UCLouvain

École polytechnique de Louvain

Software-defined Bluetooth Low Energy transceiver for reconfigurable IoT smart sensors

Author: **Renaud Gonce**
Supervisors: **David Bol, Jérôme Louveaux**
Readers: **Antoine Paris, Mathieu Xhonneux**
Academic year 2019–2020
Master [120] in Electrical Engineering

Acknowledgements

I would like to thank my supervisors, Professor David Bol and Professor Jérôme Louveaux, as well as Mathieu Xhonneux, for their guidance throughout this work.

Abstract

Technology is more and more present around us. In particular, the Internet of Things (IoT) has been emerging for a few years and is now in huge expansion. Along with it comes a massive deployment of smart sensors. This is full of promise for a lot of domains (social, economical, etc.). However, it raises some sustainability concerns, particularly from an environmental point of view. Firstly, this huge deployment puts a lot of pressure on the natural resources. Secondly, the ecotoxicity of the sensors battery is an issue, especially as their lifetime is very limited. This short lifetime comes from the fact that the wireless communication standards that these sensors use are constantly evolving, but due to their lack of reconfigurability the sensors cannot keep up with this evolution.

A possible way towards sustainability is the use of a software-defined radio (SDR) for the wireless communication of those smart sensors. An SDR has the advantage of being reconfigurable through over-the-air update, and of being able to support multiple communication standards, which makes it versatile for a large variety of IoT applications. The big challenge about SDRs is their power consumption, which should remain as low as possible from the sustainability point of view.

With this in mind, this thesis investigates the low-power implementation of a software-defined transceiver following the Bluetooth Low Energy (BLE) communication standard. BLE is the fourth generation of Bluetooth and has been designed to meet the requirements of the IoT applications.

This thesis proposes a receiver implementing non-coherent differential detection, which offers a very good tradeoff between low-complexity and performance. Moreover, the receiver mitigates non-idealities such as symbol timing offset and carrier frequency offset with very low-cost algorithms, based on an IIR filter and a Taylor series approximation respectively. This design allows to stay under the 0.1-% bound of bit error rate at a 13-dB signal to noise ratio (SNR) with a probability of 99.3 %, and with a probability of 99.9 % at a 14-dB SNR.

Contents

1 Introduction **1**
- 1.1 Context . . . 1
  - 1.1.1 IoT Smart Sensors Expansion . . . 1
  - 1.1.2 Challenges For A Sustainable IoT . . . 3
  - 1.1.3 Appeal For SDR . . . 3
  - 1.1.4 Bluetooth for IoT . . . 3
- 1.2 Research Question . . . 4
  - 1.2.1 Contributions . . . 4
- 1.3 Related Work . . . 5
  - 1.3.1 SDR for IoT . . . 5
  - 1.3.2 BLE Transceivers . . . 5
  - 1.3.3 Techniques for Non-Idealities Mitigation . . . 6
- 1.4 Outline . . . 6

2 Methodology **8**

3 BLE Overview **9**
- 3.1 Physical Layer . . . 9
  - 3.1.1 Channel Arrangement . . . 9
  - 3.1.2 Modulation Scheme . . . 10
- 3.2 Link Layer . . . 10
  - 3.2.1 Link Layer States . . . 10
  - 3.2.2 Physical Channels . . . 11
  - 3.2.3 Packet Format . . . 12
- 3.3 Specifications . . . 14

4 Transmitter **15**
- 4.1 GFSK modulation . . . 15
  - 4.1.1 Classic FSK . . . 15
  - 4.1.2 GFSK . . . 16
- 4.2 Summary and Theoretical Computational Complexity . . . 21
  - 4.2.1 Constellation Mapping . . . 21
  - 4.2.2 Pulse-Shaping Filter Convolution . . . 22
  - 4.2.3 Integration . . . 22
  - 4.2.4 Phase Modulation and Upconversion . . . 23
  - 4.2.5 Putting It All Together . . . 23

5 Receiver **26**
- 5.1 Analog-to-Digital Conversion . . . 26
- 5.2 Low-Pass Filter . . . 26
- 5.3 Demodulation . . . 28
  - 5.3.1 Differential Demodulation . . . 29
  - 5.3.2 Delay Choice . . . 32
  - 5.3.3 Decimation . . . 34

5.3.4 BER without non-idealities . . . 35
5.4 Preamble Detection . . . 36
5.4.1 IIR Filter . . . 37
5.4.2 Detection Threshold . . . 38
5.5 Symbol Timing Recovery . . . 39
5.6 Carrier Frequency Offset Correction . . . 45
5.6.1 CFO Mitigation Algorithm Derivation . . . 46
5.6.2 CFO Mitigation Algorithm Performance . . . 50
5.7 Additional non-idealities . . . 54
5.7.1 Modulation Index Offset . . . 55
5.7.2 Sampling Timing Offset . . . 55
5.8 Overall System . . . 57
5.8.1 System Performance . . . 58
5.9 Summary and Theoretical Computational Complexity . . . 62
5.9.1 Digital Low-Pass Filter . . . 62
5.9.2 Hermitian Product . . . 62
5.9.3 Preamble Search . . . 63
5.9.4 Symbol Timing Recovery . . . 63
5.9.5 Carrier Frequency Offset Estimation . . . 64
5.9.6 Demodulation . . . 65
5.9.7 Entire Receiver . . . 65

6 Quantization **68**

7 Conclusion **71**
7.1 Conclusion . . . 71
7.2 System Limits and Future Work . . . 71
7.2.1 Frequency Drift . . . 71
7.2.2 Sampling Timing Offset . . . 72
7.2.3 Channel Response . . . 72

1 Introduction

1.1 Context

1.1.1 IoT Smart Sensors Expansion

Smart sensors are progressively taking place all around us in our everyday life. They are used in a large variety of applications, going from house automation and autonomous cars to manufacturing and supply chain monitoring.

This is all based on sensing, communicating, sharing and actuating. The devices that are involved are called smart sensors as they sense and record a large variety of elements and perform some kind of processing on the collected data. The resulting information can then be directly used by the same device to perform actuation, or it can be passed on to another device for further processing and/or actuation. The data can also be transferred to a network of interconnected objects. It then becomes a part of the so-called Internet of Things (IoT), which can be seen as a network of networks, each of them identifying smart objects and connecting them together to enable data sharing.

This Internet of Things is expanding very fast, with a huge number of underlying machine to machine (M2M) connections. Some figures from Cisco Annual Internet Report confirm this. Among the global connected devices and connections, the share of M2M connections will grow from 33 % in 2018 to 50 % by 2023, with up to 14.7 billion connections. As far as mobile devices are concerned, the M2M category will grow from 1.2 billion in 2018 to 4.4 billion by 2023, a 30-% compound annual growth rate (CAGR), much larger than the smartphones CAGR of 7% [1].

This is illustrated in Figure 1 and Figure 2 from [1].

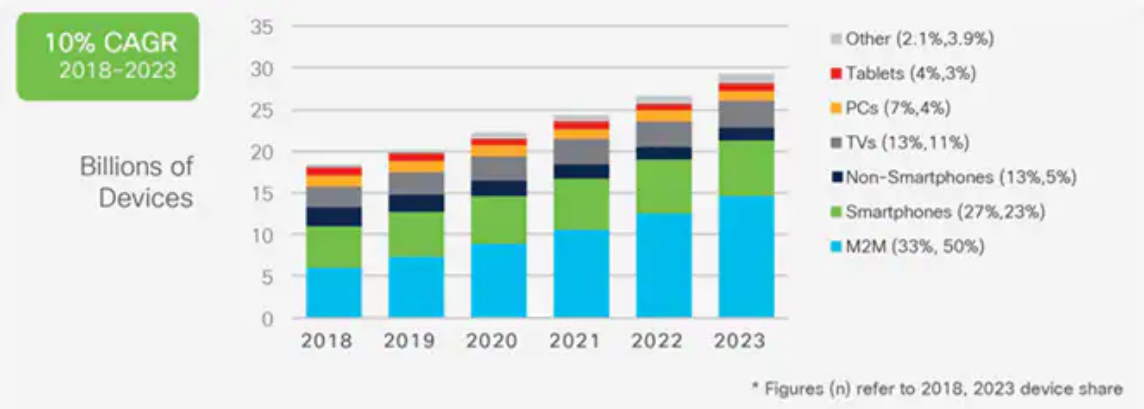


Figure 1: Global device and connection growth.

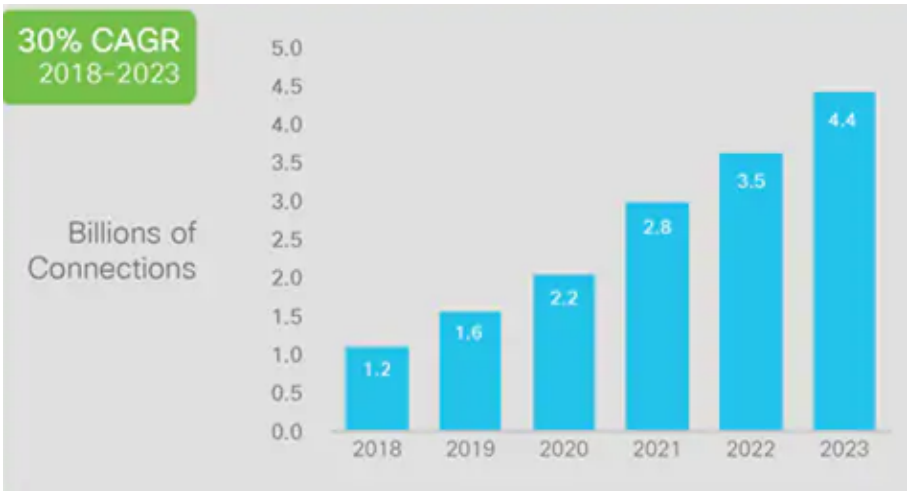


Figure 2: Global mobile Machine-To-Machine growth.

From an economical point of view, this will represent a huge market, with a potential impact of $ 3.9 trillion to 11.1 trillion per year in 2025 [2].

This is represented in Figure 3 from [2], along with its sector breakdown.

Figure 3: IoT market.

1.1.2 Challenges For A Sustainable IoT

This IoT expansion will require the deployment of a huge amount (possibly trillions [3]) of smart sensors. In order to perform such a massive deployment in a sustainable way, some challenges need to be carried out. Focusing on the environmental side, low energy consumption and natural resources preservation are a major concern.

A first issue is that smart sensors are not able to keep up with technology evolution, particularly as wireless communication standards are concerned. This significantly reduces their lifetime as they need to be regularly replaced, which is problematic in terms of resources. A solution would be to make the sensors reconfigurable.

A second issue is the battery lifetime and the way the energy is produced. Energy harvesting would be a way to reduce the pressure on natural resources. Moreover, a low-power design should be considered in order to extend the battery lifetime.

1.1.3 Appeal For SDR

As already mentioned, reconfigurability is a key aspect for a sustainable IoT. With this in mind, a software-defined radio (SDR) represents a particularly attractive solution as it allows over-the-air update of the communication standard it supports. Furthermore, an SDR offers the possibility to support multiple standards. This versatility is of particular interest for IoT devices for which a lot of different standards are emerging, each having its pros and cons depending on the targeted application. On the other hand, SDRs typically consume much power for wireless communication, which goes against the second key aspect for a sustainable IoT.

1.1.4 Bluetooth for IoT

In order to meet the requirements of the different IoT applications, numerous wireless communication standards are emerging. Each of them has its specificity, but their main characteristic is their way to balance the tradeoff between range and data rate. The reason behind this tradeoff is the low-power motivation. The most popular standards include Bluetooth, Zigbee, ANT, LoRaWAN and Sigfox.

Bluetooth is particularly popular as it was the pioneer of the Wireless Personal Area Network (WPAN, a network enabling wireless connections between devices in a small area), and as its technology has already been being incorporated in more and more devices for the last two decades.

It all began in 1998, when Ericsson, IBM, Intel, Nokia and Toshiba formed the Bluetooth Special Industry Group (SIG) to work on a global solution for short-range wireless

communication. One year later they were joined by 3Com, Lucent, Microsoft and Motorola, but Lucent was soon replaced by one of its spin-offs: Agere [4]. Another remarkable company to join the Bluetooth SIG was Apple in 2015 [5].

In 2010, the Bluetooth SIG adopted Bluetooth Core Specification Version 4.0 with Low Energy technology (BLE, formerly marketed as Bluetooth Smart). But a particularly relevant milestone came in 2013-2014 with the release of version 4.1 which includes the first features for IoT, such as the ability for peripherals to communicate independently, without the intermediate of the host [6].

As for the last version, Bluetooth 5.0, it came in 2016-2017 with a doubled speed and a quadrupled range as major improvements.

1.2 Research Question

In order to achieve an environmentally sustainable massive deployment of IoT smart sensors, this work proposes a low-power implementation of a software-defined transceiver, based on the Bluetooth Low Energy communication standard.

More precisely, this thesis addresses the following question:

How to design a fully working implementation of a software-defined Bluetooth Low Energy transceiver while maximizing its versatility to target a multi-standard SDR, and minimizing its operational cost (hence its energy consumption)?

The emphasis is put on the receiver design, as receivers typically consist in the counterpart of all the transmitter blocks, to which additional processing blocks are added for non-idealities mitigation. Moreover, this work only focuses on the digital part of the communication as the interest is put on an SDR design.

1.2.1 Contributions

The main contribution of this thesis work is the implementation of a very low-cost BLE transceiver for SDR.

Moreover, innovative techniques to mitigate transmission non-idealities are proposed, such as the reuse of the IIR filter output from the preamble detection for the symbol timing recovery, or still the simple data-aided carrier frequency offset estimator.

1.3 Related Work

In this section, existing work related to this thesis subject is browsed through. First, the topic of SDRs for IoT applications is investigated. Next come state-of-the-art and sub-optimal BLE transceivers implementations. Finally, typical techniques for non-idealities mitigation are discussed.

1.3.1 SDR for IoT

In [7], the first demonstration of a baseband SDR achieving the mW-level power consumption is realized, showing that SDR can be used for IoT applications.

An ultra-low power SDR for IoT based on a 32-bit RISC-V core is presented in [8]. Their work proposes a rather generic digital baseband processor, with an architecture for FSK demodulation including a Start of Frame Delimitor (SFD) search algorithm. This architecture can serve as a basis for this thesis research.

1.3.2 BLE Transceivers

State-of-the-art BLE transceivers are based on an all-digital phase-locked loop (ADPLL) structure, and feature a TX/RX switchable on-chip matching network [9], [10]. In [10], a high-IF discrete-time receiver is proposed in 28nm CMOS technology, achieving 2.75mW power consumption, while their transmitter reaches 3.6mW consumption. The approach from [9] includes a hybrid-loop receiver in 65nm technology, with a RX consumption of 2.3mW and a TX consumption of 2.6mW.

However, in this thesis framework, a multi-standard approach is considered in the BLE transceiver design. This means that the operational blocks that are involved should be as generic as possible in order for the SDR to be able to reuse them for the other standards it might support. For this reason, the state-of-the-art ADPLL-based transceiver design is not suitable, as it is very specific to the BLE standard.

As developed later on in Section 4.1, the BLE modulation scheme is Gaussian frequency shift keying (GFSK) with a modulation index equal to 0.5, which actually makes it a Gaussian minimum shift keying (GMSK) scheme.

GFSK and GMSK receivers designs are abundant in the literature. As the optimum receiver is based on a matched filter bank, its complexity is too high for most of the applications. It is thereby particularly not suitable for IoT applications. Therefore, multiple suboptimal techniques have been proposed.

Among them, a widespread technique is a limitor-discriminator followed by an integrate-and-dump filter (LDI receiver). Although it is a low-cost implementation, its power effi-

ciency is not optimal [11]. Another simple architecture is the phase differential detector [12]. This one appears to be the best in terms of low-complexity.

1.3.3 Techniques for Non-Idealities Mitigation

Demodulation is a key part of a receiver architecture, but the mitigation of transmission non-idealities is also essential. As detailed later on in Section 5, symbol timing offset (STO) and carrier frequency offset (CFO) are the critical non-idealities to be compensated for.

Techniques for STO correction often include operations like correlation or Fourier transform [13], [14]. The issue is that these operations are prohibitively costly. Another typical structure is a Gardner's timing error detector in an all-digital feedback loop also containing a decimator, a loop filter, a numerically-controlled oscillator and an interpolator. It is presented in [15] as a low-complexity alternative. Even though this approach is cheaper than the correlation-based or Fourier-transform-based techniques, the digital loop it requires is still quite complex.

Fourier transform can also be used to mitigate CFO [14], but this is again not suitable for a low-cost design. There also exist decision-aided least squares or maximum likelihood methods [16], or still a data-aided approach based on the access address from the BLE packet [17]. They have interesting ideas but the access address is actually variable, depending on the device and on the link state. However, another part of the BLE packet, the preamble, can be used since it is always the same. The decision-aided method can also be modified to use this known preamble data rather than the decision data which are only an approximation of the actual one.

1.4 Outline

This thesis is organized in the following way.

First, Section 2 introduces the methodology that has been followed to carry out this work.

Then, Section 3 presents an overview of the BLE standard with a description of its Physical Layer and Link Layer, along with its main specifications.

Section 4 is dedicated to the transmitter design, with some background about the modulation scheme that BLE uses and a qualitative estimation of its computational complexity.

Next comes Section 5 which presents the receiver architecture and details each of its blocks, from their derivation to their qualitative computational complexity analysis. It also concludes with the evaluation of the overall system performance.

Section 6 discusses the quantization of the digital signals involved in the system.

Finally, Section 7 presents a conclusion, as well as trails for future work.

2 Methodology

The main goal of this work is to propose digital processing blocks for the BLE communication standard that are very cheap from an energy consumption point of view. To this end, three main steps have been followed in terms of methodology.

Firstly, related work in the domains of low-power SDR and GFSK receivers has been examined. Moreover, the BLE standard has been investigated to understand how it works.

Secondly, algorithms have been derived so as to try and answer the research question. In order to assess their individual performance as well as the global system one, numerical simulations have been conducted. Those simulations have been performed through Matlab. It is noteworthy that a sequence of 10^6 symbols is used whenever a bit error rate performance is evaluated. This allows reliable results down to a bit error rate value of 10^{-5}, which is the required value in this framework to be able to properly analyze the bit error rate performance.

The Matlab codes I have written are available through the following URL: `https://forge.uclouvain.be/gonce/master-thesis-codes`.

Thirdly, a qualitative analysis of the system computational complexity has been realized, based on a representation of the system in the form of a block diagram.

3 BLE Overview

In this section, the main characteristics of the BLE standard are presented. As this work focuses on the transmitter and receiver design, the relevant information lies in the first two layers of the OSI model protocol stack, i.e. the Physical Layer and the Link Layer (LL). The Physical Layer defines how the symbols are converted into electrical signals as well as the physical support for these signals transmission, while the Link Layer is responsible for the frames transmission and reception between two devices and handles the sharing of the physical support [18]. The corresponding stack architecture is displayed in Figure 4 from [19].

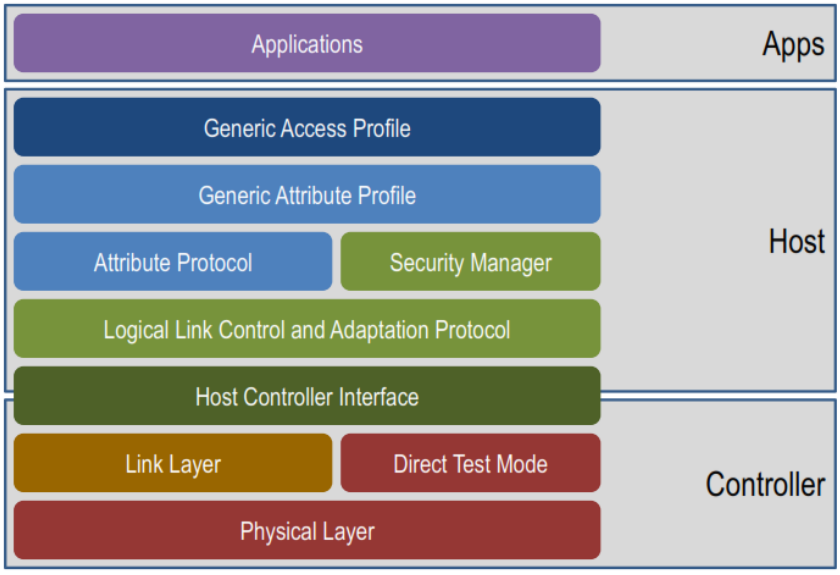


Figure 4: BLE stack architecture.

3.1 Physical Layer

3.1.1 Channel Arrangement

BLE operates in the unlicensed 2.4 GHz ISM band, which is dedicated to industrial, scientific and medical purposes, hence its name. More precisely, it uses 40 2-MHz-bandwidth RF channels, as displayed in Figure 5 from [19], whose center frequencies are given by the following expression:

$$f = 2402 + k * 2 \text{ MHz}, \; k = 0, ..., 39.$$

Furthermore, BLE uses frequency hopping in order to mitigate interference and fading [20].

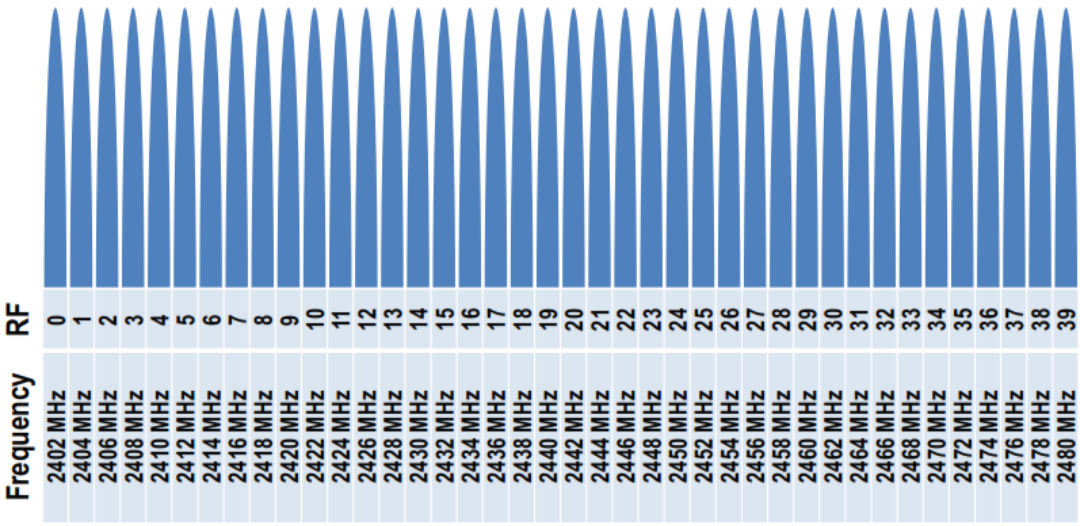


Figure 5: RF channels arrangement.

3.1.2 Modulation Scheme

For the data transmission, the modulation scheme that is used is Gaussian Frequency Shift Keying (GFSK). As developed later on in Section 4.1.2, two parameters need to be specified: the modulation index h and the bandwidth-bit period product BT_b. BLE uses the values $h = 0.5$ and $BT_b = 0.5$ [20]. The modulation scheme is discussed into more details in Section 4.1.

3.2 Link Layer

3.2.1 Link Layer States

The Link Layer can be seen as a state machine, with 5 different states [20]:

- Standby State: the Link Layer does not transmit or receive any packet. This state can be entered from any other state.

- Advertising State: the Link Layer transmits advertising packets and can listen to and respond to responses that they would trigger. This state can be entered from the Standby State.

- Scanning State: the Link Layer listens for advertising packets. This state can be entered from the Standby State.

- Initiating State: the Link Layer listens for advertising packets from one or several specific advertisers and responds to initiate a connection with another device. This state can be entered from the Standby State.

- Connection State: the Link Layer is either in the Master Role or in the Slave Role. In the Master Role, entered from the Initiating State, the Link Layer communicates with a device in the Slave Role and defines the timings of transmissions. In the Slave Role, entered from the Advertising State, the Link Layer communicates with a single device in the Master Role.

The corresponding state diagram is shown in Figure 6 from [20].

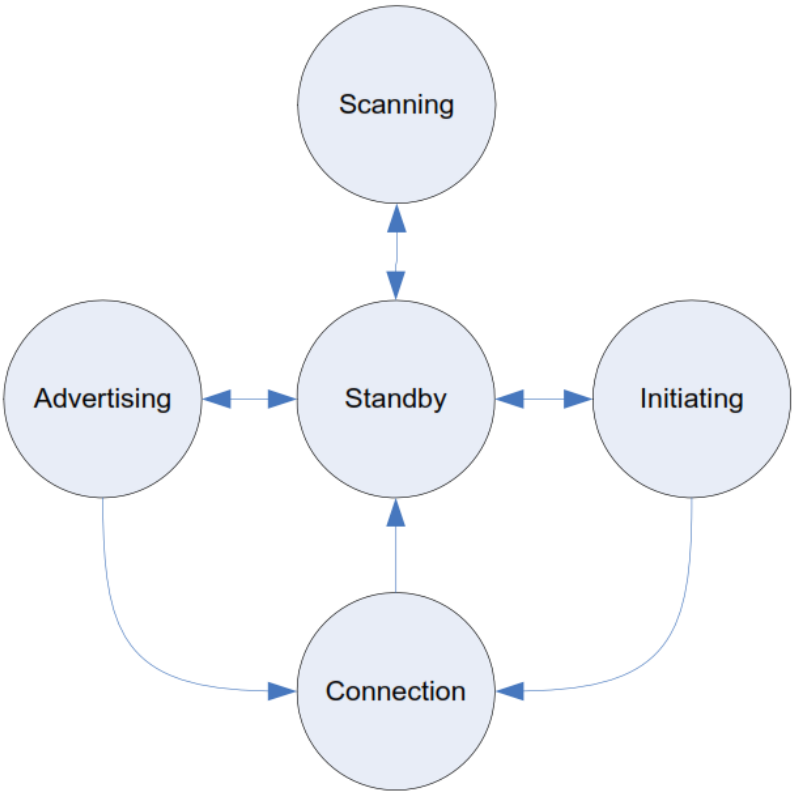


Figure 6: State diagram of the Link Layer state machine.

