UNIVERSITY OF TWENTE.

Faculty of Electrical Engineering,
Mathematics & Computer Science

Direct-Sequence Spread Spectrum in Backscatter Wireless Sensor Networks

N.J. ten Brinke BSc
September 2020

Supervisors:
R.J. de Jong Msc
dr. ir. A.B.J. Kokkeler

Faculty of Electrical Engineering,
Mathematics and Computer Science
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

Voorwoord

Het typen aan deze master thesis zorgt ervoor dat ik terugkijk op de afgelopen tijd. Per slot van rekening is dit het werk waar je al die jaren naar toe heb gewerkt en het afronden van dit afstudeerproject betekent het afronden van een levensfase. In de afgelopen jaren heb ik me in veel verschillende onderwerpen en ook deze afstudeeropdracht was weer een ontdekkingstoch waarbij ik me in veel nieuwe dingen moest verdiepen. Naast dat ik veel kennis heb vergaard, ben ik ook als persoon enorm gegroeid. Mensen die mij nog kennen van de middelbare school zullen dit zeker beamen. Het komt natuurlijk voor dat studenten na het afronden van hun master nog verder gaan studeren voor een PhD maar ik ben van plan om de wetenschappelijke wereld achter me te laten en me te richten op het bedrijfsleven. Ik ben erg benieuwd naar de uitdagingen die me daar te wachten staan.

Ik wil Hans de Jong en André Kokkeler bedanken voor hun begeleiding. Jullie stonden altijd voor mij klaar en ik vond het erg prettig om met jullie te kunnen sparren en jullie scherpe inzichten hebben me vaak op de goeie weg gezet. Verder wil ik ook alle andere mensen bedanken die me hebben geholpen of gesteund in de afgelopen tijd! Voor nu wens ik jullie veel leesplezier!

Niek ten Brinke

Enschede, 28 augustus 2020.

Abstract

Backscatter Radio is a new, emerging technology that potentially allows long-lasting Ultra-Low Power (ULP) Wireless Sensor Networks. In this technology, the power for wireless communication is provided by an external device, instead of by the sensor nodes. Unfortunately, current State-of-the-Art Backscatter technology has a very limited communication range.

This thesis explores the usage of Direct-Sequence Spread Spectrum (DSSS) in Backscatter Nodes. DSSS is a coding technique to spread the signal across a larger bandwidth. Using simulation, a case is made that DSSS indeed improves communication range, especially when there is a lot of interference from other wireless transmissions. Furthermore, an Ultra-Low Power implementation of DSSS is proposed that only slightly increases energy consumption, compared to conventional Backscatter receivers.

Keywords: Backscatter radio, wireless sensor network, ultra-low power receiver, direct-sequence spread spectrum

Contents

List of acronyms

DSSS	Direct-Sequence Spread Spectrum
CW	Continuous Wave
ASK	Amplitude-Shift Keying
AP	Access Point
WSN	Wireless Sensor Network
FSK	Frequency-Shift Keying
ASF	Adaptive Spreading Factor
BER	Bit Error Rate
PN	Pseudo-Noise
BPSK	Binary Phase-Shift Keying
FPGA	Field-programmable Gate Array
SNR	Signal-to-Noise Ratio
MAC	Medium Access Control
ULP	Ultra-Low Power
ADC	Analog-to-Digital Converter
SIR	Signal-to-Interferer Ratio
RF	Radio Frequency
TDMA	Time-Division Multiple Access
CDMA	Code-Division Multiple Access
AGC	Automatic Gain Control
AWGN	Additive White Gaussian Noise

Chapter 1

Introduction

1.1 Backscatter Radio

Internet of Things is a popular concept where many electronic devices are interconnected to the internet [5]. One important aspect is deploying a lot of sensors to better understand and control our environment. A challenge herein is developing a long-lasting Wireless Sensor Network (WSN) with low maintenance costs that can be easily deployed. Reducing energy consumption is key for this. This is because conventional sensor nodes use power consuming radio transmitters with for example power-hungry amplifiers. A simple solution would be to remove the power amplifier from the Wireless Sensor Node. This is exactly what Backscatter does. In Backscatter, a node only scatters (absorb or reflect) an incoming Radio Frequency (RF) wave. This RF wave can be ambient (i.e. existing wireless signals such as WiFi) or generated using dedicated hardware, an exciter. At the same time, it selectively changes amplitude, frequency and/or phase of the signal to modulate data. While nodes using conventional wireless technologies (such as WiFi or Bluetooth Low Energy) have a power consumption in the order of milliwatts, Backscatter reduces this to the order of microwatts, around a 1000 times smaller! [24]

1.2 Parking Sensor Networks

A real-life example where Backscatter would be beneficial to use is a Network of wireless Parking Sensors. Many parking garages are nowadays equipped with sensor nodes that measure occupancy of parking spots. This information is transmitted to a central gateway [13]. With the upcoming trend of autonomous cars, this is only expected to increase, as these cars can be guided towards empty parking slots. Currently, these nodes are battery-operated devices with a limited life-span of three to five years [14]. Because parking lots can potentially span hundreds of devices, this results in high maintenance costs. As mentioned before, Backscatter features very low power usage so Backscatter can increase the life-span of these nodes greatly. However, current Backscatter devices have a severely limited operating range. A conventional sensor node has a range of approximately 35m [14], whereas Backscatter devices have a typical operating range of only a few meters [24]. Exceptions are possible, but they either use a high-sensitivity receiver or very high exciter transmit power. This thesis studies a different technique to increase communication range.

1.3 Direct-Sequence Spread Spectrum

The technique that is researched in this thesis is Direct-Sequence Spread Spectrum (DSSS). This is a coding technique that spreads the signal across a larger bandwidth by encoding (chipping) bits with a longer Pseudo-Noise (PN) sequence. As a result, narrowband interferers have less overlap with the signal and will therefore cause less disturbance. Because of this, it is expected that using this technique will improve the communication range, at the expense of an increase in energy consumption.

1.4 Contributions & Outline

This thesis contributes by presenting a practical implementation of DSSS in Wireless Backscatter Networks. Chapter 2 will start by explaining how DSSS works and what its advantages and disadvantages are. Furthermore this chapter will discuss related work and present the research questions that form the basis for this thesis. Chapter 3 will provide a theoretical foundation for DSSS and detail the system concept that is used for the simulation. Chapter 4.1 and 4.2 present the results from the simulation described in chapter 3. Chapter 5 then proposes an Ultra-Low Power (ULP) quantization method and explores other problems that need to be solved before DSSS can be applied in practice. Finally in chapter 6, conclusions are drawn from this thesis and possible improvements are discussed.

Chapter 2

Background, Related Work and Research Questions

2.1 Backscatter

This section explains the concept of Backscatter by explaining the basic techniques used in Backscatter Radio. Figure 2.1 shows a schematic overview of a Backscatter system. A Backscatter system consists of three main components: Exciter, Backscatter Tag and Receiver.

- Exciter: Transmits a single-tone Continuous Wave (CW).
- Backscatter Tag: Reflects (scatters) signals by modulating the antenna load. This can be done by using both real (resistive) and reactive (capacitive, inductive) loads which results in either Amplitude or Phase Modulation.
- Receiver: Receives signal and extracts the data from the backscattered signal. Often a combination of Amplitude-Shift Keying (ASK) Modulation and passive Diode Envelope Detectors is used, eliminating the need for active components operating at RF frequencies (even though active receivers have superior performance, they consume a lot of power).

Two types of Backscatter systems can be distinguished: bi-static and mono-static. In a bi-static system, the receiver and exciter are two separate devices whilst in a mono-static system the receiver and exciter are integrated into the same device. Figure 2.1 shows a bi-static system. Bi-static systems generally give longer communication distance. In ambient backscatter, the data is piggybacked on an existing RF signal (then the exciter is for example a TV tower or a WiFi router). Other Backscatter systems have a dedicated exciter. This thesis focuses on bi-static systems with a dedicated exciter.

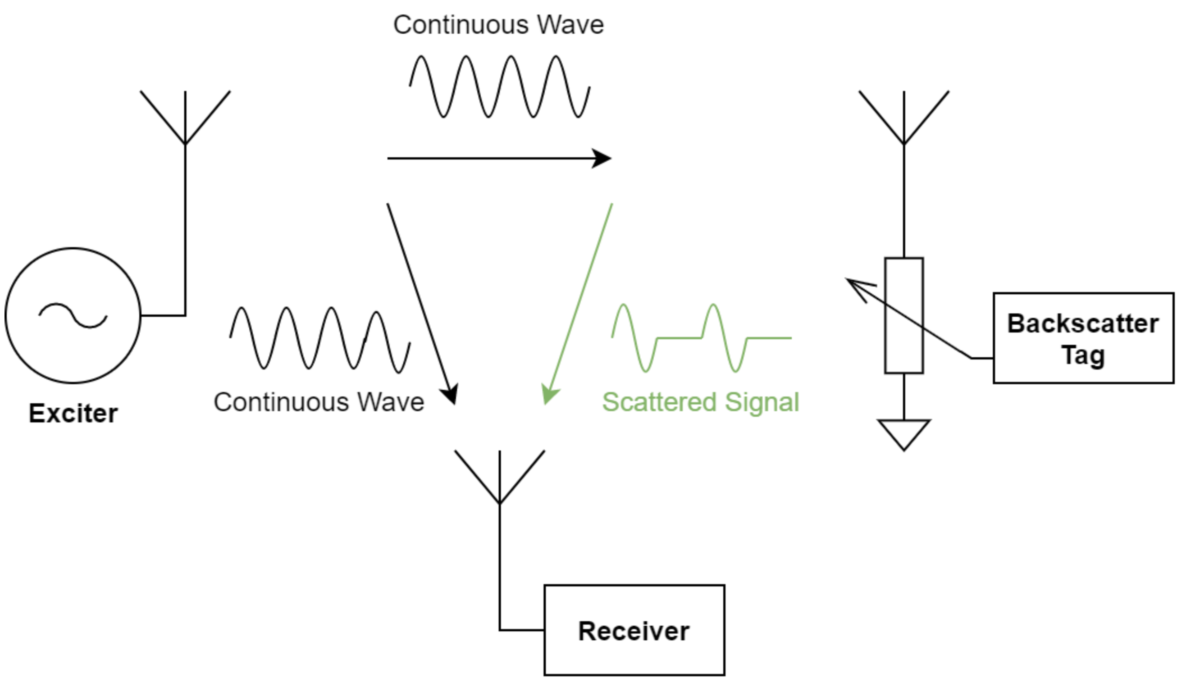

Figure 2.1: Backscatter Basic System Model

The current observation is that ultra-low power hardware exists, but their performance is often poor, i.e. they have low bit rates and/or low range [24]. One of the reasons that Backscatter has a low-range is because it has two RF channels with a single excitation source (i.e. the exciter signal has to get to the transmitter ánd from the transmitter to the receiver). This means that the path loss is much higher than with conventional wireless techniques. Furthermore, the Envelope Detector-based receivers have a low sensitivity (around -50dBm) [17]. Finally, Backscatter receivers have to deal with strong in-band interference from the Carrier Wave.

When exciter power is not sufficient to transmit data all the way back to the Access Point (AP), a mesh network (or multi-hop) can be created in which data can be routed via other nodes all the way to the final destination (in most cases the AP but other nodes can also be the destination). This, of course, has the disadvantage that nodes need to be actively listening more often (to receive incoming traffic from other nodes). A schematic overview of such a mesh network is shown in Figure 2.2.

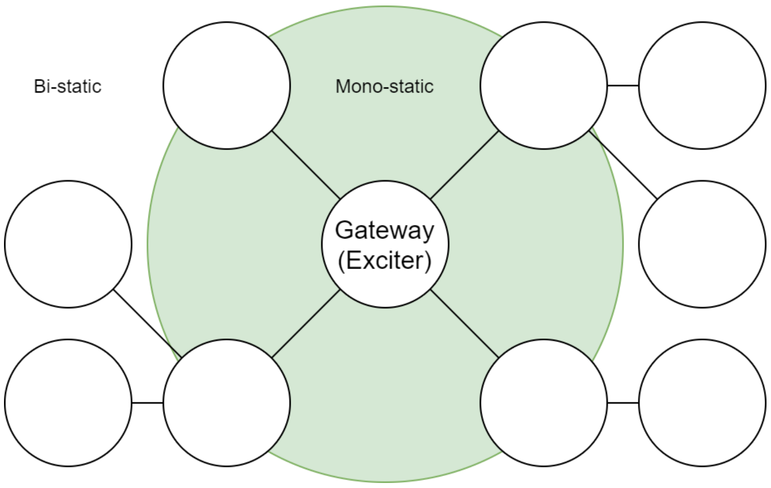

Figure 2.2: Backscatter Mesh Network

The large green circle indicates the area where all communication is mono-static. This is because these nodes communicate directly with the Gateway (and so the receiver and the exciter are integrated in the same device). Outside this circle, the exciter power is not high enough and bi-static systems are created to transmit data from the outer nodes to the Gateway.

Creating a mesh network is one option of increasing the range between nodes / gateways. Other options are: increasing the exciter strength, lowering the bit-rate (also called Rate Adaptation), increasing receiver sensitivity or using modulation techniques that are more robust against narrow-band interference and/or noise. Increasing exciter strength and receiver sensitivity are not always an option (for example because of regulations for maximum exciter strength or because of hardware limitations, imposed by the need for low power consumption). Finally, lowering the bit-rate is constrained by the application and not always an option. This leaves one option for improving transmission range: the usage of advanced modulation schemes. This thesis investigates the benefits of employing spread spectrum techniques to improve the range.

2.2 Direct-Sequence Spread Spectrum

One of these spread spectrum techniques is DSSS. This is a widely used method to allow multiple transmitters to send simultaneously over a common carrier (CDMA, such as 802.11b, GPS, Zigbee etc. etc.), but because it is much more robust to narrowband interference (from other wireless transmissions) it is also expected to improve the range. In short, DSSS converts a narrowband signal to wideband by encoding (chipping) bits with a longer PN sequence. The transmission time of a bit stays the same, but chipping introduces high frequency signals (because the chips are short pulses). A longer sequence results in shorter chip pulses and hence in a larger bandwidth. A larger bandwidth means that there is less signal overlapping with the narrowband interferer, which in turn means that the robustness of the system against narrowband interference increases. A schematic overview of this is shown in Figure 2.3. The User Data is multiplied with a PN Sequence which results in a wideband signal. A more detailed explanation will be given in chapter 3. Another advantage is that DSSS is more resilient against multipath fading [20] [10], but this thesis won't go into details on this topic.

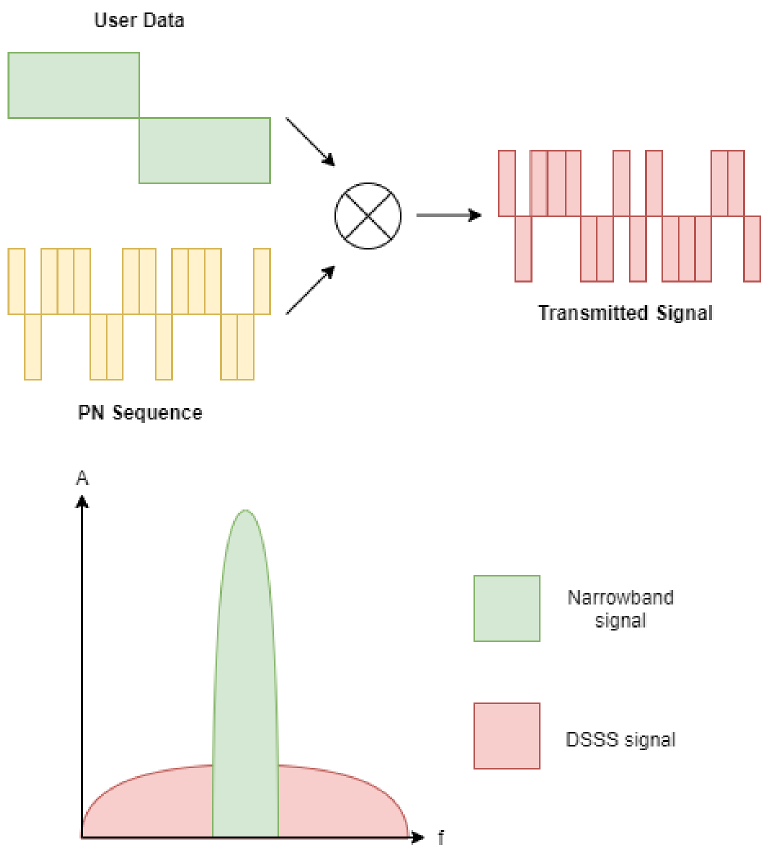

Figure 2.3: Direct-Sequence Spread Spectrum Overview

2.2.1 Downsides of Direct-Sequence Spread Spectrum

What are the downsides of Direct-Sequence Spread Spectrum that make it less practical to use in Backscatter?

- Direct-Sequence Spread Spectrum has quite a high power consumption because the receiver and transmitter need to be synchronized. This is because the received signal needs to be despreaded. A more detailed explanation of how this despreading operation works is given in chapter 3. An easy, often used, synchronization method is cross-correlation. However, this is computationally intensive (and results in a higher power consumption). According to Xu et al. [24] and [27], a Field-programmable Gate Array (FPGA) is more energy-efficient than a microcontroller when performing computations so it might be possible to create an energy-efficient implementation of a receiver in a Hardware Description Language (VHDL/Verilog) in such a way that it is usable in Backscatter. A downside of an FPGA is that it has a higher standby power consumption, compared to a microprocessor. Also, it might be possible to synchronize using different methods to simplify the despreading computation.

- Typical backscatter receivers use an envelope detector with a comparator to decode the received bits. However, one of the aspects of DSSS is that the peak power level is much lower and as a result, the signal-to-noise ratio in the channel is much lower. For proper decoding, more resolution is most likely needed in the Analog-to-Digital Converter (ADC) stage as the sampling is done closer to the noise floor. On the other hand, high-speed ADCs generally consume too much power to be used in Backscatter nodes.

- DSSS suffers from a so-called near-far effect. This happens whenever multiple transmitters are transmitting simultaneously to the same receiver but one of them is much closer to the receiver. DSSS transmissions exhibit noise-like interference to other transmissions, so the tranmission over longer distance will suffer from having a much lower SNR at the receiver input. Traditionally this is solved by reducing transmission power for shorter distances. However, this can't be done easily in Backscatter systems because it would require a more extensive load switching network.

Next to researching the range improvement resulting from DSSS, this thesis will also research methods that provide solutions to the problems mentioned above.

2.3 Related Work

2.3.1 Alternative Modulation Techniques

Many attempts have been made to implement more advanced modulation schemes in Backscatter Radio. An example of this is the work of Ensworth et al. [7] They use a combination of Frequency-Shift Keying (FSK) and a fixed frequency offset to generate signals that are compatible with the Bluetooth Low Energy standard. FSK however requires a local, power-consuming oscillator. It is also mentioned that the generation of FSK signals also creates unwanted signals that cause interference on other nearby transmissions.

LoRaWAN uses Chirp Spread Spectrum (CSS) to greatly improve the transmission range. Talla et al. [21] ported this to a Backscatter system. However, to achieve this great range, they also used specific LoRa receivers with a very high sensitivity. This makes it hard to compare their technique with comparable research (as it is unclear whether CSS or the receivers are the cause of the range improvement).

Another commonly seen method is using ASK with a fixed frequency offset such as done by Zhang et al. [26]. However, this method still results in a low range. They report a maximum communication distance of only three meters.

2.3.2 Node Synchronization in WSN

Cross-correlation is often used for synchronizing DSSS transmissions but is computationally intensive, as mentioned before. Instead, synchronization can be moved from the Physical Layer to the Medium Access Control (MAC) layer. Normally, MAC-level synchronization with Spread Spectrum can be difficult, because it needs to be quite tight (that is, the local PN sequence and in the incoming data need to be aligned correctly) to perform the despreading operation. Because of the low chip rates (compared to conventional wireless systems that use much more bandwidth), these MAC protocols might already provide synchronization that is accurate enough to perform DSSS. This is especially the case when DSSS is used in combination with Rate Adaptation, meaning that the overall bandwidth will stay more or less constant.

When analysing various MAC protocols as done by Bachir et al. [2], it can be seen that synchronization is indeed commonly done on the MAC layer. It can be done using schedules (as happens in TDMA-based MAC protocols) or with Preamble Sampling and simultaneously with other network coordination. A third option could be something similar as what is done in EkhoNet [27]: the exciter could for example send a synchronization pulse when transmission errors are detected, or even at every frame to make sure that the synchronization is tight. Furthermore, these MAC protocols could be extended for Rate Adaptation and a Adaptive Spreading Factor Scheme (ASF). This thesis will not explore various MAC protocols in detail, but they form a basis for the recommendations done in subsection 6.2.1.

2.4 Research Questions

This thesis researches methods to apply DSSS to Backscatter radio while keeping the sensor node as simple as possible (and hence energy-efficient). The main research question of this thesis is:

Does the usage of a practical Direct-Sequence Spread Spectrum implementation combined with lowering the bit rate improve the communication range compared to just lowering the bit rate?

Practical means an implementation that can be used on a Backscatter node, so uses only low-cost components and has a low power-consumption. It is hard to characterize exact range improvement because it depends on many factors. See the work of De Jong [9] for a detailed characterization. In this thesis, the goal is to minimize the required Energy-per-bit for a achieving a specfic Bit Error Rate (BER). For proper comparison, the required BER will be fixed at 10^{-3}. The sub-research questions are:

1. How much will the usage of DSSS improve the robustness against noise? (section 4.1)
2. How much will the usage of DSSS improve the robustness against narrowband (self-)interference? (section 4.2)
3. What is the sampling resolution needed for correctly decoding DSSS signals? (section 5.1)
4. How can we sample the signal in an ultra-low power fashion with the required resolution? (section 5.1)

In the next chapter, we will start by looking at DSSS from a theoretical perspective.

Chapter 3

System Overview

First of all, this chapter attempts to show that DSSS is indeed a suitable method for reducing the Energy-per-bit needed to achieve a BER of 10^{-3} (and increasing the communication distance). To get an idea of how much DSSS could improve the performance of Backscatter, this thesis looks at three different kinds of interference: Additive White Gaussian Noise (AWGN), interference and self-interference of the CW.

3.1 Theory

Why does DSSS increase the communication range? As explained in section 2.2, DSSS spreads a signal using a certain spreading factor, thereby increasing the bandwidth. The resulting bandwidth is the original signal bandwidth times the spreading factor. This spreading would not have any benefits when the frequency band is only occupied by the transmission of the signal. However, it becomes useful when different wireless signals are present that (partly) overlap the same frequencies.

