Escola Tècnica Superior d'Enginyeria de Telecomunicació de Barcelona

UNIVERSITAT POLITÈCNICA DE CATALUNYA

KTH Information and Communication Technology

MASTER THESIS

Chain simulation of DS-CDMA communication system

Simulación de una cadena de comunicaciones DS-CDMA

Simulació d'una cadena de comunicacions DS-CDMA

Author: Albert Armisen Morell

Advisor: Josep Sala Alvarez

Year: 2010

Acknowledgements

First of all, I would like to thank project director Dr. Josep Sala for his full commitment and support though the length of this project. He actively helped me going from planning to management. I would also like to specially praise the rigorousness that he brought to this endeavor. Thank you for everything I have learned and your willingness to help me and teach me anytime.

This project culminates a learning phase from my college. I've met many people who boosted my energy while allowing me to grow at a personal and professional level. The experiences with will remain in my memory and live in me. So, I thank all of those who knowingly or unknowingly contributed to support me, encourage me and an overall vision and hope to complete these studies.

At an educational level I had a great opportunity to enjoy and learn from talented teachers and inspiring as Miguel Angel (UPC), Mats (KTH), Anna (SSE), Beth (Berkeley) and Kathryn (Stanford). Indeed, some family friends guided me in this world as Jordi.

It is my wish to thank those who have crossed my path and helped to complete my training in a more humanistic and professional sense. At association level, people from “Student Union”, “Company Fair”, “Debate club” and all those in the Omega building located in UPC. At professional level, I wish to emphasize Dr. Jesus Alvarez who gave me the first opportunity to work in a very interesting research project, co-workers who excelled as Marc, Jordi, Nivard, Anna and Elena. To Ismael with whom I had the opportunity to work in a large number of projects in his company and co-workers (Toni and Valerie) from whom I learned so much.

Thanks to my friends, who have always provided great moral and social support. To childhood friends Asier, Pere, Alex, Sergi, Xavier and Victor. To UPC friends as Oriol, David, Marina, Marta, Francesc and Manolo. To KTH friends as Jona, Victor, Mario, Emma, Hedieh, Syed and Mohammed. And finally, to travelling friends as Mohammed, Lars, Priyanka and Mikhal.

Thanks to my family, my parents, my grandparents and uncles, because they transmitted values, knowledge and gave me a solid structure which allowed to get here. I owe what I am and even much of what I will be.

To all, thank you very much.

Abstract (English)

This project has analyzed and implemented a system based on DS-CDMA with a common receiver and multiple transmitters on a modular platform in Matlab, which is used for theoretical validation tool.

This platform has been chosen over a DSP implementation due to the economic cost of DSP boards. So, it was decided to implement it using Matlab considering the inherent constraints in a DSP board.

Project's main objective is to validate this system by having a simulation at a sample level which has no memory constraints. The next step would be to implement this in DSP boards; however this is beyond the scope of this project. A system has been designed that can process data with few resources in Matlab environment. The system developed is highly configurable using some input parameters. The transmitter consists of several modules that are invariant which are encoder, modulator, spreader, zero padder, pulse shaper and converter. These chained modules generate each user transmitted signal.

Once these transmitters' signals have been generated, they pass through a slowly fading channel with additive Gaussian noise which models a means of mobile communications.

Ultimately the receiver gets all signals and processes them in a series of independent modules consisting of a low pass filter, downconverter, matched filter, synchronizer, downsampler, equalizer, despreader, demodulator and decoder.

This work can be seen in the "Results" section where there are screens of the signal in each of the phases followed by a brief justification.

Abstract (Spanish)

En este proyecto se ha analizado e implementado un sistema basado en DSSS-CDMA con un receptor común y varios transmisores sobre una plataforma modular en Matlab, siendo ésta una herramienta de validación teórica.

Se ha primado esta sobre una implementación en DSP por el coste económico de las placas DSP. Así que se ha decidido hacer una implementación en Matlab con las constricciones propias de una placa DSP.

El objetivo principal del proyecto es la validación del sistema mediante la simulación a nivel de muestra sin restricciones de memoria. El siguiente paso sería la implementación en placas DSP pero esto se escapa del objetivo de este proyecto. Para ello se ha diseñado un sistema que pueda procesar los datos con pocos recursos en Matlab, marcados por una serie de variables.

El transmisor se compone de varios módulos invariantes que son el codificador, modulador, spreader, zero padder, pulse shaper y el up converter que encadenados generan la señal a transmitir de cada uno de los distintos usuarios.

Todas estas señales pasan por un canal con desvanecimientos lentos y ruido aditivo gaussiano que modeliza un medio de comunicaciones móvil.

Finalmente el receptor recibe todas las señales y las procesa en una serie de módulos independientes formados por un filtro paso bajo, downconverter, filtro adaptado, sincronizador, downsampler, equalizador, despreader, demodulador y decodificador.

En este trabajo se puede observar en la sección “Resultados” las capturas de la señal en cada una de las distintas fases seguida de una breve explicación. Para finalmente llegar a la sección de “Conclusiones” y “Futuras líneas de investigación”.

Abstract (Catalan)

En aquest projecte s'ha analitzat e implementat un sistema basat amb DSSS-CDMA amb un receptor comú y diversos transmissors sobre una plataforma modular en Matlab, essent aquesta una eina de validació teòrica.

S'ha primat aquesta per sobre d'una implementació en DSP principalment pel cost ecònomic de les plaques DSP. Així, s'ha decidit fer una implementació en Matlab amb les restriccions pròpies d'una placa DSP.

El principal objectiu del projecte es la validació del sistema mitjançant la simulació a nivell de mostra sense restriccions de memòria. El proper pas seria la implementació en plaques DSP, peró això s'escapa del objectiu d'aquest projecte. És per això que s'ha dissenyat un sistema que pugi processar les dades amb pocs recursos mitjançant Matlab, tots marcats per una serie de variables.

El transmissor es composa de diversos mòduls invariants que son el codificador, modulador, spreader, zero padder, pols conformador i el up converter que estan encadenats per generar la senyal a transmetre per cada un dels diversos usuaris.

Totes aquestes senyals passen per un canal d'esvaniment lent amb soroll Gaussià blanc que modelitza un medi de comunicacions mòbil.

Finalment el receptor rep totes les senyals y les processa en una serie de mòduls independents formats per un filtre pas baix, downconverter, filtre adaptat, sincronitzador, downsampler, equalitzador, despreader, demodulador y decodificador.

En aquest treball es pot observar en la secció de "Resultats" les captures de la senyal a cada una de les diverses fases seguides d'una breu explicació. Finalment es tracten les conclusions i les properes vies d'investigació.

Content index

Figure index

Table index

1. Introduction

This project has analyzed and implemented a system based on DS-CDMA with a common receiver and multiple transmitters on a modular platform in Matlab, which is used for theoretical validation tool.

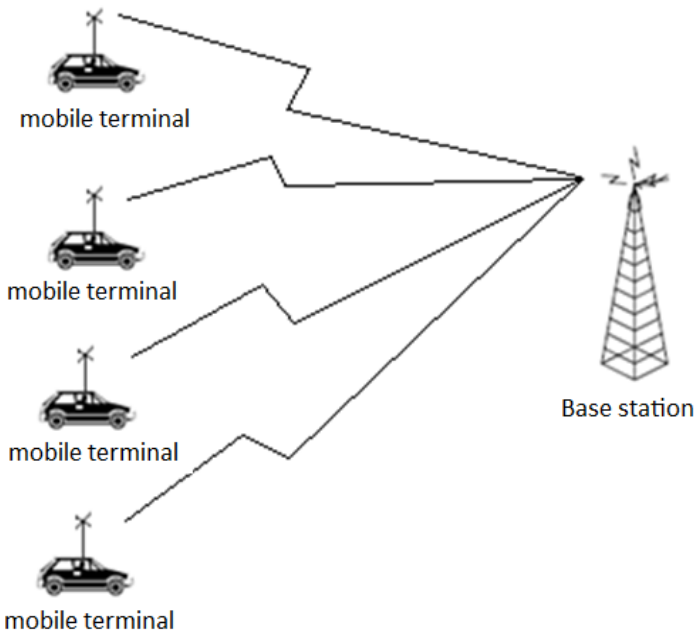


Figure 1: "System of several transmitters to a common base station"

1.1 Project context

This platform has been chosen over a DSP implementation due to the economic cost of DSP boards. For this very same reason, an implementation with Matlab was preferred.

1.2 Objectives

Project's main goal is to validate this system by having this system by having a simulation at a sample level which has no memory constraints. The next step would be to implement this in DSP boards; however this is beyond the scope of this project. A system has been designed that can process data with few resources in Matlab environment. The system developed is highly configurable using some input parameters. The transmitter consists of several modules that

are invariant which are encoder, modulator, spreader, zero padder, pulse shaper and converter. These chained modules generate each user transmitted signal.

1.3 Memory structure

Once these transmitters' signals have been generated, they pass through a slowly fading channel with additive Gaussian noise which models a means of mobile communications.

Ultimately the receiver gets all signals and processes them in a series of independent modules consisting of a low pass filter, downconverter, matched filter, synchronizer, downsampler, equalizer, despreader, demodulator and decoder.

This work can be seen in the results section where there are screens of the signal in each of the phases followed by a brief justification.

The last two sections are"future lines of work" and "references".

2. State of the art

CDMA or Code Division Multiple Access is a channel access method which uses different radio-communication technologies. CDMA is a wide spread spectrum technique for multiple access. A wide spread spectrum technique consists on extending data's bandwidth uniformly through a range of the spectrum using a pseudo-random code. This code consists of several chips that operate at a much faster rate than data transmitted

$$f_{chip} = N \cdot f_{symbol}$$

This technique allows multiplexing, which is one of the basic concepts in data communication consisting of allowing several transmitters to send information simultaneously in a single communication channel. This is achieved by having multiple users share a range of bandwidth in different frequencies. CDMA employs spread spectrum technology and a special coding scheme (where each transmitter is assigned a code) to allow multiple users to be multiplexed over the physical channel. By contrast, time division multiple access (TDMA) divides access by time, while the frequency division multiple access (FDMA) is divided by frequency. CDMA is a form of spread spectrum signaling, since the code modulated signal has a higher bandwidth than the data being transmitted.

$$B_{symbol} = \frac{1}{f_{symbol}} = \frac{1}{N \cdot f_{chip}}$$

CDMA can be applied to various applications such as use in the GPS. Currently used in satellite communications.

