehat{STO}_{frac}$. Data demodulation can now start. Since STO_{int} has been found in step 3, it can be used to perform the correct alignment of the contents of the next processed block on the start of the data samples, illustrated by the block $r_7\prime\prime[k]$ in the middle plot of Figure 6. Symbol demodulation can be achieved in the following manner:

$$f_{symbol} = \{argmax(|FFT(r_b'[k] \times B[k]^* \times e^{-j2\pi k(\widehat{CFO}_{frac}+\widehat{fe})/BW})|) - \widehat{CFO}_{int} \times \frac{N}{BW}\} \bmod N \tag{16}$$

7) Sampling frequency error compensation: As discussed in II-C, even after compensating for both STO and CFO, a sampling frequency error will gradually introduce an error in the samples contained in consecutive blocks, with the symbol energy associated with a complete chirp incrementally spreading to a duration greater or lesser than $N \times f_{s_{min}}$. If the RF carrier and the sampling clock are generated from the same crystal-based reference clock, CFO and SFO are related by the expression $SFO = f_{s_{min}} \times CFO/f_c$, where f_c is the RF carrier frequency. Thus, once CFO has been estimated in steps 3 and 4 above, the corresponding SFO estimate, $\widehat{SFO}$, can be calculated. Depending on the sign of $\widehat{SFO}$, the received signal will suffer from

[5]This claim has not been verified in this present work.

an incremental delay (or advance) of $SFO_{delay} = N \times \widehat{SFO}/f_{s_{min}}$ samples every block of N samples. The fractional part of this delay (or advance) can be compensated by the error tracking loop by summing SFO_{delay} and the *timing error* calculated in step 5 above. The integer part of this delay (or advance) can be compensated by removing (or duplicating) a sample when necessary, as proposed in [16]. However, differently from [16], compensating the fractional timing error using the error tracking loop avoids having to over-sample the signal by a factor of 2, an approach which comes at a very high energy and complexity cost.

The compensation of the incremental delay (or advance) is necessary to avoid losing symbol synchronization while demodulating the data symbols. However, the presence of SFO means that the slope of the received chirp will be slightly different from the expected one. This will tend to shift the demodulated symbol energy away from a single frequency bin, hence lowering demodulation performance. In order to recenter the symbol energy, it is possible to adjust the frequency slope of the reference base chirp $B[k]$ used in (16) to match the slope of the received chirps, as discussed in [19] and [16].

V. LoRa frame synchronization for ULP-SDR

The frame synchronization and demodulation algorithm presented above can certainly be further improved, i.e. using soft demodulation [19] or adding CFO and SFO drift compensation mechanisms for very long frames, etc. However, the aim of this section is to discuss the inherent computational complexity of the above algorithm and the adaptations that would be required in an ULP-SDR implementation.

A. Discussion on algorithm complexity

While the algorithm described in IV-B can appear relatively complex, we observe that, for each block, the basic algorithm consists in a multiplication of the block's samples with some variant of the base sequence $B[k]$, followed by an FFT of size N, followed by an absolute value calculation on the FFT outputs and, finally, the search for the argument of the resulting maximum. This is true in all phases of the algorithm (with small variants, e.g. IIR filtering during preamble detection). Computational complexity therefore essentially consists of N sine

and cosine calculations to adjust the angle of the base sequence $B[k]$ with respect to $\widehat{fe}$, $\widehat{CFO}_{frac}$, and SFO, followed by N complex multiplications, a complex FFT of size N, and N magnitude calculations.

B. Algorithm variants in the ULP-SDR context

Part of the difficulty in the synchronization algorithm presented above is due to the fact that sampling at $f_{s_{min}}$ makes it more difficult to distinguish timing and frequency errors due to the folding over of the frequency signal. In an SDR context, since oversampling avoids this effect, at first thought, oversampling might be considered a good approach for lowering the complexity of the synchronization algorithm. However, higher sampling rates necessarily increase the required digital baseband (DBB) processing clocks, potentially leading to greater energy expenditure.