3.2.2 Physical Channels

The 40 RF channels presented in Section 3.1.1 are used for two kinds of physical channels: the advertising channels and the data channels. As depicted in Figure 7 from [21], there are 3 advertising channels and 37 data channels.

The purposes of advertising channels are device discovery, connection initiation and data broadcasting, while data channels are used for actual data communication between

devices that are connected together.

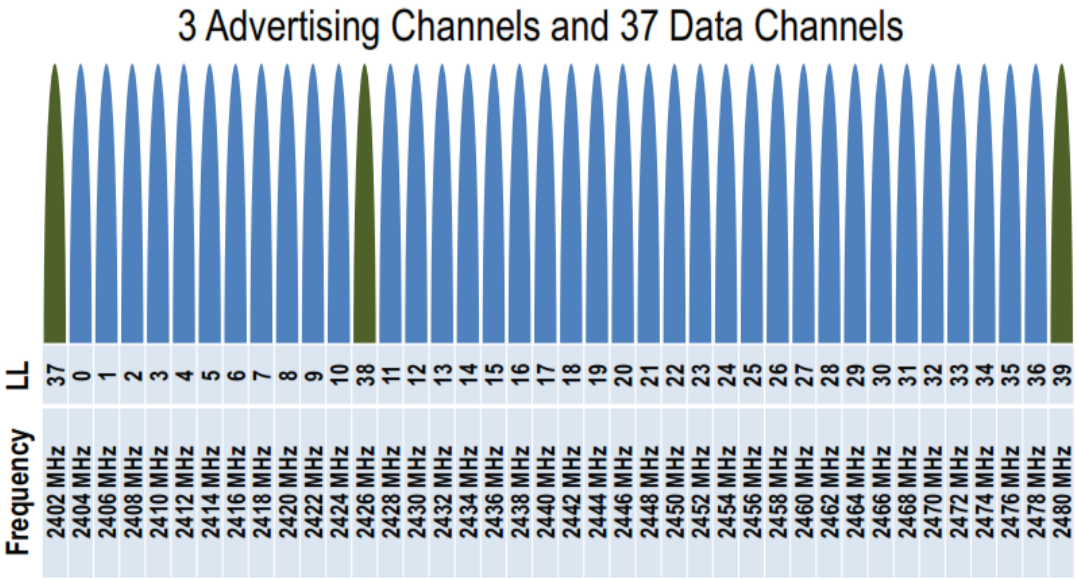


Figure 7: BLE physical channels arrangement.

In order for two devices to communicate, their transceivers must be tuned to the same RF channel. Since a physical channel collision (independent devices tuned to the same RF channel) is very likely, transmissions always start with an Access Address used as a code to identify which connection a packet belongs to [20].

3.2.3 Packet Format

During a transmission, some extra information (such as the Access Address introduced in Section 3.2.2) needs to be conveyed in addition to the application data that is sent. All this information is gathered together in a packet, which follows a well-defined structure. This structure, presented in Figure 8 is made of four fields: the preamble, the Access Address, the Payload Data Unit (PDU), and the Cyclic Redundancy Check (CRC). The order of transmission goes from right to left, the first preamble bit (the one to the right) being the least significant bit (LSB) and the last CRC bit (the one to the left) being the most significant bit (MSB) [20].

This section details the roles and structure of these four fields.

MSB LSB

CRC	PDU	Access Address	Preamble
3 bytes	2 to 33 bytes	4 bytes	1 byte

Figure 8: BLE packet format.

3.2.3.1 Preamble

The preamble is 1-byte long and is made of 8 alternating bits. It is always 10101010 for advertising channel packets, while for data channel packets it depends on the Access Address LSB, to which the preamble MSB must not be equal [20].

The role of the preamble is to help the receiver performing frequency synchronization, symbol timing estimation and Automatic Gain Control (AGC) training [20]. However, the AGC is not covered in this work.

3.2.3.2 Access Address

The Access Address is 4-byte long and is the same for all advertising packets. It is always the following one, expressed in the hexadecimal way: 0x8E89BED6 [20].

However, for data channel packets, the Access Address is different for each Link Layer connection and is used to identify each of them. It is a pseudo-random sequence generated by the device in the Initiating State, but it cannot be any sequence as some conditions still need to be satisfied [20].

At packet reception, the Access Address is checked and, depending on its correctness, the packet is either rejected or considered received [20].

3.2.3.3 PDU

The Protocol Data Unit is the field which includes the useful data, referred to as payload. It also contains different kinds of headers, as well as an optional Message Integrity Check (MIC). The exact structure of the data packet PDU is displayed in Figure 9 [22].

Inside the payload, it is actually the ATT payload which contains the application data. It can hold up to 20 bytes of data [22].

<table>
<tr><th>LL Header</th><th colspan="3">Payload</th><th>MIC (optional)</th></tr>
<tr><td rowspan="3">2 bytes</td><td>L2CAP Header</td><td colspan="2">ATT Data</td><td rowspan="3">4 bytes</td></tr>
<tr><td rowspan="2">4 bytes</td><td>ATT Header</td><td>ATT Payload</td></tr>
<tr><td>3 bytes</td><td>0-20 bytes</td></tr>
</table>

Figure 9: PDU structure.

3.2.3.4 CRC

The last field corresponds to the Cyclic Redundancy Check, which is 3-byte long. The CRC protects the PDU with an error-detecting code. After the Access Address check, if the packet is considered received, comes the CRC, which decides whether the packet is rejected or considered valid. The validness of the packet is the condition for it to be processed [20].

3.3 Specifications

In this section, a summary of the main specifications from the BLE standard is proposed. They are presented in Table 1.

Parameters	**Values**
carrier frequency	2.45 GHz
symbol rate	1 Msymb/s
modulation index	0.5
modulation index tolerance	10 %
bandwidth-bit period product	0.5
BER_{max}	0.1 %
CFO_{max}	150 kHz

Table 1: BLE standard specifications.

The tolerance on the modulation index applies to the transmitter. This 10-% value means that the modulation index h is such that $0.45 \leq h \leq 0.55$ [20].

The maximum allowable bit error rate (BER_{max}) is set to 0.1 % [20].

The carrier frequency offset (CFO) shall not exceed 150 kHz during the transmission of one packet. This value also includes the tolerance on the frequency drift, which is set to 50 kHz [20].

4 Transmitter

In order to be able to design a BLE transmitter, it is necessary to understand how its modulation scheme works. This scheme is Gaussian Frequency Shift Keying (GFSK), which is a special case of Frequency Shift Keying (FSK).

First, this section presents some background about FSK, and then extends it to GFSK. Next, a summary of the required operational blocks is proposed, as well as their corresponding block diagram, which will serve as a basis for their qualitative computational complexity evaluation.

4.1 GFSK modulation

4.1.1 Classic FSK

FSK is a type of frequency modulation. This means that the information to be transmitted is inserted into the frequency of an analog waveform. More precisely, a binary 1 is represented by a positive frequency deviation (Δ_f) from the carrier frequency (f_c), and a binary 0 is represented by a negative frequency deviation. This means that only two symbols a_m are used: $a_0 = -1$ and $a_1 = 1$, leading to two possible waveforms, which are

$$s_m(t) = \sqrt{\frac{2\varepsilon}{T_b}}\cos(2\pi f_c t + 2\pi a_m \Delta_f t)$$

$$s_m(t) = \begin{cases} \sqrt{\frac{2\varepsilon}{T_b}}\cos(2\pi f_0 t) & \text{m=0} \\ \sqrt{\frac{2\varepsilon}{T_b}}\cos(2\pi f_1 t) & \text{m=1} \end{cases}, \tag{1}$$

where $f_0 = f_c - \Delta_f$ and $f_1 = f_c + \Delta_f$, ε is the waveform energy, and T_b is the bit period[1].

The baseband representation of this waveform can be written as

$$s_{bb,m}(t) = \sqrt{\frac{2\varepsilon}{T_b}}\, \mathrm{e}^{j\phi_m(t)} ,$$

where the phase signal $\phi_m(t)$ is such that $\phi_m(t) = 2\pi a_m \Delta_f t$.

This phase signal expression can be extended to account for the transmission of more than one symbol. In order to get a continuous symbol signal out of a discrete symbol sequence $I[n]$, the first operation is to convolve the symbols with a pulse-shaping filter

[1] In this modulation scheme, the symbol period is equal to the bit period as a symbol is composed of only one bit. Moreover, strictly speaking, the wording bit refers to a unit of information, but in this thesis it will be used to refer to a binary symbol.

$g(t)$, which is a rectangular one in classic FSK. The second operation is an integration in order to get a continuous phase, hence avoiding a widening of the phase spectrum which goes along with discontinuities.

The resulting phase signal $\phi(t)$ is given by

$$\phi(t) = 2\pi\Delta_f \int_{-\infty}^{t} \sum_{k=-\infty}^{+\infty} I[k] g_{rect}(\tau - kT_b) d\tau \ ,$$

where $g_{rect}(t) = \begin{cases} 1 & -\frac{T_b}{2} \leq t \leq \frac{T_b}{2} \\ 0 & \text{otherwise} \end{cases}$.

It is often written

$$\phi(t) = \pi \frac{h}{T_b} \int_{-\infty}^{t} \sum_{k=-\infty}^{+\infty} I[k] g_{rect}(\tau - kT_b) d\tau \ ,$$

where the modulation index $h = 2\Delta_f T_b$ appears.

4.1.2 GFSK

Gaussian Frequency Shift Keying uses a Gaussian pulse $g_{Gauss}(t)$ in addition to the rectangular one, hence its name. This pulse is given by

$$g_{Gauss}(t) = K\sqrt{\frac{2\pi}{\ln(2)}} B e^{-2\frac{(B\pi)^2}{ln(2)}t^2} \ , \tag{2}$$

where K is an amplitude parameter and B is the 3-dB bandwidth.

The phase signal then becomes

$$\phi(t) = \pi \frac{h}{T_b} \int_{-\infty}^{t} \sum_{k=-\infty}^{+\infty} I[k] g_{GFSK}(\tau - kT_b) \ d\tau \ , \tag{3}$$

where

$$\begin{aligned} g_{GFSK} &= g_{Gauss} * g_{rect} \\ &= -\frac{K}{2\sqrt{ln(2)}} \left[Q\left(2B\pi \frac{t - \frac{T_b}{2}}{\sqrt{ln(4)}} \right) - Q\left(2B\pi \frac{t + \frac{T_b}{2}}{\sqrt{ln(4)}} \right) \right] \end{aligned} \tag{4}$$

and $Q(x)$ is the error function $Q(x) = \frac{2}{\sqrt{x}} \int_0^x e^{-t^2} dt$.

Finally, the transmitted signal in baseband $s_{bb}(t)$ is given by the following expression (where the energy ε has been chosen so as to normalize the signal amplitude: $\varepsilon = \frac{T_b}{2}$):

$$s_{bb}(t) = \mathrm{e}^{j\phi(t)} \ . \tag{5}$$

In the remaining part of this section, the different parameters of the modulation are investigated, starting with the so-called bandwidth-bit period product BT_b which will reveal the advantage of GFSK over ordinary FSK, and going on with the modulation index h and the amplitude parameter K.

4.1.2.1 Bandwidth-bit period product

The bandwidth-bit period product is an important parameter as it defines the shape of the Gaussian filter.

As illustrated in Figure 10, the higher the BT_b value, the narrower the pulse. An infinite value reduces GFSK to FSK as the pulse shape then tends to a rectangle.

On the other hand, in the spectral domain, the lower the BT_b value, the lower the sideband power. This effect, shown in Figure 11, demonstrates the advantage of using a Gaussian filter in the framework of a narrow-channels system such as BLE, as the interference between neighbouring channels (which are 2-MHz wide) is reduced.

However, this interchannel interference (ICI) reduction is done at the expense of the introduction of intersymbol interference (ISI) as the pulse is no longer concentrated into one single symbol period, but overlaps with the previous and following symbols, as displayed in Figure 12.

In short, GFSK allows to trade ISI performance for ICI improvement. BLE uses a BT_b value of 0.5, which produces better ISI but poorer ICI characteristics with respect to GSM for example (which uses $BT_b = 0.3$ [23]). On the other hand, with respect to the case $BT_b = 0.3$, the ICI becomes lower when further neighbouring channels are considered, as shown in Figure 11, so that ultimately a BT_b value of 0.5 appears to offer a nice ICI performance.

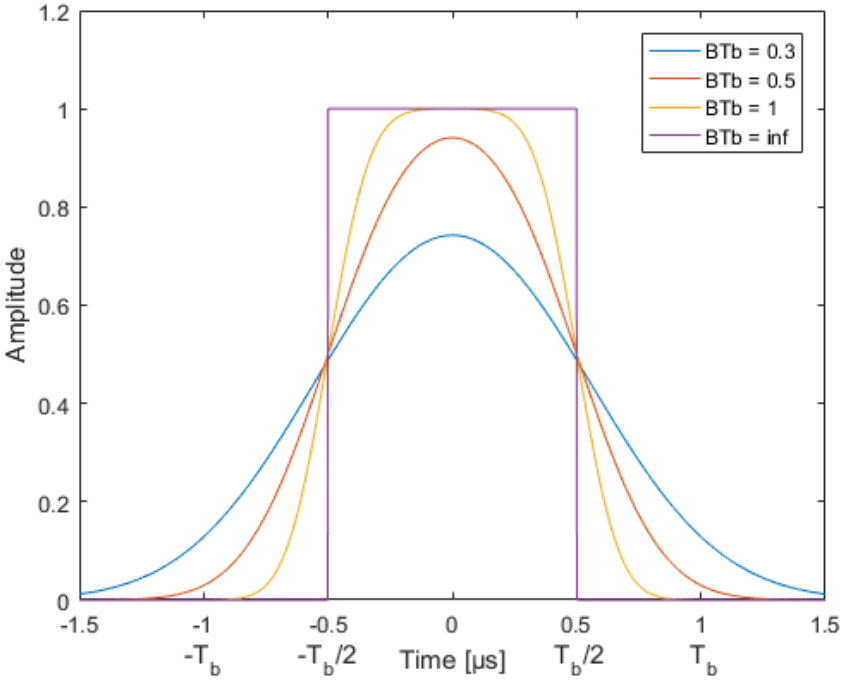


Figure 10: GFSK pulse shaping filter - Time domain.

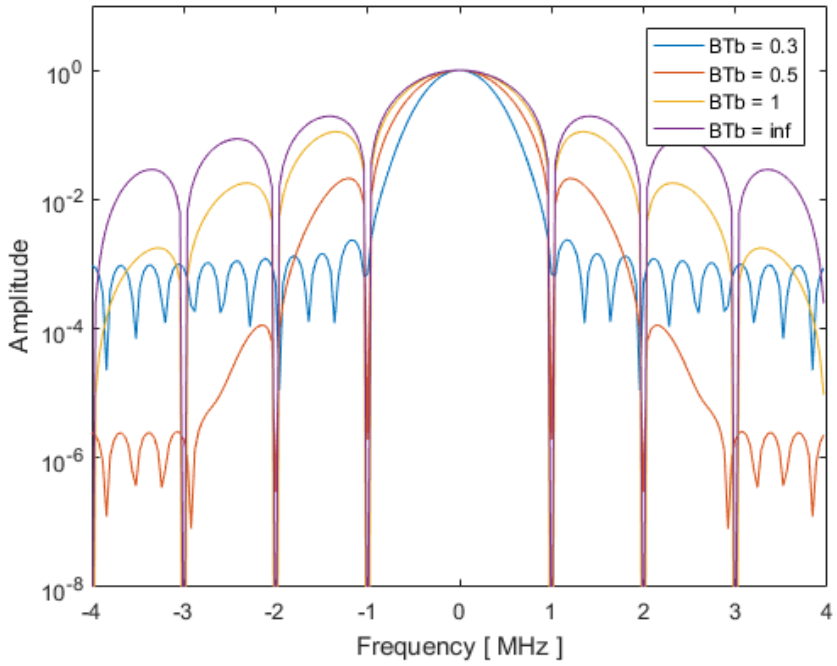


Figure 11: GFSK pulse shaping filter - Frequency domain.

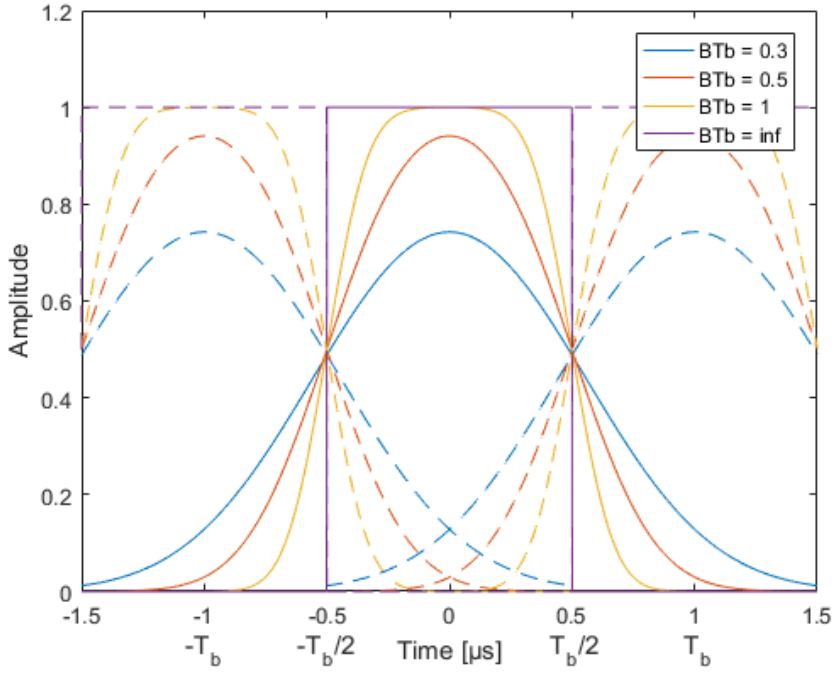


Figure 12: GFSK pulse shaping filter - Time domain: ISI introduction.

4.1.2.2 Modulation index

The modulation index is related to the symbol period (T_b) and the frequency deviation (Δ_f) and shows that the gap between f_0 and f_1 is given by

$$f_1 - f_0 = 2\Delta_f = \frac{h}{T_b} \ .$$

Going back to the FSK waveforms $s_0(t)$ and $s_1(t)$ from Equation (1), their correlation coefficient ρ_{01} can be computed as

$$\begin{aligned}
\rho_{01} &= \frac{1}{\sqrt{\varepsilon\varepsilon}} \int_0^{T_b} s_0(t) s_1(t) dt \\
&= \frac{2}{T_b} \int_0^{T_b} \cos(2\pi f_c t - 2\pi\Delta_f t)\ \cos(2\pi f_c t + 2\pi\Delta_f t)\ dt \\
&= \frac{2}{T_b} \int_0^{T_b} \frac{1}{2} \Big(\cos(4\pi f_c t) + \cos(4\pi\Delta_f t) \Big)\ dt \\
&= \frac{1}{T_b} \int_0^{T_b} \cos(4\pi\Delta_f t)\ dt \\
&= \frac{1}{T_b 4\pi\Delta_f} \sin(4\pi\Delta_f T_b) \\
&= \mathrm{sinc}(4\pi\Delta_f T_b) \\
&= \mathrm{sinc}(2\pi h) \ .
\end{aligned}$$

Therefore, in order to get orthogonal waveforms (i.e. $\rho_{01} = 0$), the modulation index h must be such that

$$h = \frac{k}{2} \qquad k \in \mathbb{Z}_0 \ .$$

This means that the modulation index used in BLE $h = 0.5$ is the minimum value for which orthogonality is obtained. In this case, (Gaussian) Frequency Shift Keying is renamed (Gaussian) Minimum Shift Keying (MSK).

Since this index sets the spacing between f_0 and f_1, a high value allows to reduce the ISI, but this is done at the expense of a larger bandwidth. Conversely, a low value is helpful to reduce the bandwidth but introduces more ISI.

4.1.2.3 Filter amplitude parameter

The filter amplitude parameter K from Equation (2) and Equation (4) must be chosen such that the integral of the filter $g_{GFSK}(t)$ is normalized to the symbol period T_b, otherwise the modulation index would not represent the true frequency deviation.

An intuitive approach is adopted to evaluate the considered integral

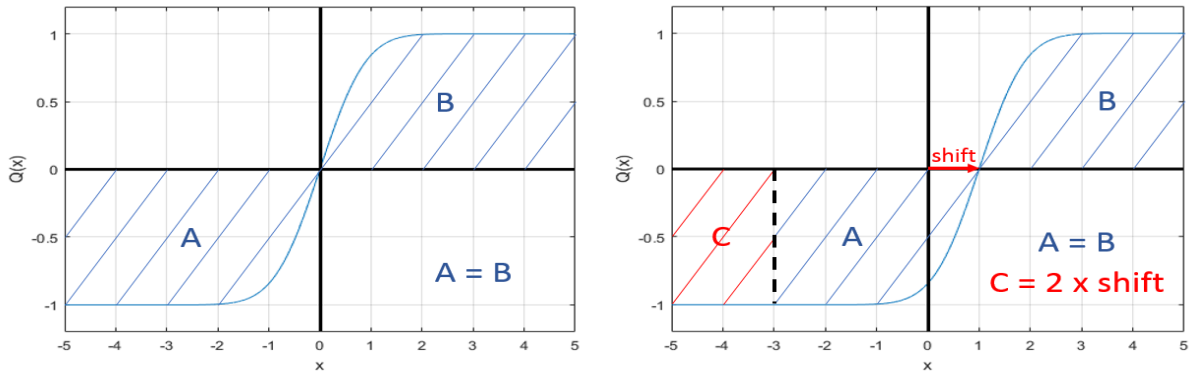
$$\int_{-\infty}^{\infty} g_{GFSK}(t)\ dt = \int_{-\infty}^{\infty} -\frac{K}{2\sqrt{ln(2)}} \left[Q\left(2B\pi \frac{t - \frac{T_b}{2}}{\sqrt{ln(4)}}\right) - Q\left(2B\pi \frac{t + \frac{T_b}{2}}{\sqrt{ln(4)}}\right) \right] dt \ . \tag{6}$$

Recalling that the error function $Q(x)$ is a sigmoid function with two horizontal asymptotes in -1 and 1, it can be seen that $\int_{-\infty}^{\infty} Q(t-\tau)dt = -2\tau$ since the shift of τ applied on this odd function unbalances its integral which is therefore no longer zero but equal to two times the shift value (in absolute value). This is illustrated in Figure 13.

Based on this, the integral in Equation (6) becomes

$$\begin{aligned} \int_{-\infty}^{\infty} g_{GFSK}(t)\ dt &= -\frac{K}{2\sqrt{ln(2)}}(-T_b - T_b) \\ &= \frac{K}{\sqrt{ln(2)}}\ T_b \ . \end{aligned}$$

The conclusion is that the amplitude parameter K must be such that $K = \sqrt{ln(2)}$.



(a) Error function.

(b) Shifted error function.

Figure 13: GFSK filter integration - Graphical resolution.

4.2 Summary and Theoretical Computational Complexity

This section presents a summary of the operations which need to be performed at the transmitter, together with their block diagram representation and a final theoretical evaluation of their computational complexity.

4.2.1 Constellation Mapping

First, the bits must be mapped to the constellation symbols. This process is called constellation mapping. In this modulation scheme, the mapping follows the one displayed in Table 2.

Bit	Symbol
0	-1
1	1

Table 2: Constellation mapping.

This mapping can be performed as follows:

$$\text{symbol} = (\text{bit} - 0.5) \times 2 \ .$$

The corresponding block diagram is depicted in Figure 14.

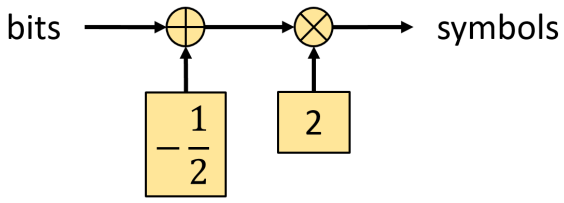


Figure 14: Constellation Mapping block diagram.

4.2.2 Pulse-Shaping Filter Convolution

Next, these symbols are shaped by the convolution with the filter g_{GFSK} from Equation (4). This convolution is performed in digital by discretizing the filter with a factor α_{reso}, representing the number of coefficients taken inside each symbol period, and by upsampling the symbol sequence using the same resolution factor. Its value is set to $\alpha_{reso} = 8$, which corresponds to the oversampling factor value used at the receiver side, which is discussed in Section 5.3.4.

This can be visualized as in Figure 15.

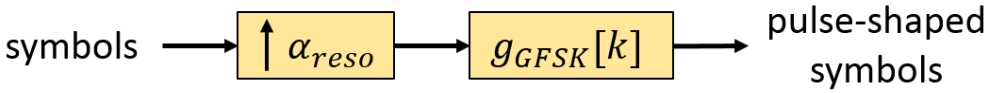


Figure 15: Pulse-Shaping Filter convolution block diagram.