The performance of CDMA is very simple, is based on XOR logic when considering that it is transmitted {0,1}, also known as exclusive OR [VITE 1995]. In Figure 2: "Relationship between data bits, pseudo-random codes and transmitted signal" displays how it is generated the spread spectrum signal from a sequence of bits and the user's chip. Data signal with pulse duration of T_b is passed to a XOR function with the transmission code with pulse duration T_c. The relationship $\frac{T_b}{T_c}$ is called spreading factor or processing gain and is a decisive factor for the upper limit of the total number of users supported simultaneously by a base station.

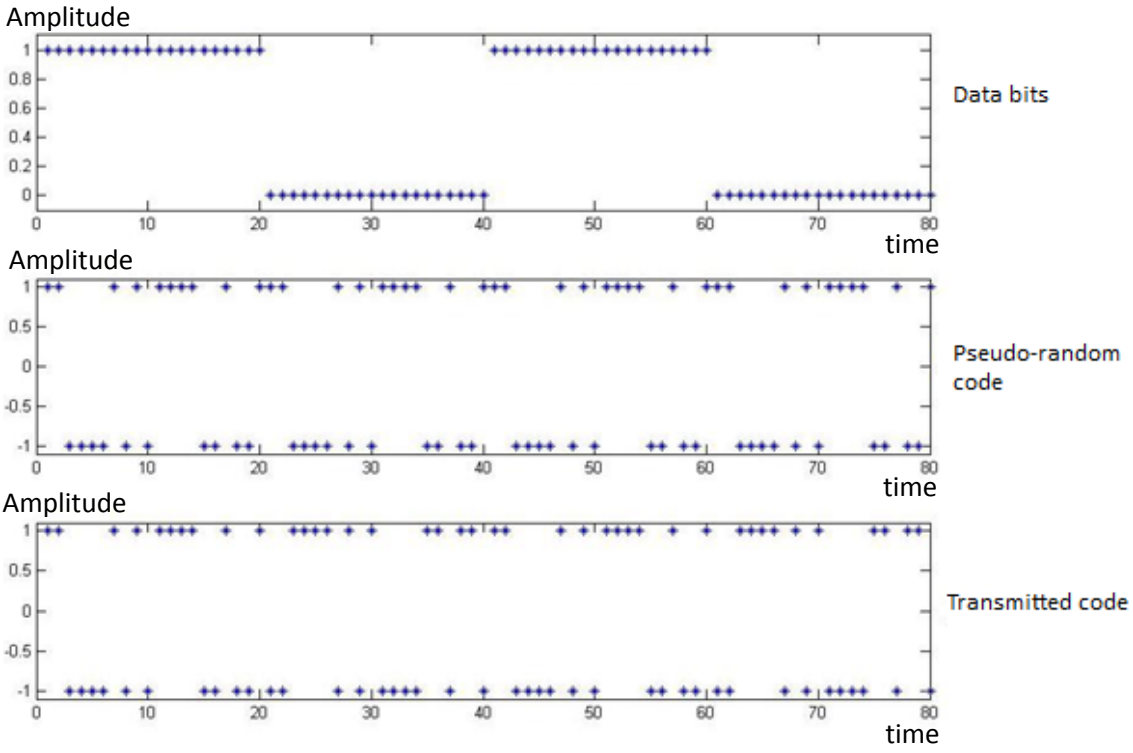


Figure 2: "Relationship between data bits, pseudo-random codes and transmitted signal"

Each user in a CDMA system uses a different code or group of codes to modulate their signal (one code if data and pilot is transmitted in the same frequency or 2 if it is done in different ones). For example one code for data channel and another for pilot channel. A key point for the performance of the system is the choice of codes used to modulate the signal from CDMA systems. The two best known are used for CDMA Hadamard and Gold used for ACDMA[1]. The best performance occurs when there is good separation between the signals of a desired user and the signals from other users (also known as MAI[2]).

[1] Asyncronous **C**ode **D**ivision **M**ultiple **A**ccess
[2] Multiple access interference

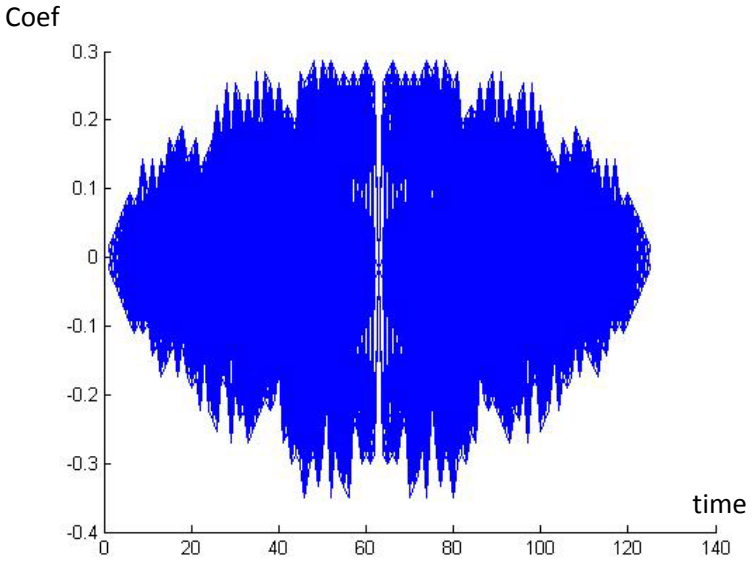


Figure 3: "Cross correlation of all users of a Gold code with a SF of 63, normalized to the correlation of a code"

The separation of the signals is performed by correlating the received signal with the local source code of the desired user. If the signal matches the desired user code then the correlation function will be high and the system can extract that signal. If desired by the user code has nothing in common with the correlation signal, this correlation should have a value close to zero. This procedure is known as auto-correlation and is used to reject multipath interference.

In general, CDMA belongs to two basic categories: synchronous orthogonal codes and asynchronous using pseudorandom sequences.

Synchronous CDMA exploits mathematical properties of orthogonality between the vectors representing data streams. For example, the binary string 1011 represents the vector$(1 \quad 0 \quad 1 \quad 1)$. These vectors can be multiplied by the scalar product, the sum of the products of their components. If the scalar product is zero, the two vectors are said to be orthogonal to each other.

Synchronous CDMA uses a code orthogonal to other codes to modulate their signal. To set an example would bea four orthogonal digital signals as shown in the following equations. Orthogonal codes have zero cross-correlation, meaning that do not interfere with each other.

$$r_{ab} = r_{ba} = a \cdot b = 0$$

$$a \cdot (a + b) = a \cdot a + a \cdot b = \|a\|^2 + 0$$

$$a \cdot (-a + b) = -a \cdot a + a \cdot b = -\|a\|^2 + 0$$

$$b \cdot (a + b) = b \cdot a + b \cdot b = 0 + \|b\|^2$$

$$b \cdot (a - b) = b \cdot a - b \cdot b = 0 - \|b\|^2$$

In the above example, an orthogonal Walsh sequences describes how two users can be multiplexed together in a synchronous system, a technique commonly known as code division multiplexing (CDM) [TONO 1992]. Indeed, a NxN Walsh matrix can be used to multiplex N users. When multiplexing is necessary that all users are coordinated so that each transmitter transmits with a delay of the channel to reach the receiver at exactly the same time. Thus, this technique is used in links to mobile base, where all transmissions originate from the same transmitter and can be coordinated perfectly.

Another type of CDMA based on the use of pseudorandom sequences is asynchronous CDMA known as ACDMA.

ACDMA links are used when the mobile to the base communication cannot be coordinated with a proper precision, mainly due to the mobility of terminals. Thus, this has a big implication which is that the chip code needs to has good properties in the exact moment and when it is delayed. A first approach would be to design a system orthogonal at all times, however this is impossible from a mathematical point of view. So, a pseudo-random sequences (PN)is used for which have these good properties. A PN code is a binary sequence that appears random but can be reproduced in a deterministic manner. These PN codes are used to encode and decode the signal from an asynchronous CDMA users in the same way that the orthogonal codes. These PN sequences are statistically correlated, and the sum of a large number of PN sequences results in multiple access interference (MAI) which is approximated by a Gaussian noise process if there is no near-far problem[3] (extracted when using Central limit theorem [4]). If all users have the same power, then you can approximate the variations of the MAI as white noise which is directly proportional to the number of users. In other words, unlike synchronous CDMA, the signals from other users appear as noise to the signal of interest and interfere slightly with the desired signal in proportion to the number of users. These signals from other users in the ACDMA are received as broadband noise which reduces the gain of the process. Since each user generates MAI, controlling the signal strength is a key issue related to CDMA transmitters.

The CDMA has a number of advantages over other systems such as TDMA and FDMA[LEWI 1993]:

[3] It is based on the receiver picks up a strong signal that makes it impossible for the receiver to detect a weaker signal.

[4] The central limit theorem states that, in very general terms, the distribution of the sum of random variables tends to a normal distribution when the number of variables is very large

1. **Efficient use of a fixed bandwidth.**

Type	Main challenge
TDMA	**Timing.** TDMA must synchronize the transmission of all users, so it is ensured that there is no co-channel interference. Since this cannot be perfectly controlled in a mobile environment, each time interval should have guard time, reducing the likelihood that users will interfere, but decreases the spectral efficiency.
FDMA	**Frequency generation**.FDMA systems must use a guard band between adjacent channels due to Doppler shift. Unpredictable shift of the signal due to user mobility. The frequency bandwidth reduces the probability that adjacent channels will interfere, but decrease the use of the spectrum.
CDMA	**Power control**. The need of a CDMA power control since it has a direct impact on signal to noise ratio (SNR). Other techniques such as SIC (Successive Interference Cancellation) can relax this restriction and improve overall system spectral efficiency.