The main difficulty in implementing the above algorithm in an ULP-SDR context lies in realizing the error tracking loop feedback signal that adjusts the delay at the input of the decimation block in the receiver's digital front-end. Implementing a decimation block in software would imply a large power burden. Alternative approaches for compensating STO_{frac} include adding a linear interpolation block at the input of the baseband receiver[6] or by using a compensation approach as in (15) but this time with a frequency error coefficient corresponding to the complete *'timing error'*. The analysis of these approaches is left to future work.

In the following sections, we propose several ideas that might be exploited by the research community to improve or create variants of the above algorithm, especially in the context of low complexity receivers.

C. Alternative preamble synchronization algorithm

The preamble synchronization algorithm proposed in step 1 above is designed to capture frames in which the preamble is as short as possible, the worse case being a single complete preamble symbol. In favorable SNR conditions, the ability to achieve very quick synchronization lowers frame transmission and reception energy overheads and link latency. In less favorable conditions

[6]A first order interpolation would require two multiplications, one sum and one division (or shift) per I and Q sample.

as shown in Figure 7, the probability of error in the value of f_b obtained using (13) without the noise reduction effect of the IIR filter can be important, potentially leading to many dropped frames due to the incorrect frame synchronization detection in step 2.

Using a larger number of preamble symbols improves performance in low SNR conditions thanks to the IIR filter discussed above. In this context, another approach that could be used to improve the preamble synchronization phase, especially in the absence of an error tracking feedback loop, consists in using a pattern matching algorithm on the successive outputs of (13): $\{f_1, f_2, f_3, ...\}$. Detecting that these values have stabilized around a single value (or two adjacent bin values since fractional time and frequency errors can lead to FFT outputs that fall between two bins) can be an alternative mean for detecting the presence of a preamble with high certainty.

D. Alternative fine frequency synchronization algorithm

Again in a context where expected frames contain more than two preamble symbols, we observe that the estimation of $\widehat{CFO}_{frac}$, previously performed in step 4 using blocks containing the two *down-chirps*, can now be operated in the preamble synchronization phase of the algorithm since identical symbols are being received. This has the advantage of producing a higher precision estimate of CFO_{frac} since the successive estimates can be averaged, reducing the impact of noise. In addition, applying an early $\widehat{CFO}_{frac}$ correction will improve both the frame and coarse synchronization phases.

To study the impact of $\widehat{CFO}_{frac}$ estimation precision in various Eb/No conditions, infinite precision Monte-Carlo simulations are run in which frame synchronization is performed assuming frames affected by a CFO randomly chosen in the range of $\pm$34 ppm. CFO_{frac} estimation is performed on the preamble symbols using the method described in Section II-E. For comparison, this plot also shows the accuracy of the CFO_{frac} estimation method proposed in [16], equation (9). We see that, not only does this last algorithm perform less accurately at low Eb/No, the algorithm itself requires 2^{SF} additional complex multiplications and sums per symbol employed.

Finally, we observe that it is also possible to estimate CFO_{frac} using the blocks containing the two frame synchronization symbols, assuming the two symbols are identical. This estimate could be averaged with the one extracted from the two *down-chirp* symbols.

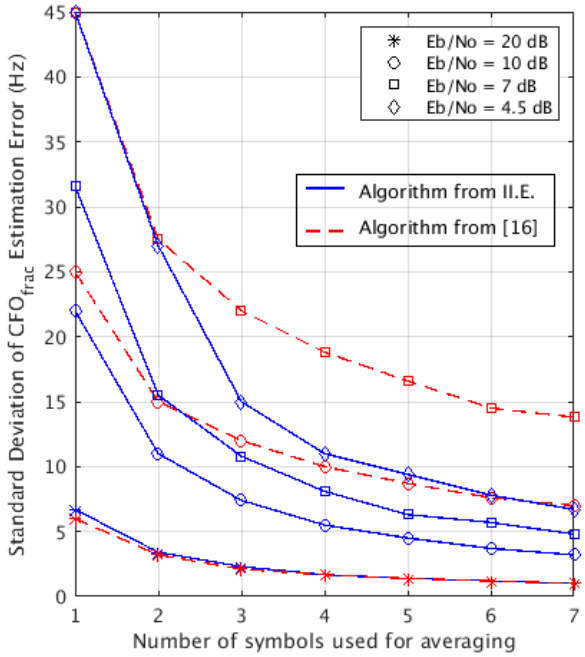

Fig. 10. Standard deviation of the CFO_{frac} estimation error versus the number of symbols over which the estimation is averaged (BW = 125 kHz, SF = 7)