4.2.3 Integration

Then, the resulting signal *convo* is integrated and multiplied by $\frac{\pi h}{T_b}$ to produce the phase signal $\phi(t)$ from Equation (3).

The integral can be computed using the trapeziums method, which consists in estimating the area under a curve segment as the area of the trapezium obtained after linearly interpolating the curve between the two segments extremities. Thus, each discrete value of the signal integral can be incrementally computed by summing the signal value at that point with the signal value at the previous point, dividing this sum by two and multiplying it by the segment length (here equal to $\frac{T_b}{\alpha_{reso}}$) to obtain the trapezium area, and finally adding this area to the integral value computed at the previous point. This is illustrated as a block diagram in Figure 16. The multiplication by a factor $\frac{T_b}{2\alpha_{reso}}$ involved in the trapezium area computation can be performed at the end and thereby directly associated with the multiplication by the phase amplitude factor $\frac{\pi h}{T_b}$ so as to end up with one single factor equal to $\frac{T_b}{2\alpha_{reso}} \times \frac{\pi h}{T_b} = \frac{\pi h}{2\alpha_{reso}}$.

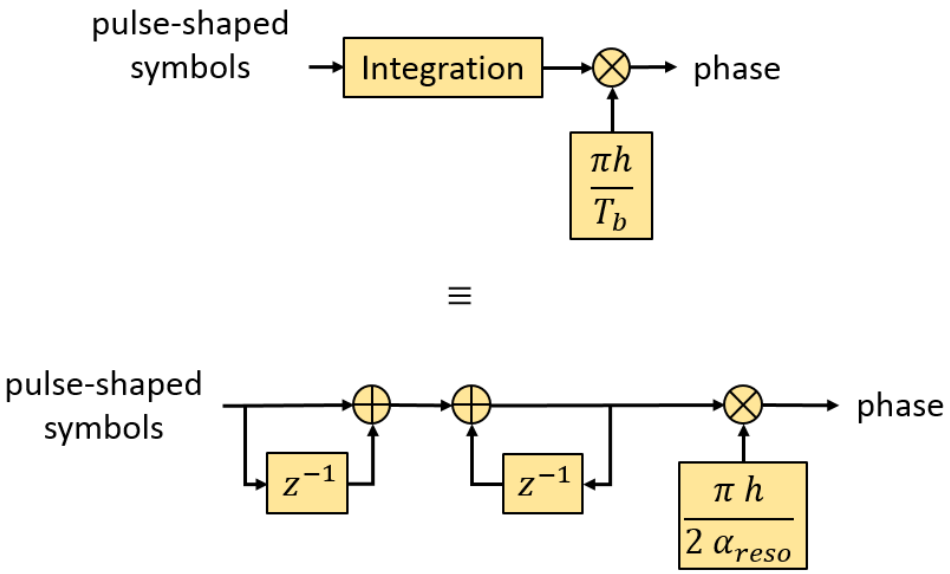


Figure 16: Integration block diagram.

4.2.4 Phase Modulation and Upconversion

Finally comes the RF portion, which handles the phase modulation and the upconversion to the carrier frequency. This part is not considered in this work as the RF block is part of the analog generic portion of the radio. However, it is important that its input signal does not contain any BLE specificity that would make it incompatible with this generic RF block.

4.2.5 Putting It All Together

Figure 18 displays a block diagram of the transmitter representing all these operations. An example of simplified analog RF portion is also presented for the sake of completeness even if it is not part of this work, but it is shaded to highlight the digital baseband portion. Single arrows indicate real signals, whereas double arrows are used for complex signals.

In terms of computational complexity, this block diagram shows that the transmitter is quite simple. The required operations are

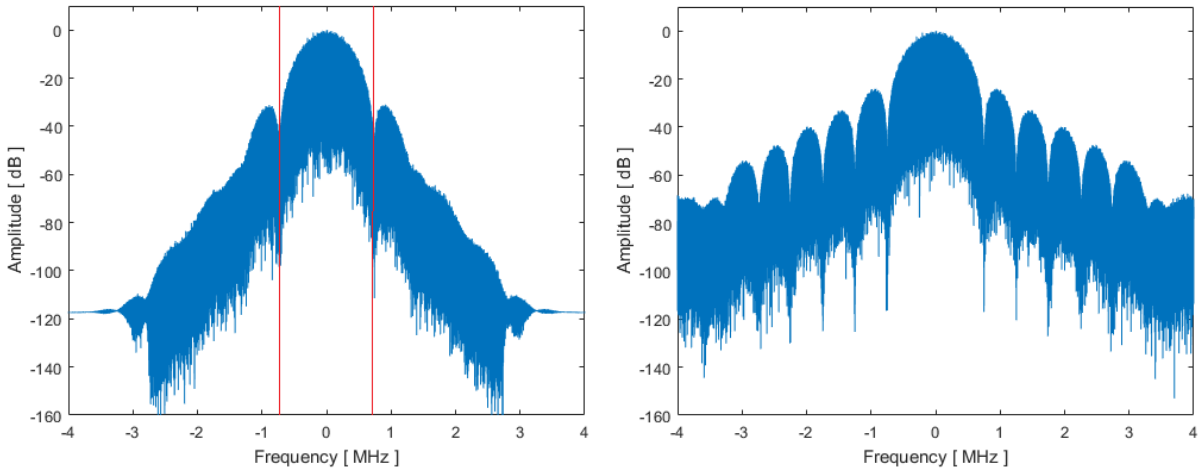
- 1 convolution;
- 1 upsampling;
- 2 delays;
- 2 multiplications;

- 3 additions.

From this list of operations, the most demanding one is the convolution from the pulse-shaping filter block. For a filter of length L, it requires itself L multiplications and $L-1$ additions, as well as $L-1$ delays (because after reception of a first sample, the convolution needs the $L-1$ next ones to complete). In this case, the length of the filter is given by $L = 3 \times \alpha_{reso} + 1$. This is the minimum number of coefficients that is necessary in order to correctly represent the Gaussian shape.

Finally, the spectrum of the transmitted signal $s(t)$ is shown in Figure 17. A comparison is made with respect to the FSK case. This demonstrates the reduction of the sideband power thanks to the Gaussian filter.

Moreover, it can be seen in Figure 17a that the main lobe extends roughly from -0.75 MHz to 0.75 MHz. This span order of magnitude is important to keep in mind for the sampling stage specification on the receiver side.



(a) GFSK case. (b) FSK case.

Figure 17: Spectrum of the transmitted signal.

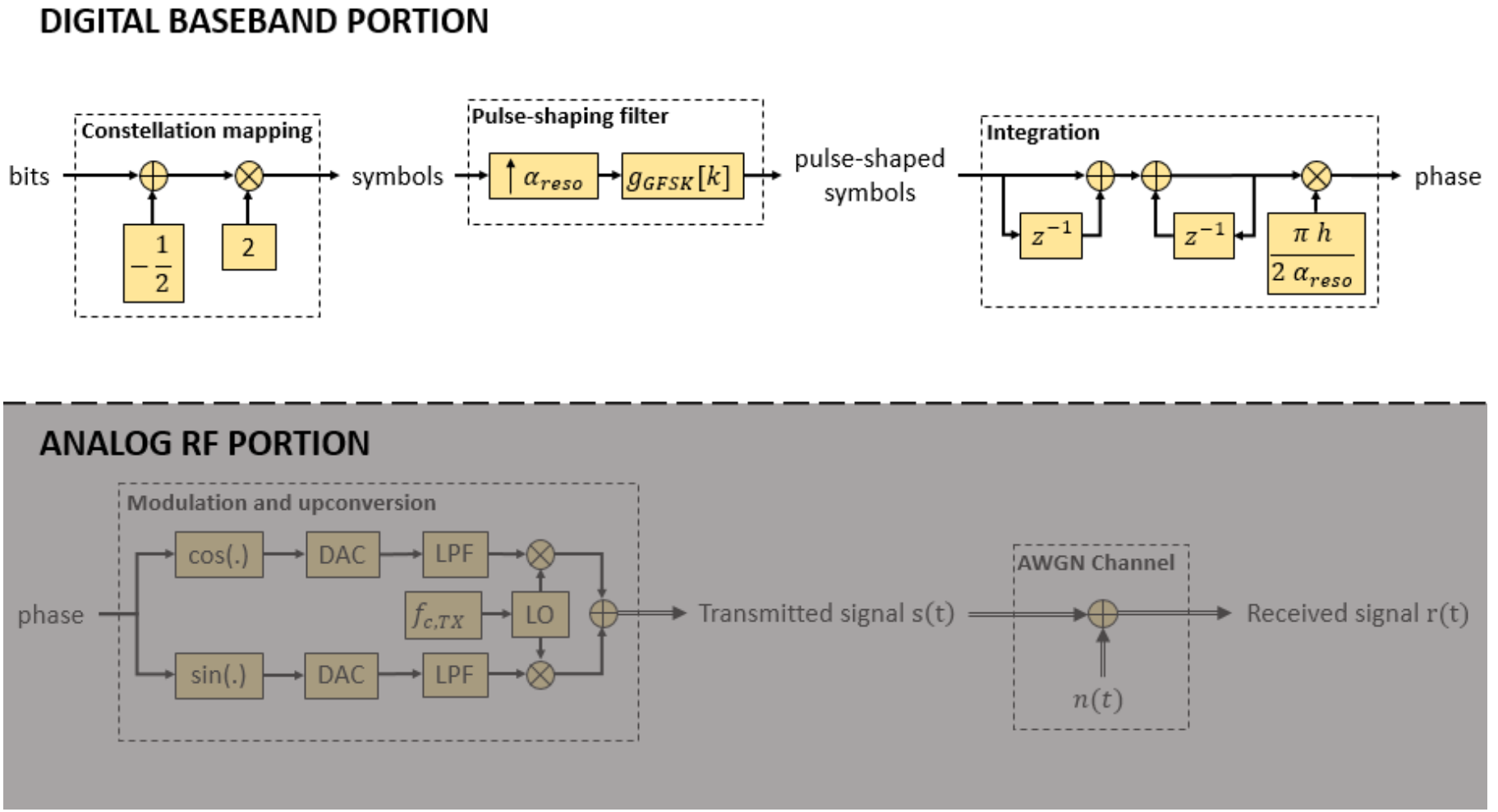


Figure 18: Transmitter block diagram.

5 Receiver

This section presents the design of the digital part of the receiver, detailing each processing block and evaluating their performance via Matlab simulations. These simulations only consider a transmission through an additive white Gaussian noise (AWGN) channel. The processing blocks consist in the demodulation and in the non-idealities mitigation algorithms. Moreover, the search for the minimum required energy per bit to noise power spectral density ratio is realized.

5.1 Analog-to-Digital Conversion

The analog-to-digital conversion (ADC) design is not part of this work as it is not a digital process. However, before entering the baseband digital processing block, it is important to have a look at the desirable spectrum of the signal to be dealt with. Therefore, it is necessary to specify what is expected from the ADC block.

First, as the analog received signal will be sampled, it is important to filter it beforehand in order to avoid aliasing from the spectrum repetition due to the sampling operation. It is considered here that this anti-aliasing filter (AAF) is such that spectrum folding is just avoided, i.e. it acts as an ideal low-pass filter with a cut-off frequency exactly equal to half the sampling rate f_s.

Next, it is important to precise that the sampling is performed at a rate $f_s = \alpha_{OS} \times \frac{1}{T_b}$, where α_{OS} is an oversampling factor. This oversampling is realized in order to get more resolution in the signals to ensure a good working of the subsequent processing steps of the reception algorithm.

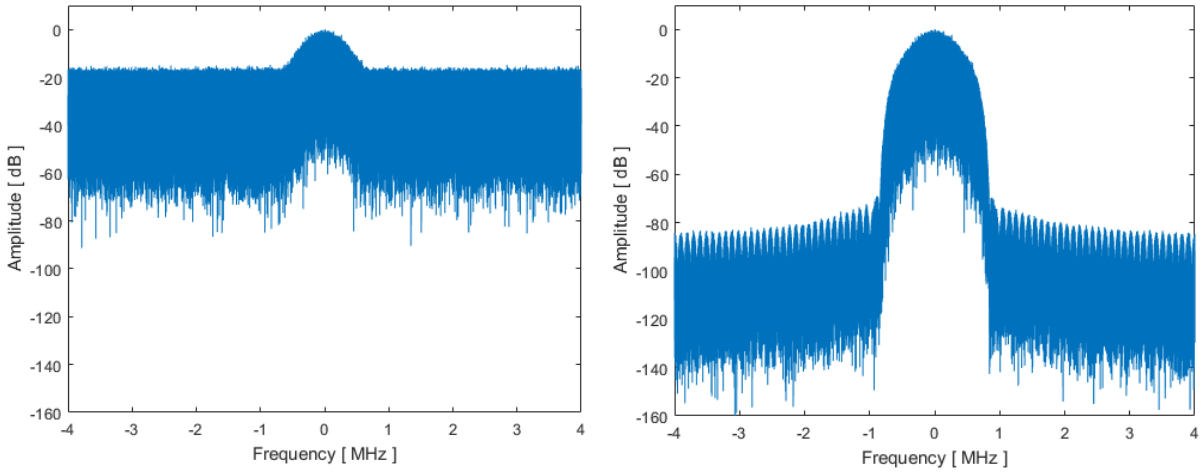
Finally, a digital low-pass filter (LPF) is required in order to filter out as much noise as possible. But this is not part of the ADC anymore, this LPF is the first digital block of the digital processing chain. It is discussed hereafter in Section 5.2.

5.2 Low-Pass Filter

The goal of the LPF directly following the ADC is to end up at its output with a spectrum as narrow as possible to remove as much noise as possible and to occupy the smallest possible bandwidth, while keeping enough useful information for the data recovery. This tradeoff in the choice of the LPF cut-off frequency has been addressed a posteriori by comparing the resulting bit error rate for different cut-off frequency values.

The result of this comparison leads the choice towards a 3-dB cut-off frequency $f_{cut} = 0.6$ MHz. This choice seems consistent because it is shown in Section 4.2 that the main lobe of the transmitted signal spectrum spans roughly from -0.75 MHz to 0.75 MHz.

This means that it is necessary to go from the spectrum of the received signal presented in Figure 19a to the spectrum after sampling and low-pass filtering presented in Figure 19b.



(a) Before ADC.

(b) After ADC and filtering.

Figure 19: Spectrum of the received signal.

To do so, the digital low-pass filter $h_{LPF}[n]$ has been implemented as a Hamming windowed-sinc filter, which consists in pointwise multiplying a truncated sinc function such as the one represented in Figure 20 with a Hamming window $w_{Ham}[n]$ as follows:

$$h_{LPF}[n] = \text{sinc}\left(2\pi f_{cut}\left(\frac{n}{M} - \frac{1}{2}\right)N_{per}T_b\right) \cdot w_{Ham}[n] \ ,$$

where $w_{Ham}[n] = 0.54 - 0.46\cos(\frac{2\pi n}{M})$, N_{per} represents the support of the filter as a number of symbol periods (illustrated in Figure 20 where $N_{per} = 10$), f_{cut} is the cut-off frequency, and $n = 0...M$ with $M = \alpha_{OS} \times N_{per}$.

Figure 21a represents the resulting LPF in time domain whereas Figure 21b represents it in frequency domain. The signal after sampling and filtering is obtained with this LPF and its spectrum is the one displayed in Figure 19b.

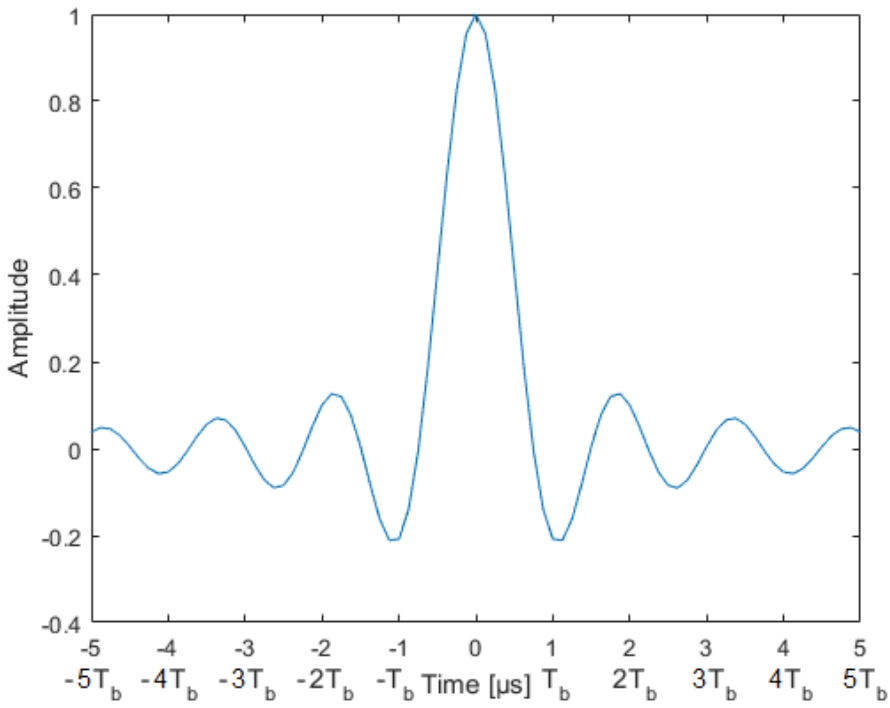
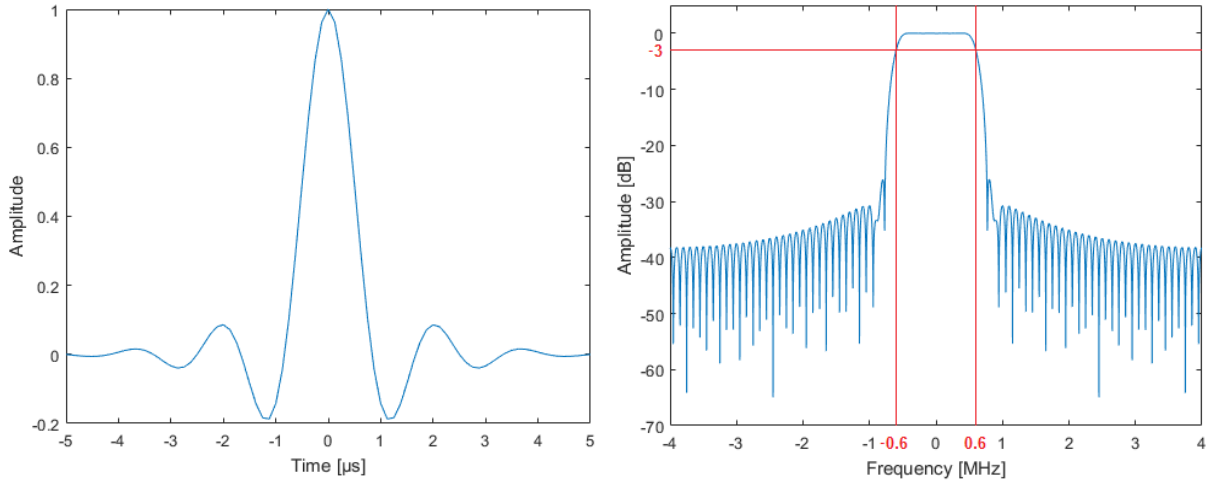


Figure 20: Truncated sinc function ($N_{per} = 10$).



(a) Time domain.

(b) Frequency domain.

Figure 21: Low-pass filter.

5.3 Demodulation

As briefly discussed in Section 1.3, there exists a large choice of GFSK demodulation algorithms. However, from a low-complexity point of view, the phase differential detector appears to be the most interesting one. It also has the advantage of being a non-coherent technique, meaning that the recovery of the phase is not necessary. Therefore, this is the

one that has been selected.

5.3.1 Differential Demodulation

The differential demodulation consists in comparing the phase of the received signal at two different instants, separated by a delay Δ. This comparison is represented by the difference of the phase at those two instants and is denoted $\delta\phi$. It is evaluated at the discrete instant $\frac{n}{\alpha_{OS}}$. This notation with the ratio with respect to the oversampling factor α_{OS} is used to highlight the presence of oversampling by considering that two integer indices are separated by a number of discrete instants equal to α_{OS}. In other words, the discrete-time version of a continuous signal $f(t)$ is defined as $f[n] = f(nT_b)$ or still, after oversampling, $f\left[\frac{n}{\alpha_{OS}}\right] = f\left(n\frac{T_b}{\alpha_{OS}}\right)$. In short, the following phase difference is considered:

$$\delta\phi\left[\frac{n}{\alpha_{OS}}\right] = \phi\left[\frac{n}{\alpha_{OS}}\right] - \phi\left[\frac{n-\Delta}{\alpha_{OS}}\right] .$$

The idea behind this is that the sign of this phase difference gives the desired information of the bit that was transmitted during the time interval $\left[\frac{(n-\Delta)}{\alpha_{OS}}T_b \; ; \; \frac{n}{\alpha_{OS}}T_b\right]$. Indeed, starting from Equation (3), the phase difference is given by

$$\begin{aligned}
\delta\phi\left[\frac{n}{\alpha_{OS}}\right] &= \left(\pi\frac{h}{T_b}\int_{-\infty}^{\frac{n}{\alpha_{OS}}T_b}\sum_{k=-\infty}^{+\infty} I[k]g_{GFSK}(\tau - kT_b)\; d\tau\right) \\
&\quad - \left(\pi\frac{h}{T_b}\int_{-\infty}^{\frac{(n-\Delta)}{\alpha_{OS}}T_b}\sum_{k=-\infty}^{+\infty} I[k]g_{GFSK}(\tau - kT_b)\; d\tau\right) \\
&= \pi\frac{h}{T_b}\int_{\frac{(n-\Delta)}{\alpha_{OS}}T_b}^{\frac{n}{\alpha_{OS}}T_b}\sum_{k=-\infty}^{+\infty} I[k]g_{GFSK}(\tau - kT_b)\; d\tau \; . \qquad (7)
\end{aligned}$$

Therefore, as the convolution $\sum_{k=-\infty}^{+\infty} I[k]g_{GFSK}(\tau - kT_b)$ gives a waveform similar to the NRZ (non-return-to-zero) binary code waveform (with smooth transitions due to the Gaussian filter as depicted in Figure 22), the sign of its integral will be the same as the sign of the symbol transmitted during the considered interval. This is always true on the condition that the bounds of the integral are well chosen, i.e. one before and one after the instant of transmission of the considered symbol, and separated by at most one symbol period. The latter condition restricts the choice of the delay Δ, but this choice is further discussed in Section 5.3.2.

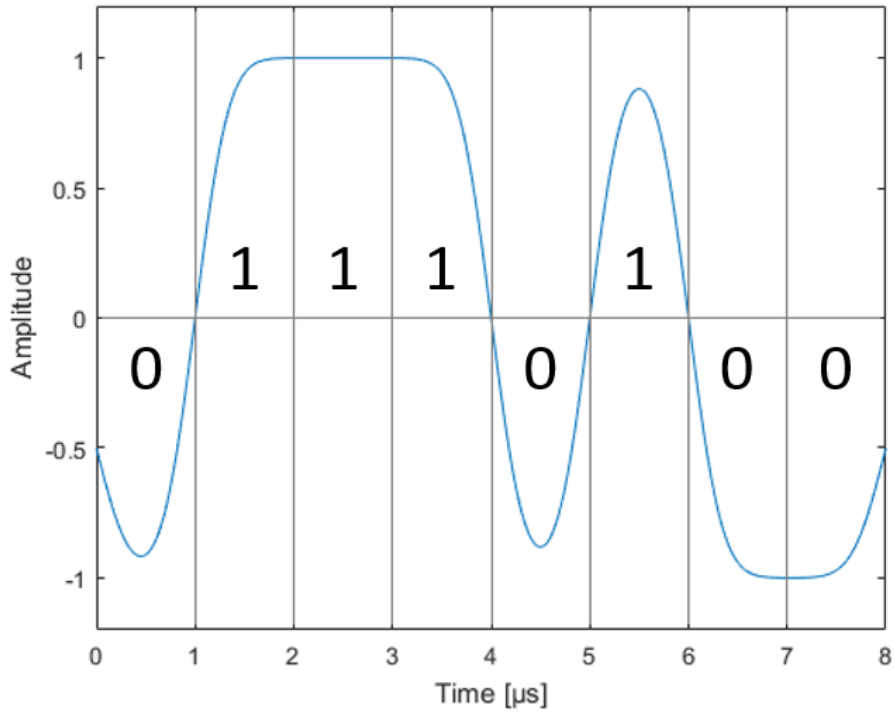


Figure 22: NRZ-like waveform of the convolution between the symbols and the pulse-shaping filter.

As a result, the demodulation can be performed by taking the sign of the phase difference $\delta\phi\left[\frac{n}{\alpha_{OS}}\right]$, yielding an ideal demodulated signal $d_{ideal}\left[\frac{n}{\alpha_{OS}}\right]$ such that

$$d_{ideal}\left[\frac{n}{\alpha_{OS}}\right] = \text{sgn}\left(\delta\phi\left[\frac{n}{\alpha_{OS}}\right]\right) . \tag{8}$$

This differential demodulation technique requires the ability to extract the phase of the received signal. This can be done through the succession of the following operations.