2. **Flexible allocation of resources**

CDMA can offer a key advantage over flexible allocation of resources, for example PN codes can be assigned to each user. In the case of TDMA and FDMA have a number of simultaneous orthogonal codes fixed slots and fixed frequency bands. This fixed number of time slots or frequency bands are underutilized especially in cases of bursts such as when data is packed. In contrast CDMA adapts to the number of users because you can add another user and the overall impact will be a decrease on SIR decreases, while if there are fewer users SIR increases.

3. **Anti-jamming capability of CDMA**

Because bandwidth is limited, it is usually common to try minimizing bandwidth. However, the use of spread spectrum techniques aims to use more bandwidth while reducing power spectral density. One of the initial reasons for doing this was military applications in communications systems. These systems were designed using spread spectrum for resistance to interference. The code makes CDMA spread spectrum signals appear random, so, these have some properties similar to noise. A receiver

cannot demodulate this transmission without knowing pseudorandom sequence used to encode data.

4. **Resistant to interference**

Both synchronous and asynchronous CDMA are resistant to interference. So if the interference is constant over the spectral width, the effective noise will be the bandwidth of the chip code over the noise bandwidth. Furthermore, the CDMA is very effective against narrowband interference since noises outside the bandwidth associated with the code chip are not affecting the signal.

Another key point is that the CDMA is resistant to multipath interference and the delayed versions of the codes will have little correlation with the original pseudorandom code, and therefore will appear as another user, which is ignored in the receiver being used a RAKE receiver. In other words, if multiple channel chip cause the least delay, the multipath signals arrive at the receiver so that travel time by at least one chip of the predicted signal.

3. System description

This section is a brief description of the implemented system. It goes from a general level description with the organization of the transmitter, receiver and channel, to a description of the blocks at a schematic level of the system, and finally a more detailed analysis is performed.

The system consists of several transmitters that communicate with a common receiver through a common medium. Communication with the common receiver can be carried out by each transmitter which uses a bandwidth set by its associated chip code.

The system is outline in Figure 4: "General Model" in general terms as each transmitter sends a signal that amount in one medium or channel, the receiver receives a signal of all transmissions over a noise introduced by the channel.

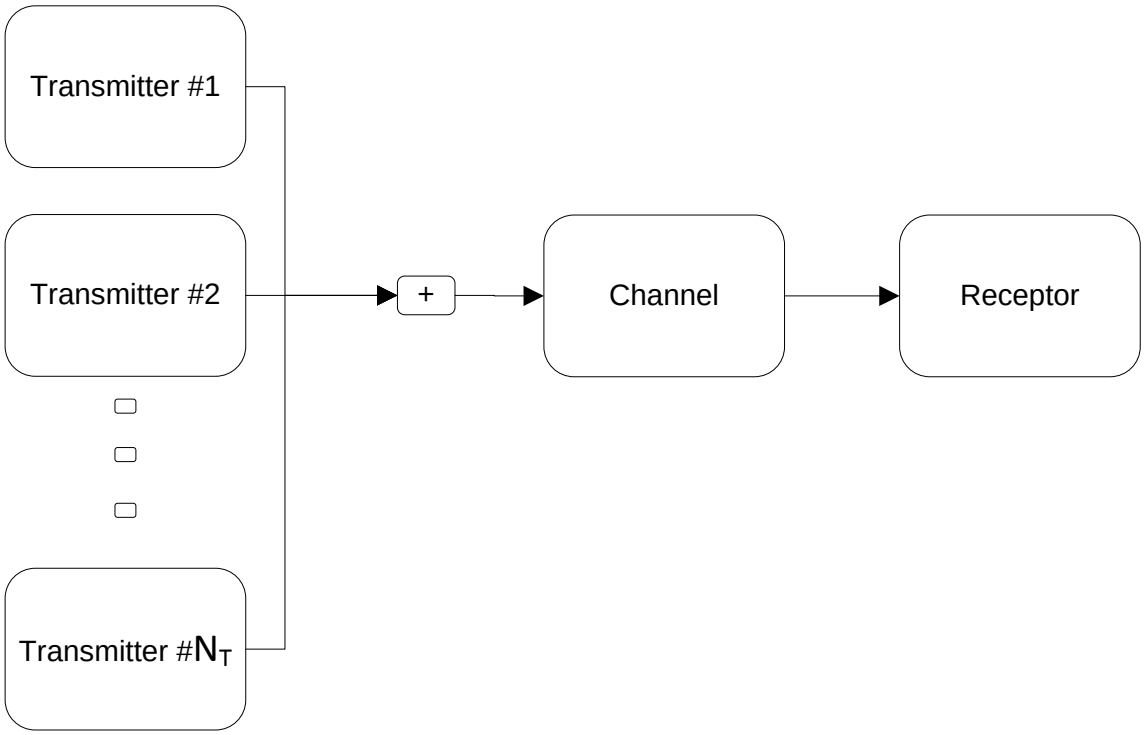


Figure 4: "General Model"

3.1. System

The implemented system is a synchronous CDMA composed of multiple transmitters and a common receptor. These transmitter and receiver consist of a series of blocks:

- Builder bits. This module generates random bits that the transmitter uses to encode and transmit.
- Encoder. The encoder includes a bits as input and adds redundancy to protect against noise and channel interference
- Modulator. This block modulates a sequence of bits in a symbol determined by the selected constellation.
- Spreader. Symbols are used as a input using single chip for each transmitter to create a widened stream.
- Zero Padder. It consists on inserting a zeros sequence to the symbols of the signal.
- Pulse shaper. It consists on modifying the signal properties to reduce the effect of intersymbol interference.
- Up converter. It consists on uploading the frequency to transmitted signal.
- Down Converter. This module is analogue to the upconverter, it lower the frequency of the received signal.
- Down sampler. Analogous to zero padder module, it consists on saving some samples of the total samples received.
- Synchronizer. This block of acquisition is to select the point at which the signal is received.
- Despreader. Module which gets symbols using user-specific chips.
- Demodulator. Analogue structure to the modulator which gets a sequence of bits based on a symbol.
- Decoder. Analogue structure to encoder which compensates for the changes introduced by the decoder.

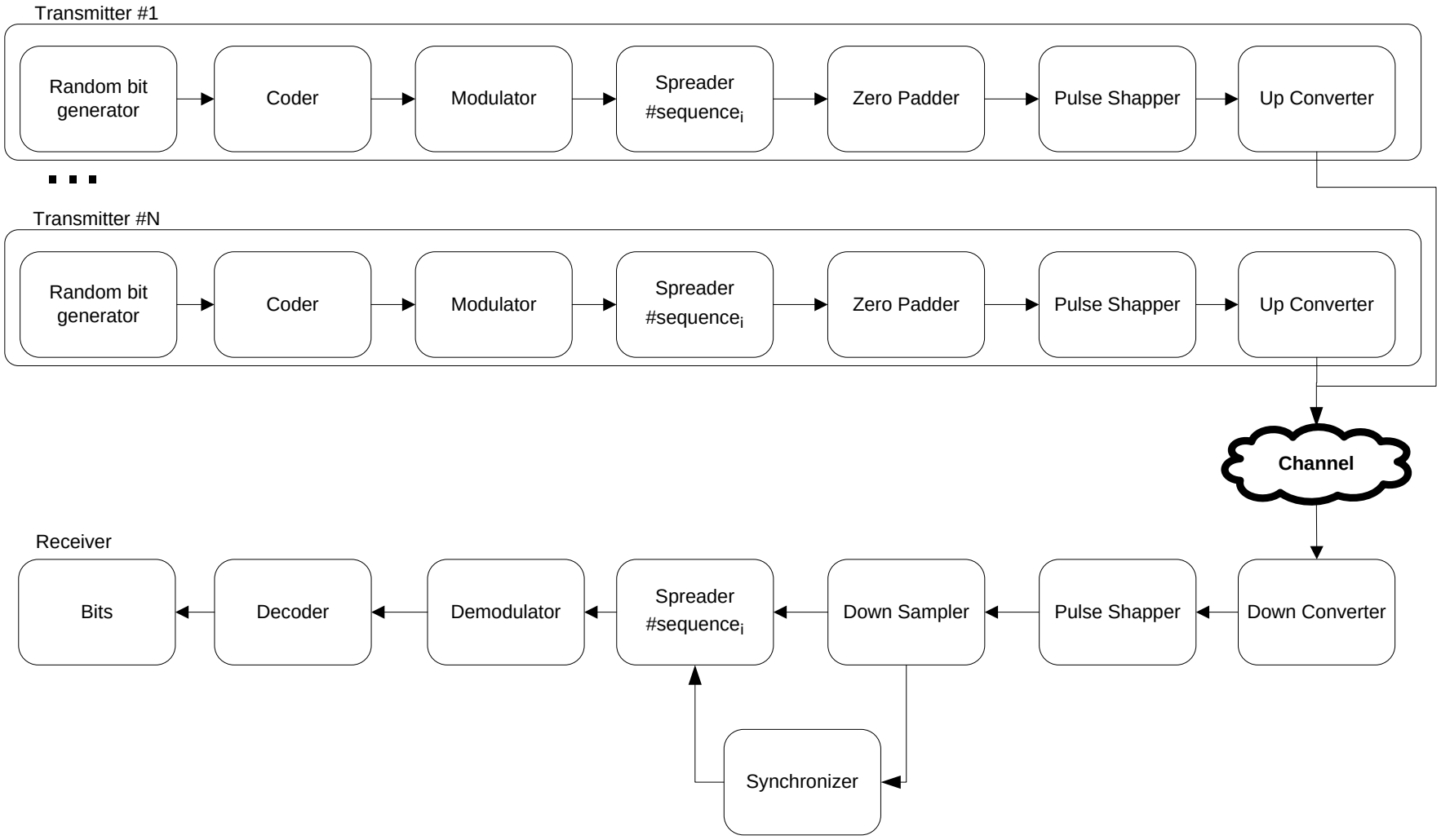


Figure 5: "General structure of receptor and transmitter chain"

3.2. Frame structure

A critical point of system design is the frame structure implemented on this CDMA model. Its design directly affects the channel estimation and actual data transmission. This project has defined this as a specific pattern, which does not correspond to any real system.

The plot consists of two different elements:

1. User bits information associated with a particular sequence spreader.
2. Pilot bits associated with a particular sequence spreader.