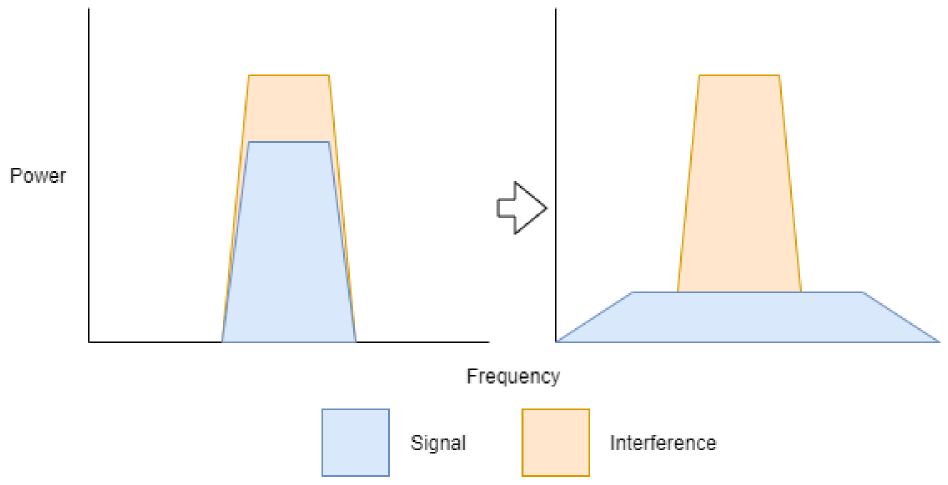

Figure 3.1: Concept of DSSS

Figure 3.1 illustrates this. On the left is the unspreaded signal overlapped by an interferer. As the spreading factor increases (figure on the right), the bandwidth starts to increase and less and less of the signal is overlapping with the interferer.

3.1.1 Time Domain Observations

Some equations can be used to show the effect of DSSS [22]. DSSS is applied in a discrete system. First of all, the spreading operation can be written as an element-wise multiplication:

$$t[n] = p[n] * d[n] \tag{3.1}$$

where $t[n]$ is the transmitted baseband waveform, $p[n]$ is the PN sequence and $d[n]$ is the data stream. $d[n]$ is upsampled such that each element is represented with an amount of samples that is equal to the length of the PN sequence. For example, if the data is $[1, 0]$ and $p[n]$ has a length of 4, then $d[n]$ will be $[1, 1, 1, 1, 0, 0, 0, 0]$.

The PN sequence has some important properties. First of all, this sequence can contain only elements of the set $\{-1, 1\}$, which are randomly selected. An example sequence with a length of 8 would be $[1, 1, -1, 1, -1, -1, 1, 1]$. Another important property is that an element-wise multiplication of the PN sequence with itself results in a sequence of 1s. This is shown in Table 3.1. This is important because it allows the despreading operation.

1	1	-1	1	-1	-1	1	1	
1	1	-1	1	-1	-1	1	1	x
1	1	1	1	1	1	1	1	

Table 3.1: Multiplying a PN sequence with itself

The same thing can be done in a binary system, in which case a PN sequence can only contain elements of the set $\{0, 1\}$. In this case, not a multiplication but a XNOR operation will result in a sequence of 1s (when doing an element-wise operation of the PN with itself). The truth table of XNOR is shown in Table 3.2.

Input A	Input B	Output
0	0	1
0	1	0
1	0	0
1	1	1

Table 3.2: XNOR Truth Table

Noise is added in the transmission channel, so the signal at the receiver will be:

$$r[n] = t[n] + n[n] \tag{3.2}$$

where $t[n]$ is the transmitted signal from Equation 3.1 and $n[n]$ is the noise.

Substituting Equation 3.1 into Equation 3.2 gives:

$$r[n] = p[n] * d[n] + n[n] \tag{3.3}$$

The receiver despreads this signal by multiplying it with the PN sequence:

$$r[n] * p[n] = p[n] * p[n] * d[n] + p[n] * n[n] \tag{3.4}$$

$p[n] * n[n]$ gives a different type of noise $\tilde{n}[n]$. And if the system is synchronized it holds that $p[n] * p[n] = 1$ (as shown in Table 3.1). So the final equation is:

$$r[n] * p[n] = d[n] + \tilde{n}[n] \tag{3.5}$$

Still, $d[n]$ is upsampled compared to the original data, so an integration step is performed to downsample the signal. As mentioned before, If the system is not synchronized or if the receiver has the incorrect PN sequence, then $p[n] * p[n] \neq 1$ and the resulting signal is just noise. This is the major reason why a DSSS transmission is difficult to intercept.

Based on Equation 3.5, we can conclude that DSSS does not improve robustness against AWGN. Why then bother using DSSS?

3.1.2 Frequency Domain Observations

Things start to change when other (other than white noise) sources are interfering (other wireless transmissions that is). This is illustrated using the frequency domain.

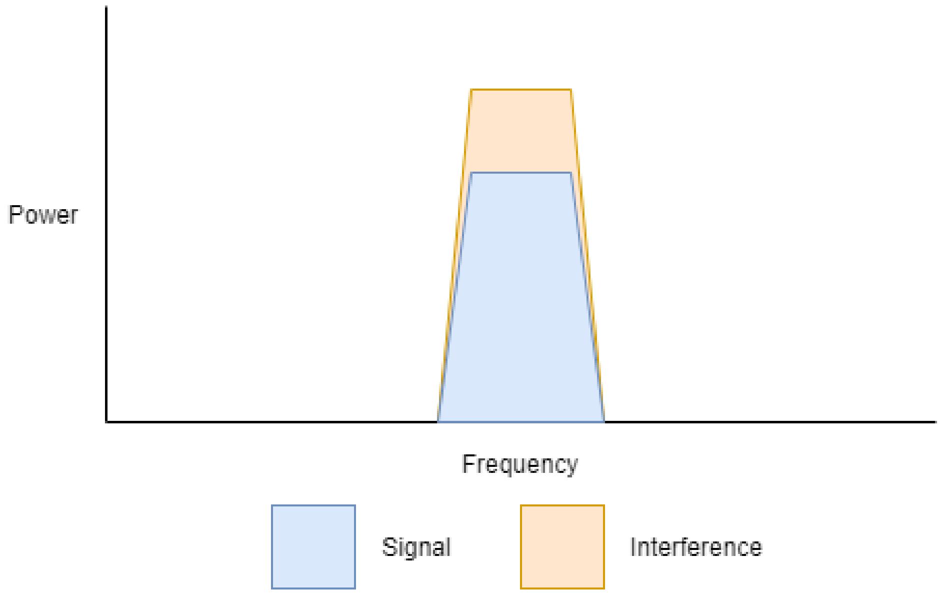

Figure 3.2: Signal overlapped by Interferer

In the worst case scenario, the signal of interest (being the DSSS signal) is fully overlapping with an interferer. This is shown in Figure 3.2. Packet Loss will probably occur often, and the expected BER is high.

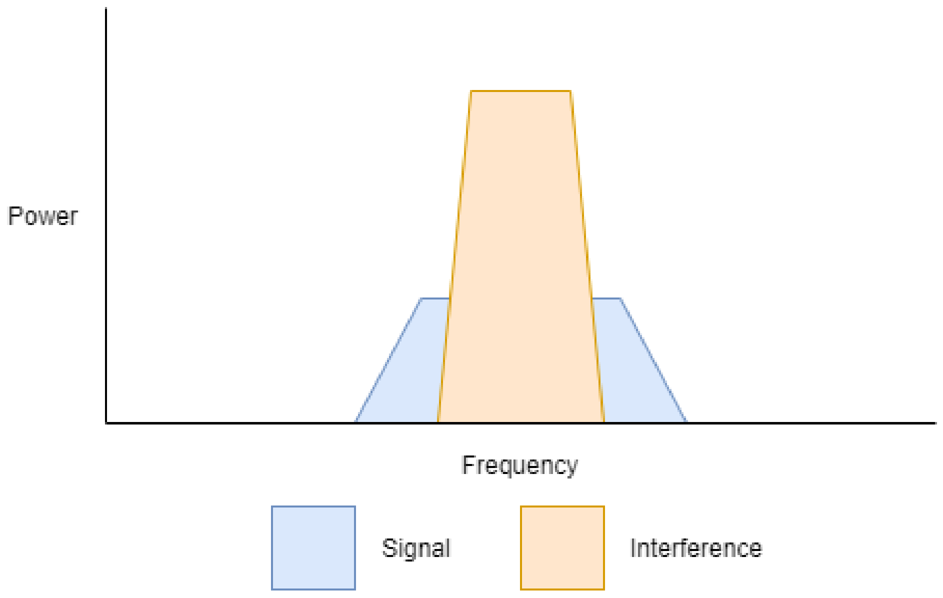

Figure 3.3: Signal partially overlapped by Interferer

Now consider the case where the signal is spreaded with a factor two as shown in Figure 3.3. Since the interferer still has the original bandwidth, half of the bandwidth (and hence half of the transmitted power) is still overlapped. But compared to Figure 3.2, the expected BER is much better.

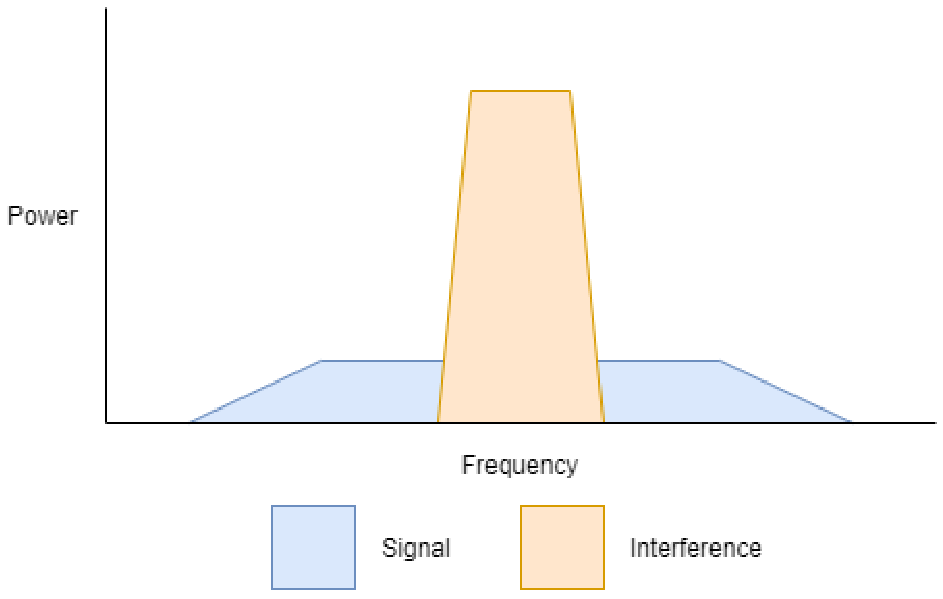

Figure 3.4: Signal partially overlapped by Interferer

Notice that the area (bandwidth times power) has not changed, so the transmitted power stays equal. This means that spreading the signal also means lowering the Signal-to-Noise Ratio (SNR). During despreading, the spreading operation is reversed by multiplying the received signal and integrating the result. This operation results into coding gain (also called processing gain) such that the resulting SNR is increased to match at least the original SNR (or higher, but this depends on the transmission channel). For a given modulation type and bit-error rate, coding gain is defined as *the ratio of the SNR per bit required for the uncoded signal to the SNR per bit required for the coded signal* [25]. For a more concise description of what coding gain actually is, also refer to [25].

Now with the situation in Figure 3.4, the signal is spreaded with a factor 4. Only a quarter of the signal is overlapped by the interferer. Note that the interferer bandwidth is unchanged in all three cases. The relative Signal-to-Interferer Ratio (SIR) is defined as the total signal power divided by the signal power that is overlapping with the interferer and can be written as:

$$SIR_{rel} = \frac{BW_{signal}}{BW_{interferer}}, \tag{3.6}$$

where BW_{signal} is the bandwidth of the signal and $BW_{interferer}$ is the bandwidth of the interferer. The relative Signal-Interferer Ratio in Figure 3.4 is 4, compared to 2 in Figure 3.3 and 1 in Figure 3.2.

This leads to the hypothesis that doubling the spreading factor corresponds to a coding gain of 3dB. To confirm this hypothesis, a wireless communication system is simulated in Matlab using a SNR (or SIR) sweep and PN sequences of different lengths. The point of interest is when the BER has a value of 10^{-3}, this is the point literature in general uses for evaluating a communication system. A more in-depth theoretical description and justification of DSSS can be found in [1].

3.1.3 Spectral Amplitude, Signal Ratio and E_b/N_0

The equations below are used to calculate the ratios resulting from the simulation. The signal and the interferer are assumed to be real signals. The equations below use SIR but the can be replaced with SNR depending on the channel (the same equations hold for an AWGN channel). The RMS Voltage levels are calculated using the formula

$$RMS = \sqrt{\frac{\sum x[n]^2}{N}}, \tag{3.7}$$

where N is the number of samples and $x[n]$ are the sample values. The $Signal$ and $Interferer$ are time-based vectors and normalized using the formulas

$$S_n = \frac{S}{RMS_S} \tag{3.8}$$

and

$$I_n = \frac{I}{RMS_I}, \tag{3.9}$$

where S and I are the $Signal$ and $Interferer$ respectively, and RMS_S and RMS_I are calculated using Equation 3.7. The resulting signal at the receiver is calculated using the formula

$$S_{receiver} = S_n + \frac{I_n}{SIR_{mag}}, \tag{3.10}$$

where SIR is converted to magnitude and S_n and I_n are calculated using Equation 3.8 and Equation 3.9 respectively.

As mentioned before, the simulation sweeps SIR_{dB}. The SIR is converted to E_s/N_0 (which is the energy-per-symbol on a dB scale) and E_b/N_0. For real signals, the generalized formula for calculating the E_s/N_0 is

$$E_s/N_0\ (dB) = 10log_{10}(\frac{0.5T_{sym}}{T_{samp}}) + SNR_{dB}, \tag{3.11}$$

where T_{sym} is the symbol period of the signal and the T_{samp} is the sampling period of the signal [12]. The formula for converting E_s/N_0 into E_b/N_0 is:

$$E_b/N_0\ (dB) = E_s/N_0 - 10 * log_{10}(k), \tag{3.12}$$

where k is the number of information bits per symbol [12]. Applying this to the DSSS Matlab simulation gives:

$$E_s/N_0\ (dB) = 10 * log_{10}(\frac{N}{2 * Nbit * SF}) + SIR_{dB} \tag{3.13}$$

and

$$E_b/N_0\ (dB) = E_s/N_0 - 10 * log_{10}(\frac{1}{SF}), \tag{3.14}$$

where SF is the spreading factor, N is the number of simulation samples (the timeline) and $Nbit$ is the number of simulated bits. So, a bit is represented by multiple (for example 50) samples. The simulation used in this thesis simulates a fixed number of samples to which all signals are upsampled. This ensures that increasing the spreading factor also means an increase in bandwidth. A numeric example: the number of samples per chip is given using the equation

$$Samples/chip = \frac{N}{Nbit * SF}. \tag{3.15}$$

Table 3.3 shows the results for different spreading factors in case $N = 256000$ and $Nbit = 100$. Clearly, the chip time decrease for higher spreading factors. This means that the bandwidth increases.

Spreading Factor	1	16	64	256
Samples per Chip	2560	160	40	10

Table 3.3: Samples per Chip for different Spreading Factors

3.2 Simulation Concept

In this thesis, the results of numerous Matlab simulations are used to show that the theory from section 3.1 is also applicable to a Backscatter receiver. De Jong [9] proposed a hardware platform and the concept of this system is used as a basis for the simulation (as it is proven to be ULP and it has a good performance) and extends this with the ideas from the previous section. In this thesis it is assumed that the system is already synchronized.

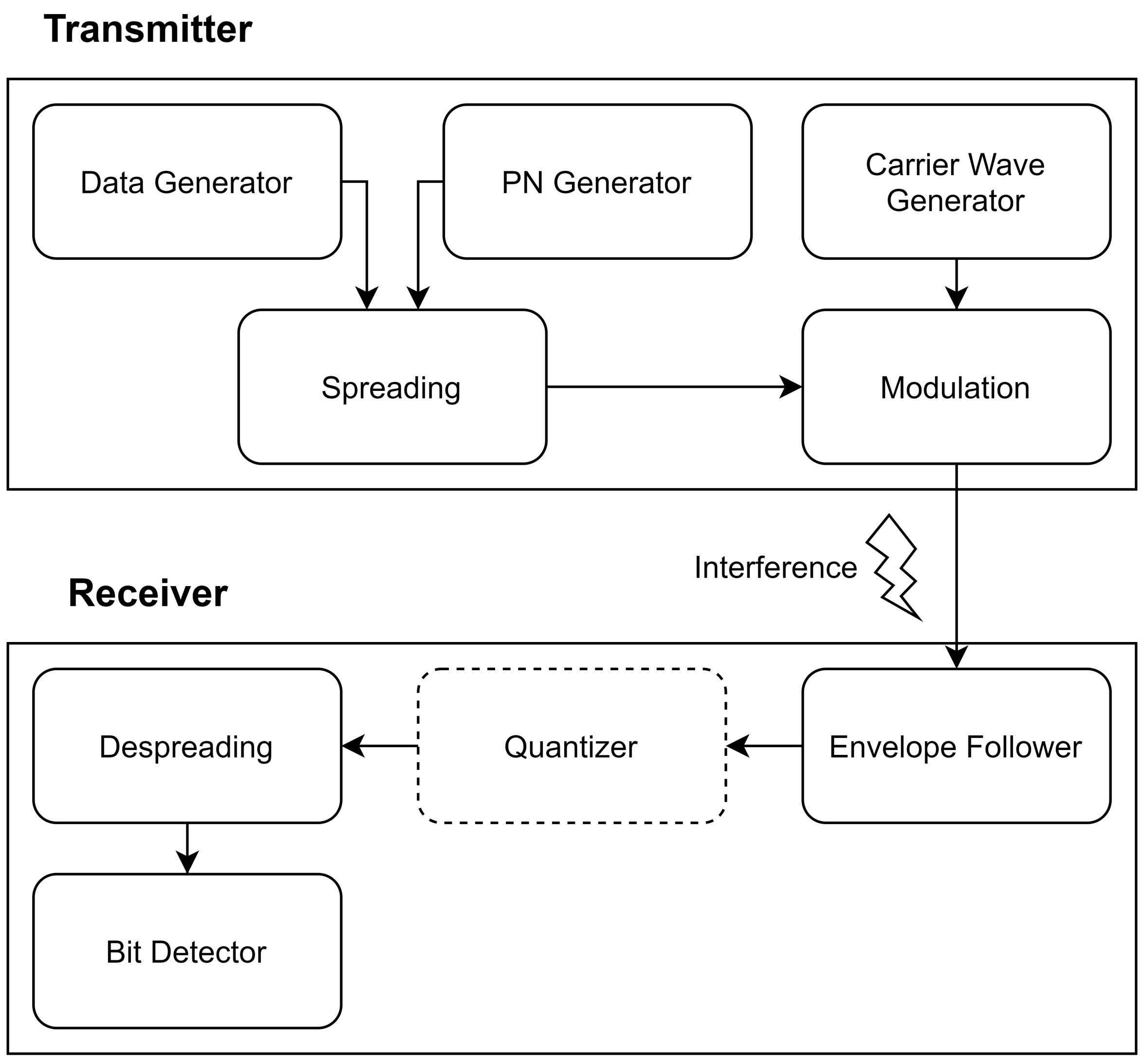

Figure 3.5: Matlab Simulation Overview

Figure 3.5 shows the block diagram of the Matlab simulation. At the transmitter-side, the data is generated, spreaded and modulated with the CW. This is transmitted to the receiver, and noise or interference is added in the channel. The Envelope Follower removes the CW and shifts the signal to baseband. In Figure 3.5, the

Quantizer is dashed because quantization is investigated *after* the basics of DSSS are confirmed to be working. This section assumes that there is no quantization noise. The effects of the latter will be investigated in section 5.1. Finally the signal is despreaded and fed into a bit detector. All blocks will be described into more detail in the following subsections.

3.2.1 Transmitter Spreading

Most of this thesis is about changing the receiver, but the transmitter also needs to be considered because it performs the spreading operation. Listing 3.1 shows a short Matlab code snippet that was used to spread the the generated data.

```
 1 function spreadedBitStream = dsss(bitStream, pnSeq)
 2     spreadedBitStream = []; % allocate empty array
 3 
 4     % scale values from [0,1] to [-1,1]
 5     bitStream = bitStream * 2 - 1;
 6     pnSeq = pnSeq * 2 - 1;
 7 
 8     % spread bitStream
 9     for i=1:length(bitStream)
10         spreadedBitStream = [spreadedBitStream bitStream(i) *
               pnSeq];
11     end
12 end
```

Listing 3.1: Matlab Spreading Code

At line 2, an empty array is allocated which will contain the spreading result. At line 4 and 5, the incoming bit stream and the PN sequence is stretched from binary to $[-1, 1]$ so all 0's are represented by -1 instead. Lines 9 till 11 contain a loop that recursively fills the array. On every iteration a copy (either multiplied with 1 (no change) or -1 (flipped)) is concatenated to the resulting array. The resulting array is then upsampled to the (fixed) number of simulation samples to ensure that a higher spreading factor (or longer PN sequence) also means a larger bandwidth.

3.2.2 Carrier Wave Modulation

To transmit the signal using RF communication, the signal needs to be modulated to a carrier frequency. The CW is a fixed-frequency sine wave. This wave is multiplied with a sequence of 0s and 1s (the spreaded data signal, that is converted from $[-1, 1]$ to $[0, 1]$) which creates a amplitude-modulated signal, like shown in Figure 3.6. This shifts the center frequency of the signal (or baseband) to the frequency of the CW.