Firstly, the so-called Hermitian product is performed on the received signal in base-band and oversampled $r_{bb}\left[\frac{n}{\alpha_{OS}}\right]$. It consists in the product between the signal and the complex conjugate of a delayed version of itself. The resulting signal H_p is given by

$$H_p\left[\frac{n}{\alpha_{OS}}\right] = r_{bb}\left[\frac{n}{\alpha_{OS}}\right] \times r_{bb}^{*}\left[\frac{(n-\Delta)}{\alpha_{OS}}\right] ,$$

which, considering the noise-free case $H_{p,nf}$, can be developed as

$$
\begin{aligned}
H_{p,nf}\left[\frac{n}{\alpha_{OS}}\right] &= s_{bb}\left[\frac{n}{\alpha_{OS}}\right] \times s^{*}_{bb}\left[\frac{(n-\Delta)}{\alpha_{OS}}\right] \\
&= \left(\cos\left(\phi\left[\frac{n}{\alpha_{OS}}\right]\right) + j\,\sin\left(\phi\left[\frac{n}{\alpha_{OS}}\right]\right)\right) \\
&\times \left(\cos\left(\phi\left[\frac{(n-\Delta)}{\alpha_{OS}}\right]\right) - j\,\sin\left(\phi\left[\frac{(n-\Delta)}{\alpha_{OS}}\right]\right)\right) \\
&= \left(\cos\left(\phi\left[\frac{n}{\alpha_{OS}}\right]\right) \times \cos\left(\phi\left[\frac{(n-\Delta)}{\alpha_{OS}}\right]\right) + \sin\left(\phi\left[\frac{n}{\alpha_{OS}}\right]\right) \times \sin\left(\phi\left[\frac{(n-\Delta)}{\alpha_{OS}}\right]\right)\right) \\
&+ j\left(\sin\left(\phi\left[\frac{n}{\alpha_{OS}}\right]\right) \times \cos\left(\phi\left[\frac{(n-\Delta)}{\alpha_{OS}}\right]\right) - \cos\left(\phi\left[\frac{n}{\alpha_{OS}}\right]\right) \times \sin\left(\phi\left[\frac{(n-\Delta)}{\alpha_{OS}}\right]\right)\right) \\
&= \cos\left(\phi\left[\frac{n}{\alpha_{OS}}\right] - \phi\left[\frac{n-\Delta}{\alpha_{OS}}\right]\right) + j\,\sin\left(\phi\left[\frac{n}{\alpha_{OS}}\right] - \phi\left[\frac{n-\Delta}{\alpha_{OS}}\right]\right) \\
&= \cos\left(\delta\phi\left[\frac{n}{\alpha_{OS}}\right]\right) + j\,\sin\left(\delta\phi\left[\frac{n}{\alpha_{OS}}\right]\right) . \qquad (9)
\end{aligned}
$$

Secondly, the imaginary part of the Hermitian product $I_{H,nf}\left[\frac{n}{\alpha_{OS}}\right]$ is extracted, which gives

$$
\begin{aligned}
I_{H,nf}\left[\frac{n}{\alpha_{OS}}\right] &= \Im\left\{H_{p,nf}\left[\frac{n}{\alpha_{OS}}\right]\right\} \\
&= \sin\left(\delta\phi\left[\frac{n}{\alpha_{OS}}\right]\right) .
\end{aligned}
$$

Lastly, the sign function is applied on $I_{H,nf}\left[\frac{n}{\alpha_{OS}}\right]$ to generate the desired demodulated signal $d_{nf}\left[\frac{n}{\alpha_{OS}}\right]$ from Equation (8) as

$$
\begin{aligned}
d_{nf}\left[\frac{n}{\alpha_{OS}}\right] &= \mathrm{sgn}\left(I_{H,nf}\left[\frac{n}{\alpha_{OS}}\right]\right) \\
&= \mathrm{sgn}\left(\sin\left(\delta\phi\left[\frac{n}{\alpha_{OS}}\right]\right)\right) \\
&= \mathrm{sgn}\left(\delta\phi\left[\frac{n}{\alpha_{OS}}\right]\right) ,
\end{aligned}
$$

where the last equality is only true if the phase difference $\delta\phi\left[\frac{n}{\alpha_{OS}}\right]$ is such that

$$\delta\phi\left[\frac{n}{\alpha_{OS}}\right] \in \left[-\pi \ ; \ \pi\right] \ .$$

This condition is also to be taken into account in the choice of the delay Δ, discussed in Section 5.3.2.

This was all done in the noise-free case, but this does not hold in practice, so that the actual demodulation signal $d\left[\frac{n}{\alpha_{OS}}\right]$ is only an approximation of $d_{nf}\left[\frac{n}{\alpha_{OS}}\right]$.

In conclusion, the actual demodulation signal is given by

$$\begin{aligned} d\left[\frac{n}{\alpha_{OS}}\right] &= \mathrm{sgn}\left(\Im\left(r_{bb}\left[\frac{n}{\alpha_{OS}}\right] \times r_{bb}^{*}\left[\frac{(n-\Delta)}{\alpha_{OS}}\right]\right)\right) \\ &\simeq \mathrm{sgn}\left(\delta\phi\left[\frac{n}{\alpha_{OS}}\right]\right) \ . \end{aligned}$$

Two elements still need to be discussed, which is done hereafter in Section 5.3.2 and Section 5.3.3.

First, the choice of the delay Δ, about which the main parts of the answer have already been pointed out in this section.

Second, the discussion about using or not all the α_{OS} samples per symbol period, about when to perform the decimation to get back to a sequence with a symbol rate equal to the original one (T_b), and about which sample among the α_{OS} ones to keep for each symbol period while decimating.

5.3.2 Delay Choice

As already pointed out in Section 5.3.1, there are two conditions on the delay Δ involved in the detection process for the demodulation signal to behave properly.

First, the delay must be such that the two instants at which the phase difference $\delta\phi\left[\frac{n}{\alpha_{OS}}\right]$ is evaluated are separated by at most one symbol period. This translates to the following condition on Δ:

$$\Delta \leq \alpha_{OS} \ . \tag{10}$$

Next, the delay must be such that the phase difference $\delta\phi\left[\frac{n}{\alpha_{OS}}\right]$ is bounded by π: $\delta\phi\left[\frac{n}{\alpha_{OS}}\right] \in \left[-\pi \ ; \ \pi\right]$.

In order to translate this condition on the phase difference to a condition on the delay, here is the expression of the maximum of the phase difference, starting from Equation (7):

$$\begin{aligned}
\max\left\{\delta\phi\Big[\frac{n}{\alpha_{OS}}\Big]\right\} &= \max\left\{\pi\frac{h}{T_b}\int_{\frac{(n-\Delta)}{\alpha_{OS}}T_b}^{\frac{n}{\alpha_{OS}}T_b}\sum_{k=-\infty}^{+\infty} I[k]g_{GFSK}(\tau - kT_b)\ d\tau\right\} \\
&= \pi\frac{h}{T_b}\int_{\frac{(n-\Delta)}{\alpha_{OS}}T_b}^{\frac{n}{\alpha_{OS}}T_b}\max\left\{\sum_{k=-\infty}^{+\infty} I[k]g_{GFSK}(\tau - kT_b)\right\} d\tau \\
&= \pi\frac{h}{T_b}\int_{\frac{(n-\Delta)}{\alpha_{OS}}T_b}^{\frac{n}{\alpha_{OS}}T_b} 1\ d\tau \\
&= \pi\frac{h}{T_b}\left(\frac{n}{\alpha_{OS}}T_b - \frac{(n-\Delta)}{\alpha_{OS}}T_b\right) \\
&= \pi\frac{h}{T_b}\frac{\Delta}{\alpha_{OS}}T_b \\
&= \pi\frac{h\Delta}{\alpha_{OS}}\ .
\end{aligned}$$

Therefore, as $h = 0.5$ and as the development for the minimum of the phase difference is identical by symmetry of the convolution result, the condition on the phase difference translates to

$$\Delta \leq 2\ \alpha_{OS}\ . \tag{11}$$

Finally, combining the two conditions from Equation (10) and Equation (11), it comes

$$\Delta \leq \alpha_{OS}\ .$$

This leads to two possible approaches.

The first one consists in maximizing the range of the phase difference, which would be beneficial to minimize the risk of getting a negative value instead of a positive one because of the noise (and vice-versa). The way to perform this maximization would be to choose $\Delta = \alpha_{OS}$, which means $\delta\phi\Big[\frac{n}{\alpha_{OS}}\Big] \in \big[-\frac{\pi}{2}\ ;\ \frac{\pi}{2}\big]$.

The second one would be to have a tradeoff between this minimization of the noise effect and the ISI by choosing for example $\Delta = \frac{\alpha_{OS}}{2}$. Indeed, if the range is maximized, the phase difference $\delta\phi\Big[\frac{n}{\alpha_{OS}}\Big]$ will be evaluated at instants just in the middle of two symbols, which means that the result will be affected by the neighbouring symbols.

5.3.3 Decimation

The demodulation signal $d\left[\frac{n}{\alpha_{OS}}\right]$ has been derived, but the question of the optimal instant to evaluate it remains. As the demodulation is based on the phase difference $\delta\phi\left[\frac{n}{\alpha_{OS}}\right]$, this optimal instant corresponds to the one at which this phase difference is the most impacted by the symbol to be demodulated. This occurs when the phase difference is exactly centered on the symbol. Referring to the phase difference expression from Equation (7), it is obtained that

$$\delta\phi\left[\frac{n}{\alpha_{OS}}\right] = \pi\frac{h}{T_b}\int_{\frac{(n-\Delta)}{\alpha_{OS}}T_b}^{\frac{n}{\alpha_{OS}}T_b}\sum_{k=-\infty}^{+\infty} I[k]g_{GFSK}(\tau - kT_b)\ d\tau\ .$$

This means that the integral bounds must be centered on the period of the symbol to be demodulated, denoted as the m^{th} symbol. Then, the condition becomes

$$\frac{1}{2}\left(\frac{n}{\alpha_{OS}}T_b + \frac{(n-\Delta)}{\alpha_{OS}}T_b\right) = \left(m - \frac{1}{2}\right)T_b$$

or still

$$n = \frac{\Delta}{2} + \left(m - \frac{1}{2}\right)\ \alpha_{OS}\ .$$

Considering the two values for Δ proposed in Section 5.3.2: $\Delta_1 = \alpha_{OS}$ and $\Delta_2 = \frac{\alpha_{OS}}{2}$, the corresponding indices n_1 and n_2 are obtained as

$$\begin{cases} n_1 = m\ \alpha_{OS} \\ n_2 = \left(m - \frac{1}{4}\right)\ \alpha_{OS}\ . \end{cases} \tag{12}$$

In the first case, the two bounds of the integral are then $m \times \alpha_{OS}$ and $\left(m-1\right) \times \alpha_{OS}$, while they are $\left(m - \frac{3}{4}\right) \times \alpha_{OS}$ and $\left(m - \frac{1}{4}\right) \times \alpha_{OS}$ in the second case. This shows that the second delay value requires to use two samples by symbol period whereas the first one requires only one. This is the drawback of the second approach.

Thanks to this analysis, it appears that only specific samples are needed for the demodulation. Therefore, decimation can be performed prior to the demodulation block, either by a factor α_{OS} or by a factor $\frac{\alpha_{OS}}{2}$ depending on the delay choice.

However, another approach is also possible without decimation and using the extra samples from the demodulation signal (before the sign extraction) in a weighted mean over one symbol period so as to reduce the noise effect. This approach has been simulated, and it produced a similar bit error rate performance, which can be seen as a sign that the noise effect is already sufficiently reduced by the noise filtering. Therefore, as this approach does not give a better performance and as it is more costly in terms of operation and symbol rate, it has not been adopted.

5.3.4 BER without non-idealities

Here is a first glimpse of the bit error rate (BER), considering perfect synchronization, perfect preamble and Start of Frame Delimiter (SFD) detection. The only imperfection that is present is the AWGN added to the signal during transmission. This first BER evaluation allows to have a bound on the system performance, to which the non-idealities correction algorithms that are presented in the following subsections will try to get as close as possible.

First, Figure 23 presents the BER curve for different values of oversampling factor ($\alpha_{OS} \in \{2, 4, 8, 16\}$) and with the arbitrary choice of $\Delta = \alpha_{OS}$. This comparison shows that the performance improvement becomes very small from an oversampling factor of 4 and becomes insignificant from a factor of 8. Therefore, the value $\alpha_{OS} = 8$ is selected.

Another important conclusion from this bit error rate simulation is that the required maximum value $BER_{max} = 0.1$ % is achieved for values of energy per bit to noise power spectral density ratio [2] $\frac{E_b}{N_0}$ down to 12 dB. This means that with the addition of non-idealities, the minimum required SNR will probably be around $\left(\frac{E_b}{N_0}\right)_{min} = 13$ dB.

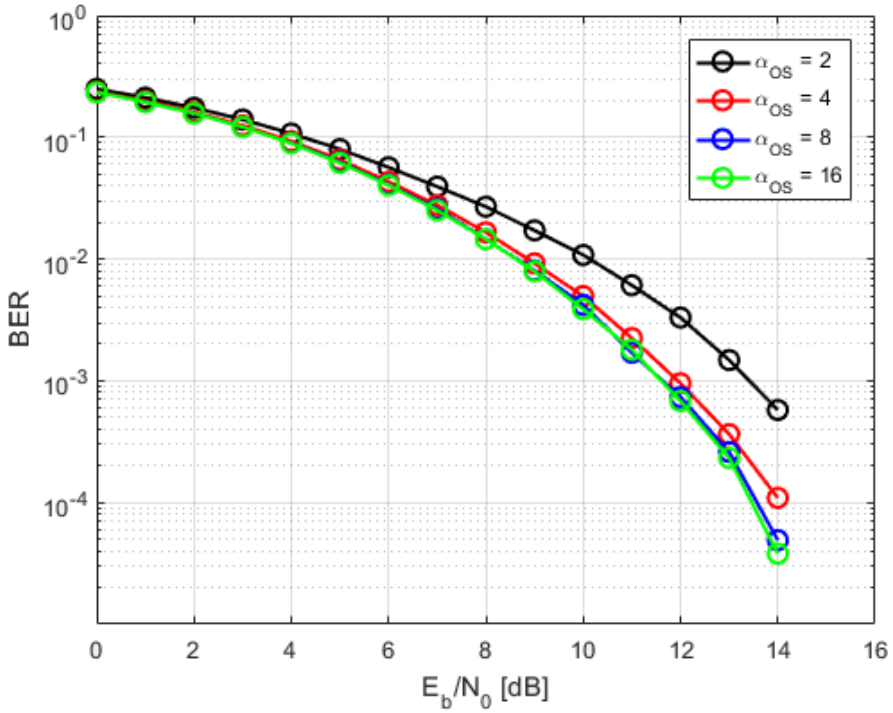


Figure 23: Bit error rate comparison with respect to oversampling factor α_{OS}.

Next, Figure 24 displays a comparison between the two values proposed for the delay

[2] From now on, this energy per bit to noise power spectral density ratio $\frac{E_b}{N_0}$ will be referred to as the signal to noise ratio (SNR) for the sake of wording simplicity. This is a misuse of language as the relationship between those two ratios is given by SNR $= \frac{S}{N} = \frac{E_b R_b}{N_0 B_n}$, meaning that they are not equal since the bit rate R_b is not exactly equal to the noise bandwidth B_n in general.

($\Delta \in \{\alpha_{OS}, \frac{\alpha_{OS}}{2}\}$). The simulation was performed with an oversampling factor $\alpha_{OS} = 8$. This proves the intuitive analysis that a delay exactly equal to the oversampling factor performs well at low signal to noise ratio (SNR) whereas a delay equal to half the oversampling factor yields better performances at high SNR as the ISI effect is then dominant over the noise effect.

As the tolerance for the BER is set to 0.1%, the minimum required SNR is 12 dB in both cases. As this value lies in the range of high SNR values, it means that from this point of view it is better to work with the second delay: $\Delta = \frac{\alpha_{OS}}{2}$. However, recalling that it requires a processing rate two times higher, the first delay $\Delta = \alpha_{OS}$ still has to be considered.

Finally, the choice of $\Delta = \frac{\alpha_{OS}}{2}$ has been made in order to optimize the BER, as discussed later on in Section 5.8.

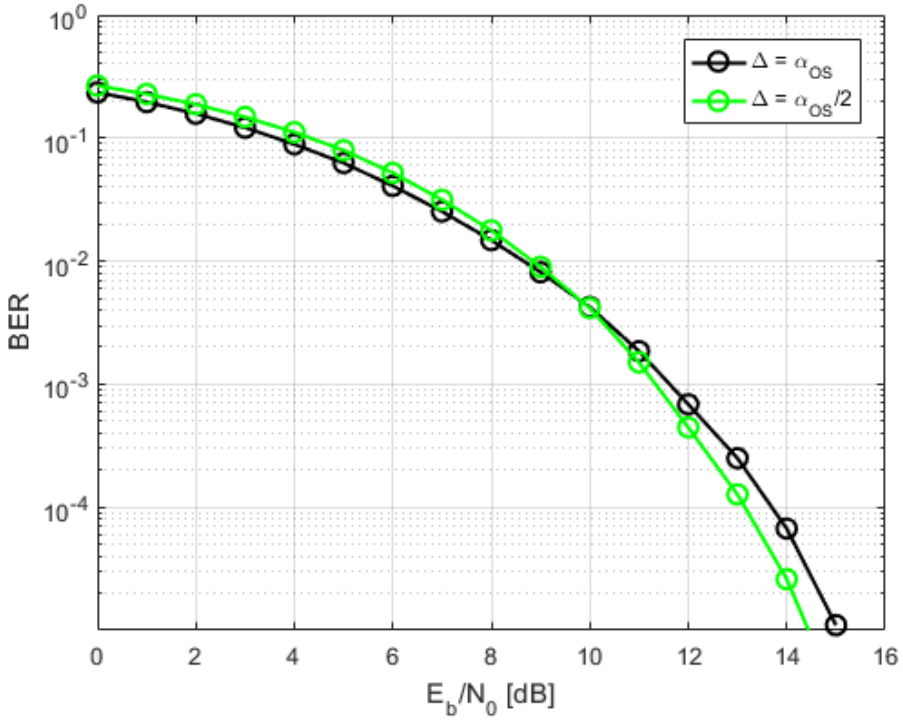


Figure 24: Bit error rate comparison with respect to delay parameter Δ.

5.4 Preamble Detection

In this section, a method to detect the start of the packet is derived. This is an important step as it allows the SDR to remain in sleep mode as long as no data is transmitted, thereby saving energy. Indeed, only the preamble search algorithm will be running, and the remaining of the processing chain will only come into play when the preamble is detected.

A simple idea to detect the preamble over the noisy signal is to perform a correlation between the incoming signal and the same sequence as the preamble. This correlation

will produce a peak at the location of the preamble, and this peak is detected when a predefined threshold is crossed.

This correlation-based method is very effective but is prohibitively costly. Therefore, another approach is preferred. It relies on an infinite impulse response (IIR) filter and comes from [8]. Its presentation is the topic of this section.

5.4.1 IIR Filter

The idea behind this IIR filter technique is to take advantage of the alternating pattern of the preamble, which is either equal to 10101010 or 01010101 (depending on the packet type). Hence, the filter is designed so as to highlight this pattern. Here is the filter proposed in [8]:

$$y[k] = x[k] - 0.875\ y[k - \alpha_{OS}]. \tag{13}$$

The corresponding impulse response is displayed in Figure 25. Its alternating behavior proves that it will highlight an alternating sequence such as the preamble.

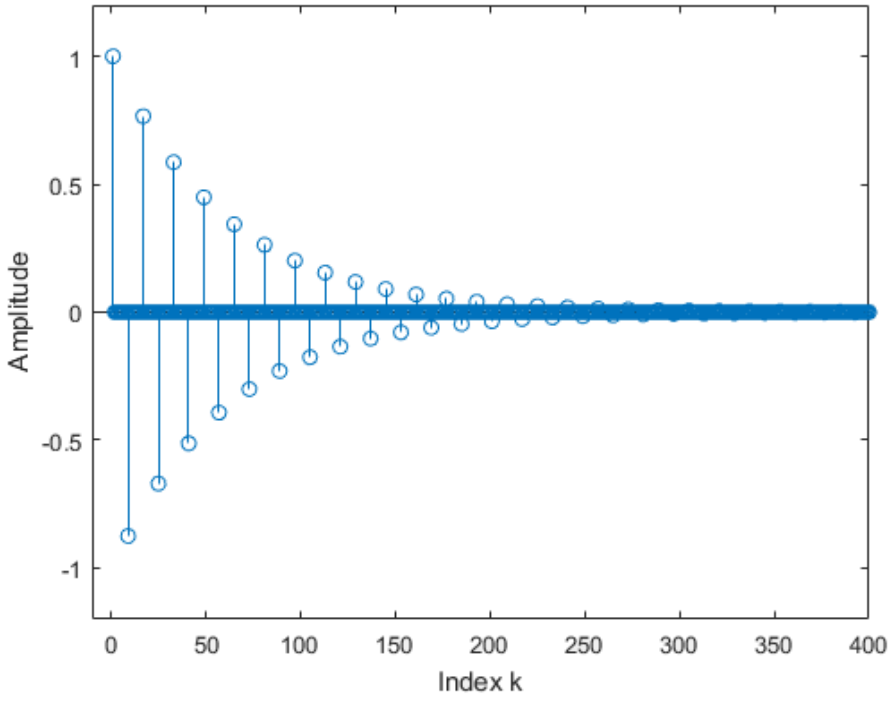


Figure 25: IIR filter impulse response.

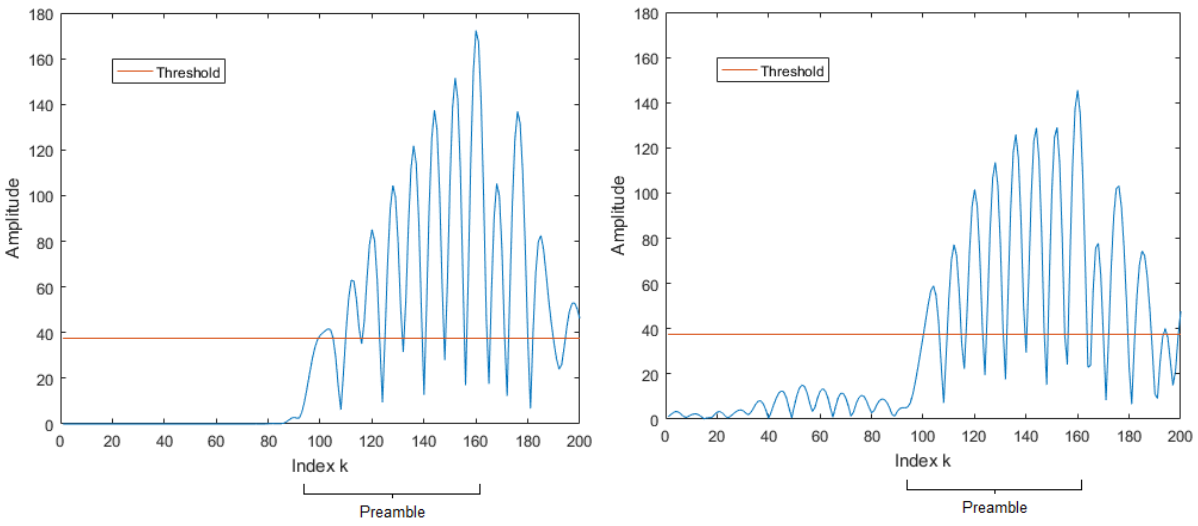
The goal of the preamble search is to avoid the waste of performing demodulation over a noisy signal when no packet is being transmitted. This means that the preamble

detection must be the first operation to take place in the digital signal processing block. Therefore, on one hand, the non-idealities such as the carrier frequency offset (CFO) are not mitigated yet, and on the other hand, the IIR filter cannot be applied to the demodulated sequence.

A first idea could be to apply this filter directly to the received signal in baseband. But this will not be very effective because the preamble signal and the noisy signal before the packet are not clearly enough distinguishable, so that the filter will have trouble to clearly highlight the preamble over the noisy signal.

A solution is to apply the filter to the output of the Hermitian product because this product makes the packet signal more distinguishable over the noisy signal. It is important to precise that the delay used in this Hermitian product is preferably equal to the oversampling factor rather than to half of it. Indeed, intersymbol interference is not a problem here. However, noise is problematic, so the larger the output range, the better.

After the signal has been filtered, its absolute value is taken and it is decided that the preamble is found when a predefined threshold is crossed. Figure 26 presents a possible filter output where the incoming signal starts with nothing but noise.



(a) SNR = ∞.

(b) SNR = 13 dB.

Figure 26: IIR filter output.

5.4.2 Detection Threshold

In order to guarantee that the sample triggering the detection is not part of a random alternating pattern from the noisy signal preceding the preamble, the detection threshold must be sufficiently high. On the other hand, if it is too high, the detection could be only triggered after the preamble. Therefore, the value of the threshold must be carefully selected.

In order to determine the optimal value, i.e. the minimum one which guarantees that the detected sample belongs to the preamble, simulations have been performed with different threshold values. Doing so, an empirical minimal value of 37.5 has been found. This is indeed the smallest value for which the 100 000 iterations of the simulation all produced a crossing of the threshold due to the presence of the preamble. This simulation considered an oversampling factor $\alpha_{OS} = 8$ and a 13-dB SNR. The resulting distribution of the detected sample is displayed in Figure 27. It can be seen that the detected sample almost always belongs to one of the first three preamble periods, which respectively span from indices 97 to 104, 105 to 112 and 113 to 120.