These bits can be differentiated on time or frequency as shown in Figure 6: "Frame structure in a time division" and Figure 7: "Frame structure in a frequency division" have been studied in both cases, concluding that the most interesting is frequency division. This report is based on the division frequency on data information and pilot information.

The first case consists on having a temporal division, firstly transmitting pilot and then transmitting the information as shown in Figure 6: "Frame structure in a time division". Its advantages are an efficient use of resources, for instance the amount of information transmitted, since you can change the number of transmitted pilot symbols vs. the transmitted information.

Figure 6: "Frame structure in a time division"

The second implementation is based on incorporating the data to transmit on one frequency, and pilot symbols on another frequency, this is achieved by assigning a code to data and a pilot. The following Figure 7: "Frame structure in a frequency division" details the structure of a frame, which uses two spread sequences for the same user. The advantages are the continuous collection of pilot data which allows estimating channel and improve synchronization.

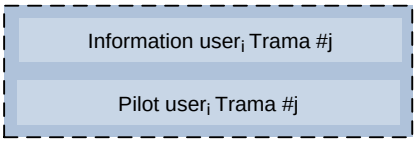


Figure 7: "Frame structure in a frequency division"

Extrapolating the second implementation in time and frequency and under the assumption that each user shares the same transmission medium. The general structure is illustrated in Figure 8: "Global structure of the frames with respect to time under the assumption that all users transmit at the same time".

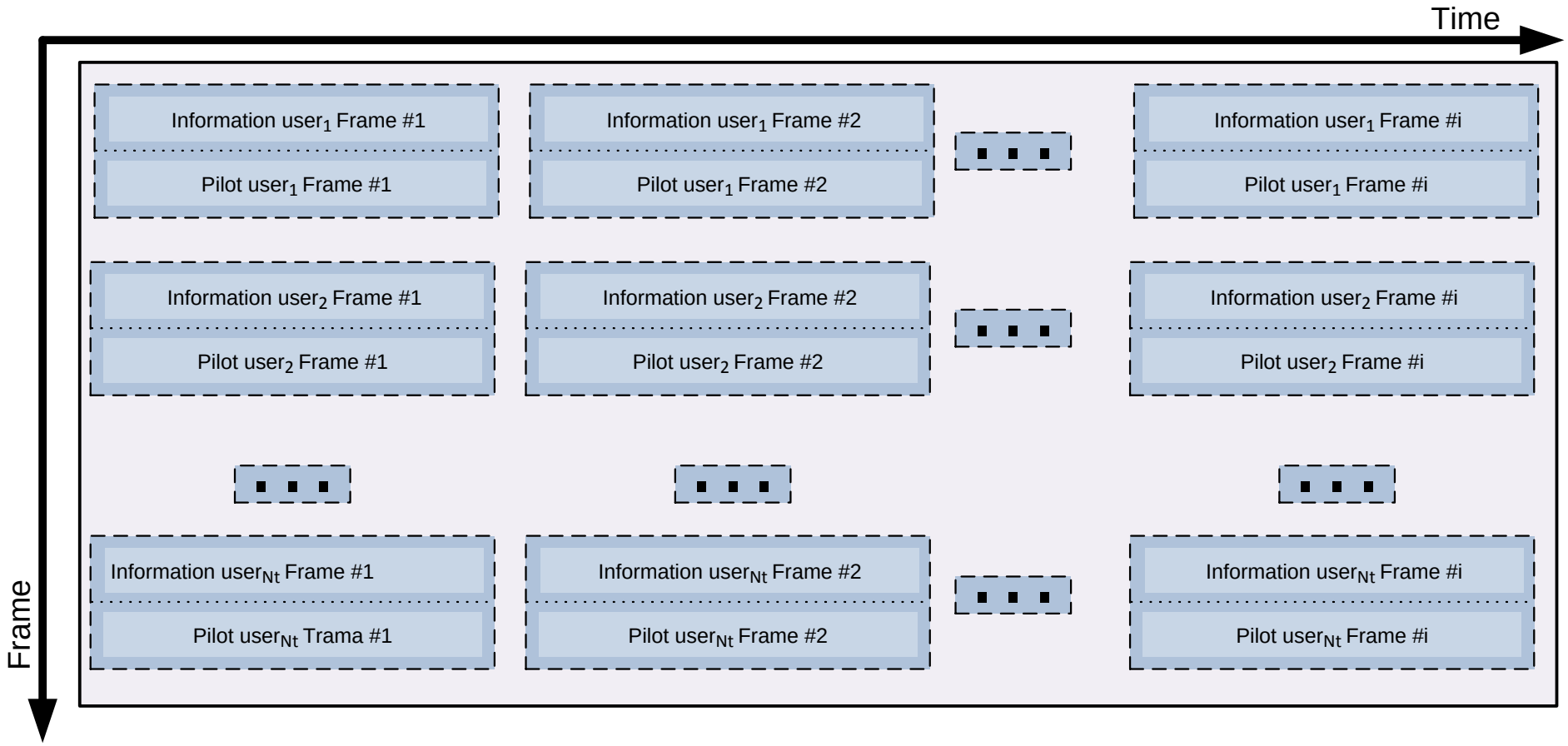


Figure 8: "Global structure of the frames with respect to time under the assumption that all users transmit at the same time"

3.3. Transmitter

The system consists of three well defined parts: transmitter, channel and receiver. Following there is going to be analyzed in detail all elements of the transmitter, from a description supported by the help of a diagram of the elements to a detailed analysis of each of elements.

The first step is to generate a series of bits with a bit generator which simulates the information transmitted by the user.

$$b_i[n] \epsilon \{0,1\}$$

These series of bits are passed through an encoder that adds some redundancy to protect against inter-symbolic interference and channel noise.

$$bc_i[n] = f(b_i[n])$$

Then the encoded bits are passed through a modulator BPSK, QPSK or 16-QAM to obtain a sequence of symbols. Following it is supposed that a BPSK modulation is used.

$$\alpha_i[n] = f(bc_i[n])$$

These symbols are repeated and multiplied by the chip associated with each user.

$$\sigma_i{}^{info}\,[n] = \sum_{l_1=0}^{SF-1} c_i{}^{info}\,[l_1] \cdot \alpha_i{}^{info}\,[n]$$

At the same time this process is done, a parallel process is done to obtain the pilot symbols using different chip sequence.

$$\sigma_i{}^{pilot}\,[n] = \sum_{l_2=0}^{SF-1} c_i{}^{pilot}\,[l_2] \cdot \alpha_i{}^{pilot}\,[n]$$

These two widened sequences are added.

$$\sigma_i[n] = \sigma_i{}^{info}\,[n] + \sigma_i{}^{pilot}\,[n]$$

Once the symbols to transmit have been proceeded Nc zeros are added between the symbols, having the following expression:

$$g[k] = \sigma_i \left\lfloor \frac{k}{\mathrm{Nc}+1} \right\rfloor \cdot \delta \left[\frac{k}{\mathrm{Nc}+1} \right]$$

Here are convolution is carried by a pulse in order to change the chip pulse shape.

$$s_i[k] = \sum_{n=-\infty}^{\infty} g[n] \cdot \varphi_c[k-n]$$

Where the pulse has a form (SRRC)

$$\varphi_c[k] = \frac{sen\left(\pi k(1-2\beta\cdot T)+8\beta\cdot k\cdot T\cdot cos\left(\pi\cdot k(1+2\beta\cdot T)\right)\right)}{\pi\cdot k\cdot T^{\frac{1}{2}}(1-(8\beta\cdot k\cdot T)^2)}$$

The frequency signal is upped in order to use the assigned broadband while protecting the signal in front of low frequency noise.

$$s_{out}[n] = Re\left\{s_i[k]\cdot e^{j2\pi\left(\frac{f_c}{f_s}\right)n}\right\}$$

The following steps are in case that an analog system should be implemented. Then, it would be need to convert the digital signal to analog as detailed in Figure 9: "Standard structure for D/A conversion".

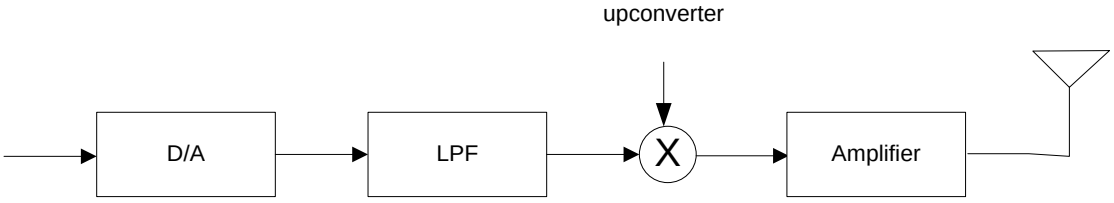


Figure 9: "Standard structure for D/A conversion"