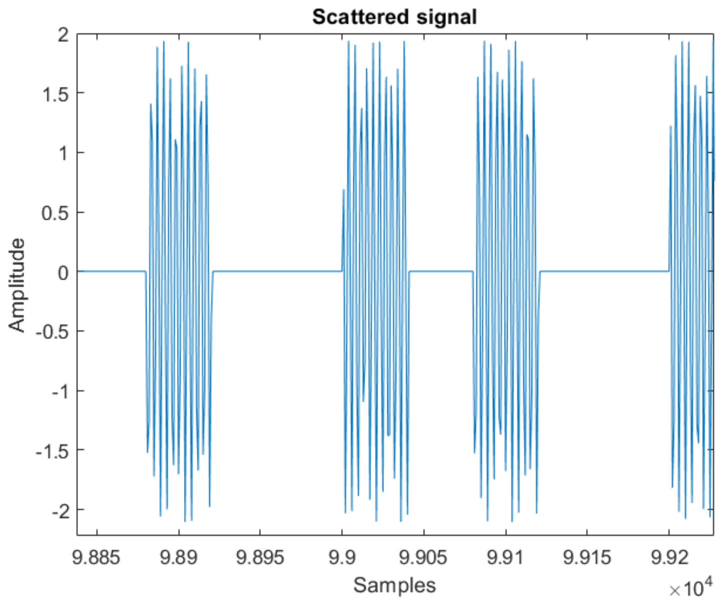

Figure 3.6: Amplitude-Modulated Signal

3.2.3 Receiver Envelope Follower

The signal from Figure 3.6 is scattered towards the receiver. Of course, the receiver also receives the CW, and interference and / or noise is also added. This signal that is received at the antenna needs to be demodulated. That is: the CW needs to be removed from the signal and the signal should be shifted back into baseband. Often, this is done with a mixer-based approach. Such a mixer however contains many power consuming, active components. Therefore, a simple passive alternative is used: the Diode Detector.

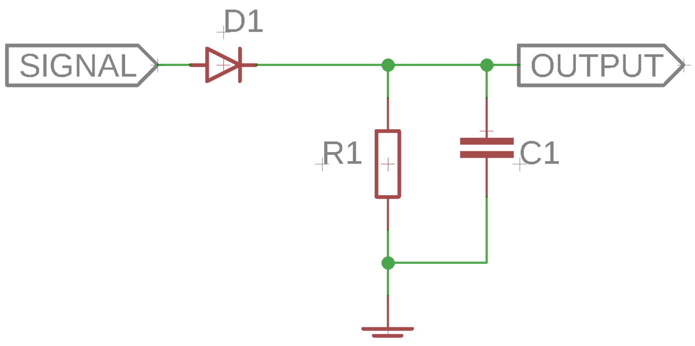

Figure 3.7: Diode Detector Schematic

Figure 3.7 shows the schematic of a Diode Detector. This Envelope Follower automatically demodulates the strongest frequency received. This requires that the CW is the strongest signal present. This is not a problem since a strong exciter signal is often present. In the simulation this is also assumed.

Figure 3.8 shows the behavior of the Envelope Follower. The blue line is the received signal, which is the sum of the CW (from the exciter) and the ASK signal. The orange line is the detected envelope.

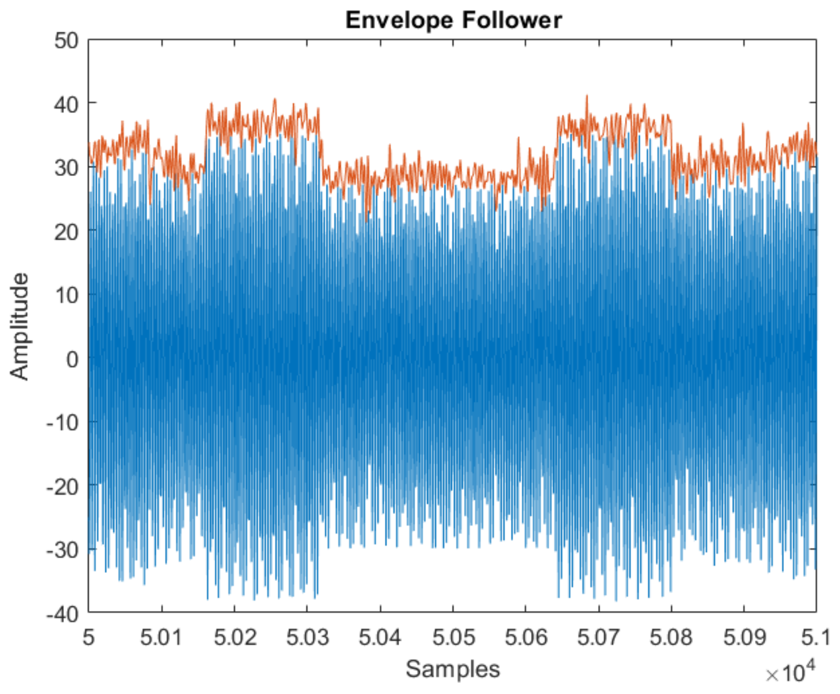

Figure 3.8: Envelope Follower

The output of the Envelope Follower is what needs to be converted to the digital domain, i.e. quantized, and despreaded.

3.2.4 Receiver Despreading

For despreading, cross-correlation can be used in case the system is not synchronized. The cross-correlation of two vectors measures similarity between the first vector and a (shifting version) of the second vector. Listing 3.2 shows a Matlab snippet that implements despreading using cross-correlation.

```
function despreadBitStream = undsss(recvSignal, pnSeq, thres)
    pnSeq = pnSeq * 2 - 1;
    xc = xcorr(recvSignal, pnSeq);
```

```
 4      posPeak = islocalmax(xc);
 5      negPeak = islocalmin(xc);
 6      peaks = [];
 7      for i=1:length(xc)
 8          if posPeak(i) | negPeak(i)
 9              peaks = [peaks xc(i)];
10          end
11      end
12      despreadBitStream = imbinarize(
          removeElementsBetweenBounds(max(peaks)*thres, min(
          peaks)*thres, peaks));
13  end
```

Listing 3.2: Matlab Cross-Correlation Code

At line 2, the PN sequence is scaled from $[0, 1]$ to $[-1, 1]$. At this point, the incoming signal is already quantized. On line 3 is the actual cross-correlation performed. This is a built-in Matlab function. It returns a series of correlation peaks. By scanning for these peaks, the optimal alignment between the incoming signal and the PN sequence can be found (each peak represents a '1' or '0'). This is done on line 4 and 5. Line 6 declares an empty array, which will contain the positive and negative peaks. This array is filled using a recursive loop on the lines 7 till 11. Finally on line 12, weak correlations are removed and the correlation peaks are scaled to binary. To illustrate the code above, Figure 3.9 shows the output from line 2.

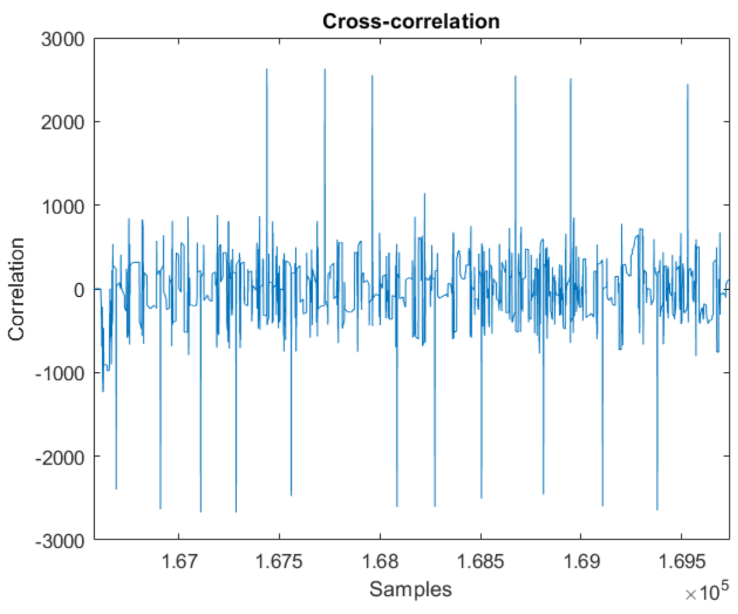

Figure 3.9: Cross-correlation output

Unfortunately, this operation requires a lot of computations. Because a synchronized system is assumed (as this can be solved at the MAC-layer), these computations can be greatly simplified to a simple multiplication. This multiplication can be highly optimized when implemented in an FPGA, because the only possible values of the PN sequence elements are -1 or 1. This means that this multiplication can be implemented with a simple two's complement calculator. Listing 3.3 shows a Matlab snippet of despreading in a synchronized system.

```
1 pnSeqUpsampled = upsampleBitStream(pnSeq, (N / length(
    bitStreamDSSS)) * SF);
2 pnSeqUpsampled = pnSeqUpsampled * 2 - 1;
3 rx = rxEnv .* repmat(pnSeqUpsampled, 1, nBit);
4 result = bitDetector(rx, length(pnSeqUpsampled));
```

Listing 3.3: Matlab Despreading Code

At line 1, the PN sequence is stretched to match the duration of a bit (because the transmitter does the same). Then on line 2, it's scaled from [0,1] to [-1,1]. At line 3, the received signal is multiplied with a repeated version of the PN sequence (because a bit is encoded with either the PN sequence or an inverted PN sequence).

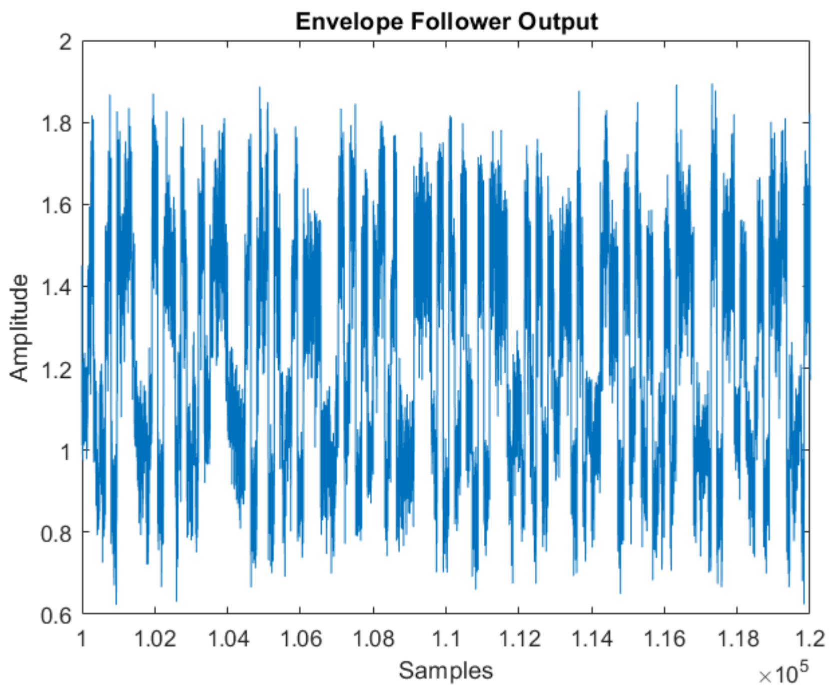

Figure 3.10: Envelope Follower Output

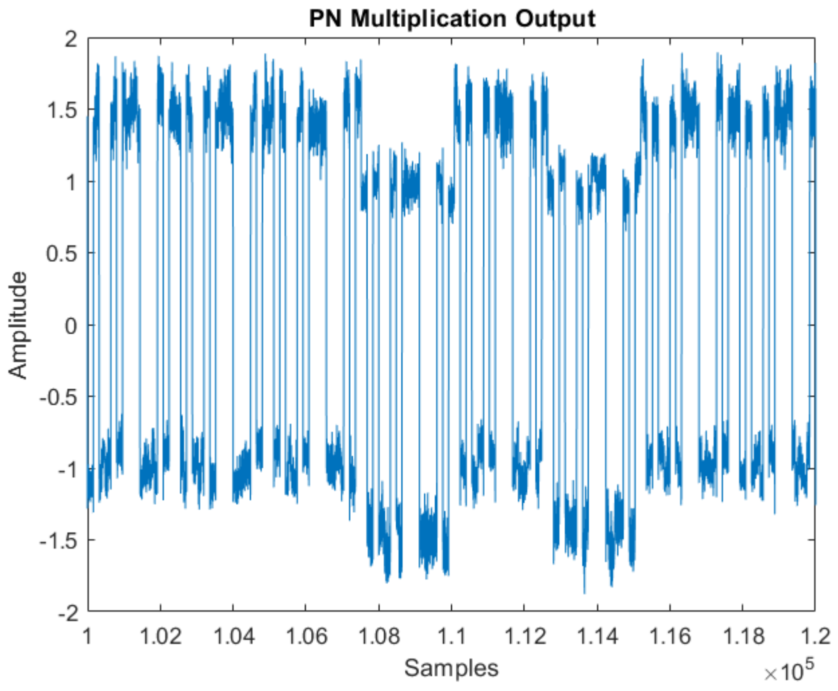

Figure 3.11: PN Multiplication Output

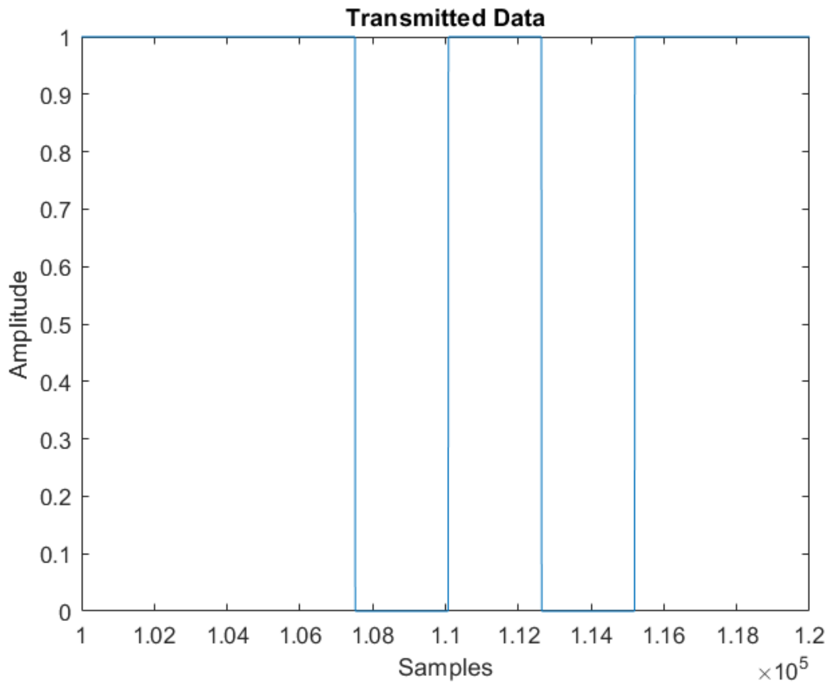

Figure 3.12: Originally transmitted data

To visualize this, Figure 3.10 shows the incoming signal (which is the rxEnv variable in Listing 3.3). Then Figure 3.11 shows the signal after multiplying with the PN sequence. In the latter figure, the shape of the original bit stream (Figure 3.12) becomes already visible. When comparing Figure 3.11 and Figure 3.12, it can be observed that 1s result in a higher mean that 0s. So by integrating the signal and comparing the mean, the original data can be recovered. The amplitudes in Figure 3.8 are quite different from Figure 3.10, this is caused by a difference in SNR. Note that in the figures above, no quantization is done yet. The effects of quantization will be described in section 5.1. The bit detector function from Listing 3.4 performs the integration step mentioned above.

```
 1 function z = bitDetector(x, window, offset)
 2     z = [];
 3     j = 0;
 4     while j < ceil(length(x)/window)
 5         bit = 0;
 6         for i = (1 + (window * j)):(window * (j+1))
 7             bit = bit + x(i);
 8         end
 9         z = [z bit/window];
10         j = j + 1;
11     end
12     z >= mean(z);
13 end
```

Listing 3.4: Matlab Bit Detector Code

Line 2 declares an empty array to contain the result and line 3 declares an iterator variable. The loop on line 4 loops until the number of (to be received) bits is reached. The window variable is essentially the bit length. On line 5 an intermediate variable is declared that holds the sum of all samples in the current window. The loop starting on line 6 loops over every sample in the window and line 7 adds the current sample value to the intermediate sum. Line 9 stores the average window value into the result vector. Line 10 then increments the iterator. The resulting vector is binarized on line 12 (so it can only contain '1' or '0' as elements). This result is an estimate of the originally transmitted data, which can be used to calculate the BER.

This concludes the description of the Matlab simulation. In the next chapters, the simulation results will be presented, starting with the robustness against noise.

Chapter 4

Simulation Results

4.1 Robustness against noise

In this section we look at how DSSS helps with robustness against noise. Following the theory from section 3.1, not much coding gain is expected when increasing the spreading factor. To test this hypothesis, a Matlab simulation was written as explained in chapter 3. This simulation simulates a AWGN channel.

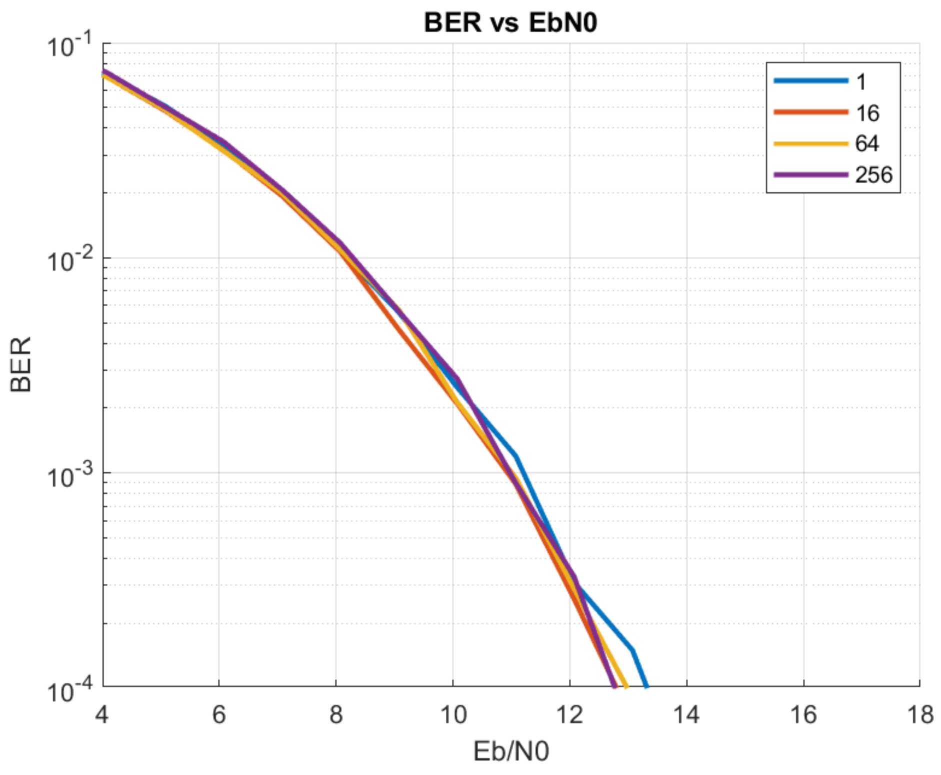

Figure 4.1: E_b/N_0 with AWGN

Figure 4.5 shows the BER plots resulting from the simulation. The BER is at the vertical axis, the E_b/N_0 on the horizontal axis. The different plots in the graph are the results from spreading with a different factor (1, 4, 64 and 256 times).

Now when we look at the differences in E_b/N_0 at a BER of 10^{-3} between the different spreading factors we can observe that the BER does not improve significantly when increasing the spreading factor. This observation matches with the hypothesis above. The tiny differences are likely because a finite number of bits is simulated (and the data and PN sequences are generated randomly).

4.2 Robustness against Interference

This section takes a look at how DSSS can improve the robustness against interference from other wireless transmissions or from the CW.

4.2.1 Interference from wireless transmissions

Figure 4.2 and Figure 4.3 show the Fast Fourier Transform of the relevant signals: the signal spreaded with DSSS and a Binary Phase-Shift Keying (BPSK) encoded interferer.

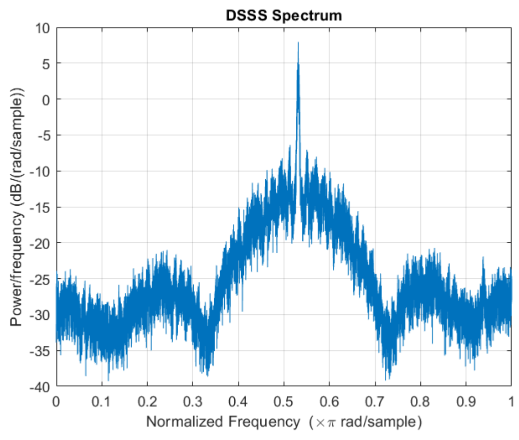

Figure 4.2: DSSS Spectrum

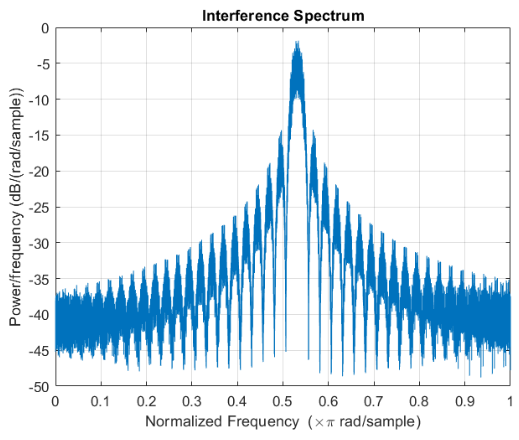

Figure 4.3: Interferer Spectrum

BPSK is a modulation type where the phase of the carrier wave is shifted 180 degrees whenever there is a transition in the data stream. Figure 4.4 shows this in schematic form. The general formula for a BPSK-modulated signal is

$$s_n(t) = \sqrt{\frac{2E_b}{T_b}} \cos\left(2\pi f_c t + \pi(1-n)\right), \tag{4.1}$$

where E_b is the energy-per-bit, T_b is the bit duration, f_c is the frequency of the carrier and n is the binary data and can be either 0 or 1 [15].

The bandwidth of the interferer in the simulations in this thesis and in Figure 4.3 is 32 times the bandwidth of the unspreaded signal. The signal in Figure 4.2 is spreaded with a factor of 256. The figures are not matched in power.