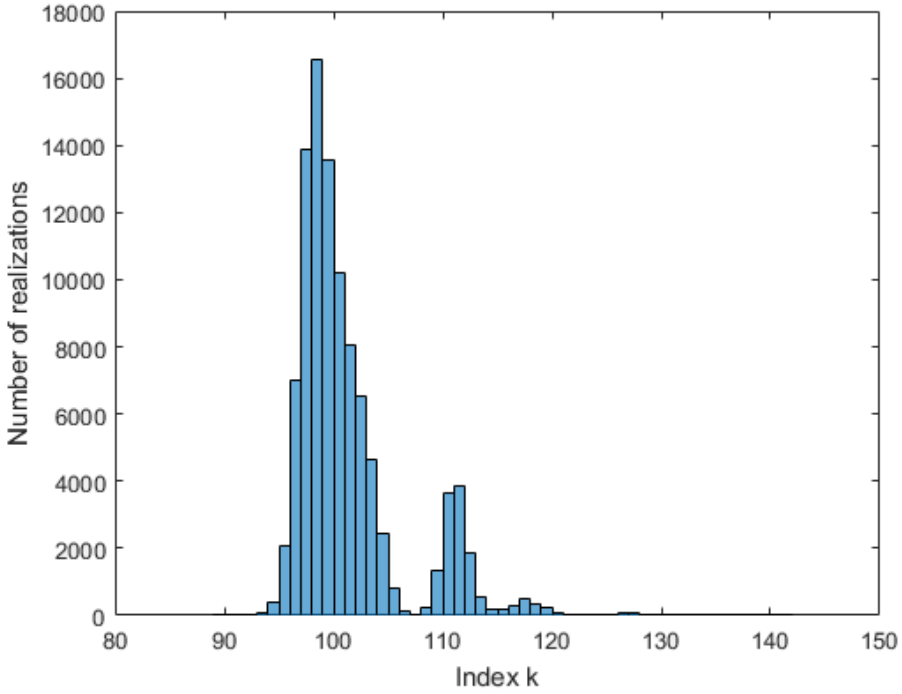


Figure 27: Detected sample index distribution.

Finally, it is noteworthy that some variations in the coefficient value of the IIR filter from Equation (13) coming from [8] have been simulated, but the initial value of 0.875 appears to be the optimal one as it is the one giving the most concentrated distribution of detected sample index.

5.5 Symbol Timing Recovery

Another important task the receiver must carry out is the symbol timing recovery. This is a synchronization process which is necessary in order to identify which of the α_{OS} samples per symbol period each considered sample is, which is crucial in order to select the right samples during the downsampling taking place before the demodulation.

The importance of this synchronization step is illustrated in Figure 28, where the BER is evaluated for three different synchronizations: a perfect one and two erroneous ones which are 1 and 2 samples away from the correct one, i.e. for symbol timing offsets STO $\in \left\{\frac{0}{\alpha_{OS}}, \frac{1}{\alpha_{OS}}, \frac{2}{\alpha_{OS}}\right\}$. An oversampling factor $\alpha_{OS} = 8$ and a delay $\Delta = \frac{\alpha_{OS}}{2}$ are considered. It can be observed that the BER gets considerably degraded when the synchronization is performed on a sample more than one sample away from the correct one, i.e. for STO $> \frac{1}{8}$.

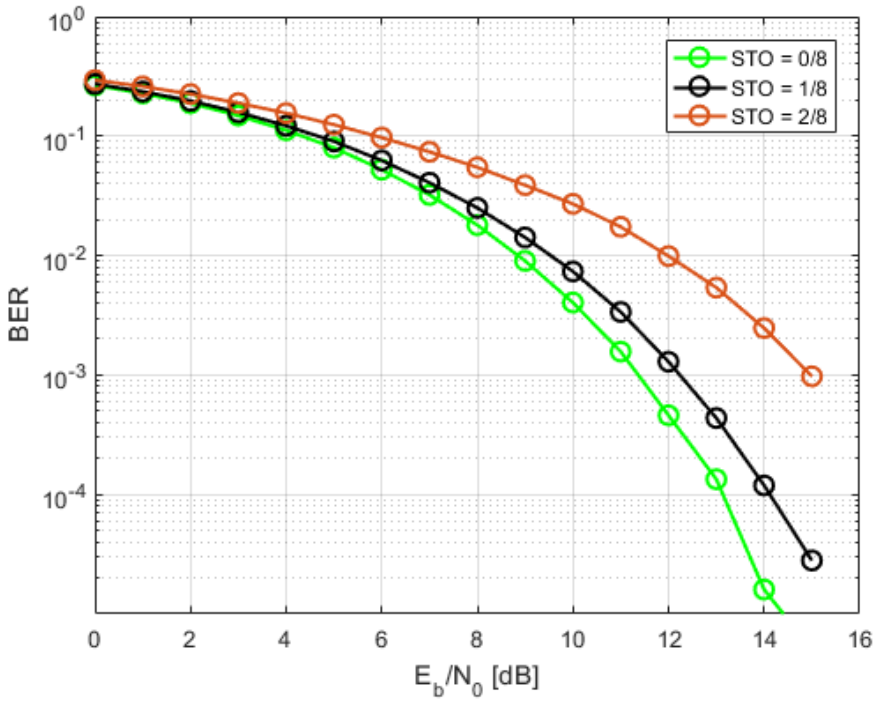


Figure 28: BER comparison between three values of symbol timing offsets: 0 (no STO), 1/8 (one sample away from correct synchronization) and 2/8 (two samples away).

As discussed in Section 1.3, a popular synchronization technique is the fully digital timing recovery loop made of an interpolator, a Gardner timing error detector, a loop filter and a numerically controlled oscillator, in a PLL-like architecture [15]. Again, this is an effective method, but it is quite costly. The goal here is to derive a much cheaper algorithm.

The idea is to make the most of the IIR filter from the preamble detection by noticing that its output presents an interesting property. Indeed, as displayed in Figure 26, the IIR filter output has one peak in each of the preamble symbol period, and each of those peaks exactly corresponds to the period index to keep while downsampling. This index is located at the end of each period, corresponding to index n_1 as derived in Equation (12), resulting from the choice of a delay $\Delta = \alpha_{OS}$, which is the one that must be used in the preamble search framework.

In short, the goal is to detect one of the local maxima from the IIR filter response of the preamble.

The first step consists in determining whether one of the eight possible peaks would be easier to detect. To do so, 100 000 simulations have been realized for each preamble period, where the goal was to detect the index of the peak. They were performed with a 13-dB SNR. The resulting distribution of the peak index is shown in Figure 29.

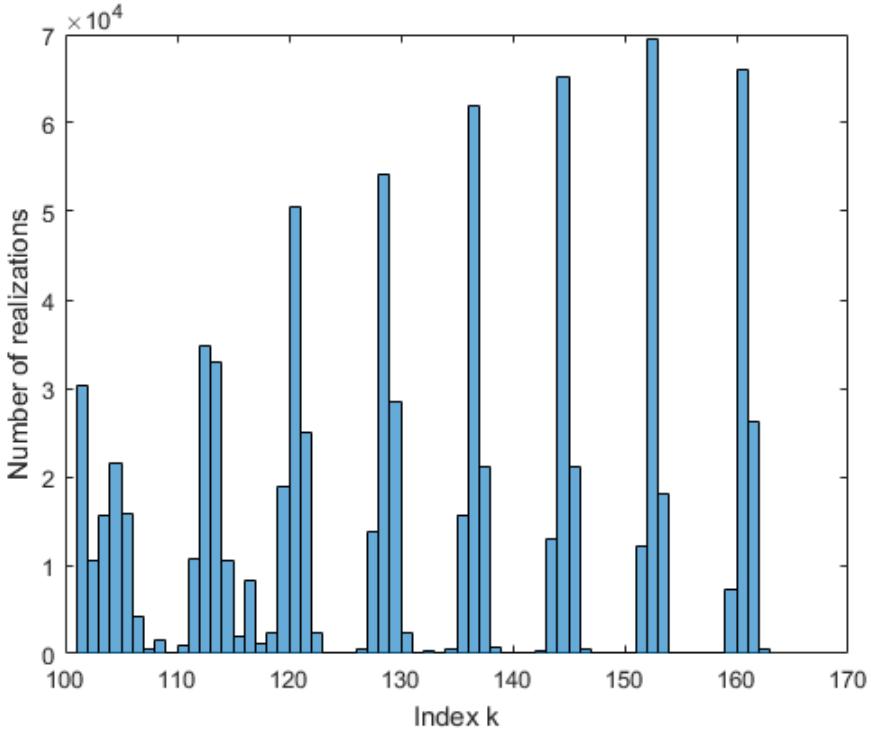


Figure 29: Detected peak index distribution.

Based on this result, the seventh period appears to be the one that maximizes the probability to find the correct period index.

The algorithm then consists in detecting the preamble, which is already done in the preamble search, and next in waiting for the seventh period to search for its local maximum. Recalling that the distribution of the detected sample index from the preamble search, presented in Figure 27, shows that most of the triggering samples are located in the first preamble period, the waiting time should correspond to six periods.

This distribution also shows that the detected local maximum will not always belong to the seventh preamble period even if it is the desired one. In order to distinguish the actual synchronizing period from the desired one, the latter is referred to as the targeted period, and its number is denoted p_t. In short, the symbol timing recovery is performed with $p_t = 7$.

It is also important to detail a little bit how the search for the local maximum is performed. It starts with setting the first considered sample (i.e. the sample located exactly six periods after the one which triggered the preamble detection) as the temporary maximum value. Then, each of the following samples is compared to this temporary

maximum and the highest of the two becomes the new temporary maximum. This goes on until 8 samples (1 period) have been tested.

Simulating this algorithm 100 000 times (with an oversampling factor of 8 and a 13-dB SNR since these are the values for which the preamble search has been designed) leads to the distribution of the detected peak index displayed in Figure 30.

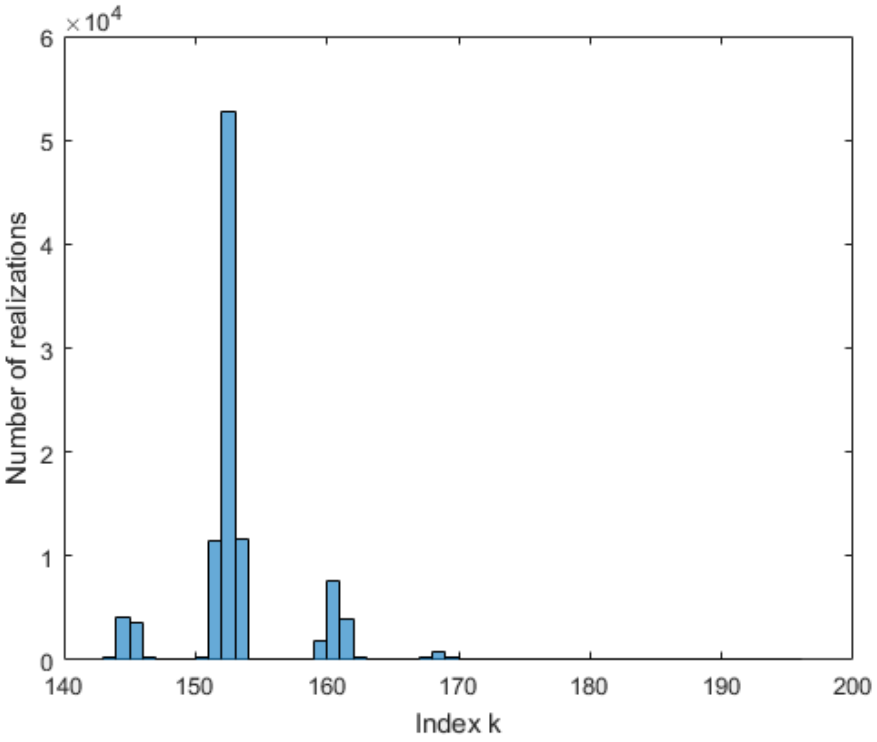


Figure 30: Detected peak index distribution - SNR = 13 dB.

Three elements must be pointed out.

Firstly, it appears that the index corresponding to the seventh period peak (index 152) has the highest probability to be detected (55%), as desired.

Secondly, indices from the previous period or from one to two periods later (or more, but with an insignificant probability) can also be detected. The fact that the detected peak might belong to the ninth period is problematic as this is just outside of the preamble, which means that the first bit of the Access Address might not be demodulated. A solution to avoid this is to start the symbol timing recovery, thus also the demodulation process, as soon as the preamble is detected, without waiting for six periods, but adapting the synchronization (which will get better and better) until six periods have passed. That way, even if the last synchronization step is performed during the first period of the Access Address, its first bit will be demodulated anyway.

Thirdly, inside each possible detected period, the correct index might not be chosen, but its direct neighbour might be instead. For example, indices 151 or 153 might be

selected instead of index 152. This can be tolerated because the BER increase due to a synchronization error of only one eighth of period is acceptable. However, a higher synchronization error makes it harder to reach a satisfactory BER. This is illustrated in Figure 28.

Going further into the details, analyzing the distribution from Figure 30 reveals that 1250 realizations out of 100 000 (1.25%) have detected the peak two (or more) indices away from the correct one, meaning that the BER target can be difficult (or impossible) to achieve in 1.25% of the time, which seems acceptable though not perfect.

One approach has been considered to try and improve this method. It consists in performing the algorithm for each preamble period, starting one period after the preamble is detected and finishing six periods after it, and averaging each period result using an increasing weight since the probability of selecting the right index increases with the considered period, as shown in Figure 29. The hope is that when a wrong index is selected for the seventh period, the selected indices from the previous periods would be correct and would allow to end up with a correct output after averaging them all.

Unfortunately, this is not what is observed from the simulation results displayed in Figure 31, where a comparison is made with respect to the original method. The detected index is presented modulo the oversampling factor ($\alpha_{OS} = 8$), which amounts to gather all the periods together in only one. Index 8 is the targeted one, and its direct neighbours, indices 1 and 7, are still acceptable, meaning that the distributions from Figure 31 show that the averaging degrades the results instead of improving them.

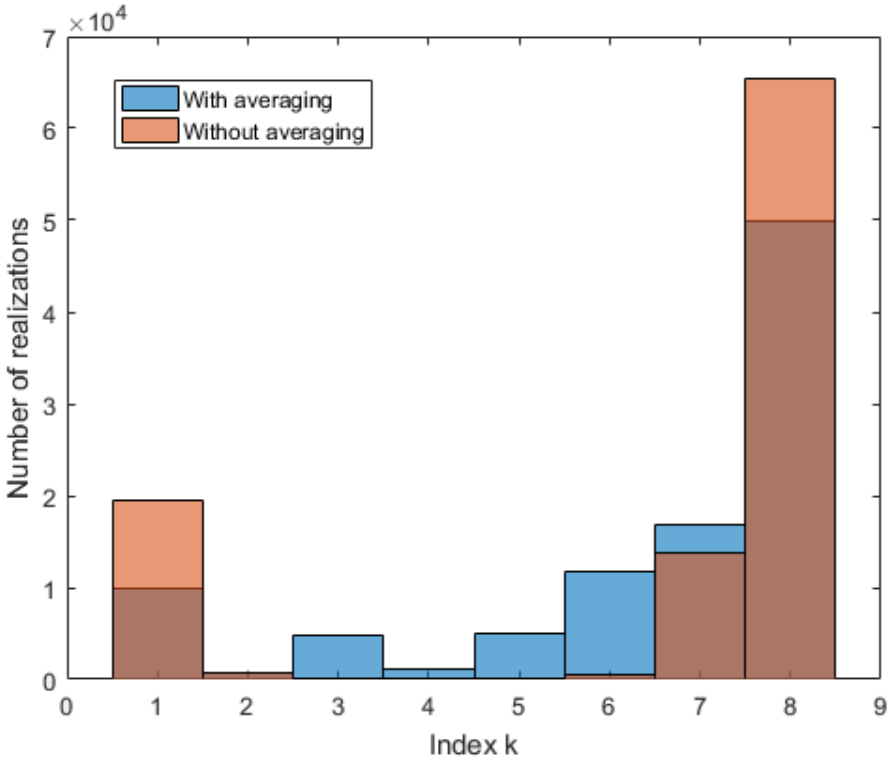
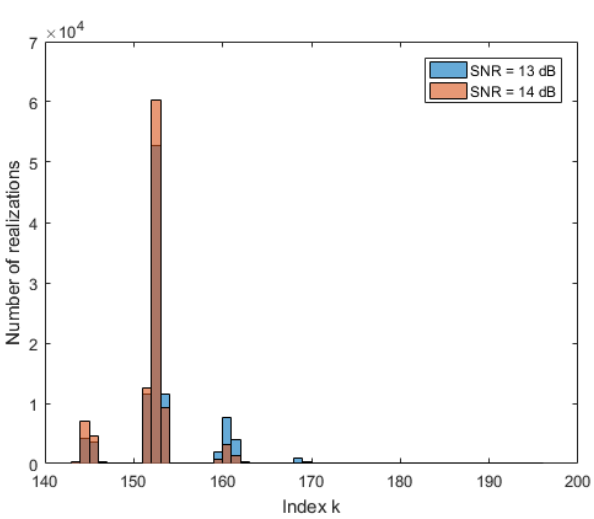
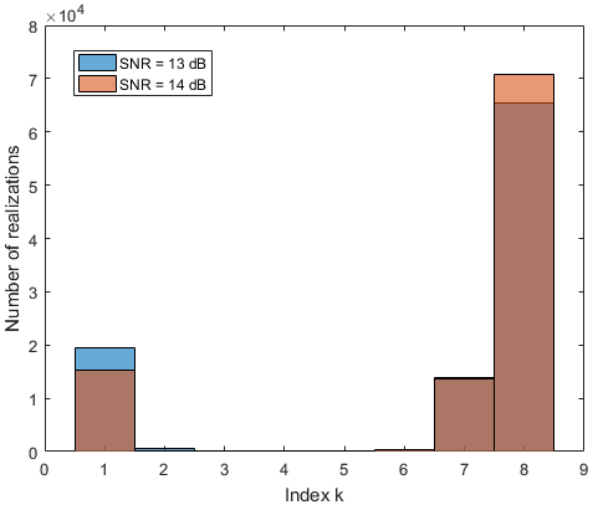


Figure 31: Comparison between the symbol timing recovery with averaging and without averaging.

Finally, here are the results in the case where it is decided to work with a higher SNR, i.e. going from 13 dB to 14 dB. This requires to redesign the preamble search block in terms of threshold value because the initial one is no longer optimal. The new empirical threshold value is 32.5. This new configuration allows to improve the symbol timing recovery performance, as shown with the detected peak index distribution in Figure 32a, and in Figure 32b with the distribution brought back into a single period. In terms of percentage of non acceptable results, it goes from 1.25 % with 13 dB to 0.5 % with 14 dB.



(a) Detected peak index distribution.



(b) Detected peak index distribution brought back into one single period.

Figure 32: Comparison between the symbol timing recovery for SNR = 13 dB and for SNR = 14 dB.

In conclusion, it might be necessary to work with a 14-dB SNR in order to stay under the targeted BER of 0.1%. But this needs to be assessed while performing the final BER evaluation, which is presented in Section 5.8.1.

5.6 Carrier Frequency Offset Correction

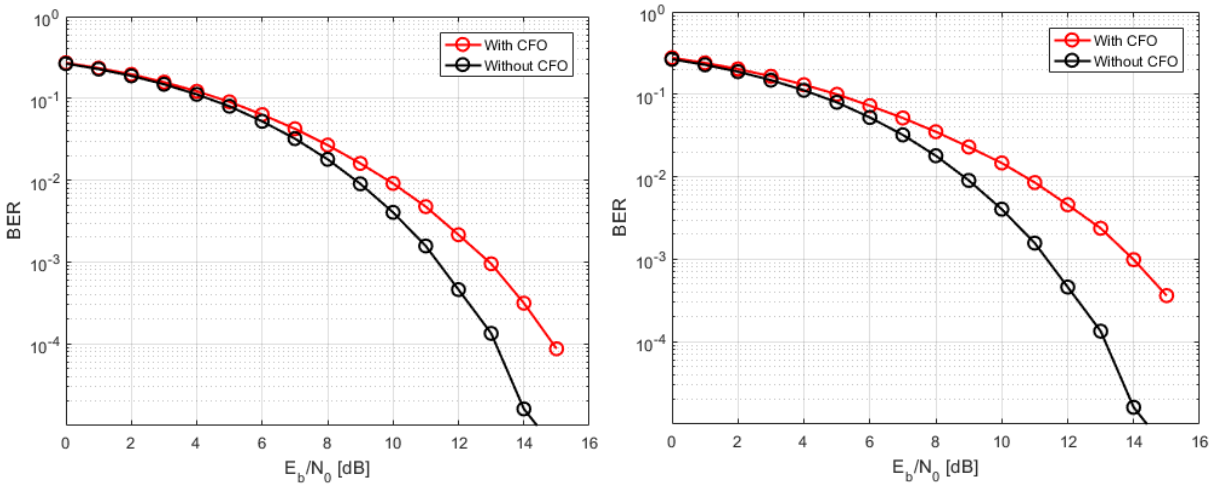
The last non-ideality to deal with is the carrier frequency offset (CFO), which comes from the fact that the local oscillators are not perfect so that the carrier frequency they produce is not exactly equal to the specified value. A typical value for an oscillator tolerance is ±20 ppm, which leads to a CFO between −40 and +40 ppm [24]. In the case of BLE, as presented in Section 3.3, the specification announces a CFO between −150 and +150 kHz (which amounts to ±60 ppm), in which a 50-kHz drift is included.

In short, the CFO f_{off} is the difference between the carrier frequencies at the transmitter $f_{\text{c,TX}}$ and at the receiver $f_{\text{c,RX}}$:

$$f_{\text{off}} = f_{\text{c,TX}} - f_{\text{c,RX}} \ .$$

The CFO needs to be compensated for, otherwise the demodulation will produce a lot of errors. This is due to the fact that, as the actual frequency is higher/lower, the phase of the signal will vary faster/slower. This means that, as time goes by, the difference between the ideal phase and the actual one will increase, leading inevitably to errors during demodulation.

To determine what is the maximum allowable CFO for the BER to remain below 0.1 %, BER simulations have been performed for different values of CFO, and considering perfect synchronization. It has been found that this maximum allowable CFO is equal to 17 ppm for SNR = 13 dB, and 23 ppm for SNR = 14 dB, as shown in Figure 33.



(a) CFO = 17 ppm is critical for SNR = 13 dB. (b) CFO = 23 ppm is critical for SNR = 14 dB.

Figure 33: BER evaluation with critical CFOs.

5.6.1 CFO Mitigation Algorithm Derivation

In order to compensate for the CFO, the first step is to estimate it. This is the hardest part. Once the estimation is done, the only thing to do is to tune accordingly the frequency of the mixer used for the downconversion.

As discussed in Section 1.3, among the existing low-cost CFO mitigation techniques, one possibility is the decision-aided maximum likelihood method presented in [16].

Starting from this technique, it is possible to rather go for a data-aided approach based on the preamble since it is known in advance. More precisely, the idea is to perform a signal approximation so as to be able to isolate the CFO contribution, and thus to obtain an expression for the CFO estimate.

First, here is how the CFO f_{off} can be explicitly expressed inside the received signal:

$$r_{bb}\left[\frac{n}{\alpha_{OS}}\right] = \cos\left(\phi\left[\frac{n}{\alpha_{OS}}\right] + 2\pi f_{\text{off}}\frac{nT_b}{\alpha_{OS}}\right) + j\,\sin\left(\phi\left[\frac{n}{\alpha_{OS}}\right] + 2\pi f_{\text{off}}\frac{nT_b}{\alpha_{OS}}\right) + w\left[\frac{n}{\alpha_{OS}}\right], \quad (14)$$

where it is assumed that the received signal is normalized and where $w\left[\frac{n}{\alpha_{OS}}\right]$ is the AWGN contribution. As the noise is a stochastic variable, from now on, it will be approximated by its mean, which is zero.