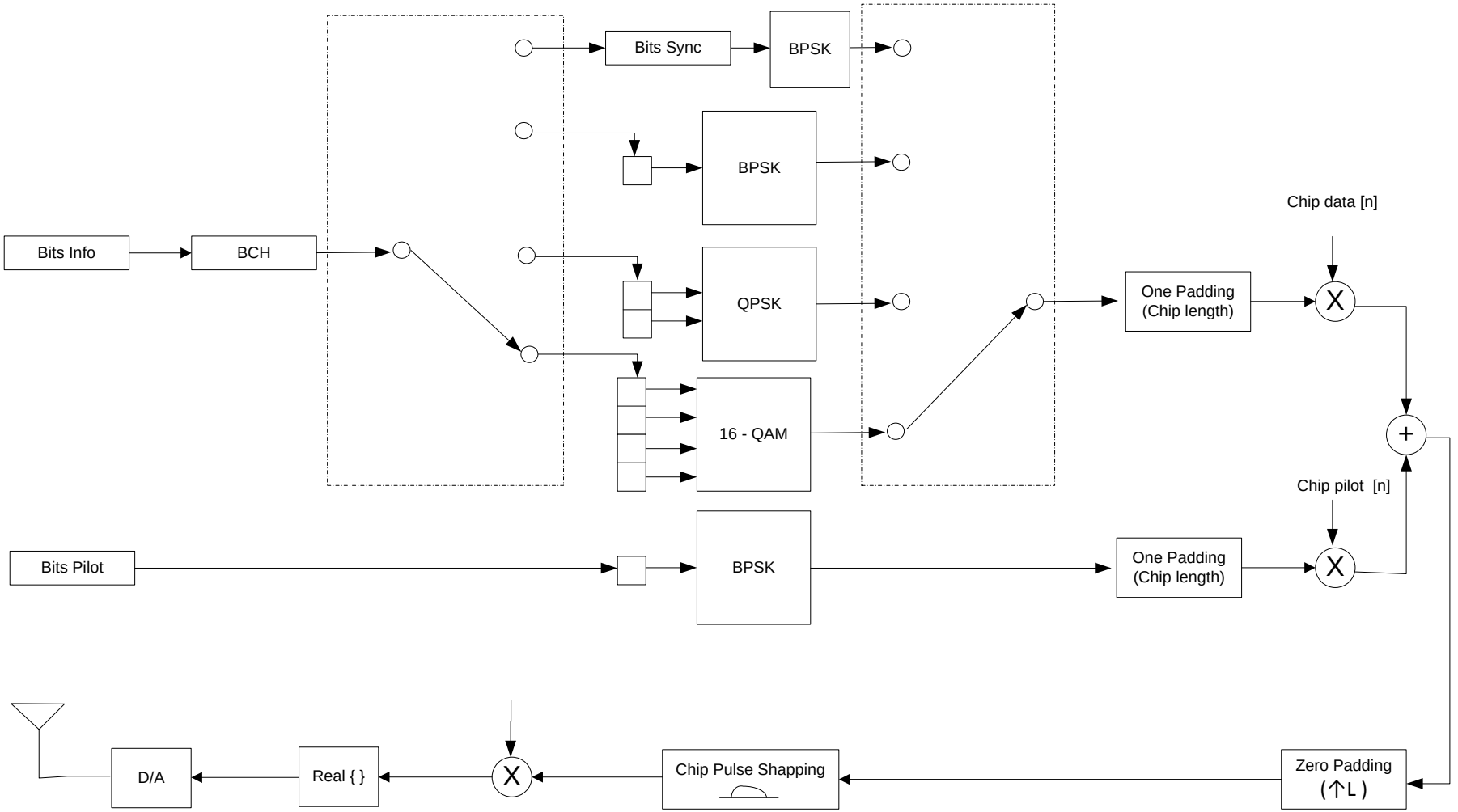


Figure 10: "Transmission chain for a user"

3.3.1. Encoder

3.3.1.1. Introduction

An encoder is a block that implements an algorithm which converts a sequence of bits into another sequence of bits with specific characteristics. In our case, we used an encoder to minimize the impact of the channel by correcting the errors introduced by the channel.

The main idea lies in the correction and data detection is to add redundancy to the message. This redundancy can be used by the receiver to check the consistency of the received message and correct message degradation.

Coding can be systematic or unsystematic. The systematic encoding is based on adding a bit fixed test obtained from deterministic algorithms. In contrast, non-systematic coding algorithms are based on obtaining a coded message.

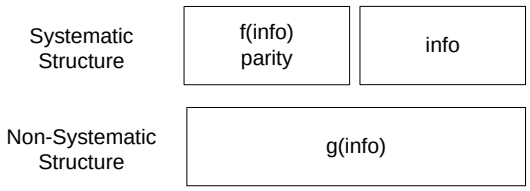


Figure 11: "Systematic vs no-systematic structure "

When choosing the encoder channel a communication characteristic needs to be studied. FEC[5] coding has been selected for being a suitable for the kind of channel implemented. Several algorithms can retransmit incorrect data(ARQ[6]) or a combination of data retransmission with a correction code (HARQ[7]) if the channel capacity cannot be estimated or changes constantly.

Following there is description of the main FEC since one of them will be implemented in this project. FEC are some systems that add redundancy to the system in order to use such redundancy to recover the original message.

There are two types of codes:

- **Convolutional codes**. Are codes that are processed bit by a bit, used mostly for hardware implementation as Viterbi code.
- **Block Code** are codes which are processed using bit sequences. The main block codes are:

[5] Forward Error Correction
[6] Automatic Repeat Request
[7] Hybrid Automatic Repeat Request

- Golay
- BCH
- Multidimensional parity check
- Hamming Codes
- Reed-Solomonor Turbo Codes
- LDPC[8]

Keep in mind that the turbo and LDPC codes are defined for convolution codes. So even if codes are included within a code block, these are a hybrid between convolutional and block codes.

Below in Figure 12: "BER-SNR comparing different types of channel" is a diagram which compares the main coders shows that the turbo code is the most efficient. LDPC results are very similar to the turbo code.

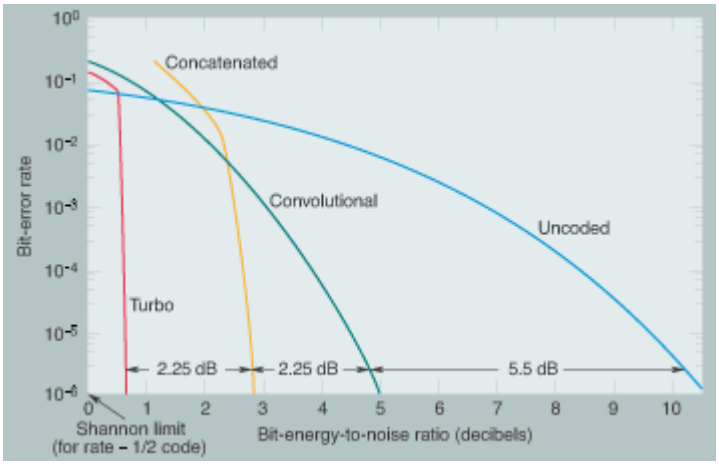


Figure 12: "BER-SNR comparing different types of channel"[9]

3.3.1.2. Motivation choice: BCH Codes

In this project we used the BCH codes as they are available in Matlab libraries. These are used in communications systems which need FEC error detection and correction of errors when the received signal can be susceptible to error or uncertainty.

The choice of such codes has been done for their properties of correctness of data, as well as easy implementation and low use of system resources.

[8] Low Density Parity Check Codes
[9] "Forward Error-Correction Coding" from Charles Wang, Dean Sklar, and Diana Johnson

3.3.1.3. Mathematical formulation

BCH encoders are implemented using a linear feedback shift register as shown in Figure 13: "Block diagram of the BCH encoder". Each input bits have K symbols, and the output have N symbols. The input message is moved to the left N-K positions and added to the first N-K parity bits positions. The parity symbols are calculated according to the module (N-K) of the message and g (x) which is a polynomial base.

$$C = W \cdot G$$

Where C is the encoding of information bits W longitude k and G is the matrix from the generator polynomial g (x)

$$G = \begin{bmatrix} g(x) \\ x \cdot g(x) \\ \vdots \\ x^{n-r-1} \cdot g(x) \end{bmatrix}$$

Indeed, it can also be represented with the following block diagram.

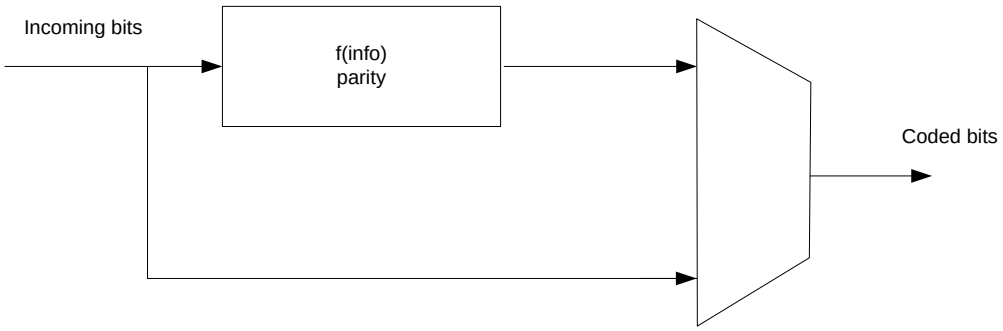


Figure 13: "Block diagram of the BCH encoder"

3.3.2. Modulator

3.3.2.1. Introduction

This modulator is a crucial process in the system. In this module the system will focus on digital modulation consisting in passing a series of bits into symbols. In all digital modulation, each of the phases, frequencies or amplitudes are assigned according to a unique pattern of binary bits. Normally, each phase, frequency or amplitude encodes have the same number of bits. Thus a certain number of bits generate a symbol.

The steps taken by the modulator for transmitting the information are:

1. Grouping the data into code words, one for each symbol to be transmitted
2. Map these words on attributes such as the amplitudes of the signals I and Q

The alphabet consists of $M = 2^N$ symbols; each symbol represents an N bit message[TSVI 2005]. There are 4 cases of fundamental digital modulation:

- **Phase Shift Keying (PSK).** This scheme is based on controlling the phase of the symbols generated. There are two types of deployment with the differential phase or the phase itself. In PSK symbols are chosen in a uniform angular distribution spread over a circle. This provides maximum phase separation and gives the best immunity to corruption of the signal. Each symbol is associated with a given energy can be fixed. Two typical cases are using two-phase BPSK and QPSK uses four phases as shown in Figure 14: "Constellation for BPSK and QPSK modulation using gray code".

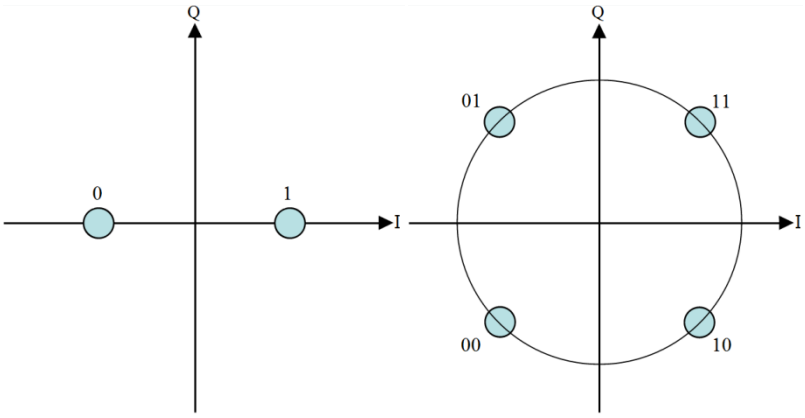


Figure 14: "Constellation for BPSK and QPSK modulation using gray code"

- **Quadruatre Amplitud Modulation** (QAM). It is a digital modulation technique that conveys data by modulating the carrier signal information in both amplitude and phase. This is achieved by modulating a single carrier, shifted 90 degrees the phase and amplitude.