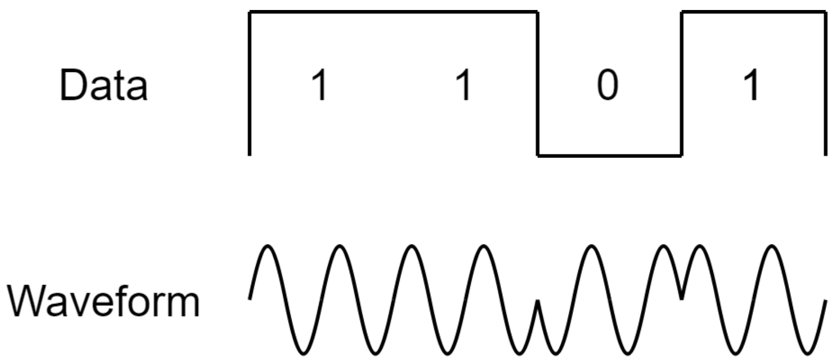

Figure 4.4: BPSK Modulation

These signals are combined and fed into the receiver simulation. Notice that this simulation deals with the worst-case scenario: the interferer has the exact same center frequency as the signal so there is maximum overlap.

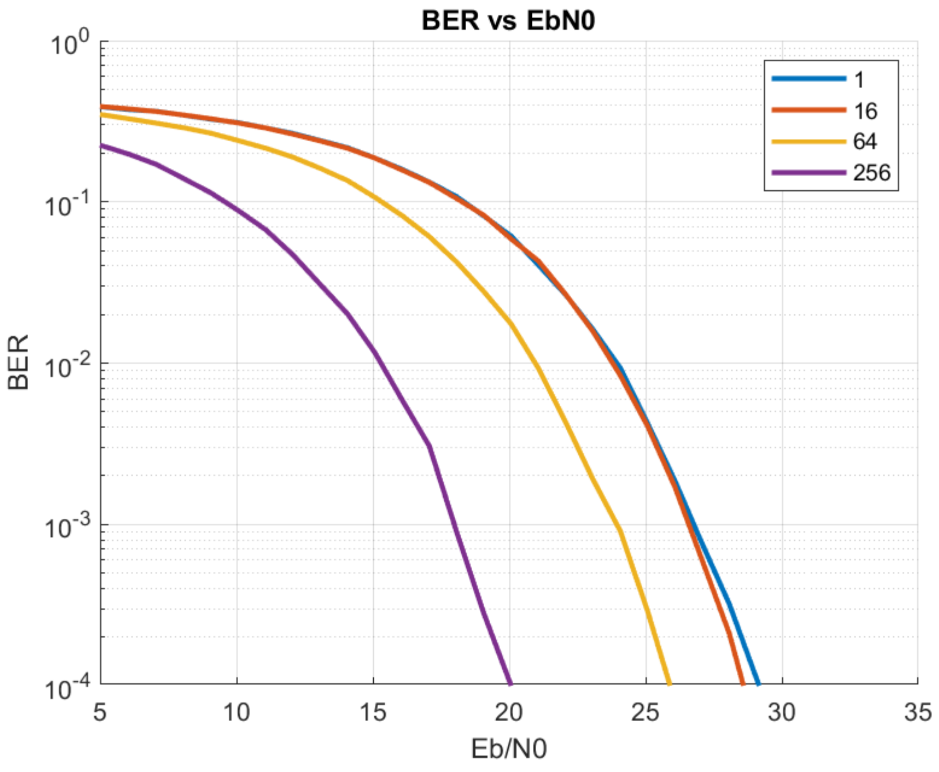

Figure 4.5: E_b/N_0 with BPSK Interferer

Figure 4.5 shows the BER plots resulting from the simulation. The BER is at the vertical axis, the E_b/N_0 on the horizontal axis. The different plots in the graph are the results from spreading with a different factor (as shown in the legenda).

Based the differences in E_b/N_0 between the different spreading factors, it can be observed that the BER does not improve when the spreading factor is increased from 1 to 16. An improvement of around 3dB can be observed when going from 16 to 64. Going from 64 to 256 yields an increase of around 6dB.

As mentioned above, the interferer had a bandwidth of 32 times the bandwidth of the signal. This explains why the results only improve in case the spreading factor is larger than 32, which is the expected result based on the hypothesis from chapter 3, where it was mentioned that doubling the spreading factor corresponds to a coding gain of 3dB.

4.2.2 Interference from Continuous Wave

To research robustness against self-interference from the Exciter, a non-ideal CW was used. The CW is wider (occupies a larger frequency band), has harmonics and a white noise floor.

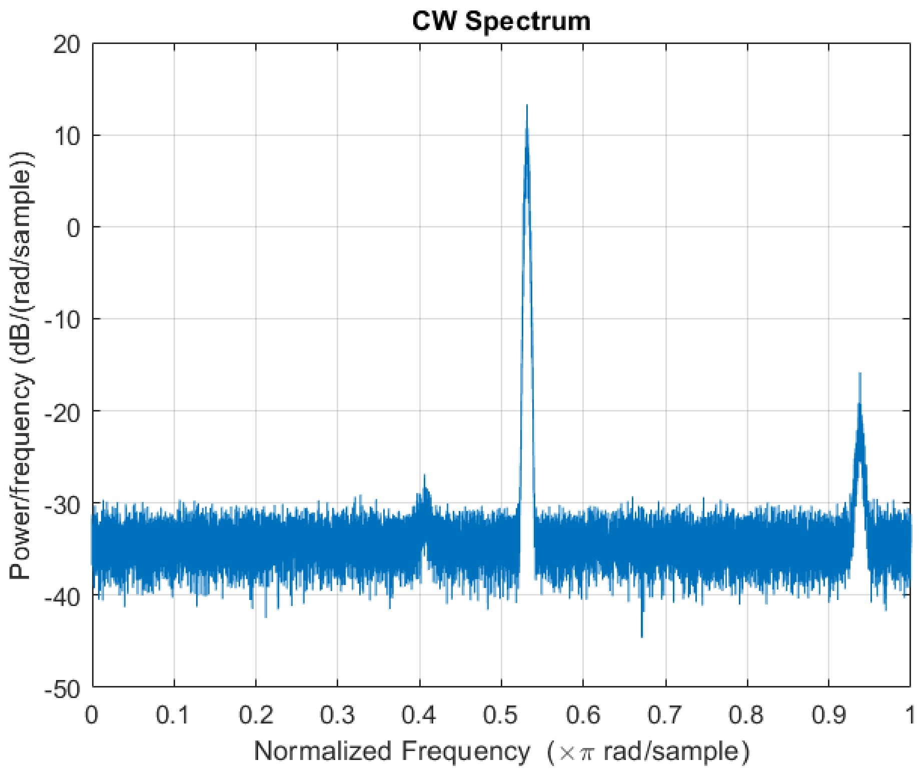

Figure 4.6: CW Spectrum

The spectrum of the CW used in the simulations from this section (and next chapter) is shown in Figure 4.6.

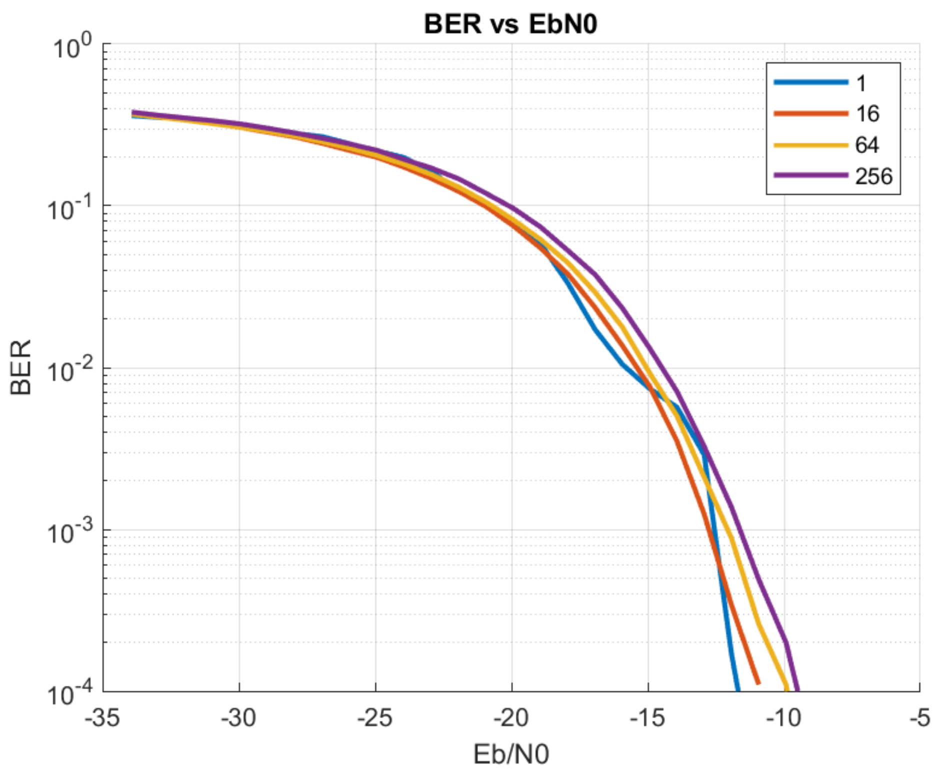

Figure 4.7: E_b/N_0 with CW

The result of this simulation is shown in Figure 4.7. The y-axis represents the BER, the x-axis is representing the E_b/N_0. When looking at the Point of Interest (BER of 10^{-3}), the curve for N=1 is not very smooth. Also, for an increasing Spreading Factor, the performance is slightly worse. It is unclear what exactly causes this. The good BER performance indicates that the system is already quite robust against self-interference, demodulating already removes most of the CW.

Figure 4.7 shows the results when the CW is in-phase with the signal. Something different can be observed when phase cancellation occurs. Phase cancellation is a phenomenon in which combining two ”out-of-phase” waves results in a weakened or lost signal. In this case, the two waves are the spreaded signal and the CW. Figure 4.8 shows what happens when the CW is fully out-of-phase and has a higher amplitude than the signal. The transmitted signal (marked as *TX*) is summed together with the CW at the receiver which results in the signal marked as *RX*. Clearly, the received signal equals 0 when the transmitted data is a 1 and the other way around. This means that all bits are inverted.

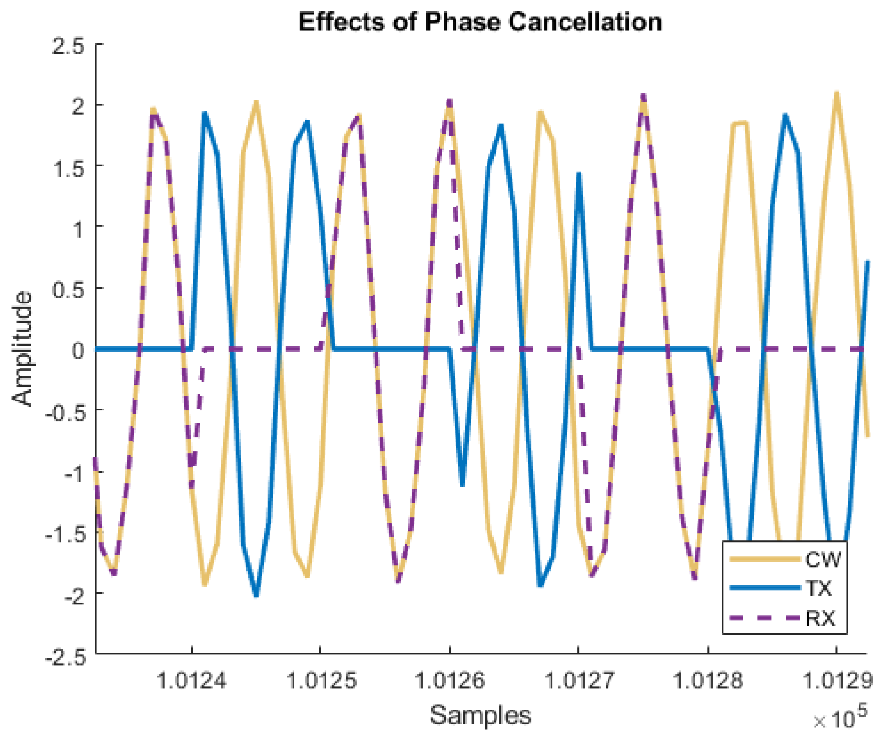

Figure 4.8: Effects of Phase Cancellation

The simulation results are shown in Figure 4.9. In this case, the CW is fully out-of-phase, which means that maximal phase cancellation is occurring.

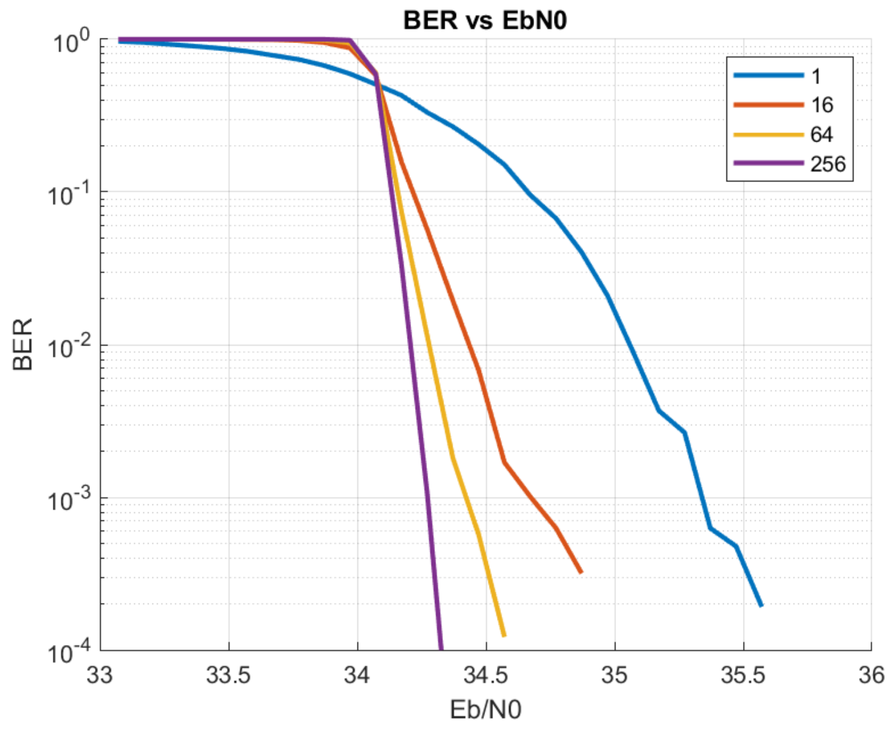

Figure 4.9: E_b/N_0 with CW out-of-phase

Figure 4.9 shows a clear dependence of the BER performance on the Spreading Factor, but there is only a gain of about 1dB when increasing the Spreading Factor from 1 to 256. Another important observation is that a much higher E_b/N_0 is needed to achieve the same BER, compared to Figure 4.7. The reason for this is the *bit inversion* explained above. This also explains why the BER is 10^0 for low E_b/N_0 values; all the bits are inverted. Apparently, the flipping point in this particular case is around an E_b/N_0 of ~ 34.1. For values higher than this, the signal is stronger than the carrier wave. The spreaded signals have steeper curves because there is less overlap with the CW. From this, it can be concluded that it is not very beneficial to use DSSS to increase the robustness against self-interference.

This chapter looked at the simulation results and the performance improvements that DSSS gives. The next chapter will present the performance of a practical DSSS implementation for Backscatter applications.

Chapter 5

Practical DSSS

In the previous chapter it is shown that DSSS does not improve robustness against AWGN, but it gives a significant improvement in robustness against interference from other wireless transmissions. This chapter provides an answer to research questions 3 and 4 from section 2.4 by researching the required number of quantization bits and proposing a new quantization method that samples the signal in an ULP fashion. Also, a different simulation framework is used to simulate the system at a more detailed level.

5.1 Sampling DSSS signals

To recap, Figure 5.1 shows the block diagram of the receiver. In this section, an implementation for the *Quantizer* block is proposed.

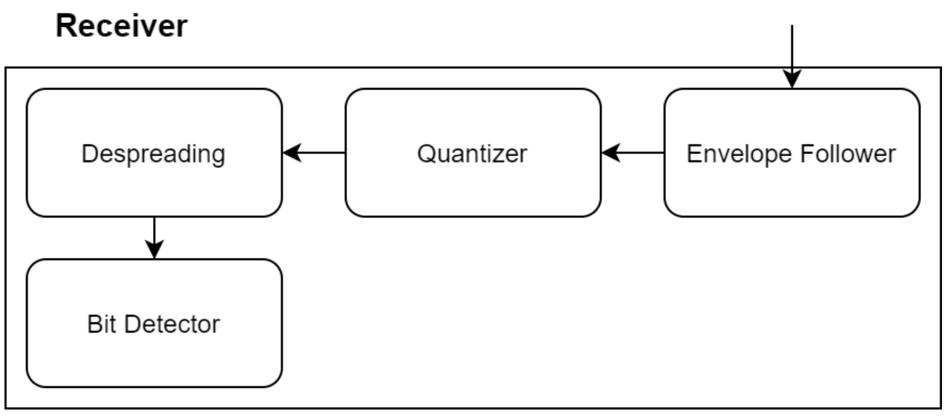

Figure 5.1: Matlab Simulation Receiver

Figure 5.2 shows what the signal (that needs to be quantized) can look like. In this case, it is the signal mixed with the CW and a BPSK interference.

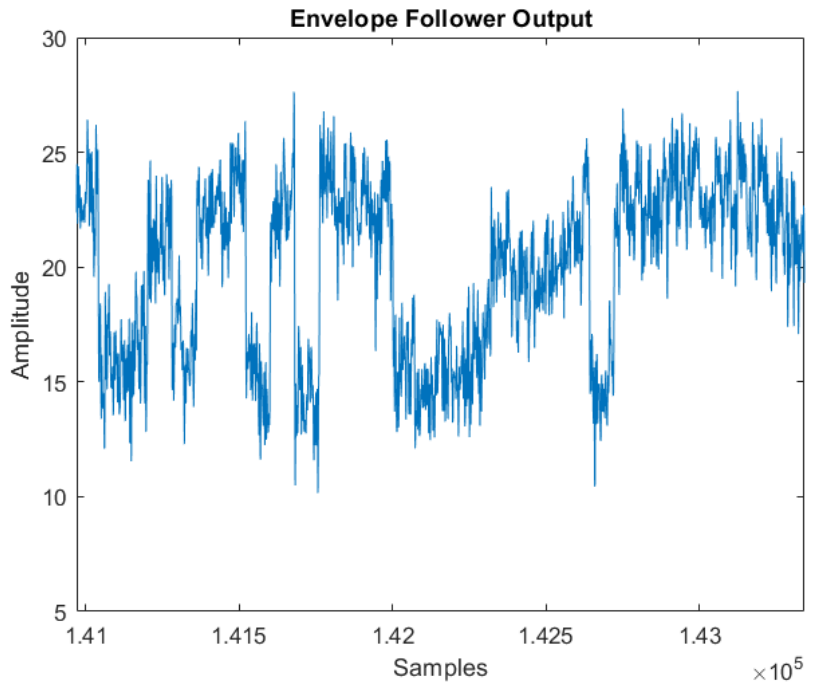

Figure 5.2: Signal to be quantized

State-of-the-art ADCs can provide a high quantization accuracy or sampling speed but require a lot of power and are therefore not practical to use in a Backscatter system.

To avoid this, Liu et al. [11] propose a quantization method that is based on a 1-bit comparator. This method was also used by De Jong [9]. The justification for this is that the surrounding RF-signals have a much higher bit rate than backscatter signals. By averaging the received signal, the variations in power (resulting from the ambient RF-signals) are removed, allowing the backscattered signal to be decoded. But, the bit detector from Listing 3.4 is an averaging-based mechanism. So with more bits, the averaging result will be more accurate and noise will cause less quantization errors. Hence this thesis proposes a new quantization method that is an expansion of the 1-bit (comparator-based) quantizer commonly seen in other Backscatter systems.

Shi et al. [18] compare a 1-bit quantizer with a high-resolution quantizer and observe a 2dB loss in E_b/N_0 for the 1-bit quantizer. In their case they use BPSK modulation (as opposed to ASK in this thesis) and two 8-bit ADCs for sampling both In-phase and Quadrature components. For an explanation of In-phase and Quadrature Signals, refer to [6]. They simulate their system with an AWGN channel. In addition, Namgoong [23] concludes that 3-bit quantization already approximates ideal sampling, under the assumption that the Automatic Gain Control (AGC) is done perfectly. Ouvry et al. [16] emphasizes the need for a good AGC and note that more sampling resolution is needed when the headroom of the ADC is not used in full. Finally, Wu et al. [8] show that using more than 4 quantization bits barely improves BER performance, but the results show that 3 bits are also already very close, and this is also the resolution used by numerous DSSS Modem chips [4].

To summarize, the expected result is that the ideal number of quantization bits is 3-4 bits, and the observed improvement in BER performance will probably be around 2dB compared to 1-bit quantization.

5.1.1 Histogram Analysis

Figure 3.8 shows the signal (in orange) that needs to be quantized. It can be observed that the Envelope Follower only outputs a positive signal (so it does not get below 0), which is an important property when designing a quantizer. More properties can be derived by looking at the histograms. The number of occurrences is positioned at the y-axis, the signal amplitude at the x-axis.

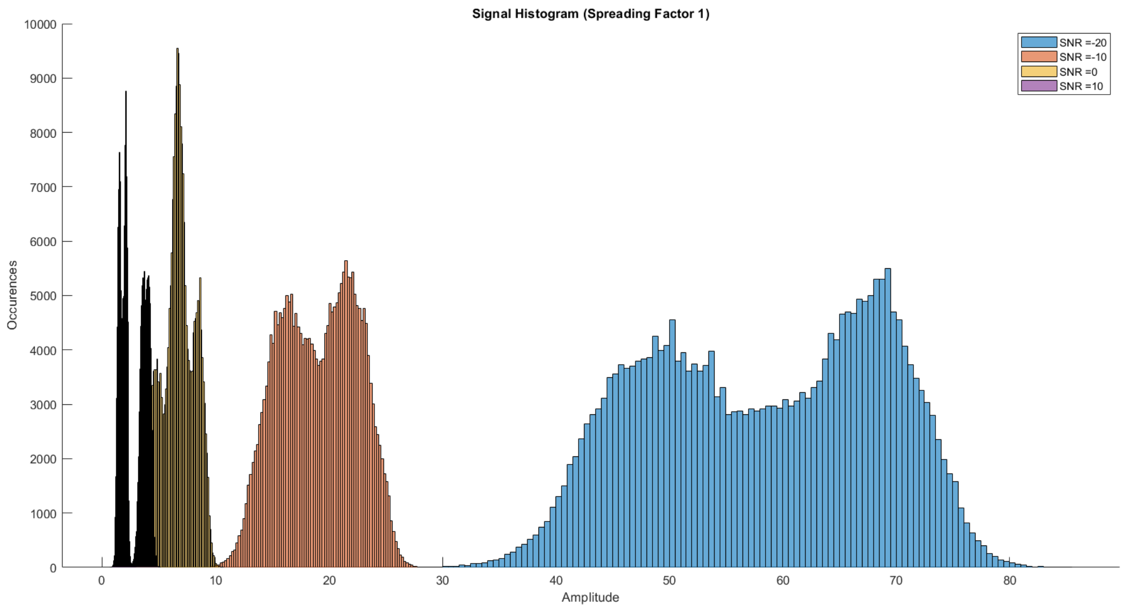

Figure 5.3: Signal Histogram (Spreading Factor 1)

In Figure 5.3 a histogram of the signal (plus added noise or interference) is shown with the Spreading Factor being 1 (so no spreading is being done). Multiple plots are presented with increasing SNR values from right to left. Both the CW and the interferer are reduced in power, but it is ensured that the CW is stronger than the interferer as this is a requirement for the demodulation to happen correctly. In this section it is assumed that the RMS is always 12dB higher than the interferer. An observation is that the mean is scaling along with the amplitude such that the mean-amplitude ratio in this communication system is more or less constant. This raises the hypothesis that the quantization bounds can be derived from the mean. This hypothesis will be explored in the remainder of this section.