However, the signal to be approximated is not directly the received signal in baseband from Equation (14), but the differential one resulting from the Hermitian product. The reason behind this is that the desired algorithm is data-aided, but the instantaneous phase is unknown, even in the preamble. However, the phase difference is known, hence the need to resort to the Hermitian product to get a differential signal, which, based on Equations (9) and (14), is obtained as

$$\begin{aligned}
H_{p,nf}\left[\frac{n}{\alpha_{OS}}\right] &= \left(\cos\left(\phi\Big[\frac{n}{\alpha_{OS}}\Big] + 2\pi f_{\text{off}}\frac{nT_b}{\alpha_{OS}}\right) + j\,\sin\left(\phi\Big[\frac{n}{\alpha_{OS}}\Big] + 2\pi f_{\text{off}}\frac{nT_b}{\alpha_{OS}}\right)\right) \\
&\times \left(\cos\left(\phi\Big[\frac{(n-\Delta)}{\alpha_{OS}}\Big] + 2\pi f_{\text{off}}\frac{(n-\Delta)T_b}{\alpha_{OS}}\right) + j\,\sin\left(\phi\Big[\frac{(n-\Delta)}{\alpha_{OS}}\Big] + 2\pi f_{\text{off}}\frac{(n-\Delta)T_b}{\alpha_{OS}}\right)\right) \\
&= \cos\left(\left(\phi\left[\frac{n}{\alpha_{OS}}\right] + 2\pi f_{\text{off}}\frac{nT_b}{\alpha_{OS}}\right) - \left(\phi\left[\frac{n-\Delta}{\alpha_{OS}}\right] + 2\pi f_{\text{off}}\frac{(n-\Delta)T_b}{\alpha_{OS}}\right)\right) \\
&+ j\,\sin\left(\left(\phi\left[\frac{n}{\alpha_{OS}}\right] + 2\pi f_{\text{off}}\frac{nT_b}{\alpha_{OS}}\right) - \left(\phi\left[\frac{n-\Delta}{\alpha_{OS}}\right] + 2\pi f_{\text{off}}\frac{(n-\Delta)T_b}{\alpha_{OS}}\right)\right) \\
&= \cos\left(\delta\phi\left[\frac{n}{\alpha_{OS}}\right] + 2\pi f_{\text{off}}\frac{\Delta T_b}{\alpha_{OS}}\right) + j\,\sin\left(\delta\phi\left[\frac{n}{\alpha_{OS}}\right] + 2\pi f_{\text{off}}\frac{\Delta T_b}{\alpha_{OS}}\right).
\end{aligned}$$

As this is the noise-free version of the Hermitian product, the actual equation is an approximation of it. Hence, it is given as

$$H_p\left[\frac{n}{\alpha_{OS}}\right] \simeq \cos\left(\delta\phi\left[\frac{n}{\alpha_{OS}}\right] + 2\pi f_{\text{off}}\frac{\Delta T_b}{\alpha_{OS}}\right) + j\,\sin\left(\delta\phi\left[\frac{n}{\alpha_{OS}}\right] + 2\pi f_{\text{off}}\frac{\Delta T_b}{\alpha_{OS}}\right).$$

Then, extracting the imaginary part gives I_H as

$$I_H\left[\frac{n}{\alpha_{OS}}\right] \simeq \sin\left(\delta\phi\left[\frac{n}{\alpha_{OS}}\right] + 2\pi f_{\text{off}}\frac{\Delta T_b}{\alpha_{OS}}\right).$$

It is now possible to extract f_{off}, but this requires the use of the inverse sine. With the aim of limiting as much as possible the algorithm to basic operations, it is chosen to approximate the sine by its Taylor series. Stopping at the second order of a sine function $\sin(x)$ developed around x_0 yields

$$\sin(x) \simeq \sin(x_0) + (x - x_0)\,\cos(x_0) - \frac{(x-x_0)^2}{2}\,\sin(x_0).$$

In this case, the development is realized around $x_0 = \delta\phi\Big[\frac{n}{\alpha_{OS}}\Big]$, which gives

$$I_H\left[\frac{n}{\alpha_{OS}}\right] \simeq \sin\left(\delta\phi\left[\frac{n}{\alpha_{OS}}\right]\right) + 2\pi f_{\text{off}}\frac{\Delta T_b}{\alpha_{OS}}\,\cos\left(\delta\phi\left[\frac{n}{\alpha_{OS}}\right]\right) - \frac{(2\pi f_{\text{off}}\frac{\Delta T_b}{\alpha_{OS}})^2}{2}\sin\left(\delta\phi\left[\frac{n}{\alpha_{OS}}\right]\right).$$

Two possible values for the CFO estimate $\hat{f_{\text{off}}}$ are then obtained as the roots of this quadratic equation:

$$\hat{f}_{\text{off,n}}\left[\frac{n}{\alpha_{OS}}\right] = \frac{\cos\left(\delta\phi\left[\frac{n}{\alpha_{OS}}\right]\right) \pm \sqrt{\sin^2\left(\delta\phi\left[\frac{n}{\alpha_{OS}}\right]\right) - 2I_H\left[\frac{n}{\alpha_{OS}}\right]\sin\left(\delta\phi\left[\frac{n}{\alpha_{OS}}\right]\right) + 1}}{2\pi\frac{\Delta T_b}{\alpha_{OS}}\sin\left(\delta\phi\left[\frac{n}{\alpha_{OS}}\right]\right)}.$$

By simulation, it turns out that only one root is valid. Here is the resulting CFO estimate:

$$\hat{f}_{\text{off,n}}\left[\frac{n}{\alpha_{OS}}\right] = \frac{\cos\left(\delta\phi\left[\frac{n}{\alpha_{OS}}\right]\right) - \sqrt{\sin^2\left(\delta\phi\left[\frac{n}{\alpha_{OS}}\right]\right) - 2I_H\left[\frac{n}{\alpha_{OS}}\right]\sin\left(\delta\phi\left[\frac{n}{\alpha_{OS}}\right]\right) + 1}}{2\pi\Delta T_b\sin\left(\delta\phi\left[\frac{n}{\alpha_{OS}}\right]\right)}, \tag{15}$$

which has been slightly simplified by setting $\Delta = \alpha_{OS}$. The reason behind this choice is the same as in the case of the preamble detection. It is relevant to maximize the output range to fight against the noise, whereas fighting against ISI is irrelevant here. Moreover, it allows to perform only one Hermitian product whose output can be used for multiple purposes.

Equation (15) shows a way to estimate the CFO. However, it requires the use of the phase difference value $\delta\phi\left[\frac{n}{\alpha_{OS}}\right]$, which is not known at any time. Nevertheless, the additional approximations necessary to estimate the phase difference can be avoided by evaluating the CFO only during the preamble. The advantage is that the preamble is known, thus so is the phase difference of the corresponding signal. Furthermore, correcting for the CFO only during the preamble allows to save energy during the remaining part of the packet. On the other hand, this also means that the CFO is estimated once and for all. Therefore, the applied CFO correction will be the same for the whole packet, even if the frequency drifts during the packet transmission, which BLE specifies as going up to 50 kHz (20 ppm), as presented in Section 3.3. But in a first approach, this possible drift is neglected.

Thanks to the knowledge of the preamble, Equation (14) allows to estimate directly the CFO from the imaginary part of the Hermitian product $I_H\left[\frac{n}{\alpha_{OS}}\right]$. However, the computation is not straightforward as it involves a square root and, more importantly, a division by the sine of the phase difference. This is problematic as, once per period, this phase difference is zero, leading to a division by zero. Therefore, one sample per period should be removed from the computation, which complicates the algorithm.

Based on this analysis, it seems relevant to consider using only the first-order development of the sine as it might lead to an algorithm with a better trade-off between simplicity and accuracy. The approximation of the I_H signal becomes

$$I_H\left[\frac{n}{\alpha_{OS}}\right] \simeq \sin\left(\delta\phi\left[\frac{n}{\alpha_{OS}}\right]\right) + 2\pi f_{\text{off}} T_b \cos\left(\delta\phi\left[\frac{n}{\alpha_{OS}}\right]\right) ,$$

which produces the following CFO estimate:

$$\hat{f}_{\text{off,n}}\left[\frac{n}{\alpha_{OS}}\right] = \frac{I_H\left[\frac{n}{\alpha_{OS}}\right] - \sin\left(\delta\phi\left[\frac{n}{\alpha_{OS}}\right]\right)}{2\pi T_b \cos\left(\delta\phi\left[\frac{n}{\alpha_{OS}}\right]\right)} . \tag{16}$$

The algorithm then consists in computing the estimate for each sample of the preamble and in outputting the final estimate as their mean, which gives

$$\hat{f}_{\text{off}} = \frac{1}{8\ \alpha_{OS}} \sum_{n=1}^{8\ \alpha_{OS}} \hat{f}_{\text{off,n}}\left[\frac{n}{\alpha_{OS}}\right] . \tag{17}$$

However, in practice, preamble detection must have been performed beforehand because CFO estimation cannot start before the preamble has been found. Therefore, recalling the distribution of the detected sample from Figure 27, it means that the 8 periods of the preamble cannot be used, but only 6 can considering that the preamble might be detected during its second period.

Actually, two elements change a bit this analysis. Firstly, CFO estimation can also use the first period after the end of the preamble because the alternating pattern always goes on with at least the first bit of the Access Address, meaning that, from a CFO estimation point of view, the preamble can be seen as 9-period long. Secondly, it should be avoided to use the first preamble period because the corresponding I_H signal is there a bit distorted since it has been computed using the end of the noisy signal preceding the packet.

Thus, it is decided that the algorithm starts one period after the preamble has been detected, and ends seven periods later.

Another very important point is that the CFO estimation algorithm does not necessarily need symbol timing recovery to have been done beforehand. This seems odd at first since the computation of $\hat{f}_{\text{off,n}}$ in Equation (16) involves the computed signal I_H and values that can be precomputed and tabled (because the phase difference is known during the preamble), meaning that a synchronization should be necessary in order to fetch the correct values from the table. But, as the final estimation is obtained from the average of $\hat{f}_{\text{off,n}}$, if this average is performed over an even number of periods, then the synchronization with the tabled values is not required. This is due to the property that the average of the difference of two vectors is equal to the difference of the averages of those two vectors, and to the fact that the signal I_H and the sine/cosine of the phase difference during the preamble are both periodic with a periodicity of 2 times the preamble period, meaning that the average of $\hat{f}_{\text{off,n}}$ will be the same even without symbol timing recovery.

This is a very interesting point, because it allows the symbol timing recovery algorithm and the CFO estimation algorithm to be independent. Otherwise, if CFO estimation needed synchronization, the system would be much less effective as a trade-off would be necessary between the number of preamble periods to be used on one side for the symbol timing recovery, and on the other side for the CFO estimation. Indeed, a better synchronization is achieved when working with the end of the preamble, but the best CFO estimation needs as much periods as possible, meaning that the synchronization would have to be done at the start of the preamble. In that case, targeting the m^{th} period for the symbol timing recovery, i.e. $p_t = m$, the number of remaining periods for the CFO estimation would be equal to $8 - m$.

But thanks to this synchronization-insensitive CFO estimation algorithm, the optimal number of periods can be used in both algorithms, i.e. $p_t = 7$ for the symbol timing recovery, and 6 periods for the CFO estimation.

Finally, Equation (17) thus becomes

$$\hat{f}_{\text{off}} = \frac{1}{6\ \alpha_{OS}} \sum_{n=1}^{6\ \alpha_{OS}} \hat{f}_{\text{off,n}}\left[\frac{n}{\alpha_{OS}}\right] . \tag{18}$$

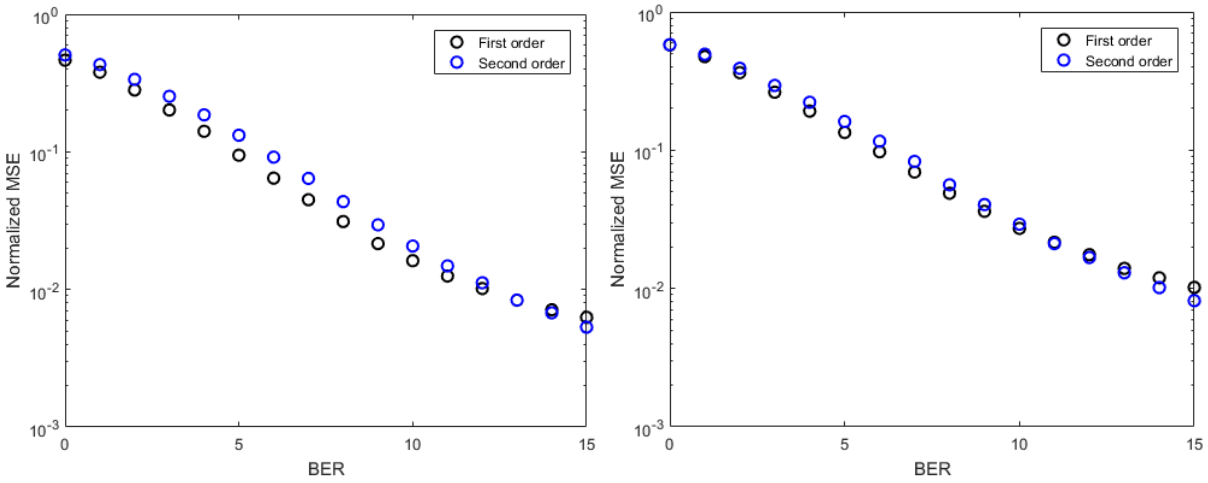
5.6.2 CFO Mitigation Algorithm Performance

A first comparison between the two CFO estimation methods (first-order versus second-order approximation) is performed in terms of normalized mean squared error (MSE), defined as

$$\text{MSE}_n = \frac{1}{N} \sum_{i=1}^{N} \left(\frac{f_{\text{off}} - \hat{f_{\text{off}}}}{f_{\text{off}}} \right)^2 ,$$

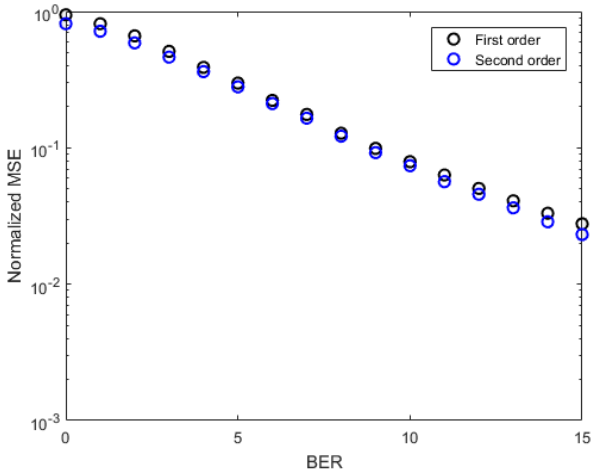
where N is the number of different simulations that are performed.

Figure 34 presents this MSE comparison with $N = 10\ 000$ for different SNR values, which required to consider a perfect preamble detection (the triggering sample is fixed to the first preamble sample) in order to be able to perform the simulation at low SNR. There is also a comparison between three different values of number of periods that are used. The simulation is realized with a CFO of 30 ppm.



(a) Averaging over 6 periods.

(b) Averaging over 4 periods.



(c) Averaging over 2 periods.

Figure 34: Comparison between CFO estimation using first-order and second-order method for a CFO values of 30 ppm.

Two important observations can be done from this simulation.

Firstly, it seems that using the first-order approximation actually gives more accurate estimates, unless only 2 periods or high SNRs are used. In any case, the best trade-off between simplicity and accuracy is clearly achieved with the first-order approximation. Therefore, from now on, the first-order method will always be used.

Secondly, the higher the number of periods that are used, the more accurate the estimates. However, going from 6 to 4 periods does not dramatically change the results, but going down to 2 periods seems to degrade a lot the algorithm performance. This confirms the intuition of choosing 6 as the number of periods and thereby the validity of Equation (18).

Another interesting comparison of MSE can be made between two values of CFO: a medium one of 30 ppm and the extreme one of 60 ppm. Figure 35 presents this

comparison.

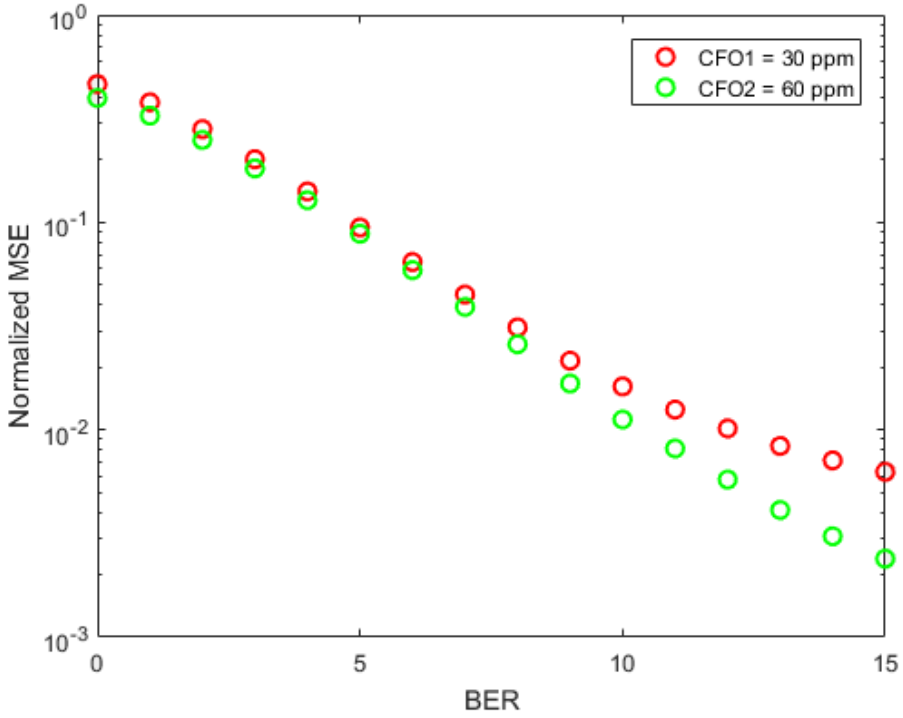


Figure 35: MSE comparison between CFO estimation for CFO values of 30 and 60 ppm.

Interestingly, the estimate accuracy appears to get better when the CFO gets higher, which is quite fortunate. As far as the SNRs of interest are concerned, the normalized MSEs are all below 1 % with 0.84 % and 0.71% for 30 ppm and SNR values of 13 dB and 14 dB respectively, and 0.41 % and 0.31 % for 60 ppm. This means that, after correction, the remaining CFO is in average below 4 ppm in any case, which is well below the maximum allowable CFOs of 17 ppm (SNR = 13 dB) and 23 ppm (SNR = 14 dB).

The algorithm performance in terms of MSE brings interesting information, but it does not give any idea about the estimate spread, which is as important as the mean. Therefore, a histogram has been realized and is displayed in Figure 36. It is based on $N =$ 100 000 simulations and a 13-dB SNR. Furthermore, it evaluates the CFO estimation in the cases where the CFO is equal to 0, 15 and 45 ppm, in addition to 30 and 60 ppm.

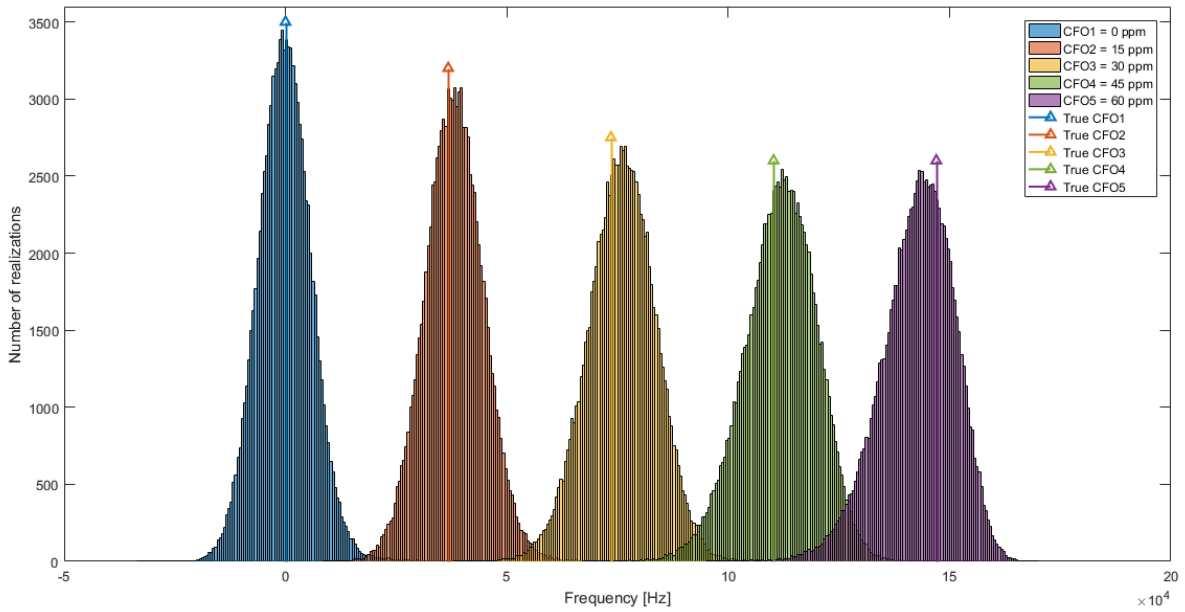


Figure 36: Comparison between CFO estimations for different CFO values.

Figure 36 shows that the spread of the CFO estimate increases when the CFO increases itself. However, when normalized with respect to its corresponding CFO value, the standard deviation is actually decreasing. This is shown in Table 3. In any case, the accuracy of the estimation is satisfactory.

CFO [ppm]	**Standard Deviation [kHz]**	**Normalized Standard Deviation [/]**
0	6.20	/
15	6.80	18.5 %
30	7.75	10.5 %
45	8.20	7.5 %
60	8.25	5.5 %

Table 3: Standard deviation of the CFO estimations.

Finally, as working with a 14-dB SNR might be preferable, it is interesting to see what can be expected in that configuration. The same simulation has been performed as in the 13-dB case, and the resulting estimates distributions are presented in Figure 37.

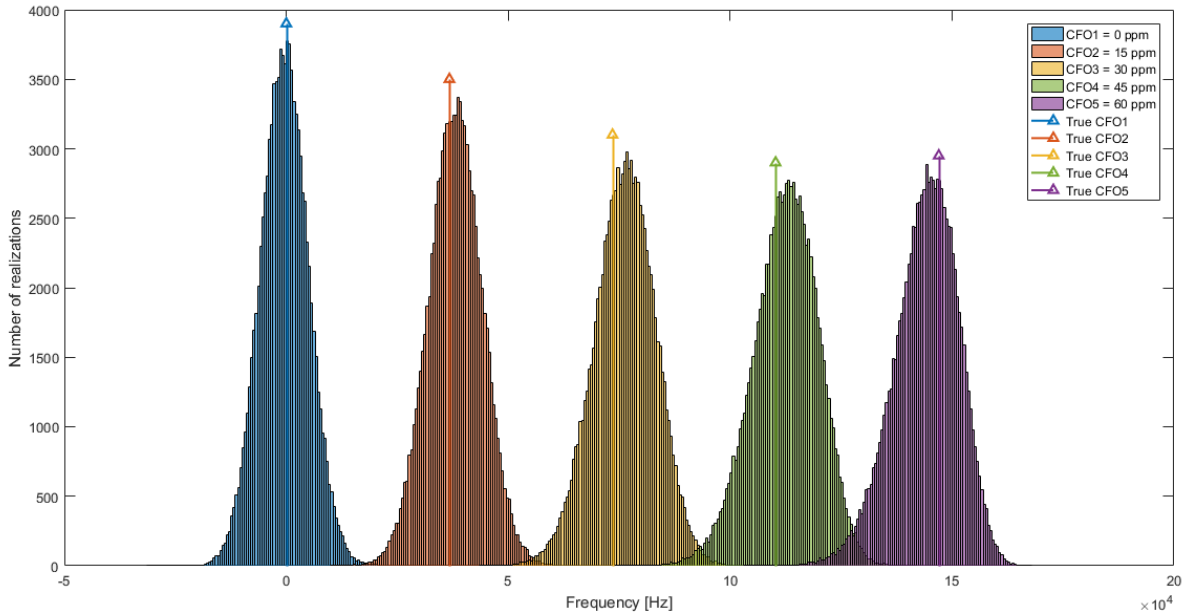


Figure 37: Comparison between CFO estimations for different CFO values for SNR = 14 dB.

As expected, the standard deviations are lower when working with a higher SNR. Table 4 summarizes their values.

CFO [ppm]	**Standard Deviation [kHz]**	**Normalized Standard Deviation [/]**
0	5.40	/
15	6.10	16.5 %
30	7.95	9.5 %
45	7.35	6.5 %
60	7.20	5.0 %

Table 4: Standard deviation of the CFO estimations for SNR = 14 dB.

In conclusion, the very low-cost CFO estimation algorithm which has been derived is efficient, with a normalized MSE below 1 % and a normalized standard deviation below 20 % (for a 13-dB SNR). Moreover, it has the very interesting property of being synchronization-insensitive.

5.7 Additional non-idealities

In this section, additional non-idealities such as modulation index offset and sampling timing offset are briefly discussed.

5.7.1 Modulation Index Offset

Modulation index offset (MIO) can occur because there is a 10-% tolerance at the transmitter, which means that the effective modulation index lies somewhere in the range [0.45 ; 0.55]. But this turns out not to be problematic as the modulation index does not need to be known at the receiver. The only effect is a slight variation in the amplitude of the phase, as demonstrated by Equation (3). In practice, a lower/higher modulation index produces more/less ISI (yielding a poorer/better BER) and shrinks/widens the bandwidth, as discussed in Section 4.1.2.2. But again, those variations are quite small, and knowing the actual value at the receiver does not matter.

5.7.2 Sampling Timing Offset

Sampling timing offset is close to symbol timing offset. It denotes the fact that the sampling is performed in an asynchronous way, as represented graphically in Figure 38, where the truncated sinc function curve is taken as an example.

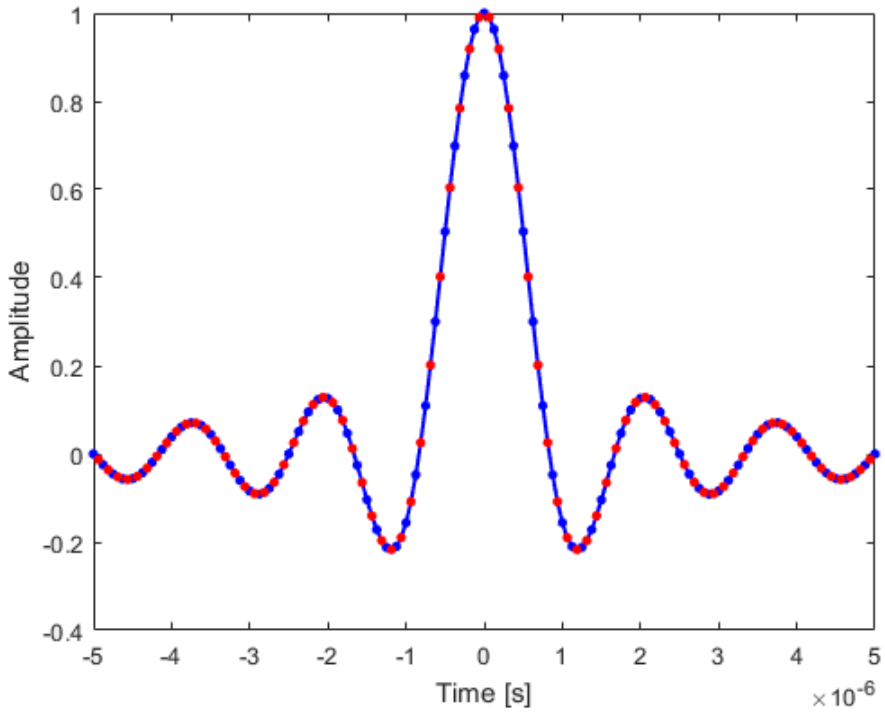


Figure 38: Sampling timing offset graphical representation.
The blue dots correspond to the expected samples, whereas the red ones correspond to the worst case sampling offset.