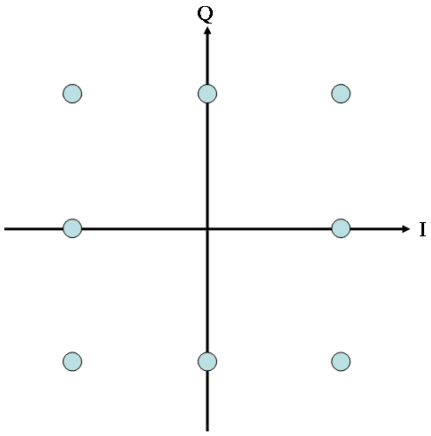


Figure 15: "Constellation from a 8-QAM modulation"

3.3.2.2. Motivation choice: BPSK, QPSK, 16QAM

We have chosen three different types of modulation to provide the system flexibility to channel interference. Thus, the BPSK modulation is easy to implement, allowing transmit signals to low SNR. 4QAM is somewhere quite efficient and can be used with a SNR average. While the 16QAM requires a high SNR and have different power transition levels, but is most efficient.

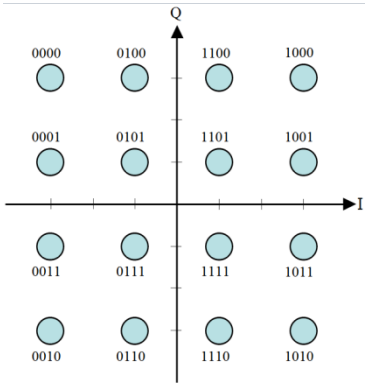


Figure 16: "Constellation from a 16-QAM modulation using Gray code"

3.3.2.3. Mathematical formulation

The mathematical expression for BPSK is:

$$s = \sqrt{\frac{E_b}{T_b}} cos(k \cdot \pi), parak = 0,1$$

Mathematical expression 4QAM modulation is given by next expression:

$$s = \frac{1}{2}\sqrt{\frac{E_b}{2 \cdot T_b}}\left(cos\left(k \cdot \frac{\pi}{2} + \frac{\pi}{4}\right) + j \cdot \sin\left(k \cdot \frac{\pi}{2} + \frac{\pi}{4}\right)\right), parak = 0,1,2,3$$

The mathematical expression for the 16QAM modulation is given by the following expression:

$$s = \sqrt{\frac{E_b}{80 \cdot T_b}}\left(cos\left(k \cdot \frac{\pi}{2} + \frac{\pi}{4}\right) + j \cdot \sin\left(k \cdot \frac{\pi}{2} + \frac{\pi}{4}\right)\right), parak = 0,1,2,3$$

$$s = \sqrt{\frac{3 \cdot E_b}{80 \cdot T_b}}\left(cos\left(k \cdot \frac{\pi}{2} + \frac{\pi}{4}\right) + j \cdot \sin\left(k \cdot \frac{\pi}{2} + \frac{\pi}{4}\right)\right), parak = 0,1,2,3$$

For implementation are calculated all possible combinations and makes a look-up table to optimize the function.

3.3.3. Spreader

3.3.3.1. Introduction

This module implements a DSSS[10] scheme. This variation uses more bandwidth than the information without modular, meaning it spreads the spectrum of the signal.

It consists in the modulation of the input signal with a sequence of chips. This sequence of chips has a form pseudo-noise that has certain properties very similar to white noise. A bit is composed by several chips.

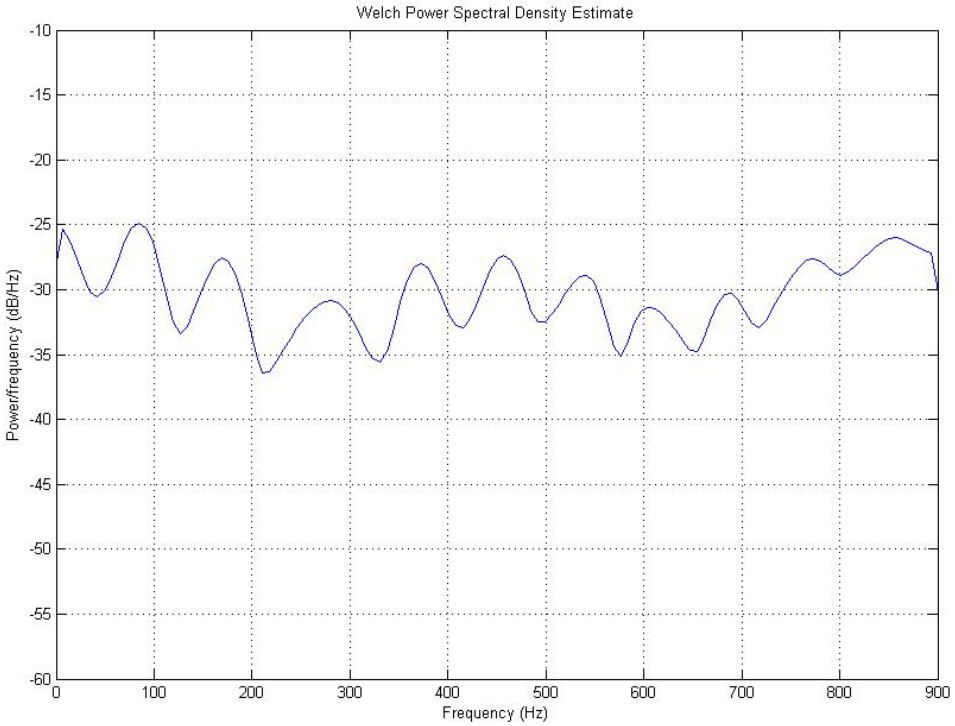


Figure 17: "Spectral Estimation of a pseudo-random Gold code where you see a tendency to white noise"

This modulation implies that chip sequence is known in advance by the receiver, and is used to reconstruct the signal sequence.

There are several types of pseudo-noise each with different properties [TONO 1992]. The most common ones are:

- **MLS**[11]. Bit sequences are generated by a linear shift registers. These sequences are periodic and undergo a series of binary sequences that can be played by the shift registers. The initial value of the records is indicated by an irreducible polynomial.

[10] Direct Sequence Spread Spectrum

They have a periodicity bigger than the re-alimented register thank to its feedback. It has the same properties as Hadamard codes.

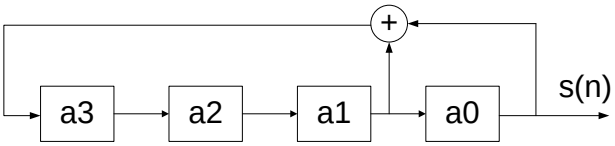


Figure 18: "Getting a sequence with noise properties through MLS

- **Gold codes**. Gold sequences are named after its discoverer. Consist of combining two m-sequences using a XOR, such that cross-correlation has only three values. The resulting sequence is a repetition of 2^m+1, instead of 2^m-1 as the original sequences.
- **Kasami codes**. Kasami sequences have optimal cross-correlation that approaches the theoretical limit of Welch.
- **Orthogonal codes**. These codes have zero cross correlation, but the problem is that they are not robust when adding a noise in the form of offset.
- **Walsh codes**. Walsh codes are generated by applying the Hadamard transform starting from scratch repeatedly.

The DSSS transmissions can multiply the information transmitted through the use of a signal "noise." This "noise" is a pseudorandom signal with values between 1 and -1, at a frequency much higher than the original sequence. Thus the signal is broadened spectrum signal occupies a wider bandwidth than the original signal.

The result seems to be white noise. From this apparent white noise, the signal can be recovered using the pseudo-random signal used for spreading it. This process is known as de-spreading.

3.3.3.2. Motivation choice: Spreader using Gold codes

This module has been implemented to share a single channel with multiple users, although the spreading also provides other benefits:

- Robustness against jamming
- Reduce the relationship between signal and background noise, allowing the transmission hide more easily.
- It provides a relative time between the transmitter and receiver.

[11] Maxium Linear Sequence

Regarding the pseudo-random noise signal has been chosen a Gold code for good cross-correlation and its robustness to offsets.

3.3.3.3. Mathematical formulation

It mainly consists on applying a Gold code to each user. There are multiples codes which are based on a common mathematical expression:

$$s[(n-1)\cdot(2^{m}+1)+i] = b[n]\cdot g_j[i]$$

Where $b[n]$is the bit sequence

$g_j(1..i..2^m+1)$ is the sequence of length 2m +1 gold by user j

$s[m]$ is the sequence after the spreader

3.3.4. Zero Padder

3.3.4.1. Introduction

The zero padder is the process of increasing the frequency of signal by including zeros between signal samples. The oversampling factor L is usually an integer or a fraction greater than unity. This factor multiplies the sampling frequency for sampling the output.

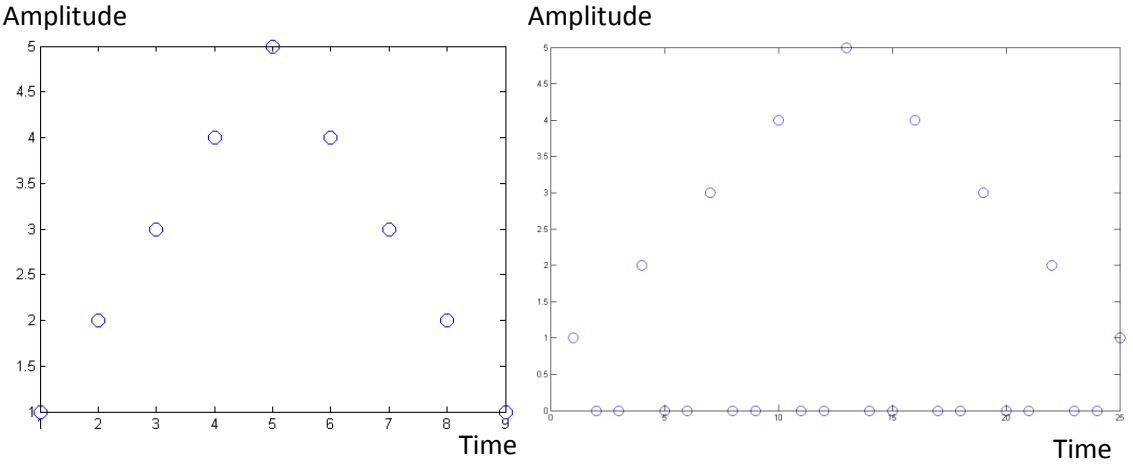


Figure 19: "Original signal and passed through the block zero padder"

This module is transparent to the Nyquist-Shannon theorem[12] as long as the original signal meets it.