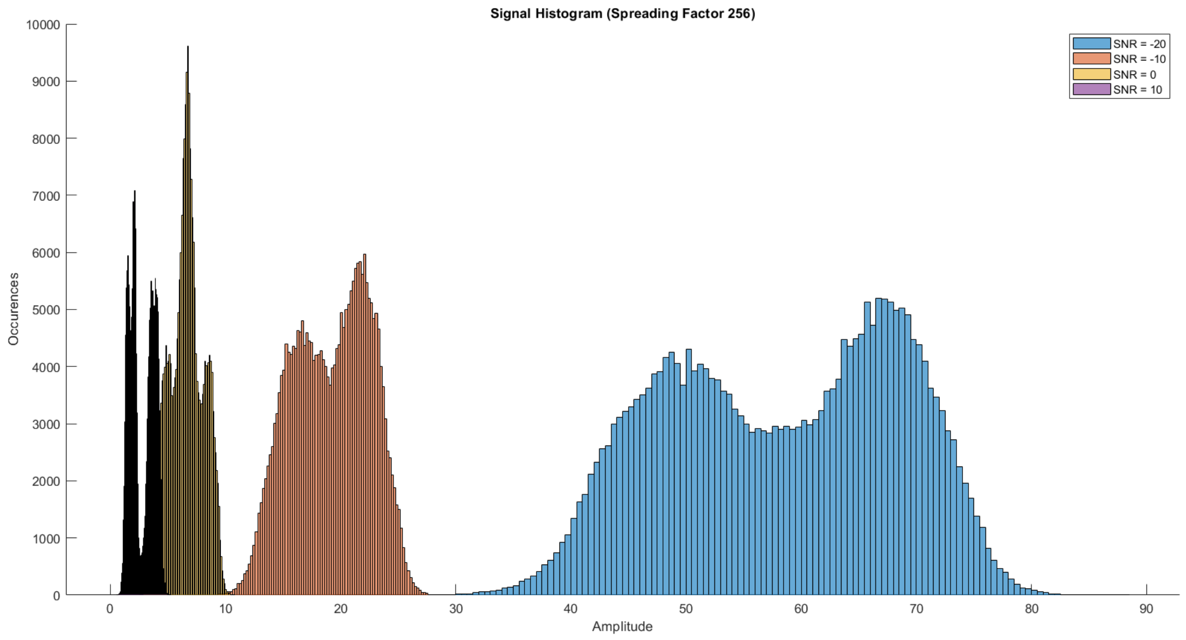

Figure 5.4: Signal Histogram (Spreading Factor 256)

Figure 5.4 shows the histograms when the Spreading Factor is increased to 256. The histograms do not change in shape which is logical since the amplitude is not changed, only the chip rate (and hence the number of occurences). Let's compare two of these histograms. Figure 5.5 shows a histogram of a signal with poor SNR, Figure 5.6 shows a histogram with a good SNR. In both cases the signal is spreaded with a factor of 256.

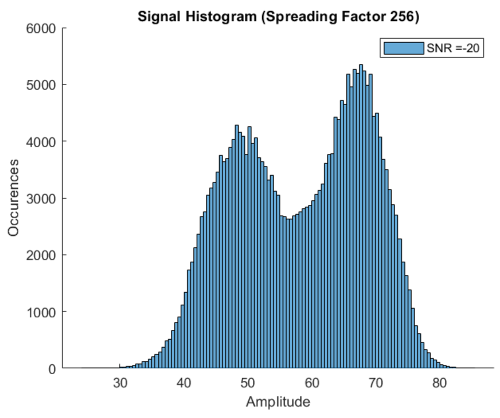

Figure 5.5: Histogram with SNR=-20

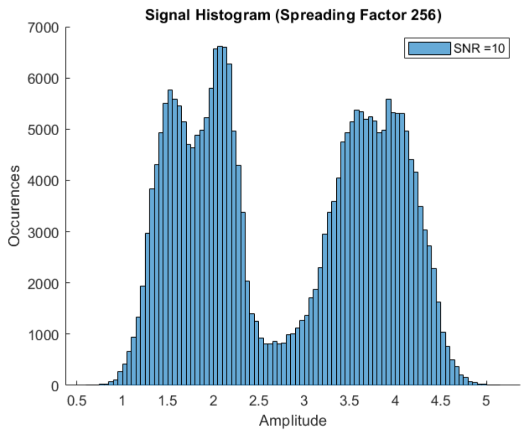

Figure 5.6: Histogram with SNR=10

We can observe that the shape is changing, and for higher SNR we see a clear peak that is caused by the CW. However, the ratio between the bound and the mean seems to remain more or less constant.

5.1.2 Quantization Proposal

The hardware design introduced by Liu et al. [11] and re-used by De Jong [9] uses a 1-bit comparator and a analog low-pass (averaging) filter for the comparator threshold. This is a simple, ULP solution and allows the software to work at signal edges rather than fixed-period sampling. In the previous subsection a hypothesis was posed that there is a relation between signal mean and amplitude. This section proposes a method to sample such a signal in an ULP fashion and is an extension of the 1-bit comparator. This proposal is then used to determine if a higher resolution ADC results in a better BER performance.

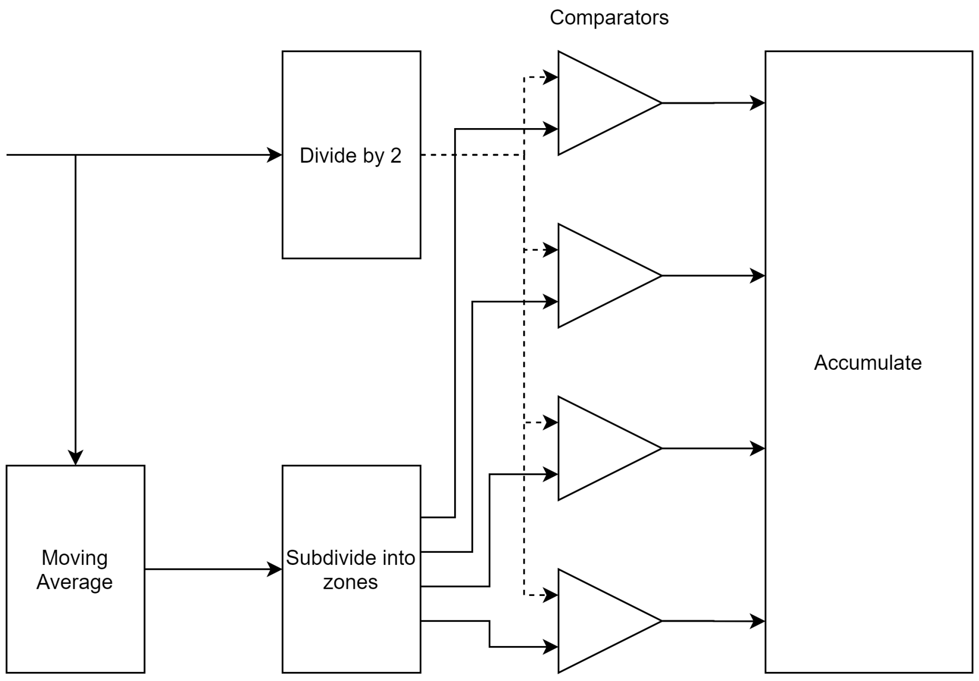

Figure 5.7: Quantization Block Diagram

Figure 5.7 shows the block diagram of the proposal. In Figure 3.8 we saw that the envelope follower demodulates the signal and removes the CW. Before looking at the histograms, we already established that a side-effect of this envelope follower is that the signal has a DC-offset and is always above 0. This signal is the input on the left side. Conventional ADCs often use DC-blocking capacitors and sample with the signal centered around 0 (or a fixed offset) and use a Programmable Gain Amplifier (PGA) to maximize resolution. However, a PGA is an active component and adds additional power consumption.

This thesis proposes to use adaptive thresholds instead of sampling around a fixed point. Because the signal bounds seem to be related to the mean, the bounds can be easily derived from the RC-filter output using simple voltage dividers. This is also reflected in Figure 5.7 where the signal is fed into a *Moving Average* block (which is an RC-filter), followed by a *Subdivide into zones* block. The area between the bounds can be divided into additional bins to sample at a higher resolution. The signal is also attenuated such that only division is needed. Figure 5.7 displays this: the signal is attenuated by a factor 2 and the comparators sample the signal based on the thresholds supplied by the *Subdivide into zones* block. To illustrate the formulas below, Figure 5.8 shows the concept schematically.

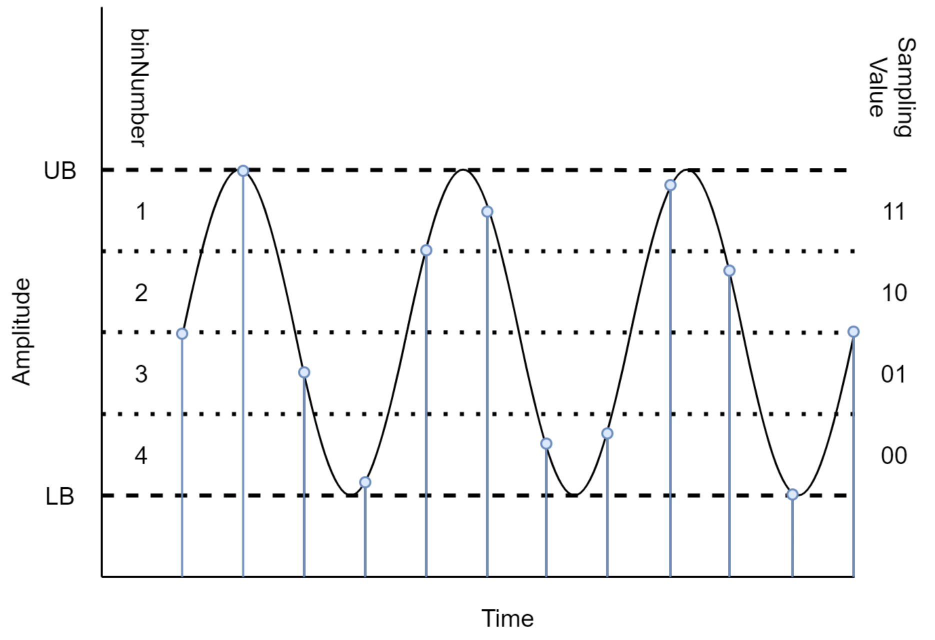

Figure 5.8: Sampling Concept

The *mean-bound ratio* is defined as the ratio between the signal mean and upper bound of the quantizer. A *bin* is the spacing between two quantization thresholds (the horizontal lines in Figure 5.8). The *mean-bound ratio* is constrained to the interval $(0.5, 1]$ (meaning that it has to be higher than 0.5 but can be equal to 1) so that the upper bound is always larger than the lower bound and the lower bound never is negative. The outer bounds are calculated using the following equations:

$$upperBound = mean * meanBoundRatio \tag{5.1}$$

and

$$lowerBound = mean * (1 - meanBoundRatio), \tag{5.2}$$

where the $mean$ is the output of the averaging filter and meanBoundRatio is the *mean-bound ratio*. Then for every bin the quantization threshold is given by

$$threshold = upperBound - (upperBound - lowerBound) * \frac{binNumber}{numberOfBins}, \quad (5.3)$$

where $\frac{binNumber}{numberOfBins}$ is a fraction that determines the bin position and size. The $numberOfBins$ is given by $2^{sampling\ bits}$. The *mean-bound ratio* and the *bin* thresholds are determined using simulation. $binNumber$ indicates which *bin* is selected, starting from the lowest bin. $binNumber$ ranges between $[1..numberOfBins]$.

5.1.3 Determining the number of quantization bits

A three step method is used for determining the optimal bounds and optimal number of quantization bits. First a *mean-bound ratio* is selected and that ratio is used to determine the optimal number of quantization bits. Then, with that number of bits the mean-bound ratio is revisited to see if there is a better ratio that can be selected. Thirdly, with the new bounds, it is checked if the result from the first step is still the optimal number of quantization bits.

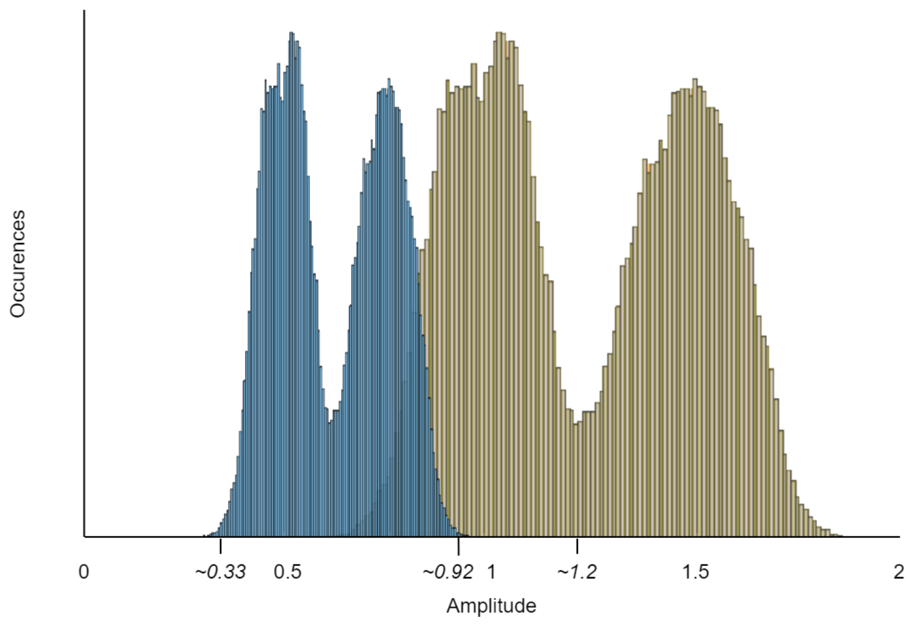

Figure 5.9: Mean-bound ratio derivation

As a starting point, a *mean-bound ratio* of 0.8 was chosen visually, based on the histogram in Figure 5.9. The yellow histogram is the original histogram. The SNR for this histogram is 10. The blue is the same signal but attenuated by 6dB. The estimated mean of the yellow histogram is ~ 1.2. The upper bound of the blue histogram is ~ 0.92. This gives an estimated *mean-bound ratio* of $\frac{0.92}{1.2} = \sim 0.8$.

With this starting point, the optimal number of bits was determined using simulation. The lines in Figure 5.10, 5.11, 5.12 and 5.13 represent the BER plots for spreading factors ranging from 1 to 256 and for varying number of quantization bits and with a BPSK Interferer. The lines marked as *Ideal* represent a perfect ADC with no quantization loss.

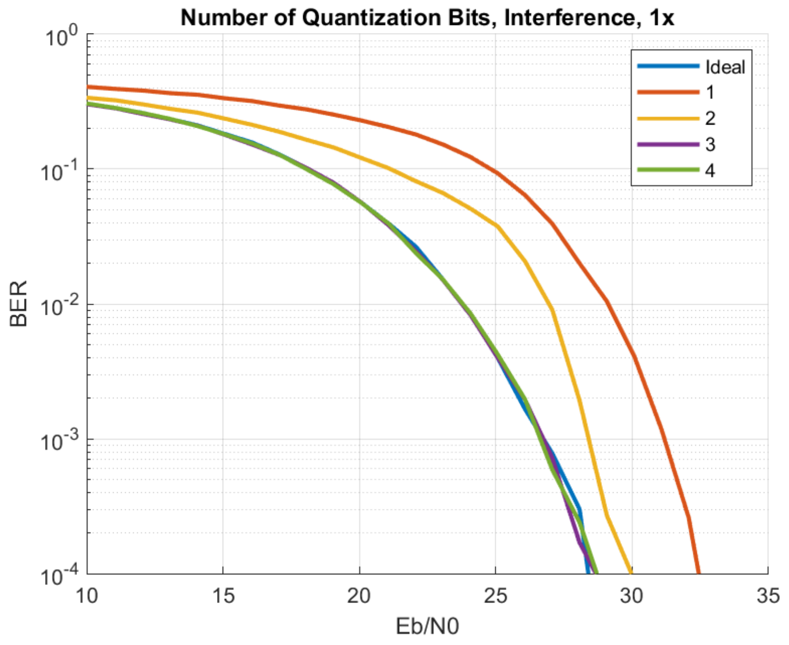

Figure 5.10: No. of Quantization bits, BPSK Interference, Spreading Factor 1

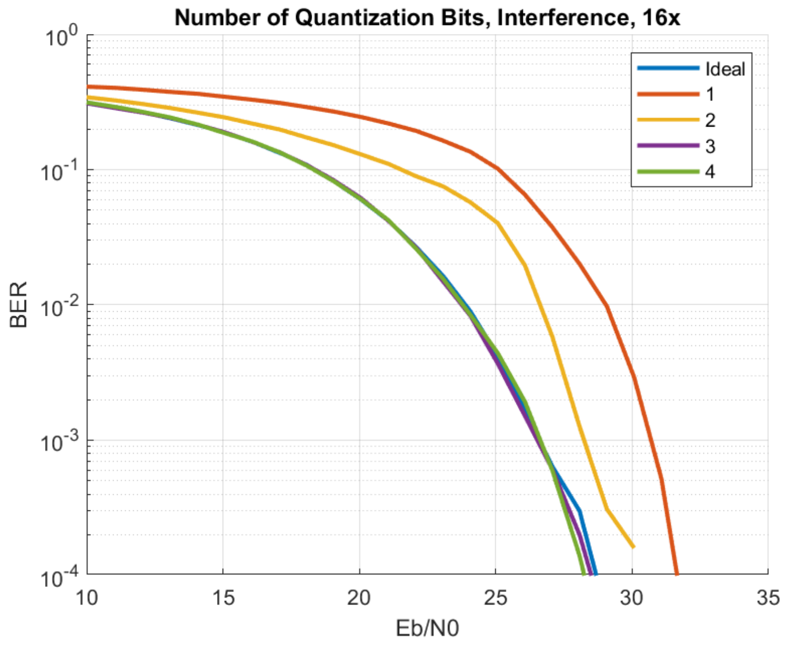

Figure 5.11: No. of Quantization bits, BPSK Interference, Spreading Factor 16

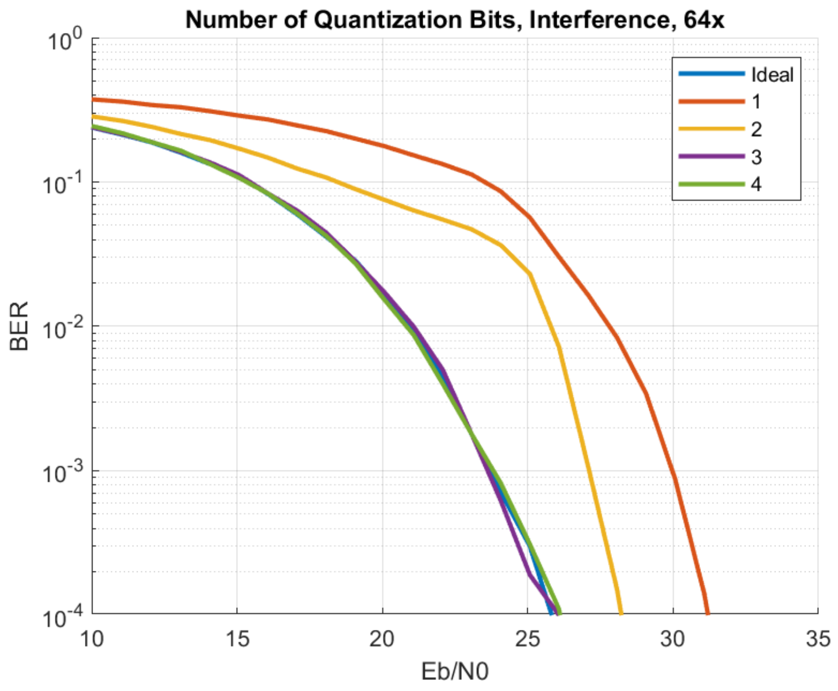

Figure 5.12: No. of Quantization bits, BPSK Interference, Spreading Factor 64

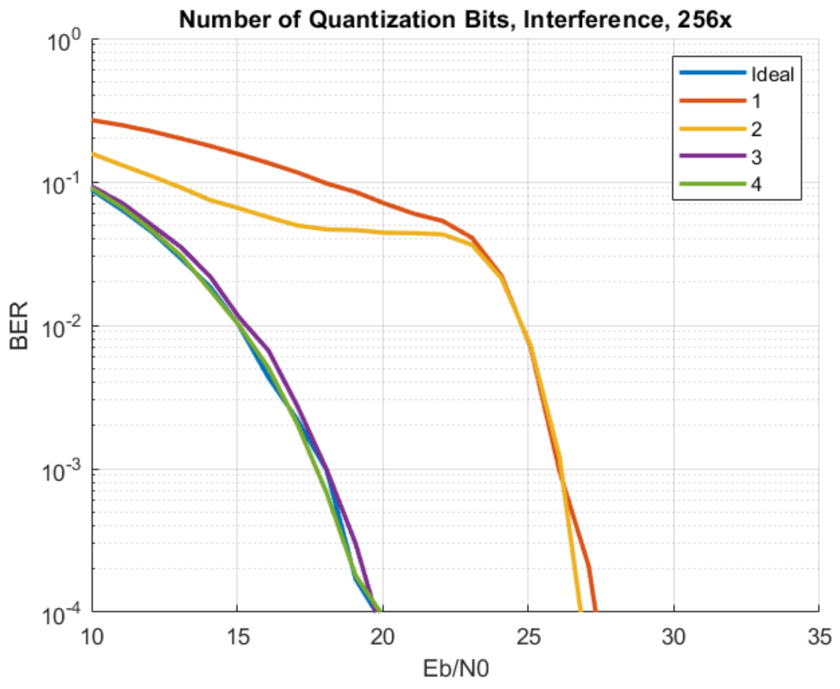

Figure 5.13: No. of Quantization bits, BPSK Interference, Spreading Factor 256

The behaviour of the 2-bit quantizer sticks out. For lower Spreading Factors, it performs somewhere in-between 1 bit and 3 bits. However, for a Spreading Factor of 256x, it does not perform any better than a 1-bit quantizer. It can be concluded that 3 quantization bits works quite well, as it is very close to the ideal line.