E. Alternative frame (sync word) synchronization algorithm

In an alternative approach to the frame synchronization algorithm presented in step 2 above, rather than searching for the frame synchronization symbols after realizing a time realignment by f_1, the search for these symbols can be done using the next, un-realigned blocks ($r_2[k]$ and $r_3[k]$ in the bottom plot of Figure 6). Assuming that the frame synchronization symbols employ a modulation value of 2, initial frame synchronization will be achieved if the receiver finds the sequence $\{..., f_1, f_1+2, f_1+2, ...\}$. However, since at this point STO_{frac} (and potentially also CFO_{frac}) errors remain and thus FFT outputs can fall between two bins, the pattern matching algorithm should accept ± 1 bin errors on the extracted bin indices. Alternatively, the output of this bin pattern matching search could also be combined with the output of the magnitude pattern matching search of the original algorithm.

F. An algorithm for resolving data start block ambiguity

We observe that the samples corresponding to the two and a quarter *down-chirps* can be spread over 3 or 4 blocks (for example, in the middle plot of Figure 6, they are contained in blocks $r'_4[k]$, $r'_5[k]$ and $r'_6[k]$). This means that, even once coarse time synchronization has been performed in step 3 and thus $\widehat{STO}_{int}$ has been calculated, there remains some uncertainty as to the block index which contains the very first data (header or payload if there is no header) symbol sample. Resolving this uncertainty is necessary for correctly applying the final block alignment (e.g. $r''_7[k]$ in the middle plot of Figure 6). Of course, it is possible to rigorously lift this uncertainty by re-aligning, thanks to $\widehat{STO}_{int}$, the samples contained in all of the blocks employed to search for the *down-chirps* (assuming these have been stored in memory) and repeating step 3. The perfect time alignment between blocks and *down-chirps* then makes it easy to identify the two complete *down-chirps* contained in the frame using the FFT magnitude. This is the approach used to obtain the ideal synchronization result shown in Figure 11. However, in a low complexity receiver, all of these additional computations should be avoided.

Here, we present a low complexity technique for resolving this ambiguity: Once steps 2, 3 and 4 have been performed, since $\widehat{STO}_{int}$ and f_{down} are known, it is easy to calculate the expected number of *down-chirp* samples that will be contained in the M successive blocks suspected of containing *down-chirp* samples. These values are stored in the $expected\_samples$ vector[7]. Assuming the FFT magnitudes for these M blocks have been stored in memory, a second vector named $measured\_magnitudes$ is constructed by extracting the FFT magnitude outputs at the index f_{down} for these M blocks. At high SNR, we indeed expect that this vector reflect the proportion of *down-chirp* samples contained in each block. Finally, if we apply a convolution of $expected\_samples$ with $measured\_magnitudes$, the maximum of the convolution result can be used to identify the block index containing the start of the data symbols.

As a further improvement, if we are certain that at least the first two of the M blocks do not contain *down-chirp* samples (e.g. this is the case if they contain frame synchronization

[7]Precisely, $expected\_samples = [.., 0, N - \widehat{STO}_{int}, N, 1.25 \times N - (N - \widehat{STO}_{int}), A, 0, ..]$, where $A = 0.25 \times N - (N - \widehat{STO}_{int})$ if $A > 0$ or 0 otherwise. For example, if N=128, M=6 and $\widehat{STO}_{int} = 0$, $expected\_samples = [0, 0, 128, 128, 32, 0]$.

samples), the content of the vector that is convolved with $expected\_samples$ can be improved using an approach similar to Dixon's test for outlier detection [21]: for a given dataset, outliers are detected by dividing the *gap* between each value and the values' expected *range*. In our case, we define the *range* as the absolute value of the difference between the two first values of the $measured\_magnitudes$ vector. Indeed, since *down-chirps* are not present, we expect these values to contain only noise. For each other element of the $measured\_magnitudes$ vector, a new value is calculated (and stored in a vector called $Dixon\_magnitudes$) by dividing the distance (*gap*) between the value (if positive) and the largest of the first two values. In a noisy context, this means that blocks containing *down-chirp* samples will be identified as outliers. Finally, the $expected\_samples$ vector can be convolved with the $Dixon\_magnitudes$ vector to identify the block containing the start of the data samples.