[12] Nyquist's theorem shows that the exact reconstruction of a continuous periodic signal from baseband samples, it is mathematically possible if the signal is bandlimited and the sampling rate is more than twice its bandwidth.

3.3.4.2. Motivation choice

The addition of this module is used to reduce inter-symbolic interference.

3.3.4.3. Mathematical formulation

The zero padding is to apply the following mathematical expression.

$$g[k] = \left\{ f\left[\frac{k}{L}\right] if \frac{k}{L} isinteger \right\} \\ 0\ otherwise$$

Where: f (k) is the input signal

g (k) is the output signal

L is the sampling factor

3.3.5. Pulse Shaper

3.3.5.1. Introduction

The pulse shaper transmits a symbol (complex) over a pulse by appropriate amplitude in channels I & Q according to its specific constellation. The pulse shaper controls the shape of the power spectral density, under the restriction of ISI = 0. Spectrum signal is determined by the type of pulse shaper used in the transmission as shown in the following mathematical expression.

$$s_h(t) = s(t) \cdot h(t)$$

$$S_h(f) = S(f) * H(f)$$

Where: s(t) is the signal

h(t) is the filter

S(f) is the signal in frequency domain

H (f) is the filter in frequency domain

3.3.5.2. Motivation choice: “SRRC”

The objective of implementing this filter is limiting bandwidth transmission, thus reducing the inter-symbolic interference produced by the channel.

The filter used was a SRRC for their good properties. Several tests have been done to select this filter.

3.3.5.3. Mathematical formulation

We have chosen the Root Raised Cosine Square selecting parameters that fit correctly into our system.

The function is given by:

$$\varphi_c[k] = \frac{sin\left(\pi k(1-2\beta\cdot T)+8\beta\cdot k\cdot T\cdot cos\left(\pi\cdot k(1+2\beta\cdot T)\right)\right)}{\pi\cdot k\cdot T^{\frac{1}{2}}(1-(8\beta\cdot k\cdot T)^2)}$$

This function is obtained from raised cosine pulse.

$$h(t) = sinc\left(\frac{t}{T}\right)\frac{\cos(\pi\beta t)}{1-4\beta^2t^2} = \text{sinc}\left(\frac{t}{T}\right)\cdot p_\beta(t)$$

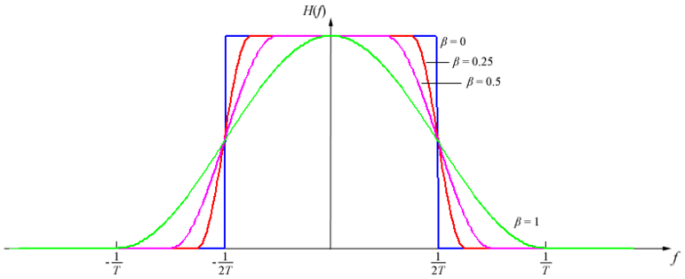


Figure 20: "Raised cosine amplitude for different parameters"

and its frequency expression is:

$$P_{RC}(f) = F\{h(t)\} = T\cdot\Pi(f\cdot T) * F\{p_\beta(t)\}$$

$$= \begin{Bmatrix} T & ,|f| \le \frac{1}{2T}-\beta \\ T\left[cos^2\left(\frac{\pi}{4\beta}\left(|f|-\frac{1}{2T}+\beta\right)\right)\right] & ,\frac{1}{2T}-\beta \le |f| \le \frac{1}{2T}+\beta \\ 0 & ,resto \end{Bmatrix}$$

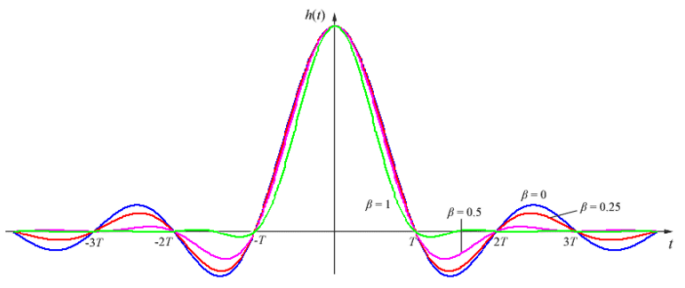


Figure 21: "Frequency transform of raised cosine with several parameters"

A way to get a neutral value-neutral basis (1) in the center is by square root to obtain the raised cosine pulse as we seek to:

$$P_{RC}(f)^{\frac{1}{2}} \cdot P_{RC}(f)^{\frac{1}{2}} = 1$$

Thus the function will be as follows:

$$\Phi_{\text{SRRC}}(f) = P_{RC}(f)^{\frac{1}{2}} = \begin{Bmatrix} T^{\frac{1}{2}} & ,|f| \leq \frac{1}{2T} - \beta \\ T^{\frac{1}{2}}\left[cos\left(\frac{\pi}{4\beta}\left(|f| - \frac{1}{2T} + \beta\right)\right)\right] & ,\frac{1}{2T} - \beta \leq |f| \leq \frac{1}{2T} + \beta \\ 0 & ,resto \end{Bmatrix}$$

Since its inverse Fourier transform:

$$\varphi_c(t) = \frac{sen\left(\pi t\left(\frac{1}{T} - 2\beta\right) + 8\beta \cdot t \cdot cos\left(\pi \cdot t\left(\frac{1}{T} + 2\beta\right)\right)\right)}{\pi \cdot T^{-\frac{1}{2}}(1 - (8\beta \cdot t)^2)}$$

Sampling this expression we obtain

$$\varphi_c[k] = \frac{sen\left(\pi k(1 - 2\beta \cdot T) + 8\beta \cdot k \cdot T \cdot cos\left(\pi \cdot k(1 + 2\beta \cdot T)\right)\right)}{\pi \cdot k \cdot T^{\frac{1}{2}}(1 - (8\beta \cdot k \cdot T)^2)}$$

3.3.6. Up converter

3.3.6.1. Introduction

Up converter places the low-pass signal in an appropriate range for the transmission, that transmission is determined according to the statutory licenses that grant the use of a spectrum.

The procedure introduces image frequency to twice the signal carrier. For this same reason that you need to filter out these frequencies images.

3.3.6.2. Motivation choice

The aim of the module is to shift the frequency modulation, as the actual communication devices are very noisy at low frequencies. In addition, this allows you to select a carrier frequency which transmit the communication

3.3.6.3. Mathematical formulation

This module follows the following mathematical expression that makes a frequency shift:

$$y(t) = x(t) \cdot \cos(2\pi f_c t)$$

Where y (t) is the output signal

x (t) is the equivalent complex input signal $x(t) = I(t) + j \cdot Q(t)$

Sampling this function we obtain

$$y[n] = x[n] \cdot \cos[2\pi f_c \cdot n \cdot T]$$

3.4. Channel

A channel is the medium through which signals travel carrying information between the transmitter and receiver. There are a multitude of channels and each channel has its specific properties:

- Nature of the signal which is capable of transmitting
- Bandwidth
- Noise generated

Electromagnetic signals can use multiple channels depending on the frequency of the transmitted signals, cables, vacuum (satellites), the atmosphere itself, among others [JAWI 1974]. This project is considered an empty channel.

Summing up, the signal at the output of channel filter that simulates the codes will no longer be orthogonal due to interference from other users.

$$\varphi_i'(t) = \varphi_i(t) * h_i(t)$$

$$\varphi_j'(t) = \varphi_j(t) * h_j(t)$$

$$\int \varphi_i'(t) \cdot \varphi_j'(t) \cdot dt \neq \delta[i-j]$$

Indeed, the following expression is true when there is an absence of synchronization and sequences are not orthogonal.

$$\int \varphi_i'(t-\tau_i) \cdot \varphi_j'(t-\tau_j) \cdot dt \neq 0 \; para \; \tau_i \neq \tau_j$$

In this case we focus on fading channels with AWGN.

3.4.1. Channel AWGN [13]

AWGN channel model is a simple linearly added white noise with a constant spectral density and a Gaussian distribution with input signal. Below is graphically in Figure 22: "Real spectral signal of white noise baseband" the main baseband signal characteristics, where BW is the bandwidth, fc is the center frequency and power is the white noise.

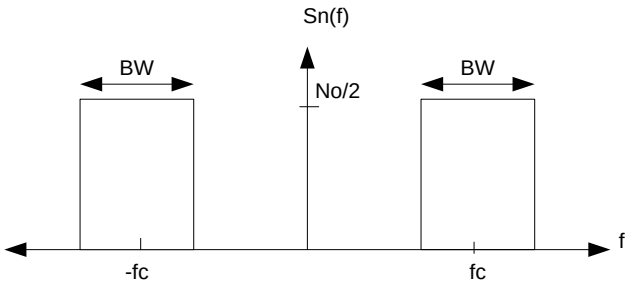


Figure 22: "Real spectral signal of white noise baseband"

This model allows an approach to analyze the system before considering dispersion, interference and frequency selective fading. The broadband Gaussian noise is used to model natural elements such as thermal noise. This model is used to level communications satellite, and together with other models such as the slow fading for modeling terrestrial communications.

3.4.2. Fading channels

The propagation of waves through wireless channels is a very complicated phenomenon characterized by various effects such as multipath and shadowing. In recent years there has been a great effort to obtain statistical models and characterization of these elements. We have obtained relatively simple statistical models and quite accurate fading channels. These models depend on the environment and the communication scenario [MASI 2000].