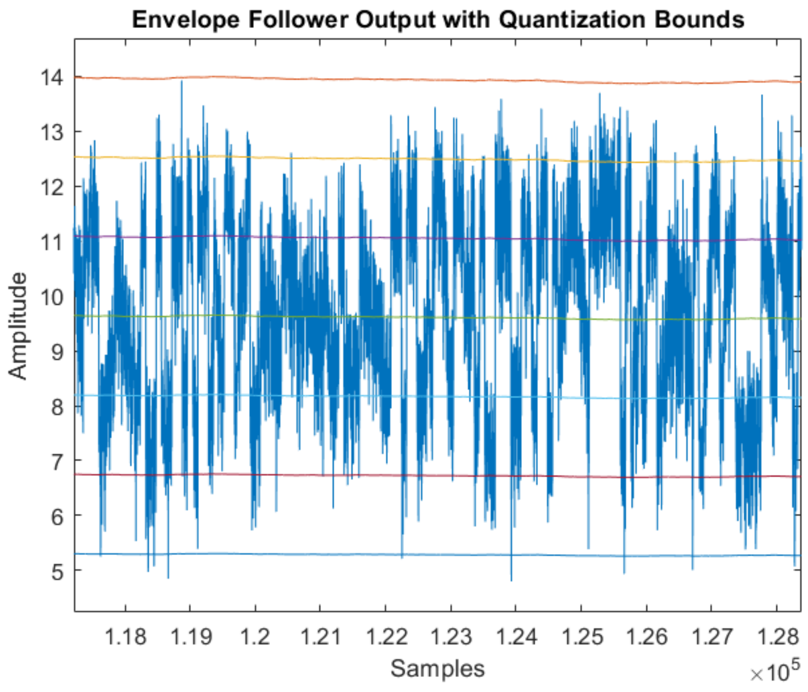

Figure 5.14: Envelope Follower Output with Quantization Bounds

Figure 5.14 shows the signal that needs to be quantized, wíth quantization thresholds for a 3-bit quantizer. The horizontal lines mark these thresholds. An observation is that it might be possible that the chosen *mean-bound ratio* is not optimal, as the signal barely comes above the upper or below the lower bound, although it seems quite a good choice already.

5.1.4 Determining the mean-bound ratio

This section will look into the second step, determining a better mean-bound ratio. The number of quantization bits is fixed at 3 for determining the optimal mean-bound ratio before revisiting it in subsection 5.1.5. Last subsection only simulated with a BPSK Interference, but in this subsection, the bounds are also simulated with AWGN to check if there is a bit difference between ratios.

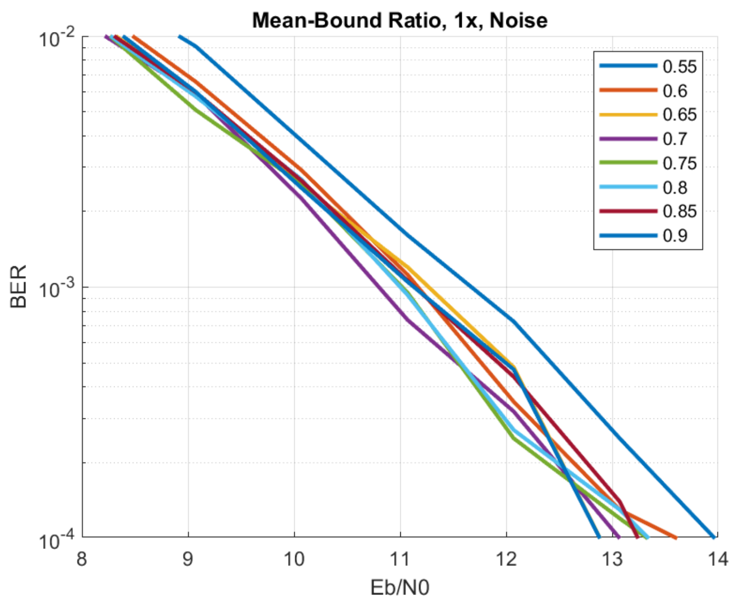

Figure 5.15: Mean-bound ratio, AWGN, Spreading Factor 1

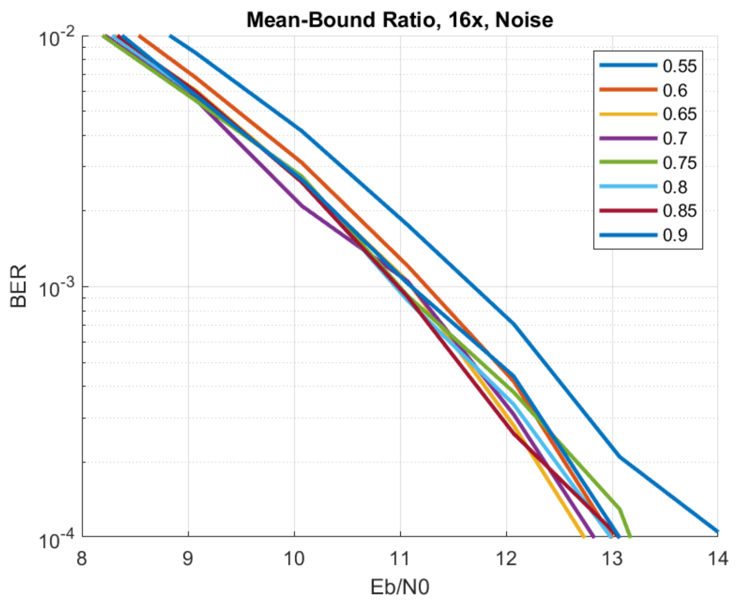

Figure 5.16: Mean-bound ratio, AWGN, Spreading Factor 16

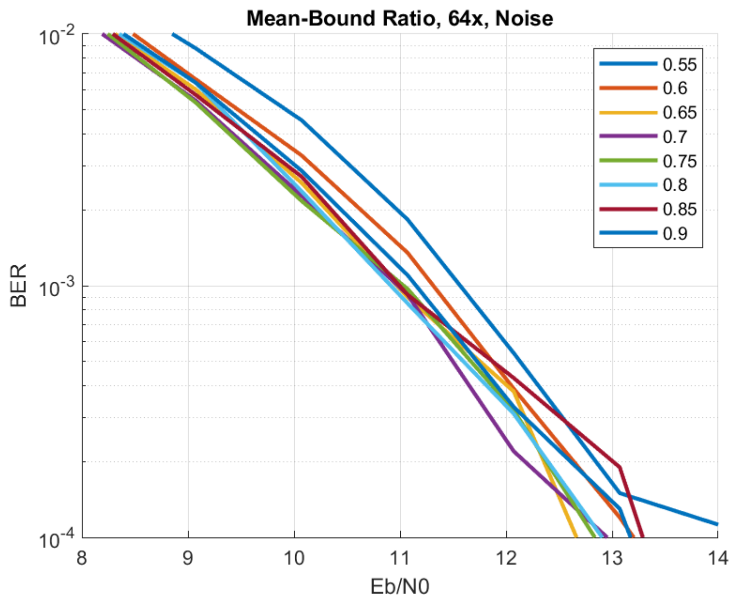

Figure 5.17: Mean-bound ratio, AWGN, Spreading Factor 64

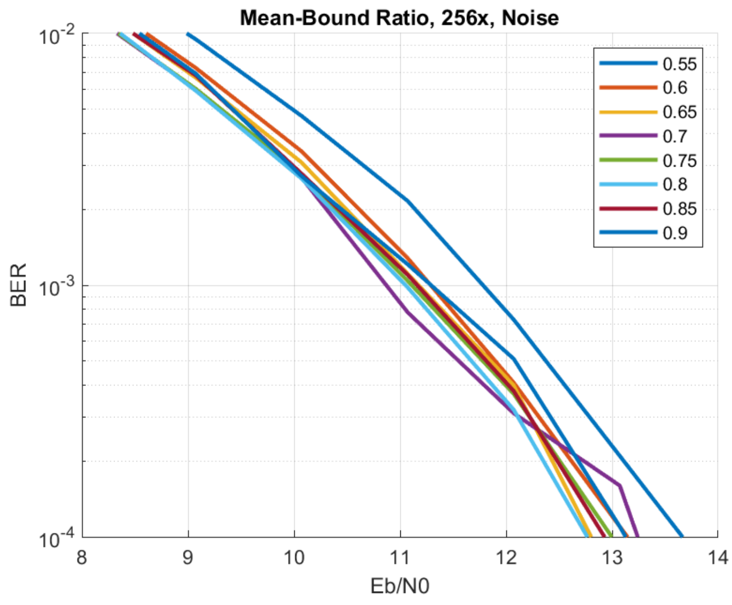

Figure 5.18: Mean-bound ratio, AWGN, Spreading Factor 256

Figure 5.15, 5.16, 5.17 and 5.18 show the results of the various mean-bound ratios, in case the interference is AWGN. Clearly, the selection of a poor bound does not cause a significant impact on the BER performance. The difference between the best ratio and the worst ratio is less than 1dB. Next, the simulation is repeated but with a BPSK interferer instead of AWGN.

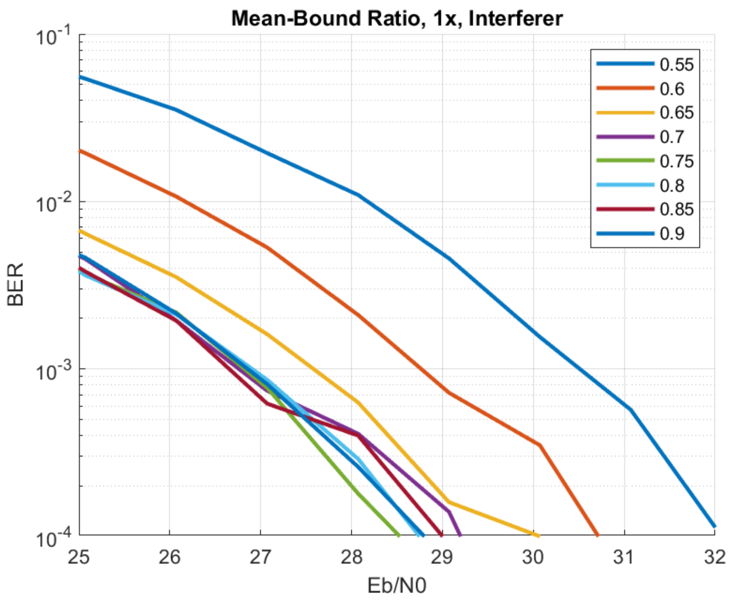

Figure 5.19: Mean-bound ratio, BPSK Interferer, Spreading Factor 1

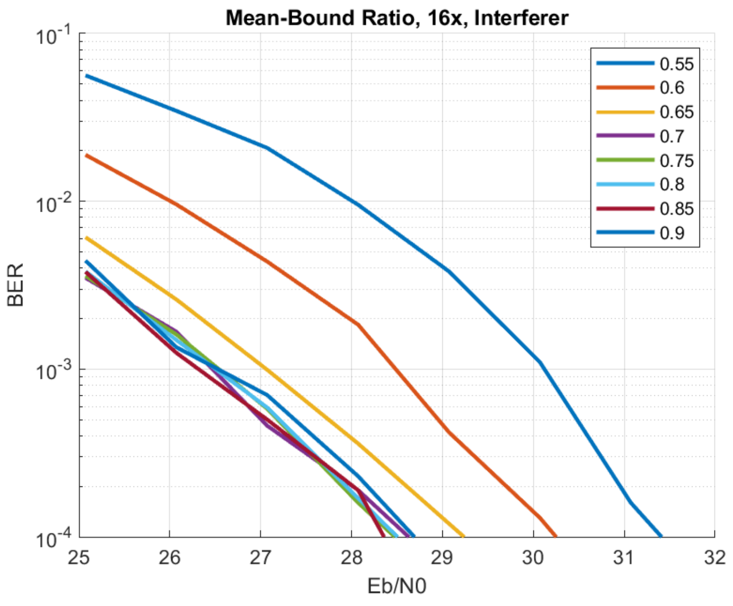

Figure 5.20: Mean-bound ratio, BPSK Interferer, Spreading Factor 16

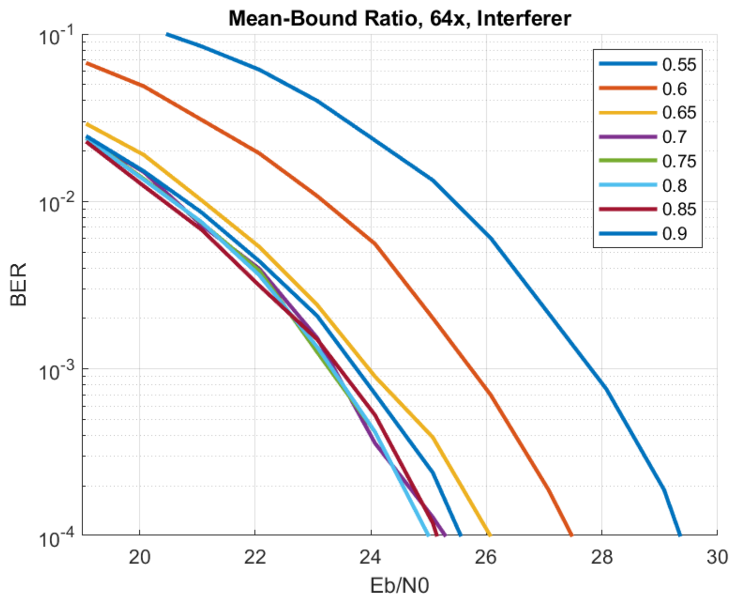

Figure 5.21: Mean-bound ratio, BPSK Interferer, Spreading Factor 64

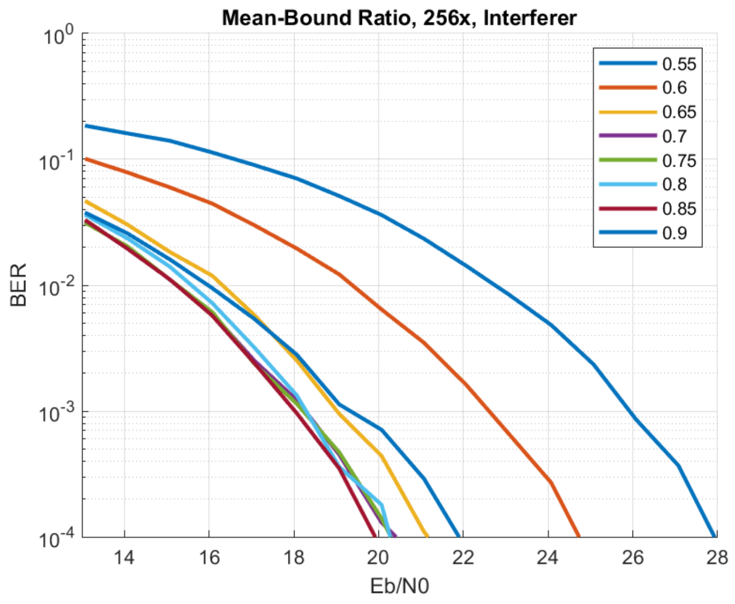

Figure 5.22: Mean-bound ratio, BPSK Interferer, Spreading Factor 256

The Figures 5.19, 5.20, 5.21 and 5.22 show the BER plots of the various mean-bound ratios, in case the interference is a BPSK interferer. Clearly, choosing a *mean-bound ratio* that is too low results in poor performance: the difference between a ratio of 0.85 and 0.55 is almost 8dB (with a Spreading Factor of 256). Another observation is that poorly chosen ratios don't benefit as much from increasing the Spreading Factor. It can be concluded that a ratio of 0.85 generally gives the best results, although it is a close call with 0.75 and 0.8.

5.1.5 Revisiting the number of quantization bits

The third step is verification: with the adjusted mean-bound ratio, it might be possible that a different number of quantization bits is more optimal so this simulation is ran again.

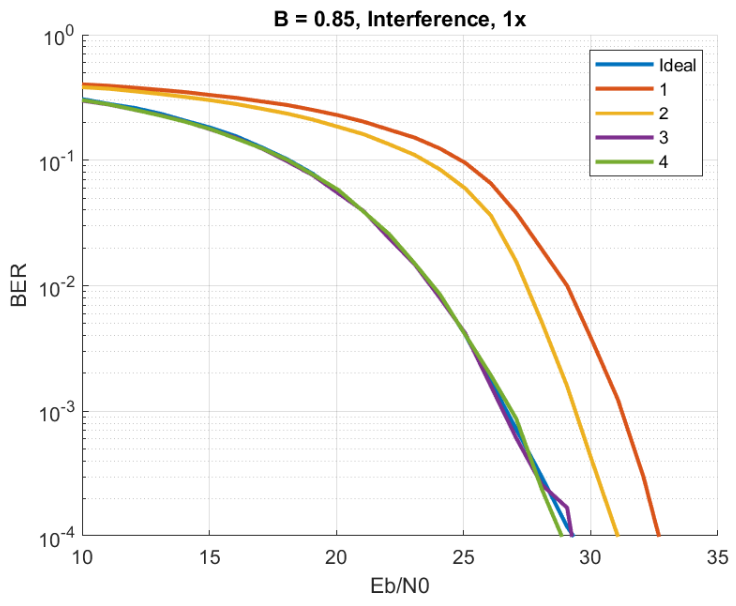

Figure 5.23: Mean-bound ratio = 0.85, BPSK Interferer, Spreading Factor 1

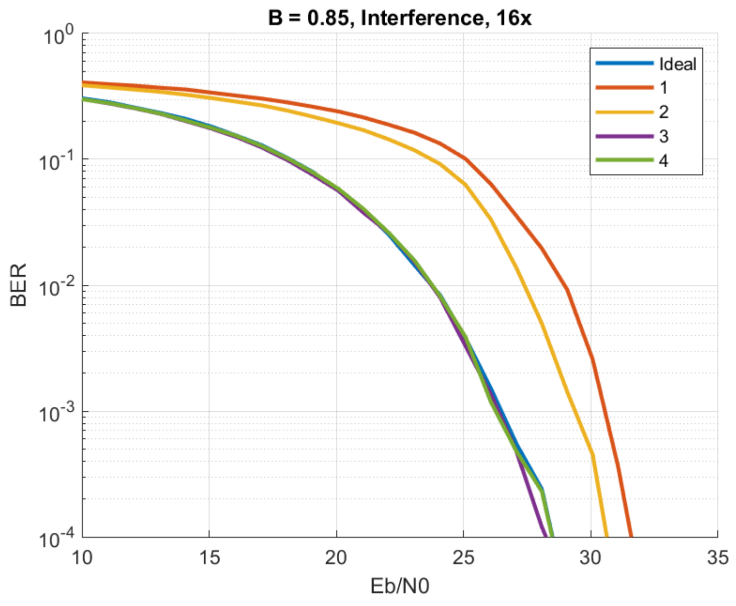

Figure 5.24: Mean-bound ratio = 0.85, BPSK Interferer, Spreading Factor 16

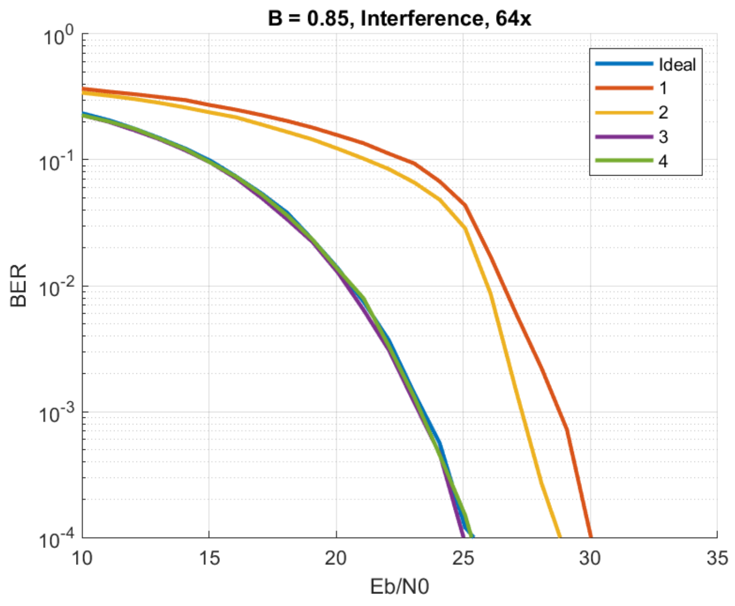

Figure 5.25: Mean-bound ratio = 0.85, BPSK Interferer, Spreading Factor 64

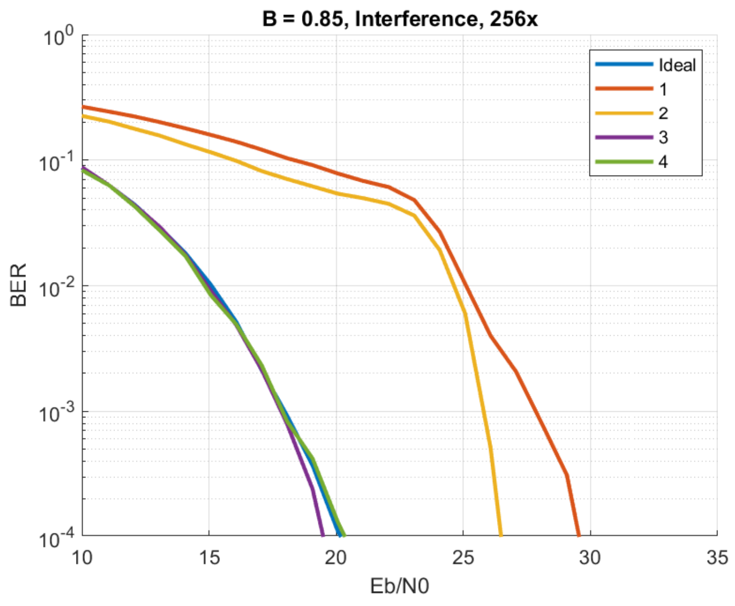

Figure 5.26: Mean-bound ratio = 0.85, BPSK Interferer, Spreading Factor 256

The figures 5.23, 5.24, 5.25 and 5.26 show the results for the quantization simulation when the mean-bound ratio is fixed at 0.85. The simulation is also ran with a AWGN channel.