This adds uncertainty to the symbol timing recovery. The worst case corresponds to sampling exactly halfway in between the expected samples, i.e. with a clock having a phase shift of 180° with respect to the ideal one. In the case the symbol timing recovery would badly synchronize the digital processing blocks with one sample error, this worst case sampling offset would amount to get up to one and a half sample error. On the other hand, by luck, it could also reduce it down to half a sample error.

The relevant question about sampling timing offset is to find out whether it changes the tolerance on the symbol timing recovery performance, which was set to one sample for $\alpha_{OS} = 8$. This amounts to analyzing the case where there would be one and a half sample of synchronization error since sampling timing offset can add up to half a sample error. The BER has been evaluated in this configuration and is displayed in Figure 39. It can be observed, quite logically, that the resulting BER lies just in between the cases where the synchronization error is of 1 and 2 samples. Going further into the details, this worst case sampling offset produces a BER just over the target of 0.1 % for a 13-dB SNR, meaning that working with 14 dB might be wiser, especially since this BER simulation considers no CFO.

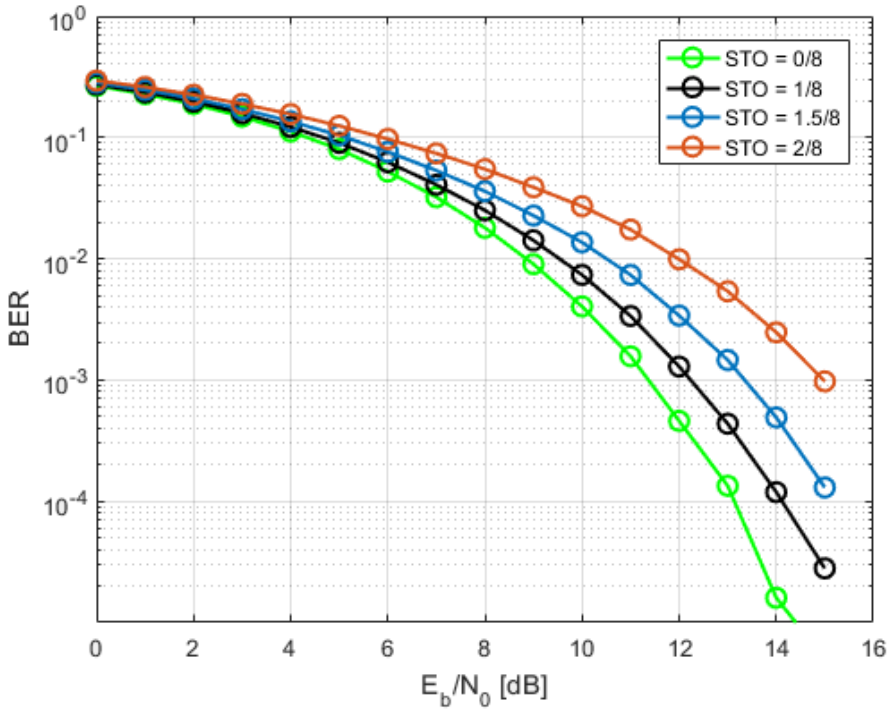


Figure 39: BER comparison between four values of symbol timing offsets: 0 (no STO), 1/8, 1.5/8 and 2/8.

In conclusion, sampling timing offset does not really change the tolerance on the symbol timing recovery performance, although the uncertainty that it adds might favour the choice of working with a 14-dB SNR rather than a 13-dB one.

5.8 Overall System

Four digital processing blocks have been derived in order to allow a proper reception of the data. These blocks are the preamble search, the symbol timing recovery, the carrier frequency offset estimation and the demodulation. When putting everything together, some trade-offs need to be made.

The first and the most important one concerns the number of periods to be used for symbol timing recovery and for CFO estimation. Fortunately, an effective solution has been found to avoid it, as described in Section 5.6.

Secondly, all the blocks have as input the Hermitian product performed on the received signal in baseband. However, the optimal delay involved in this product is not the same for all the blocks. Preamble search, symbol timing recovery and CFO estimation all require a delay $\Delta = \alpha_{OS}$, whereas the demodulation performs better at the working SNRs with half this delay. Therefore, the two solutions are either realizing two different

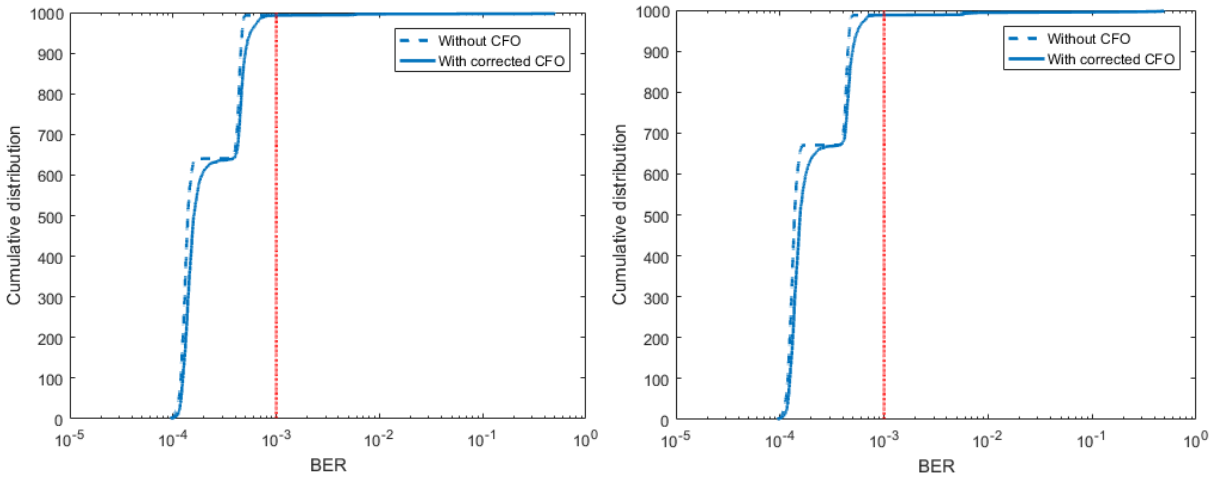
Hermitian products, either realizing only one with a delay equal to the oversampling factor, which reduces the BER performance, as shown in Figure 24.

In order to maximize the BER performance, it has been chosen to implement two different Hermitian products.

5.8.1 System Performance

In order to assess the global system performance, simulations have been realized to evaluate the BER. As the symbol timing recovery is a critical block that can produce very different results, it is necessary to perform more than one simulation and to present statistical results. Therefore, 1000 simulations have been made, and their results are presented as a cumulative distribution. The analysis then consists in observing the number of simulations which produce a BER lower than 0.1 %.

The first set of simulations has been performed for a 13-dB SNR. The resulting cumulative distribution is displayed in Figure 40. Moreover, two degrees of comparison are presented. The first one compares the BER with corrected CFO with the BER without CFO. The second one compares through the BER the symbol timing recovery performances when targeting the 6^{th} or the 7^{th} preamble period.



(a) $p_t = 6$. (b) $p_t = 7$.

Figure 40: Cumulative distribution of the BER with 1000 realizations and a 13-dB SNR.

Figure 40 exhibits two interesting elements. First, the CFO correction is very efficient as its BER cumulative distribution is very close to the CFO-free one. Second, targeting the 6^{th} period for the symbol timing recovery shows slightly better performance when reaching the targeted BER than targeting the 7^{th} one. This difference in performance between $p_t = 6$ and $p_t = 7$ is very small with a cumulative distribution of 993/1000 versus 988/1000. Still, this shows that it is actually preferable to work with $p_t = 6$ rather

than $p_t = 7$. This goes against the results from Section 5.5, which are yet confirmed in Figure 41, where the symbol timing recovery performances with those two p_t values are compared and give the advantage to $p_t = 7$ with 98.75 % of acceptable synchronization versus 98.58 % for $p_t = 6$.

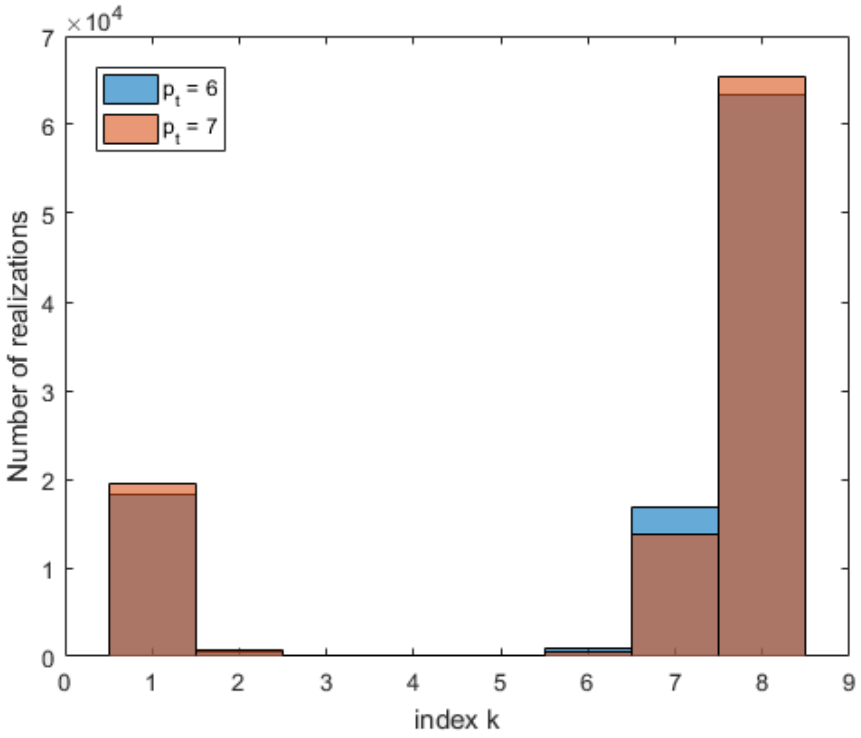
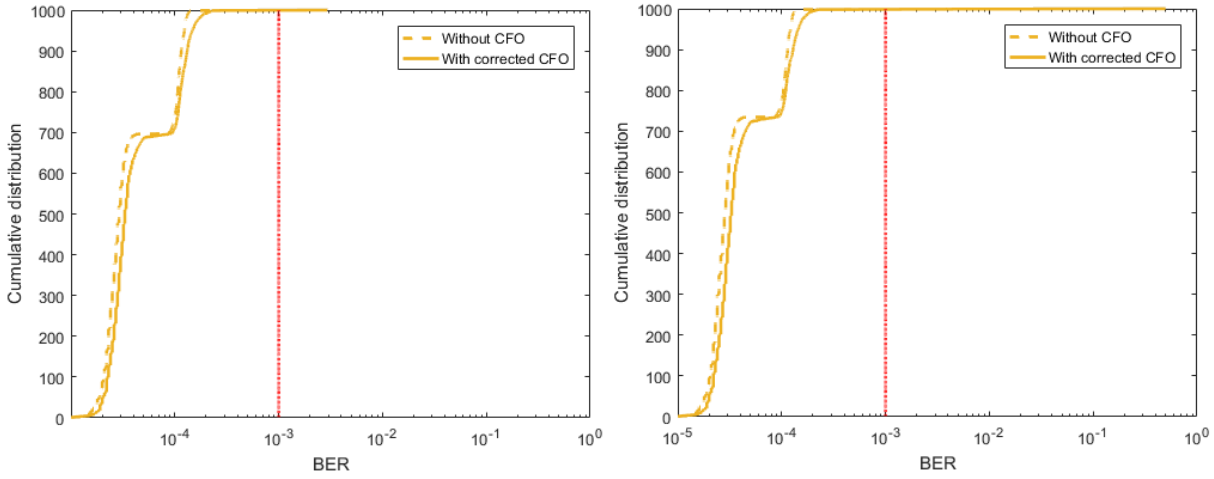


Figure 41: Comparison between the symbol timing recovery with $p_t = 6$ and with $p_t = 7$.

Thanks to this analysis of BER cumulative distribution, it comes that, with a 13-dB SNR, the global system produces a BER below 0.1 % with a probability of 99.3 %.

In the case where this probability would not be high enough, it can be considered to work with a 14-dB SNR. The corresponding BER evaluation is presented in Figure 42, where the same comparisons as in the 13-dB case are realized.



(a) $p_t = 6$. (b) $p_t = 7$.

Figure 42: Cumulative distribution of the BER with 1000 realizations and an SNR of 14 dB.

Figure 42 shows that once again targeting the 6^{th} period for the symbol timing recovery is preferable. This time, the number of satisfactory BERs is 999/1000 for $p_t = 6$ versus 998/1000 for $p_t = 7$. Again, this is unexpected given the symbol timing recovery performance for those two p_t values, which is presented in Figure 43. Indeed, the case $p_t = 6$ yield 99.32 % of acceptable synchronization versus 99.53 % for $p_t = 7$.

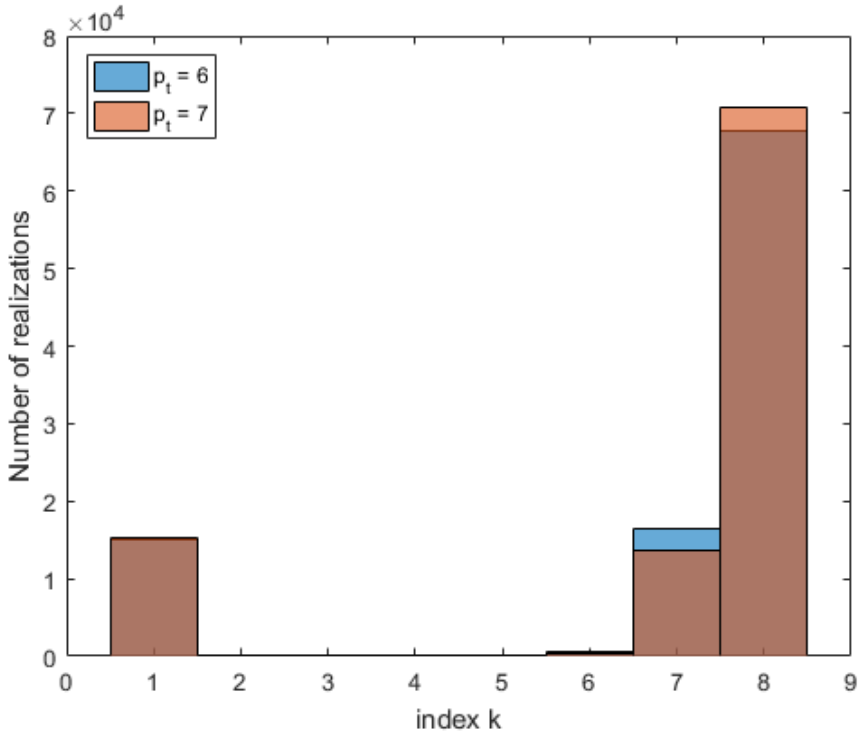


Figure 43: Comparison between the symbol timing recovery with $p_t = 6$ and with $p_t = 7$ for an SNR of 14 dB.

To summarize, Table 5 presents the global system configuration and its performances for SNR values of 13 dB and 14 dB.

	SNR = 13 dB	**SNR = 14 dB**
Preamble detection threshold	37.5	32.5
Targeted preamble period for symbol timing recovery	6^{th} period	
Probability of acceptable synchronization	98.58 %	99.32 %
Number of preamble periods used for CFO estimation	6	
Normalized MSE of the CFO estimation (for CFO = 30 ppm)	0.84 %	0.71%
Normalized standard deviation of the CFO estimation (for CFO = 30 ppm)	10.5 %	9.5 %
Probability of acceptable BER	99.3 %	99.9 %

Table 5: System parameters and performances.

It is important to discuss the difference between the probability of acceptable synchronization and the probability of acceptable BER. In Table 5, the second one appears to be better than the first one, which does not make sense since it should be the contrary. Indeed, in the case where there would be no CFO, those two probabilities should be equal because the synchronization is defined as acceptable when it leads to an acceptable BER, as discussed in Section 5.5 where no CFO is considered. But, given that there is some CFO, even in the configuration of an acceptable synchronization, the resulting BER might be above 0.1 % if the CFO is not estimated well enough.

The probability of acceptable synchronization results from the symbol timing recovery and corresponds to the probability of performing the synchronization with the correct sample or with one sample error since this only slightly increases the resulting BER. The values from Table 5 for this probability come from a simulation with 100 000 repetitions.

On the other hand, the probability of acceptable BER comes from a simulation with only 1000 repetitions. This lower number of repetitions is probably the reason behind the fact that the probability of acceptable BER is higher than the probability of acceptable synchronization.

Therefore, in order to get the best insight about the system performance, it is probably best to look at the probability of acceptable synchronization, especially as the CFO is very well corrected.

In any case, this proves that the CFO correction performs very well, which also justifies the choice of looking at the probability of acceptable synchronization.

In conclusion, the system presents a probability of around 99 % to stay below the maximum allowed BER of 0.1 % when working with a 13-dB SNR, and a probability of around 99.5 % when working with a 14-dB SNR.

5.9 Summary and Theoretical Computational Complexity

This section presents a summary of the digital operations which need to be performed at the receiver, together with their block diagram representation. In those block diagrams, a single arrow indicates a real signal, whereas a double arrow indicates a complex signal.

5.9.1 Digital Low-Pass Filter

The digital LPF is the first digital block and is applied to the unfiltered oversampled received signal in baseband $r_{\text{bb,unf}}$ obtained directly after the downconversion and the ADC.

Its block diagram representation is straightforward as it is simply a convolution with the LPF impulse response. It is presented in Figure 44.

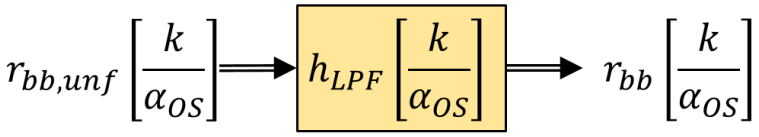


Figure 44: Digital low-pass filter block diagram.

5.9.2 Hermitian Product

As discussed in Section 5.8, two versions of Hermitian products are implemented.

The first one, denoted H_{p1}, is the input for the preamble search, the symbol timing recovery and the CFO estimation. It is computed with a delay equal to the oversampling factor α_{OS}.

The second one, denoted H_{p2}, is the input for the demodulation and is obtained with a delay equal to half the oversampling factor.

They can be represented as in the block diagram displayed in Figure 45, in which the signal *Demod* controls the multiplexer to compute the first Hermitian product or the second one. This control signal is set to 1 to enable the demodulation when 6 periods have passed since the preamble has been found.

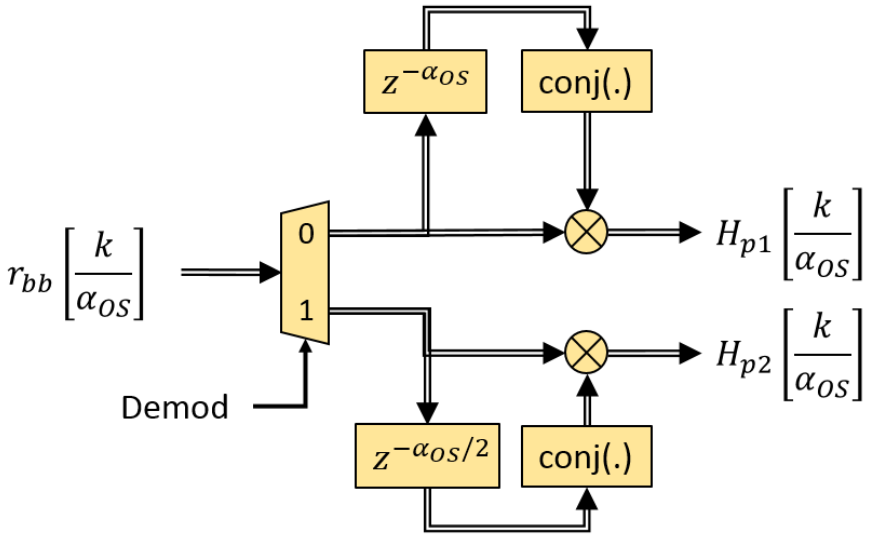


Figure 45: Hermitian products block diagram.

5.9.3 Preamble Search

The first algorithm to come into play is the preamble search. It looks for the first sample coming out the IIR filter whose complex magnitude *IIRoutMagn* crosses a predefined threshold *Thresh*.

The corresponding block diagram is shown in Figure 46. The output *PFnd* is a control signal informing whether the preamble has been found.

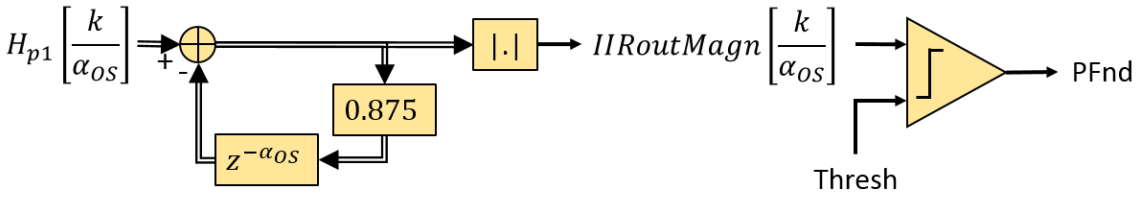


Figure 46: Preamble search block diagram.

5.9.4 Symbol Timing Recovery

The second algorithm is the symbol timing recovery. As $p_t = 6$, it waits for 5 periods and then looks for the maximum value of the IIR filter output during one period. Finally, it outputs the synchronizing index (modulo the oversampling factor), corresponding to this maximum.

To do so, as depicted in the block diagram presented in Figure 47, a control signal *MaxTempFnd* goes high whenever a temporary maximum is found. This signal controls

a switch which, when closed, forwards the value of the counter *Cnt2* to the output *Index*, this counter value representing the temporary maximum index. If the tested value following the temporary maximum is not higher than the latter, then the switch opens and its index is not forwarded to the output.

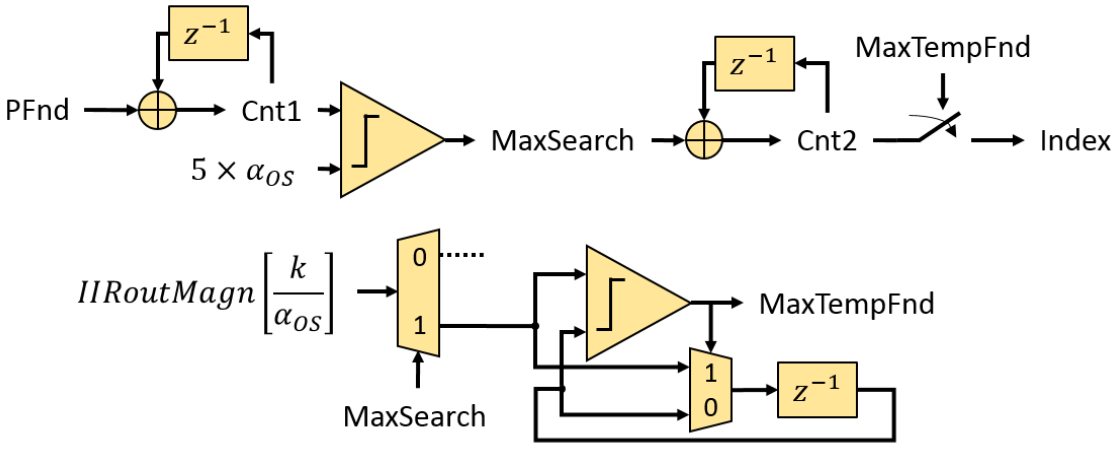


Figure 47: Symbol timing recovery block diagram.

5.9.5 Carrier Frequency Offset Estimation

The last non-ideality correction algorithm is the CFO estimation. Once the preamble is found, it estimates the CFO value using Equation (18). The average is performed over 6 periods, after which the demodulation is enabled.

More precisely, the algorithm starts by normalizing the Hermitian product, which is necessary because it is supposed in Equation (16) that everything is normalized.

Then, the imaginary part of the Hermitian product is extracted to produce the signal I_p from Equation (16).

Next, the precomputed and tabled values corresponding to the sine and the inverse of the cosine multiplied by the factor $2\pi T_b$ from Equation (16) are fetched from two Look-Up Tables (*LUT 1* and *LUT 2*).

Finally, the average is realized.

This is represented in the block diagram shown in Figure 48.