In Figure 11, we present simulation results of frame synchronization failure, meaning that the algorithm was unable to identify the first data sample hence leading to complete frame loss, for different synchronization algorithms. Simulations are run in the following conditions: BW=125 kHz, SF = 7, frames contain 6 preamble symbols and the two frame synchronization symbols (*sync word*) are set to 0. CFO is chosen randomly in the range $\pm$34 ppm, STO_{int} is chosen randomly in the range $[0, 127]$, SFO and STO_{frac} are set to 0. The proposed low complexity algorithm employs the alternative techniques described in Sections V-C and V-D as well as the technique based on Dixon's test presented above (M=6). We compare the performance of this algorithm with the ideal, high complexity, synchronization algorithm discussed above and with another low complexity algorithm based on a simple heuristic rule for choosing the block containing the first data sample: the two largest values of the $measured\_magnitudes$ vector, presumably corresponding to blocks containing the largest number of *down-chirp* samples, are compared. If these values increase with block index, the block index following that of the greater value is assumed to contain the first data sample. If these values decrease with block index, the block index of the smaller value is assumed to contain the first data sample. Such a simple rule is shown to result in very poor results.

The good performance of our proposed low complexity algorithm lies in the fact that, if f_{up} and f_{down} are extracted with high certainty (which is the case when the frame contains more than

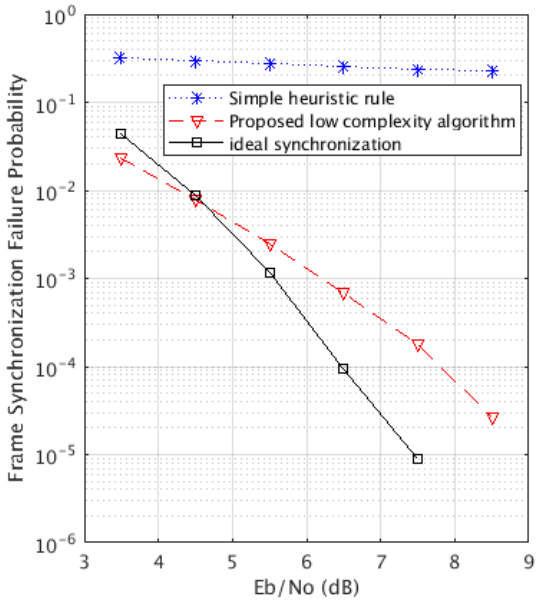

Fig. 11. Simulation results of frame synchronization failure (BW = 125 kHz, SF = 7)

two preamble symbols), then so is $\widehat{STO}_{int}$ and so is the $expected\_samples$ vector. Since the convolution with the $Dixon\_magnitudes$ vector gathers information from several blocks, this leads to very high synchronization accuracy even in very low SNR conditions. The computing cost of this technique is very low and is essentially due to the memory that is required to store FFT magnitude outputs for the M blocks.

VI. Conclusion

The importance of the LoRa physical layer for the IoT community has prompted the authors' efforts to provide a clear explanation of the timing and frequency synchronization algorithms required to detect LoRa-modulated frames in the context of minimum sampling rate optimum receivers. We describe how robust frame detection can be performed while focusing on minimal complexity implementations of the proposed algorithms. In particular, for the first time, a method for resolving integer symbol timing and carrier frequency offset ambiguity is described. We also provide simulation and measurement results showing the relationship between number of received preamble symbols and both frame detection performance and fractional CFO estimation error. Finally, we propose several ideas for lowering the complexity of certain synchronization

mechanisms, allowing these to be implemented on recently proposed ultra-low power software-defined receivers.

ACKNOWLEDGMENT

The authors would like to thank Reda Bekkar of CEA, Sylvie Charbonnier and Chhayarith Heng-Uy of Gipsa-Lab and Mathieu Xhonneux of UCLouvain for fruitful discussions.