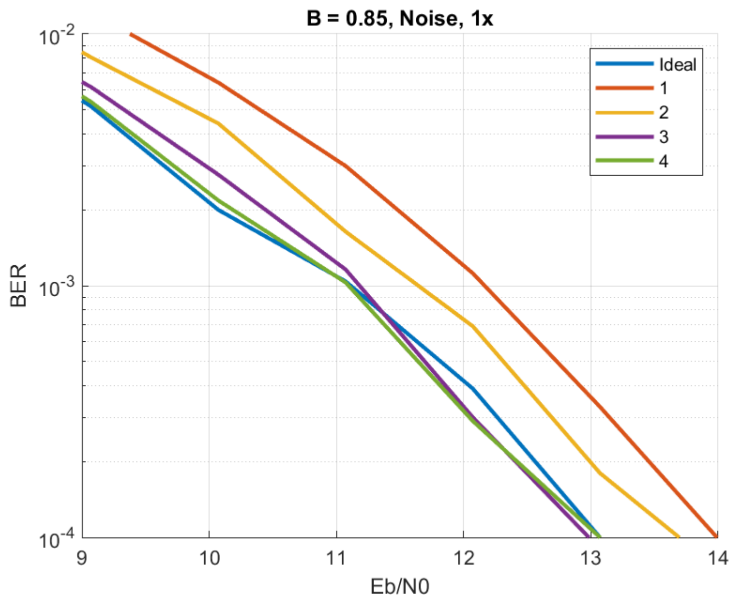

Figure 5.27: Mean-bound ratio = 0.85, AWGN, Spreading Factor 1

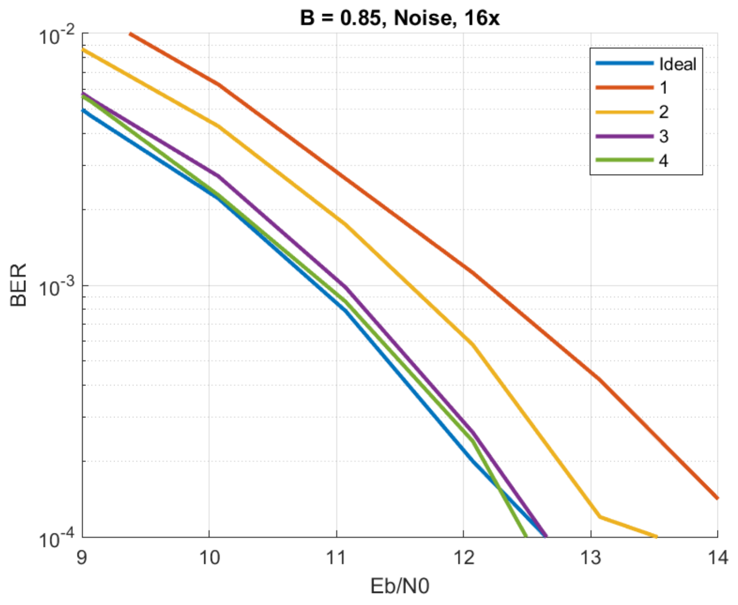

Figure 5.28: Mean-bound ratio = 0.85, AWGN, Spreading Factor 16

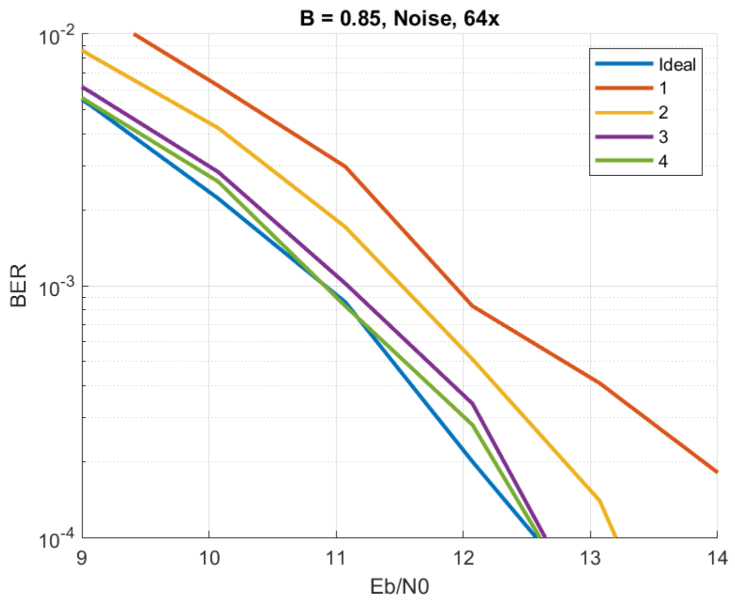

Figure 5.29: Mean-bound ratio = 0.85, AWGN, Spreading Factor 64

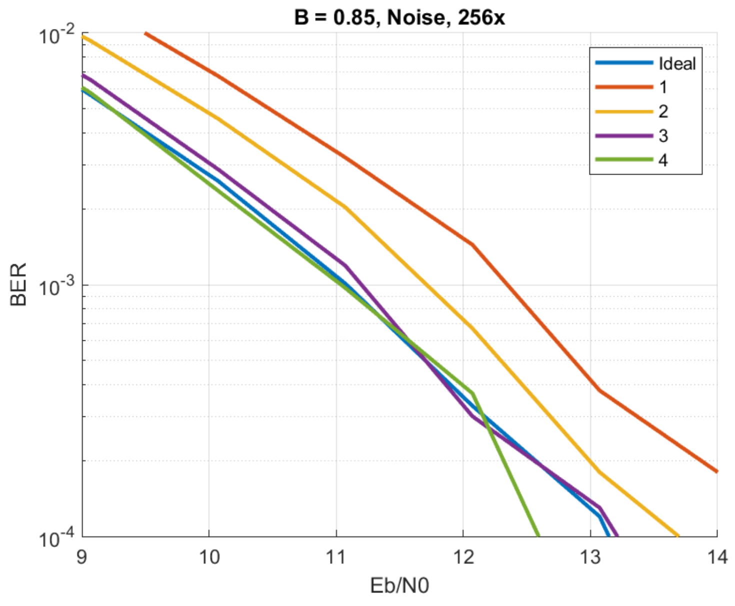

Figure 5.30: Mean-bound ratio = 0.85, AWGN, Spreading Factor 256

At the start of section 5.1, the hypothesis was formulated that the ideal number of quantization bits is 3 or 4 and that the expected improvement would be around 2dB for a AWGN channel, compared to a 1-bit quantizer. Figures 5.27, 5.28, 5.29 and 5.30 show that the BER performance improvement is not 2dB (for a AWGN channel) but slightly lower (at least for the simulated Spreading Factors). With a BPSK interferer, it is much more beneficial to increase the number of bits. This is especially the case with higher PN sequence lengths (for a Spreading Factor of

256, the improvement can be around 10dB, as shown in Figure 5.26), which can be explained by the fact that there is a lower SNR on the transmission channel.

5.1.6 Effects of varying the averaging filter time constant

In the start of this section, it was assumed that the moving average filter had a window length that was equal to the length of the simulation (so the average of the fully captured signal was calculated). This is however not suitable for practical applications. To simulate the effects of different RC-times, a simulation model for an RC-filter was used, created by Sankar et al. [19]. The transfer function of the low-pass filter is given by

$$H(z) = \frac{kz^{-1}}{1-(1-k)z^{-1}}, \tag{5.4}$$

where k is the filter constant which is inversely proportional to the RC-time of the filter. So a low k corresponds to a long RC-time and vice versa. Figure 5.31, 5.32, 5.33 and 5.34 show the simulation results for different values of k with different spreading factors. In all cases, a BPSK interference is added in the channel.

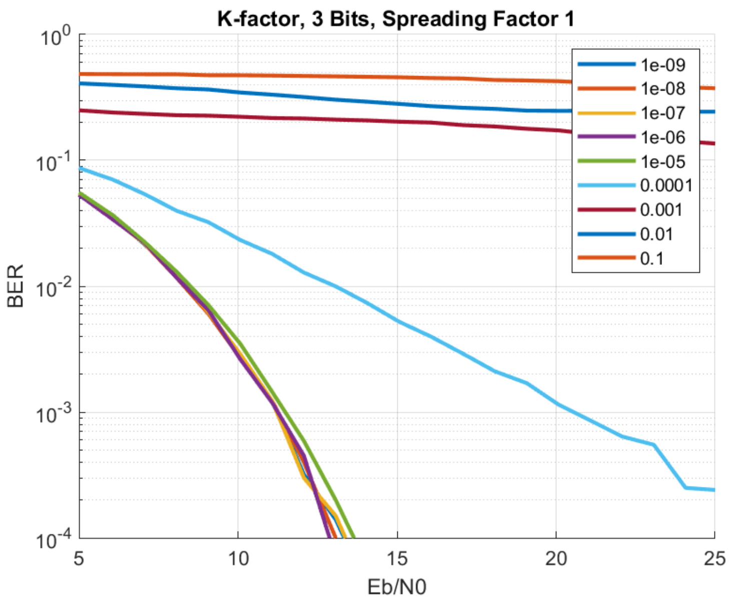

Figure 5.31: k-constant, BPSK Interferer, Spreading Factor 1

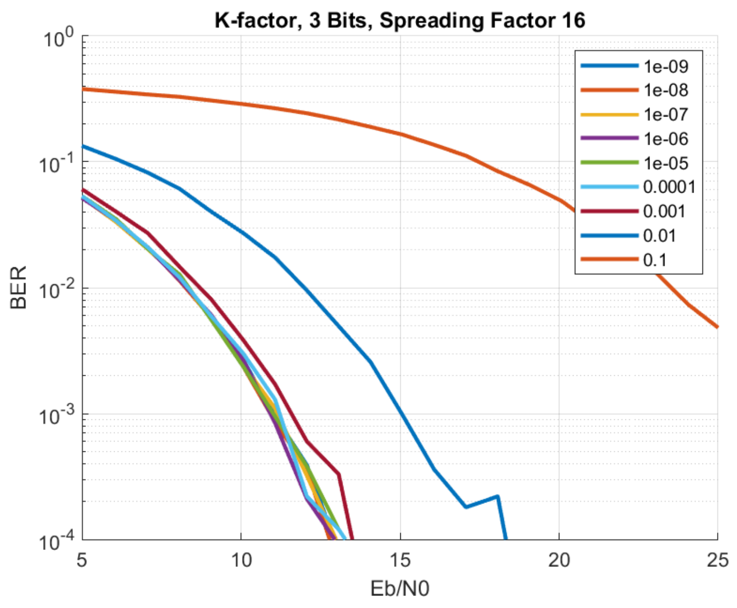

Figure 5.32: k-constant, BPSK Interferer, Spreading Factor 16

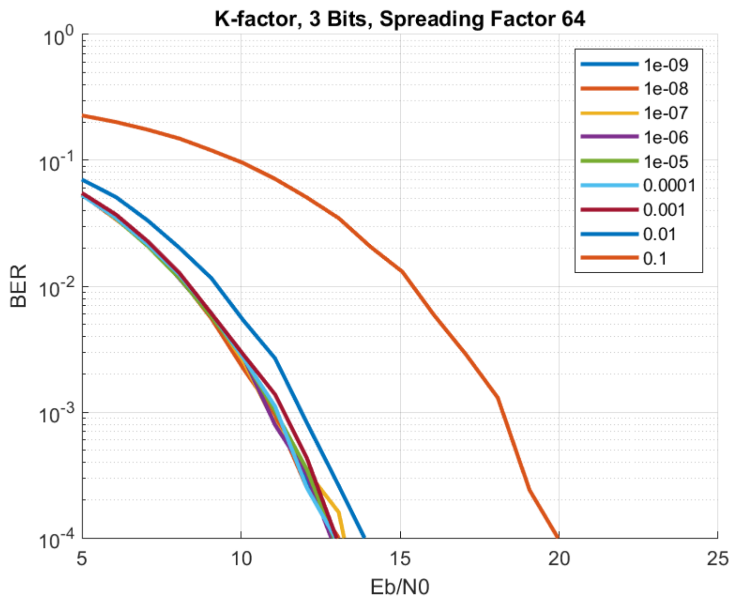

Figure 5.33: k-constant, BPSK Interferer, Spreading Factor 64

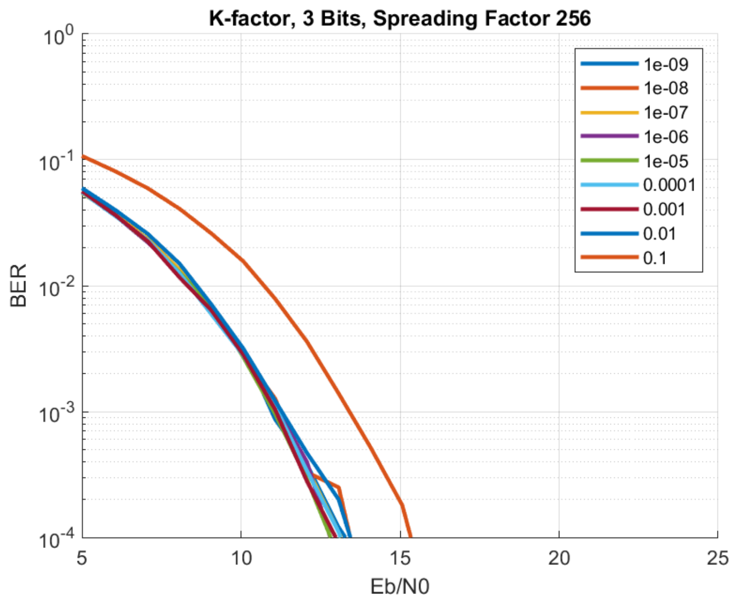

Figure 5.34: k-constant, BPSK Interferer, Spreading Factor 256

When looking at these graphs, it can be observed that high values of k do not work well with low spreading factors. From this we can conclude that the RC-filter should be sufficiently slow to ensure that individual chips do not change the mean too much, so the RC-filter should be as slow as possible. But on the other hand it should be able to adapt to the environment fairly quickly to deal with short-term interferences. The exact value depends on the total bandwidth chosen for the system and the maximum spreading factor.

5.2 Tag-to-Tag Simulation

To simulate the system at a more detailed level, a basic implementation of DSSS was written in C++ in the Tag-to-tag simulation framework by De Jong (not published at the time of writing). There are a number of differences between this simulation and the Matlab simulation from earlier sections.

First of all, The tag-to-tag simulation framework only uses baseband equivalent models (so there is no demodulation and no interferer). It does however simulate receiver hardware in much more detail (with for example amplifier noise and comparator properties) than the Matlab simulation does. It also simulates the receiver firmware. The phase noise from the CW is assumed to be white noise. Because the simulations work a little different, the despreading algorithm had to be changed (to work with the receiver firmware simulation).

```
 1 if(transition) {
 2     int pulse = ticks - lastTicks - 1;
 3     int count = countChipsInPulse(pulse, chipLength);
 4     while (count > 0) {
 5         result = multiplyCmpValWithPN(cmp, pnIdx);
 6         updateWindowAverage(result);
 7         if(windowEndReached()) {
 8             if (windowAverage > 0)
 9                 writeToBuffer(1);
10             else
11                 writeToBuffer(0);
12         }
13         decrementCount();
14         incrementPnIdx();
15     }
16     lastTicks = ticks;
17 }
```

Listing 5.1: C++ Despreading Code

The pseudo-code in Listing 5.1 shows the functionality of the despreading algorithm. The current firmware simulation works on transitions of the comparator. This is reflected in line 1 in the if-statement. So it does not use a fixed sample rate. Then on line 2, the pulse length (the time since last transition) is determined. In line 3, the number of chips are counted. So for example, if a pulse lasts 300 ticks and the chip length is 100, this function returns 3, as 3 chips fit in the pulse. Then the loop starting on line 4 multiplies every chip with the corresponding element in the PN sequence. The PN sequence in this simulation is implemented as a circular buffer, so line 5 multiplies the current comparator output with the currently selected element in the PN sequence. This result is added to the Window Average on line 6 and if a bit time is expired (the if-statement on line 7 checks for this) then the output result is written to the buffer (1 or -1, depending on if the window average is larger or smaller respectively than 0). Lines 7 till 13 essentially implement the bit detector from Listing 3.4. Finally, the iterator variables are updated on line 14 and 15, and the time is updated on line 17. The resulting BER plots from this simulation are shown in Figure 5.35. The different curves indicate different spreading factors. The simulation was performed using 1, 5 and 10 as spreading factors.

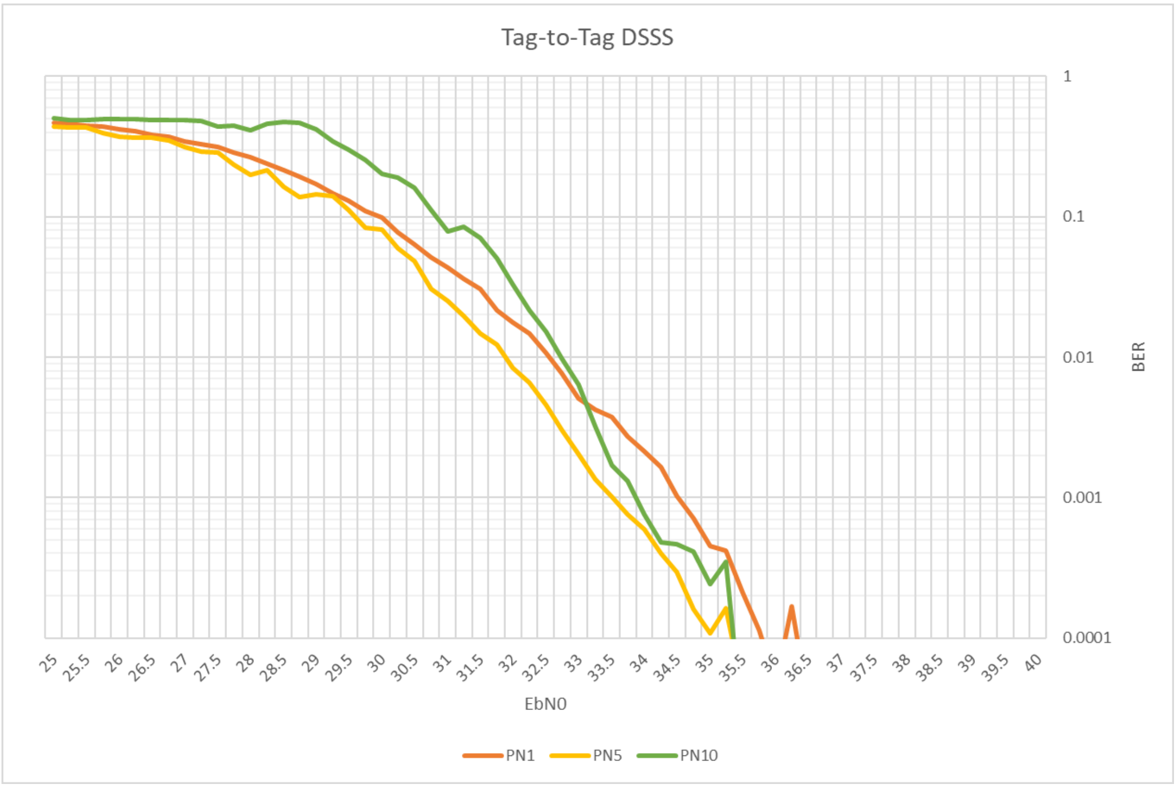

Figure 5.35: Tag-to-Tag Simulation results

The plots show that increasing the spreading factor from 1 to 5 results in a 2dB gain. Increasing the spreading factor from 5 to 10 results in a performance degradation. A spreading factor of 20 was also tested and showed even worse results. What the cause of this is, is at this point unclear and researching this issue is something that could be done in future work. The simulation framework already used a 1-bit comparator to digitize the signal, for this simulation it was left in place. The E_b/N_0 values on the x-axis are fairly high, this is because the simulated interference is white noise only. Concluding, increasing the Spreading Factor does not make much of a difference when it comes to dealing with white exciter noise and white noise interference. Based on section 4.1, this is expected.

5.3 Energy Consumption

We've seen in chapter 3 that introducing DSSS involves additional computations and additional hardware. However, it's hard to quantify what this exactly means for the energy consumption, because it depends on many factors.

First of all, the need for synchronization via the MAC protocol gives some overhead, but it does not need additional hardware and will be disregarded in this section.

Secondly, additional comparators (in case of 3 bits, a total of 7 comparators are needed) and resistor dividers are needed. This gives some additional energy consumption, but compared to a state-of-the-art ADC this is much less.

To give a rough estimation of the increase in energy: De Jong [9] uses an LPV7215 comparator which has a typical power consumption of $1.04 \mu W$ with a biasing network that consumes $1.62 \mu W$ which gives a total of $2.66 \mu W$. The estimated increase equals this value multiplied with 6 (as 7 is the required amount of comparators for quantizing with 3 bits). This gives an increase of $6 * 2.66 = 15.96 \mu W$. The total measured board consumption of De Jong [9] is $134 \mu W$, which means that the estimated board consumption *with* 3-bit quantizer will be $150 \mu W$. Table 5.1 shows this in table format.

Please note that Table 5.1 only estimates the increase as a result of hardware changes in the quantizer and neglects any changes in software. This means that the fourth row is a bit of an underestimation and the Power Consumption of a board in reality will be higher.