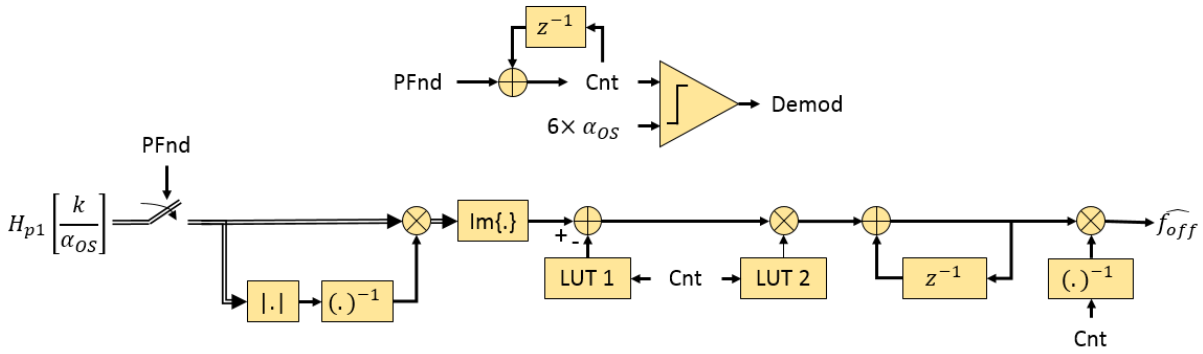


Figure 48: CFO estimation block diagram.

5.9.6 Demodulation

Finally, the demodulation takes place to recover the transmitted bits.

It starts by decimating the second Hermitian product, denoted H_{p2}, using the synchronizing index from the symbol timing recovery.

Next, the imaginary part is extracted.

Finally, the sign function *sgn(x)* is applied to get the symbols ($+1$ or -1), and the constellation demapping is realized to recover the bits from the symbols. These two operations can actually be performed using a single comparator, outputting 1 if the sample is positive and 0 otherwise.

The block diagram of the demodulation is shown in Figure 49.

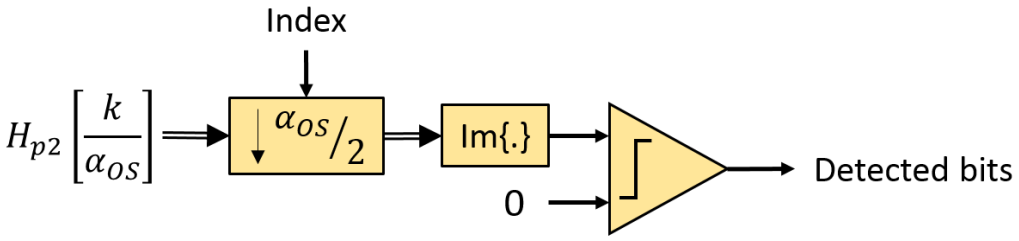


Figure 49: Demodulation block diagram.

5.9.7 Entire Receiver

Putting everything together, the global receiver block diagram is presented in Figure 50. An exemplary simplified analog RF portion is also represented for the sake of completeness. It includes the downconversion, which is tuned thanks to the CFO estimate, and the ADC.

Based on this diagram, a first estimation of the receiver computational complexity can be realized. The following operations are required:

- 1 convolution;
- 3 multiplexers;
- 8 delays;
- 2 complex conjugate computations;
- 2 complex multiplications;
- 6 additions;
- 4 real multiplications;
- 2 complex magnitude computations;
- 5 comparators;
- 2 switches;
- 2 multiplicative inverse computations;
- 2 imaginary part extractions;
- 2 LUT value extractions;
- 1 downsampling.

As for the transmitter, the receiver computational complexity is quite low as it involves only one convolution in addition to rather basic operations. The convolution takes place in the Low-Pass Filter block. The minimum required filter length L_{LPF} to ensure a proper filtering has been obtained by simulation as $L_{\mathrm{LPF}} = 8 \times \alpha_{OS} + 1$.

Actually, the IIR filter cannot be reduced to only one addition, multiplication and delay, but rather as a second convolution since those 3 operations need to be performed more than once. The actual number can be determined by considering the impulse response of the filter and its minimum required number of coefficients L_{IIR}. By simulation, a number of $L_{\mathrm{IIR}} = 14 \times \alpha_{OS} + 1$ coefficients has been obtained, which are actually 15 nonzero coefficients, spaced by $\alpha_{OS} - 1$ zeros.

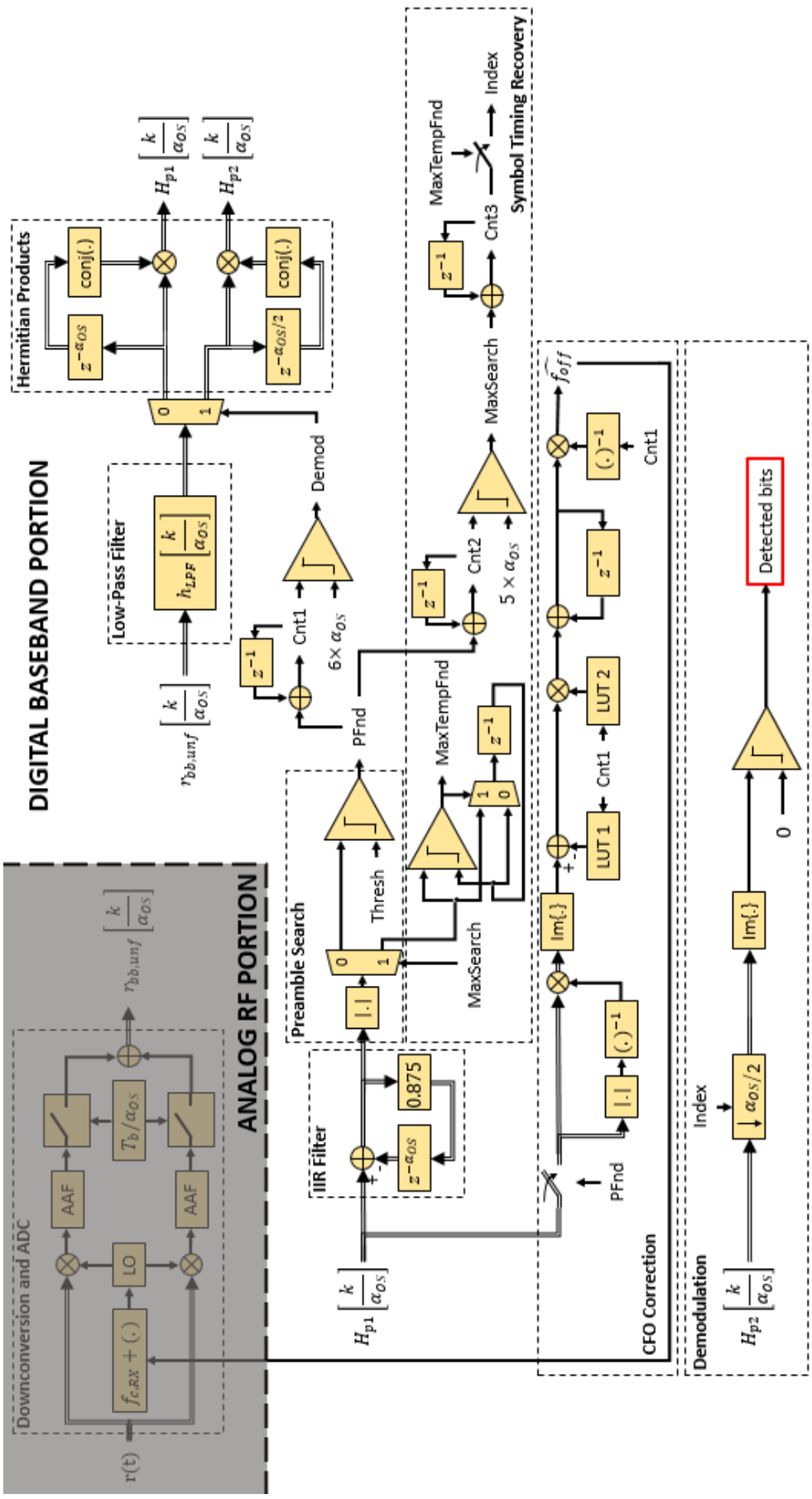


Figure 50: Receiver block diagram.

6 Quantization

In the digital domain, numbers cannot be represented with an infinite precision. The common digital representation of numbers is the floating-point one. However, from a hardware point of view, the fixed-point representation is less costly to manipulate. Therefore, as the goal is eventually to implement the digital part of the transceiver on a software platform such as the C platform, this fixed-point representation must be used.

In the fixed-point representation, binary numbers are represented using the 2's complement representation and are normalized so as to lie in the range [-1;1[. The 2's complement representation performs the following mapping from a binary number $B = b_{N-1}b_{N-2}...b_1b_0$ ($b_i \in \{0, 1\}$) to its decimal value D :

$$D = -b_{N-1}\ 2^{N-1} + b_{N-2}\ 2^{N-2} + ... + b_1\ 2^1 + b_0\ 2^0\ .$$

The normalization then consists in dividing the number by 2^{N-1} so that the implied binary point is located just after the first bit (which is the sign bit), which produces the fractional representation $F(B)$ as

$$F(B) = -b_{N-1}\ 2^0 + b_{N-2}\ 2^{-1} + ... + b_1\ 2^{-(N-2)} + b_0\ 2^{-(N-1)}\ .$$

This representation can be denoted using the Q-format QN or still Q1.N-1, meaning that the only bit assigned to the integer portion of a number is the sign bit and that the remaining N-1 bits are used for the fractional portion.

In order to save computational effort, the lowest level of quantization for each signal in the system should be used, while preserving sufficient precision so as to keep the initial BER performance. However, on a software platform like the C platform, the formats are limited to generic ones such as Q1.31, Q1.15 and Q1.7. This means that the signals whose required precision is the lowest (e.g. the constants) are represented in Q1.7, whereas the other ones are in Q1.15 or Q1.31. Moreover, it also means that saturation and truncation must be used during the operations on those signals in order to stick to one of those three formats.

The quantization levels at each step of the digital processing blocks have been determined by simulation after having mapped the code of the system implementation from floating-point to fixed-point. They are represented in lighter type in the block diagrams of the transmitter and of the receiver, displayed in Figure 51 and Figure 52 respectively. In these block diagrams, a quantization level of N stands for the Q1.N-1 format.

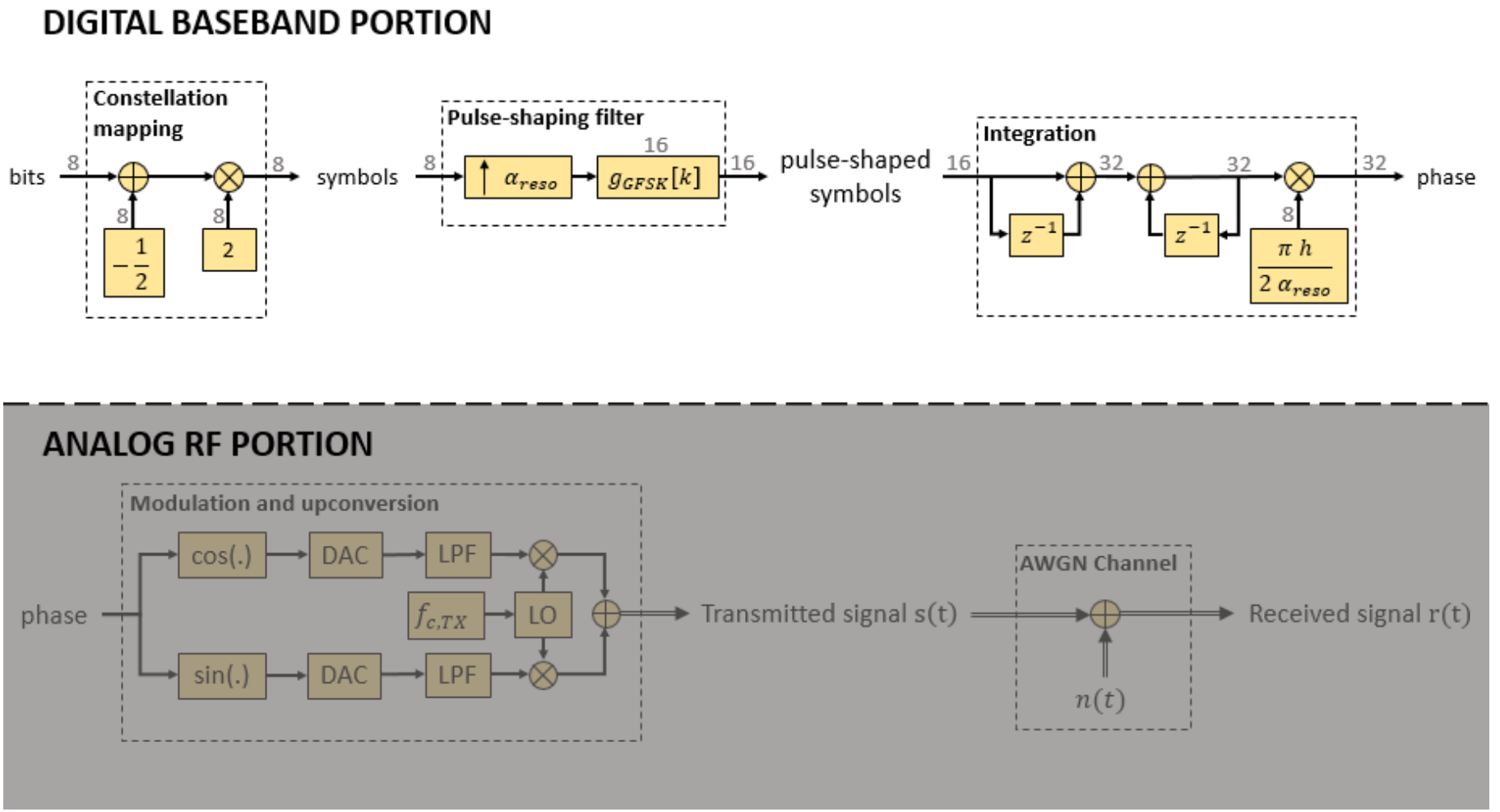


Figure 51: Transmitter block diagram with signal quantization.

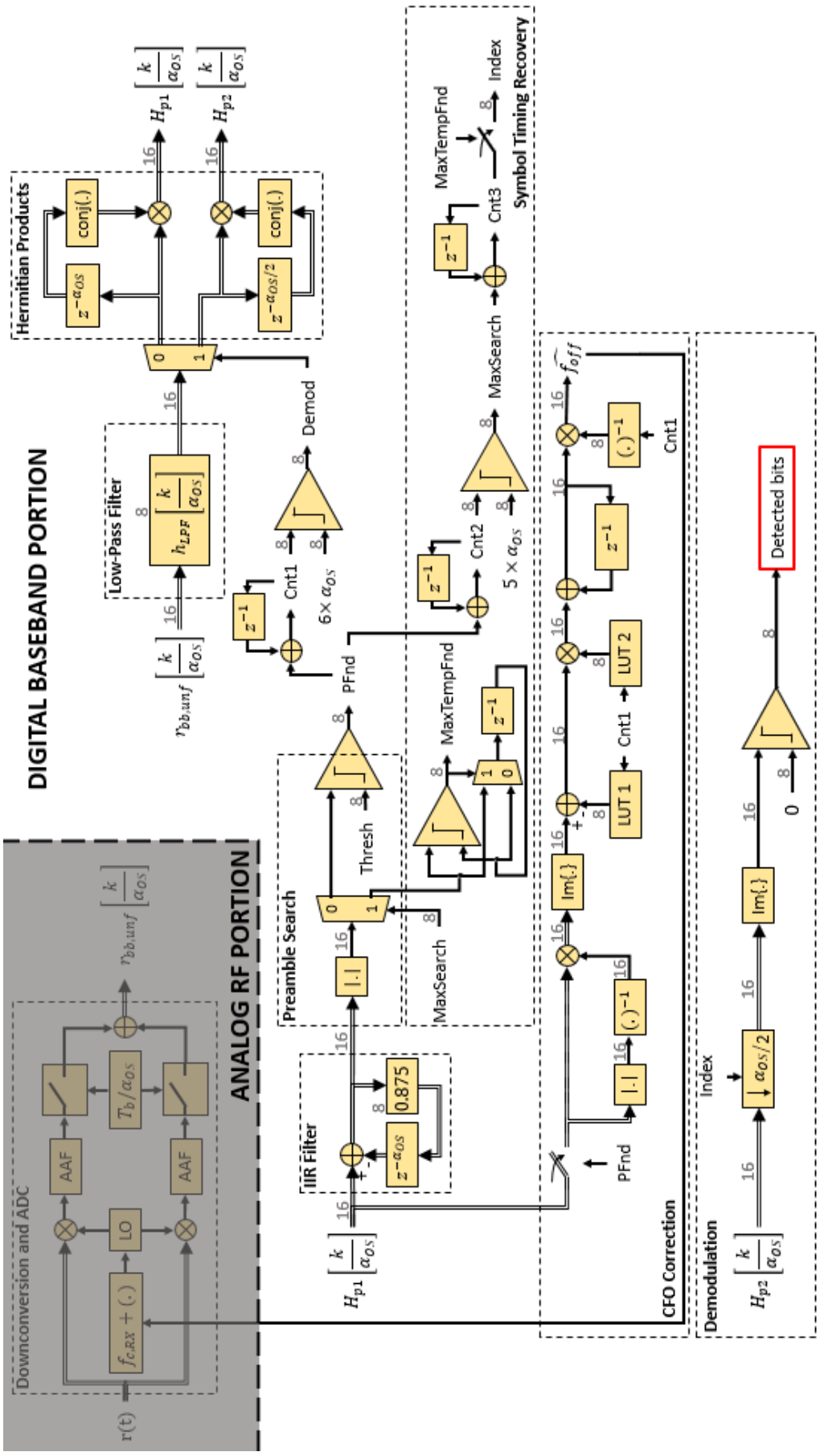


Figure 52: Receiver block diagram with signal quantization.

7 Conclusion

7.1 Conclusion

This thesis addresses the question of the low-energy implementation of a software-defined BLE transceiver in view of IoT applications. To this end, a GFSK receiver has been implemented with non-coherent differential detection. Furthermore, it features a preamble detection and a symbol timing recovery algorithm based on an IIR filter, as well as a data-aided carrier frequency offset mitigation through a simple Taylor series approximation.

In terms of performance, the system has been designed to meet the requirement of a 0.1-% bound on the bit error rate. This is almost perfectly reached using a 13-dB or 14-dB SNR since the system yields a 99.3-% and 99.9-% probability of satisfactory BER respectively. These values come from a bit error rate evaluation that has been run 1000 times.

The computational complexity of the system implementation has also been discussed, from a qualitative point of view. The conclusion is that the required operations are very basic. The most demanding ones are the convolutions, but only three are necessary.

7.2 System Limits and Future Work

First, an important task to be carried out in a future work is the quantitative evaluation of the system computational complexity in order to realize a proper energy consumption analysis of the system. This requires to implement the system on an actual software platform such as the C platform.

Moreover, the current system has room for improvement. The three main elements that may degrade its performances are the frequency drift, the sampling timing offset and the channel response.

7.2.1 Frequency Drift

The receiver is currently not robust against frequency drift since the CFO is only estimated once, during the preamble. As presented in Section 3.3, this drift can go up to 50 kHz during the transmission of one packet. This amounts to 20 ppm, which is exactly the critical value that makes the BER go beyond the targeted one, as illustrated in Figure 33.

A future work may consist in trying to mitigate this drift. An idea could be to periodically wake up the CFO mitigation algorithm after the first correction. But this time,

the algorithm cannot be data-aided as only the preamble data are known in advance. However, a decision-aided method could be used, as proposed in [16].

7.2.2 Sampling Timing Offset

Currently, nothing is done in order to mitigate the sampling timing offset that is discussed in Section 5.7.2. As this synchronization issue adds some uncertainty that can degrade (or sometimes improve) the bit error rate, it would be interesting to be able to suppress it, or at least to reduce it.

With this in mind, it would be interesting to analyze in a future work the tradeoff between the synchronization gain and the complexity that a structure like the Gardner's timing error detector from [15] could bring.

7.2.3 Channel Response

The last important non-ideality that is not yet taken into account is the channel response. In practice the transmitted signal gets distorted by more than the additive white Gaussian noise. For example, there can be fading or still multipath. Therefore, it would be interesting to add a mechanism for channel equalization.

References

[1] Cisco, *Cisco Annual Internet Report (2018-2023) White Paper*, Mar. 2020. Accessed: Apr. 9, 2020. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html

[2] McKinsey Global Institue, *The Internet of Things: Mapping the value beyond the hype*, Jun. 2015. Accessed: Apr. 9, 2020. [Online]. Available: https://www.mckinsey.com/~/media/McKinsey/Industries/Technology Media and Telecommunications/High Tech/Our Insights/The Internet of Things The value of digitizing the physical world/Unlocking_the_potential_of_the_Internet_of_Things_Executive_summary.ashx

[3] J. Bryzek, *The Trillion Sensors (TSensors) Foundation for the IoT*, 2015. Accessed: Apr. 12, 2020. [Online]. Available: http://www.iot-inc.com/wp-content/uploads/2015/11/2-Janusz.pdf

[4] C. Bisdikian, "An Overview of the Bluetooth Wireless Technology," *IEEE Communications Magazine*, pp. 86-94, Dec. 2001.

[5] Bluetooth SIG, *Our History*. Accessed: Apr. 10, 2020. [Online]. Available: https://www.bluetooth.com/about-us/our-history/

[6] Trusted Reviews, *Bluetooth 4.1: what is it and why should you care?*, Dec. 9, 2013. Accessed: Apr. 10, 2020. [Online]. Available: https://www.trustedreviews.com/opinion/bluetooth-4-1-what-is-it-and-why-should-you-care-2916997

[7] Y. Chen, et al., "A low power software-defined-radio baseband processor for the Internet of Things," in *Proceedings of the IEEE International Symposium on HPCA*, pp. 40-51, 2016.

[8] Hela Amor, Carolynn Bernier. "Software-Hardware Co-Design of Multi-Standard Digital BasebandProcessor for IoT," in Design, Automation and Test in Europe (DATE), Mar. 2019, Florence, Italy.

[9] Z. Sun, et al., "A 0.85mm2 BLE Transceiver with Embedded T/R Switch, 2.6mW Fully-Passive Harmonic Suppressed Transmitter and 2.3mW Hybrid-Loop Receiver," *ESSCIRC Conf.*, pp. 310-313, 2018.

[10] F.-W. Kuo, et al., "A Bluetooth Low-Energy (BLE) Transceiver with TX/RX Switchable On-Chip Matching Network, 2.75mW High-IF Discrete-Time Receiver, and 3.6mW All-Digital Transmitter," *VLSI Circ. Symp.*, 2016.

[11] B. Yu, L. Yang and C.-C. Chong, "Optimized Differential GFSK Demodulator," *IEEE Transactions on Communications*, vol. 59, pp. 1497-1501, Jun. 2011.

[12] M. K. Simon and C. C. Wang, "Differential Versus Limiter-Discriminator Detection of Narrow-Band FM," *IEEE Trans. Commun.*, vol. com. 31, pp. 1227-1234, Nov. 1983.

[13] M. Morelli and U. Mengali, "Joint Frequency and Timing Recovery for MSK-Type Modulation," *IEEE Trans. Commun.*, vol. 47, pp. 938-946, Jun. 1999.

[14] Y.-L. Huang, K.-D. Fan and C.-C. Huang, "A Fully Digital Noncoherent and Coherent GMSK Receiver Architecture with Joint Symbol Timing Error and Frequency Offset Estimation," *IEEE Trans. Veh. Technol.*, vol. 49, pp. 863-874, May 2000.

[15] M. Azam, et al., "An improved, Non-Data Aided Symbol Timing Recovery for GMSK Modulated Signals," in *Proc. IEEE INMIC*, pp. 107-111, 2003.

[16] D.-C. Chang, "Least Squares/Maximum Likelihood Methods for the Decision-Aided GFSK Receiver," in *IEEE Signal Processing Letters*, vol. 16, pp. 517-520, Jun. 2009.

[17] Lu, et al., "Bluetooth Low Energy Frequency Offset and Modulation Index Estimation," U.S. patent 9,722,845 B2, Aug. 1, 2017.

[18] Wikipedia, *Pile de protocoles*, Jul. 14, 2019. Accessed: Apr. 21, 2020. [Online]. Available: `https://fr.wikipedia.org/wiki/Pile_de_protocoles`

[19] Robin Heydon, *Bluetooth Low Energy*, 2012. Accessed: Apr. 21, 2020. [Online]. Available: `https://www.cl.cam.ac.uk/teaching/1819/MobSensSys/mobile-10.pdf`

[20] Bluetooth SIG, *Bluetooth Core Specification version 4.0*, Jun. 30, 2010. Accessed: Apr. 21, 2020. [Online]. `https://www.bluetooth.com/specications/bluetooth-core-specication`

[21] J. Decuir, "Introducing Bluetooth Smart, Part 1: A Look at Both Classic and New Technologies," *IEEE Consumer Electronics Mag.*, Jan. 2014.

[22] M. Afaneh, *Bluetooth 5 speed: How to achieve maximum throughput for your BLE application*, Sep. 6, 2017. Accessed: Apr. 27, 2020. [Online]. Available: `https://www.novelbits.io/bluetooth-5-speed-maximum-throughput/`

[23] J. G. Proakis and M. Salehi, *Digital Communications*, 5th edition, McGraw-Hill, New York, 2008.

[24] Wikipedia, *Carrier frequency offset*, Mar. 30, 2020. Accessed: May 23, 2020. [Online]. Available: `https://en.wikipedia.org/wiki/Carrier_frequency_offset`

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain
Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | **www.uclouvain.be/epl**