REFERENCES

[1] I. Markit. "Connectivity technologies, an in-depth view into the competition, applications and influencers driving the foundation of iot." [Online]. Available: https://ihsmarkit.com (2018) [Apr. 11, 2019].

[2] J. Haxhibeqiri, E. De Poorter, I. Moerman, and J. Hoebeke, "A survey of lorawan for iot: From technology to application," *Sensors*, vol. 18, no. 11, 2018. [Online]. Available: http://www.mdpi.com/1424-8220/18/11/3995

[3] P. Robyns, P. Quax, W. Lamotte, and W. Thenaers, "A multi-channel software decoder for the lora modulation scheme," in *3rd International Conference on Internet of Things, Big Data and Security*, Jan. 2018, pp. 41–51.

[4] B. Sikken. "Decodinglora." [Online]. Available: https://revspace.nl/DecodingLora [Apr. 9, 2019].

[5] M. Knight, "Reversing lora: Exploring next- generation wireless." in *GRCon*, 2016.

[6] J. Blum. "Lora modem with limesdr." [Online]. Available: https://myriadrf.org/news/lora-modem-limesdr/ (2016) [Apr. 11, 2019].

[7] S. Wu, S. Kang, C. Chakrabarti, and H. Lee, "Low power baseband processor for iot terminals with long range wireless communications," in *2016 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, Dec 2016, pp. 728–732.

[8] Y. Chen, S. Lu, H. Kim, D. Blaauw, R. G. Dreslinski, and T. Mudge, "A low power software-defined-radio baseband processor for the internet of things," in *2016 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, March 2016, pp. 40–51.

[9] H. Belhadj Amor and C. Bernier, "Software-hardware co-design of multi-standard digital baseband processor for iot," in *2019 Design, Automation Test in Europe Conference & Exhibition (DATE)*, March 2019.

[10] C. Heng Uy, C. Bernier, and S. Charbonnier, "Energy Efficient Channel State Classification for Lifetime Enhancement of LPWA Networks," in *11th International Conference on COMmunication Systems & NETworkS*, Bangaluru, India, Jan. 2019.

[11] ——, "Design of a Low Complexity Interference Detector for LPWA Networks," in *IEEE I^2MTC 2019. IEEE International Instrumentation and Measurement Technology Conference*, Auckland, Australia, May 2019.

[12] L. Vangelista, "Frequency shift chirp modulation: The lora modulation," *IEEE Signal Processing Letters*, vol. 24, no. 12, pp. 1818–1821, Dec 2017.

[13] C. Gupta, T. Mumtaz, M. Zaman, and A. Papandreou-Suppappola, "Wideband chirp modulation for fh-cdma wireless systems: coherent and non-coherent receiver structures," in *IEEE International Conference on Communications, 2003. ICC '03.*, vol. 4, May 2003, pp. 2455–2459 vol.4.

[14] K. Huang, Z. Wang, and R. Tao, "Study of incoherent demodulation technique in chirp spread spectrum communication systems," in *2008 9th International Conference on Signal Processing*, Oct 2008, pp. 1926–1929.

[15] B. M. Popovic, "Generalized chirp-like polyphase sequences with optimum correlation properties," *IEEE Transactions on Information Theory*, vol. 38, no. 4, pp. 1406–1409, July 1992.

[16] R. Ghanaatian, O. Afisiadis, M. Cotting, and A. Burg, "Lora digital receiver analysis and implementation," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2019, pp. 1498–1502.

[17] J. G. Proakis, *Digital Communications*. McGraw-Hill, 1995.

[18] H. Tanaka, "A frequency and timing synchronization circuit making use of a chirp signal," Japanese Patent 12 969 998, April 24, 1998.

[19] O. Seller and N. Sornin, "Low complexity, low power and long range radio receiver," European Patent 3 264 622, July 1, 2016.

[20] ——, "Low power long range transmitter," European Patent 2 763 321, Feb. 5, 2013.

[21] W. J. Dixon, "Analysis of extreme values," *Ann. Math. Statist.*, vol. 21, no. 4, pp. 488–506, 12 1950. [Online]. Available: https://doi.org/10.1214/aoms/1177729747