Hardware	**Power Consumption** (μW)
Comparator + biasing network	2.66
7 Comparators + biasing network	18.62
Board Consumption (1-bit, CPU Active, RX)	134
Board Consumption (3-bit, CPU Active, RX)	150

Table 5.1: Estimated Power Consumption of different elements and system configurations

In percentage, the increase in power consumption is

$$\frac{150 - 134}{150} * 100\% = \frac{16}{150} * 100\% = \ 10.7\%. \tag{5.5}$$

Finally, the processing device performs more calculations and operates at a higher frequency because of the increased number of symbols. An indication of how much this increases the energy consumption is very hard to give (for example because the processing device performs a lot more operations than just encoding and decoding data) and as such will be left to future work. As mentioned before, the impact of this can be reduced by using a ULP FPGA instead of a microprocessor (see subsection 2.2.1, first bullet point), at the expense of a higher standby consumption.

This chapter discussed how a DSSS implementation can be made that is usable in Backscatter networks. The next chapter will draw some conclusions from this thesis and some recommendations are given for improving the results of this thesis.

Chapter 6

Conclusion and Recommendations

6.1 Conclusion

This paper presented first steps into integrating Direct-Sequence Spread Spectrum into a Backscatter Wireless Node. In this chapter, the research questions from section 2.4 will be answered, starting with the sub-questions.

6.1.1 Sub-questions

1. How much will the usage of DSSS improve the robustness against noise?
In chapter 3, the theory showed that there was no expected improvement when the signal is transmitted across an AWGN channel and indeed, section 4.1 showed that increasing the Spreading Factor did not yield any improvement in BER performance.

2. How much will the usage of DSSS improve the robustness against narrowband (self-)interference?
In contrary to the first question, the usage of DSSS will improve robustness against narrowband interference. The BER performance improvement can get up to the spreading factor, meaning that doubling the spreading factor leads to a improvement of 3dB. Improvement against self-interference from the exciter is however marginal.

3. What is the sampling resolution needed for correctly decoding DSSS signals?
In the beginning of section 5.1, the hypothesis was posed that 3 bits would be enough to correctly decode DSSS signals without a significant performance loss. In the remainder of the section, it was indeed shown that 3 bits is the optimal number of quantization bits.
4. How can we sample the signal in an ultra-low power fashion with the required resolution?
Subsection 5.1.2 proposed an ULP quantization method for sampling the received signal. This method is based on a number parallel comparators with adaptive thresh-

olds that are derived from the mean of the signal. By increasing the number of comparators, the number of quantization bits can be increased to match the needed resolution. In section 5.3, it was shown that increasing the resolution from 1 to 3 bits leads to only a $\sim 10\%$ increase in energy consumption, whereas the improvement in BER performance can be up to 10dB (for a Spreading Factor of 256) as shown in subsection 5.1.5.

6.1.2 Main Research Question

The following main research question was formulated in section 2.4:

Does the usage of a practical Direct-Sequence Spread Spectrum implementation combined with lowering the bit rate improve the communication range compared to just lowering the bit rate?

Following the answers to the sub-questions, it can indeed be concluded that the usage of DSSS will indeed improve the receiver sensitivity and hence increase the communication range. When the channel is dominated by noise or when the interference is wideband, not much improvement is gained, as shown in section 4.1. However, DSSS *does* improve the robustness against narrowband interference from external wireless transmissions as shown in section 4.2, at the expense of an increased energy consumption.

The conclusion holds even when using a practically usable, ULP receiver implementation, as discussed in chapter 3 and chapter 5. How much the actual increase in communication distance will be, depends highly on external factors.

6.2 Recommendations and Future Work

This thesis proposed a practically usable foundation for a DSSS receiver, but there are still a couple of things that need to be researched before it is fully-functional. There are also some options to improve the results shown in this thesis. For some of them, this thesis already gives some ideas.

6.2.1 Adaptive Spreading Factor Scheme

First of all, some recommendations are done for an Adaptive Spreading Factor (ASF) scheme (and how it can be constructed and communicated across the network) and some mechanisms that might be useful at the MAC layer. Please note that none of the ideas in this section are tested in practice or even simulated. Saad Biaz et al.

[3] studied and compared several Rate Adaptation Algorithms that can be used for 802.11. The goal of Rate Adaption is determining the optimal data transmission rate for the current wireless condition. A similar concept can be used for determining the optimal Spreading Factor.

In many Backscatter networks, the network has a convergent shape (such as in the Parking Lot example). This means that the traffic flows from the outer boundaries of the network towards a central gateway (or multiple). See Figure 6.1 for a schematic representation of the network.

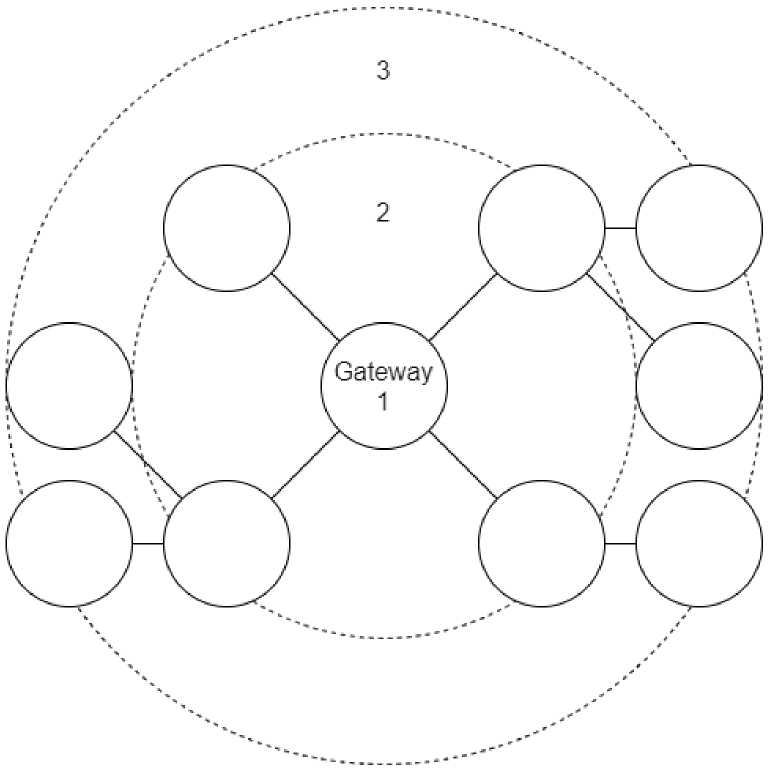

Figure 6.1: Convergent Network

This shape can also be represented as a tree where the gateway is positioned at the first (or highest) level and the nodes sit at lower levels where the nodes at the outer boundaries sit at the lowest level. The circles in Figure 6.1 represent the levels in which the nodes are placed. This tree-representation can be used to determine the routing paths when nodes can not reach the gateway directly. This tree can be expanded to also include Spreading Factor and Bit Rate.

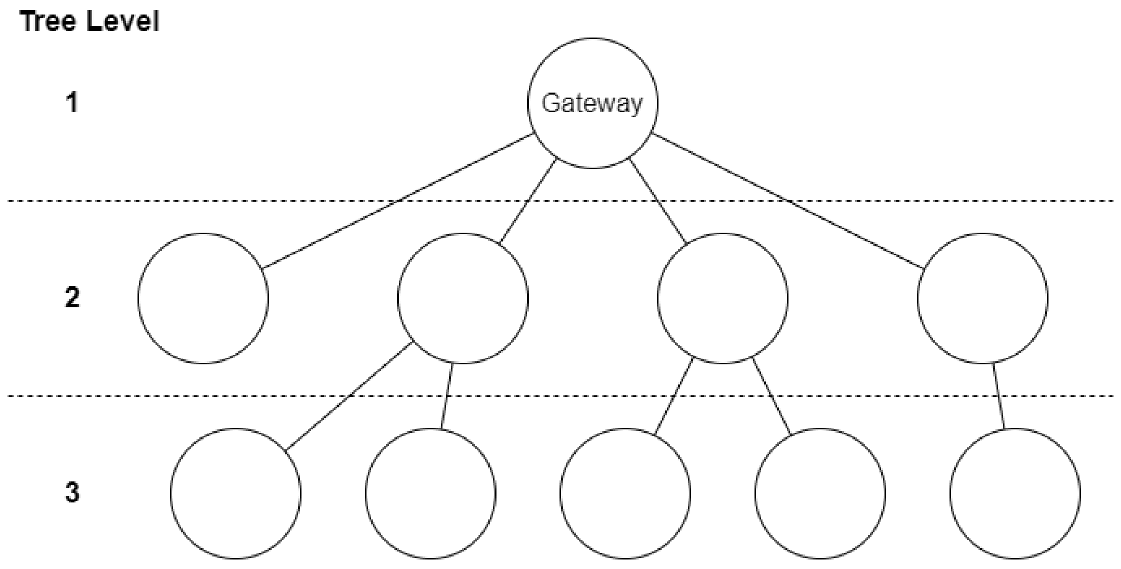

Figure 6.2: Tree Representation of a Convergent Network

The nodes at the higher levels need to receive and transmit more data than the nodes at lower levels. These nodes would benefit of a lower spreading factor so they can use more bandwidth to increase their data rate. However, as shown by De Jong et al. [9], the lower-level nodes have less Reach-Range Budget and would benefit from a higher spreading factor to increase their communication distance to increase the outer boundaries or improve the network robustness.

With this knowledge, the spreading factor can be made dependent on the tree level. Communication of this tree can be easily implemented in the MAC protocol and coordinated across all nodes during the network setup phase.

However, a disadvantage of this ASF is that it introduces additional overhead computations on the nodes. Also, the spreading factor needs to be known on both the transmitter *and* the receiver side, because otherwise despreading is not going to work very well (unless the receiver tries to despread on all spreading factors, but this is computationally expensive). Hence, it is better to determine a fixed Spreading Factor for every single node in the network setup phase and use that until changing wireless conditions force a re-evaluation of the used Spreading Factor.

A (small) added benefit is that the power consumption is more constant across the network (since there is less difference in bandwidth used by the nodes) so better battery-lifetime estimations can be made across the whole network.

It is best to use a high spreading factor during the network setup phase to ensure that all nodes can be reached.

Considering Code-Division Multiple Access

An advantage of DSSS is that it allows for Code-Division Multiple Access (CDMA) schemes. This technique uses orthogonal PN sequences to allow for parallel transmissions. However, having a large number of possible PN sequences greatly increases the number of computations done when despreading. It is much less energy-consuming to use a global PN sequence for the (limited amount of) possible spreading factors and use a time-based MAC scheme. A major reason for this is that in WSNs, the interval between transmissions is often large (of course, depending on the size of the network). An example of such a time-based MAC protocol could be Time-Division Multiple Access (TDMA). An additional advantage of using a time-based MAC is that it also mitigates the near-far effect. The tight synchronization needed for TDMA is not a disadvantage, because it is also needed for despreading. Other, more sophisticated, options are studied by Bachir et al. [2], but not all are applicable to Backscatter Networks (because Backscatter has a different nature than conventional networks, as radio transmission and reception is no longer the energy bottleneck).

A TDMA/CDMA Hybrid proposal

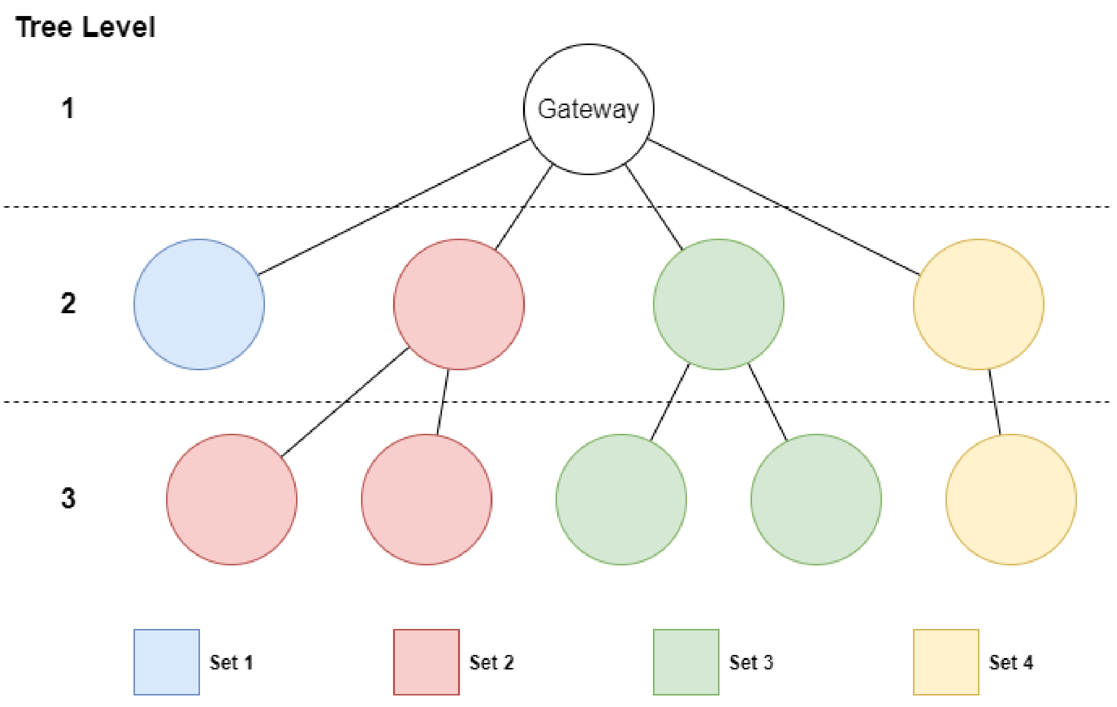

Figure 6.3: CDMA Tree

Because the gateway is assumed to have a lot of energy budget (and is capable of doing a lot more computations), an advanced extension of the idea described above could be to assign a set of PN sequences to every sub-tree of the gateway and implement CDMA at gateway level. This is shown in Figure 6.3. Within sub-trees, the nodes still use the same PN sequences so CDMA is not possible. A different MAC scheme such as TDMA needs to be used here.

The near-far effect can be mitigated by creating the sub-trees in such a way that the physical distance between them is relatively large.

6.2.2 Security

Commonly, DSSS is also used for covert transmissions (because to receivers that do not have the correct PN sequence, the signal is very noise-like). However, as argued above it is much more energy efficient to only use a small set of PN sequences. An idea could be to only use a predefined sequence to set up the network. And as soon as a connection is made, the sequences are changed. The changed sequences can be transmitted across the network ,but it would be more secure to generate them locally, for example based on some seed.

The problem remains though that, in combination with an Adaptive Spreading Factor Scheme, the transmissions closer to the gateway can be intercepted more easily because the PN sequences are much shorter. So the choice can be made to either remove the Adaptive Spreading Factor Scheme (or pose a fixed bottom limit on the shortest PN sequence length) or implement encryption on a higher network layer.

6.2.3 Other Recommendations

Section 5.3 explained why it is difficult to quantify the increase of Energy Consumption that comes with using DSSS. It is however useful to do this, because the increase might be too high (depending on the demands of the target application).

Initial tests suggested that the usage of differential encodings did not yield any performance improvement when used in combination with DSSS. However, there might be other options out there that could work. A benefit from such an encoding is the elimination of a DC bias. This might improve quantization as the mean will fluctuate less (especially in cases where the PN sequence consists of many consecutive 0s or 1s).

One thing that this thesis did not research in depth was the so-called Near-far effect. This thesis recommended the use of a Time-based MAC scheme in section 6.2.1 but there might still be options to solve this at the physical layer.

This thesis did look at self-interference from the CW but the model of the CW was somewhat simplistic, in both the Tag-to-tag and the Matlab simulation. This model could be expanded for a better indication of system robustness against self-interference.

Another benefit from DSSS is the resilience against multi-path fading. It could be investigated if this benefit can be utilized in Backscatter Radio.

In section 5.2, a simulation framework was used to simulate a DSSS implementation. However, simulating with longer PN sequences caused an unexpected performance degradation. Researching and solving this issue is another item for future work.

Finally, section 5.1 proposed a quantization method, but many alternatives exist for this. Researching these might yield less quantization loss and hence an improved communication range.

Bibliography

[1] *An Introduction to Spread-Spectrum Communications - Maxim Integrated*. (Visited: 2020-07-14). Feb. 2003. URL: https://www.maximintegrated.com/en/design/technical-documents/tutorials/1/1890.html.

[2] A. Bachir et al. “MAC Essentials for Wireless Sensor Networks”. In: *IEEE Communications Surveys Tutorials* 12.2 (Feb. 2010), pp. 222–248. DOI: 10.1109/SURV.2010.020510.00058.

[3] Saad Biaz and Shaoen wu. “Rate adaptation algorithms for IEEE 802.11 networks: A survey and comparison”. In: Aug. 2008, pp. 130–136. ISBN: 978-1-4244-2702-4. DOI: 10.1109/ISCC.2008.4625680.

[4] H. M. Chang and M. H. Sunwoo. “Implementation of a DSSS modem ASIC chip for wireless LAN”. In: *1998 IEEE Workshop on Signal Processing Systems. SIPS 98. Design and Implementation (Cat. No.98TH8374)*. 1998, pp. 243–252.

[5] Per Christensson. *Internet*. https://techterms.com/definition/internet (Visited: 2020-07-09). Sept. 2015.

[6] All About Circuits. *Understanding I/Q Signals and Quadrature Modulation: Radio Frequency Demodulation: Electronics Textbook*. URL: https://www.allaboutcircuits.com/textbook/radio-frequency-analysis-design/radio-frequency-demodulation/understanding-i-q-signals-and-quadrature-modulation/.

[7] J. F. Ensworth and M. S. Reynolds. “BLE-Backscatter: Ultralow-Power IoT Nodes Compatible With Bluetooth 4.0 Low Energy (BLE) Smartphones and Tablets”. In: *IEEE Transactions on Microwave Theory and Techniques* 65.9 (Sept. 2017), pp. 3360–3368. DOI: 10.1109/TMTT.2017.2687866.

[8] Jen-Shi Wu et al. “A 2.6-V, 44-MHz all-digital QPSK direct-sequence spread-spectrum transceiver IC [wireless LANs]”. In: *IEEE Journal of Solid-State Circuits* 32.10 (1997), pp. 1499–1510.

[9] R.J. de Jong. “Range characterization of backscatter wireless sensor networks”. 2019.

[10] C. Kchao and G. L. Stubeer. "Analysis of a direct-sequence spread-spectrum cellular radio system". In: *IEEE Transactions on Communications* 41.10 (1993), pp. 1507–1517.

[11] Vincent Liu et al. "Ambient Backscatter: Wireless Communication out of Thin Air". In: *SIGCOMM Comput. Commun. Rev.* 43.4 (Aug. 2013), pp. 39–50. ISSN: 0146-4833. DOI: 10.1145/2534169.2486015. URL: https://doi.org/10.1145/2534169.2486015.

[12] Matlab. *AWGN Channel*. (Visited: 2020-09-09). URL: https://nl.mathworks.com/help/comm/ug/awgn-channel.html.

[13] *Nedap SENSIT Parking Sensors*. https://www.nedapidentification.com/products/sensit/ (Visited: 2019-10-22).

[14] *Nedap SENSIT Relay node datasheet*. https://www.nedapidentification.com/products/sensit/sensit-relay-node/ (Visited: 2019-10-22).

[15] Bourdillon O. Omijeh and Tedje Oteheri. *Binary Phase Shift Keying Digital Modulation Technique for Noiseless and Noisy Transmission*. Oct. 2016. URL: http://article.sciencepublishinggroup.com/html/10.11648.j.cssp.20160503.11.html.

[16] L. Ouvry, C. Boulanger, and J. R. Lequepeys. "Quantization effects on a DS-CDMA signal". In: *1998 IEEE 5th International Symposium on Spread Spectrum Techniques and Applications - Proceedings. Spread Technology to Africa (Cat. No.98TH8333)*. Vol. 1. 1998, 234–238 vol.1.

[17] R. Piyare et al. "Ultra Low Power Wake-Up Radios: A Hardware and Networking Survey". In: *IEEE Communications Surveys Tutorials* 19.4 (Oct. 2017), pp. 2117–2157. DOI: 10.1109/COMST.2017.2728092.

[18] Qicai Shi, S. R. Korfhage, and R. J. O'Dea. "A low-cost receiver design for DSSS location system". In: *IEEE 2002 International Conference on Communications, Circuits and Systems and West Sino Expositions*. Vol. 1. 2002, 220–224 vol.1.

[19] Krishna Sankar. *Digital implementation of RC low pass filter*. (Visited: 2020-01-17). Dec. 2007. URL: http://www.dsplog.com/2007/12/02/digital-implementation-of-rc-low-pass-filter/.

[20] E. A. Sourour and M. Nakagawa. "Performance of orthogonal multicarrier CDMA in a multipath fading channel". In: *IEEE Transactions on Communications* 44.3 (1996), pp. 356–367.

[21] Vamsi Talla et al. "Lora backscatter: Enabling the vision of ubiquitous connectivity". In: *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 1.3 (2017), p. 105.

[22] *Understanding Spread Spectrum for Communications*. (Visited: 2020-06-15). 2019. URL: https://www.ni.com/nl-nl/innovations/white-papers/06/understanding-spread-spectrum-for-communications.html.

[23] Won Namgoong. "ADC and AGC requirements of a direct-sequence spread spectrum signal". In: *Proceedings of the 44th IEEE 2001 Midwest Symposium on Circuits and Systems. MWSCAS 2001 (Cat. No.01CH37257)*. Vol. 2. 2001, 744–747 vol.2.

[24] C. Xu, L. Yang, and P. Zhang. "Practical Backscatter Communication Systems for Battery-Free Internet of Things: A Tutorial and Survey of Recent Research". In: *IEEE Signal Processing Magazine* 35.5 (Sept. 2018), pp. 16–27. DOI: 10.1109/MSP.2018.2848361.

[25] Randy Yates. *A Coding Theory Tutorial*. http://www.digitalsignallabs.com/downloads/tutorial.pdf (Visited: 2020-08-14). Aug. 2009.

[26] PENGYU ZHANG et al. "Enabling Practical Backscatter Communication for On-body Sensors". In: *Proceedings of the 2016 ACM SIGCOMM Conference*. SIGCOMM '16. Florianopolis, Brazil: ACM, 2016, pp. 370–383. ISBN: 978-1-4503-4193-6. DOI: 10.1145/2934872.2934901. URL: http://doi.acm.org/10.1145/2934872.2934901.

[27] Pengyu Zhang et al. "EkhoNet: High Speed Ultra Low-Power Backscatter for next Generation Sensors". In: *Proceedings of the 20th Annual International Conference on Mobile Computing and Networking*. MobiCom '14. New York, NY, USA: Association for Computing Machinery, 2014, pp. 557–568. ISBN: 9781450327831. DOI: 10.1145/2639108.2639138.