The main effect when a signal goes through a fading transmission is the fluctuation over time of amplitude and phase components. For coherent modulation, a fading effect in phases can have a negative impact on the reception making it impossible any communication. This effect has been corrected in this project. For non-coherent modulations, the phase information is not necessary for the reception and may well neglect their effect. So in the analysis of coherent and non-coherent modulation, an attention must be paid just to the amplitude or power, since

[13] Additive White Gaussian Noise

the phase has no effect, either because it has been already corrected or because it has no effect when the signal reception.

So the first classification of these models is the behavior of the powers, separating them into two types of models:

1. **Large-scale models**. These types of models explain the behavior of the powers at distances much greater than the wavelength range distances of kilometers. There are several different models including:
    a. Open space
    b. Okumura-Hata
    c. Blocking: Log-distance and Log-normal
2. **Small-scale models**. These types of models explain the behavior of the powers at distances comparable to the wavelength of the order of meters. In this project we will focus on implementing this type of models. These models are based on the type of multipath due to the Doppler Effect.

Following an explanation of basic knowledge about channels, then several models for frequency-flat fading channels and frequency-selective fading channels are described.

3.4.2.1. Background

Multipath channels arise when the signal reaches the receiver via multiple paths as shown in Figure 23: "Multipath signal", where a base station communicates with a mobile terminal receiving the signal directly over the reflections produced by two buildings [JEMI 2000].

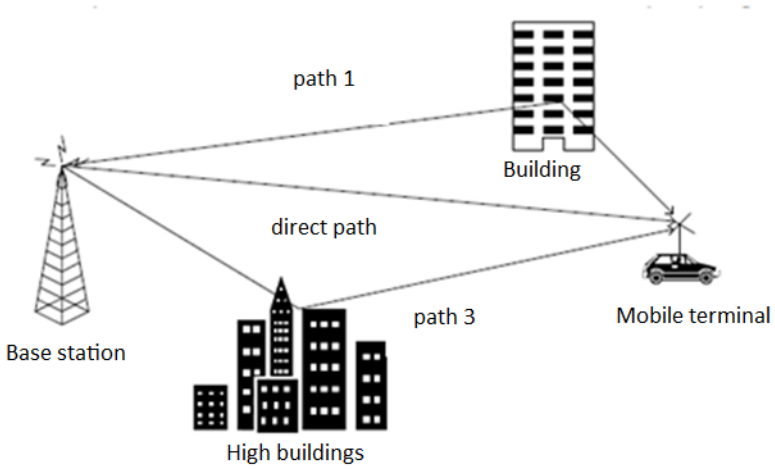


Figure 23: "Multipath signal"

This can be modeled as the following mathematical expression to the response:

$$H(z) = \sum_{l=0}^{L} H_l \cdot z^{-l}$$

Where L is the number of independent paths through which the signal arrives, and the attenuation H_l associated to that path.

Being the equivalent baseband channel between transmitter i and receiver antenna is given by the following expression:

$$h_{i,j}[n] = \sum_{l=0}^{L} \alpha_l \cdot \delta[\mathrm{n} - \mathrm{n_l}]$$

Where α_l usually modeled with Gaussian complex variables to simulate a frequency selective Rayleigh channel. Its variance is defined by the Power Delay. The Power Delay Profile (PDP) gives the intensity of the signal received through the multipath channel as a function of time delay (τ). The time delay (τ) is the difference between the time delay between direct visualization and the way through several paths.

$$\alpha_l = E[|\alpha_l|^2]$$

Table 1: "Correlation and spectral properties of different types of fading processes taken from Mason" is the most common models to model the environment and their specific approaches.

Table1: "Correlation and spectral properties of different types of fading processes taken from Mason"

Type of fading spectrum	Fading autocorrelation (ρ)	Normalized PSD[14]
Rectangular	$\frac{\sin 2\pi f_d T_S}{2\pi f_d T_S}$	${2f_d}^{-1}, \lvert f\rvert \le f_d$
Gaussian	$e^{-(\pi f_d T_S)^2}$	$e^{-\left(\left(\frac{f}{f_d}\right)^2\right)}\left(\sqrt{\pi} f_d\right)^{-1}$
Earth moving	$J_o(2\pi f_d T_S)$	$\left(\pi^2\left(f^2 - {f_d}^2\right)\right)^{-\frac{1}{2}}, \lvert f\rvert \le f_d$
First order Butterworth	$e^{-(2\pi \lvert f_d T_S\rvert)}$	$\left(\pi f_d\left(1 + \frac{f}{f_d}\right)^2\right)^{-1}$
Second order Butterworth	$e^{-\left(\frac{\pi \lvert f_d T_S\rvert}{\sqrt[2]{2}}\right)}$	$\left(1 + 16\left(\frac{f}{f_d}\right)^4\right)^{-1}$

[14] Power Spectral Density

3.4.2.2. Channel Characterization

3.4.2.2.1. Multipath propagation

The multipath propagation is due to constructive and destructive combination of delays caused by signal components with different amplitudes [MASI 2000]. These are mainly classified into two types: flat fading and frequency selective fading.

$$h_{i,j}[n] = \sum_{l=0}^{L} \alpha_l \cdot \delta[\mathrm{n} - \mathrm{n_l}]$$

An important feature of channel fading is that it is a selective in frequency. If all the spectral components of a transmitted signal are affected in the same way by the channel, this channel is said to be afrequency non-selective or flat fading. This is the case of systems where the bandwidth of the transmitted signal is much smaller than the coherent bandwidth f_c. This bandwidth is measured by the frequency range for which the fading is correlated and is defined as the frequency bandwidth over which the correlation function of two samples of a response channel on the same time but at different frequencies have a value similar.

On the other hand, if the spectral components of the transmitted signal are affected in different frequency and phase, it is said to be a frequency selective fading causing inter-symbolic interference and not keeping the transmitted signal spectrum.

In Figure 24: "Flat Fading vs. Frequency Selective Fading" shows the main feature where Bc is the bandwidth of the channel, and B is the bandwidth of the signal.

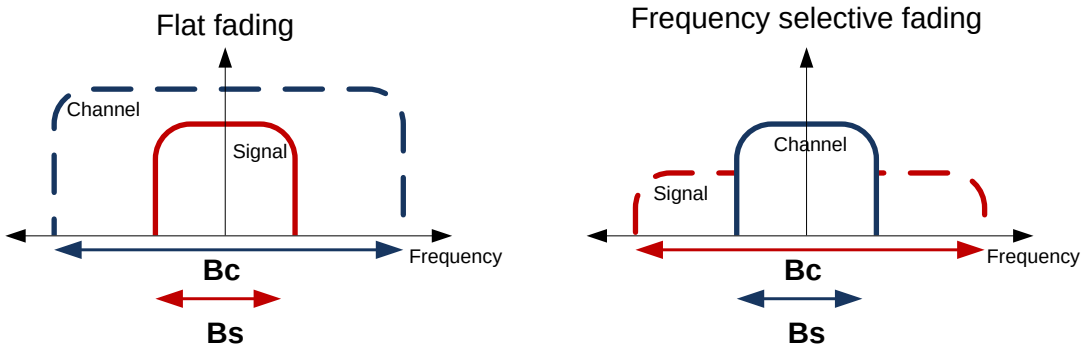


Figure 24: "Flat Fading vs. Frequency Selective Fading"

3.4.2.2.2. Doppler dispersion

The Doppler spread is another differentiator. There are two types of fading according to it: slow and fast fading. The main distinction is given by the coherence time $Tc = \frac{1}{Bc}$ which measuresthe time period over which the fading process is correlated. It is said to be slow fading if symbol time Ts is smaller than the coherence time, if not it is considered a fast fading. In Figure 25: "Fast fading vs slow fading" compares the bandwidth of the signal (B_C) with the Doppler bandwidth (B_D) allowing seeing the principal distinction between high dispersion fading fast fading and slow fading with low scattering fading.

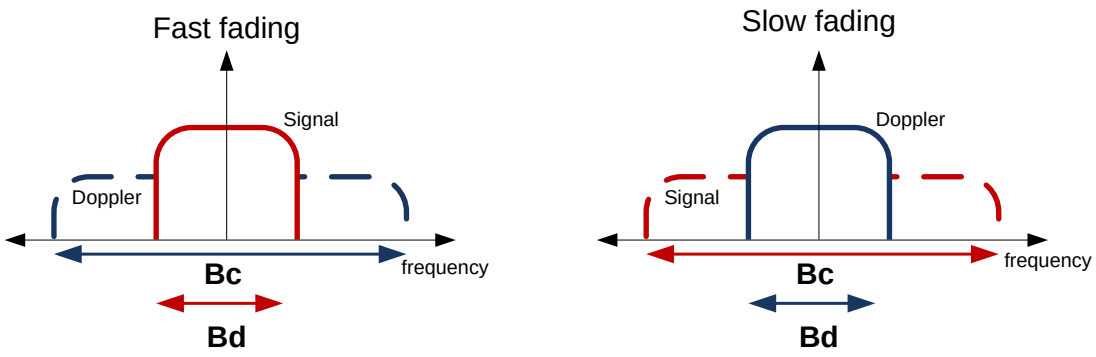


Figure 25: "Fast fading vs slow fading"

The effect of a fast fading channel is a symbol decorrelation over another symbol. One must consider this variation of the fading channel for an interval of one symbol to the next in order to compensate this type of error and consider receiver communication elements' decisions based on an observation of two or more signal symbols. This is done with a series of correlated models which mainly depends on specific propagation of the environment in a particular scenario.

In channels with slow fading, a fading of a certain level may affect successive symbols, leading to error burst. So, if this wants to be avoided, the channel needs to be estimated by adding a header which the receiver knows a priori.

3.4.3.Implemented model

When a signal is propagated by a frequency selective fading channel spectrum is affected by the channel transfer function, it results on a dispersion in time of the signal. This type of fading can be modeled with a linear filter characterized by the following impulse response:

$$h(t) = \sum_{l=0}^{L_p} \alpha_l \cdot e^{-j\theta_l} \cdot \delta(t - \tau_l)$$

L_p is the number of possible realizations δ is a Dirac delta, l is the channel index θ_l, α_l and τ_l are random variables representing the phase, amplitude and delay of the channel.

This model assumes a number of features and simplifications of the channel:

- It has been assumed slow fading
  - L_p is constant for a certain length of time longer than the symbol.
  - θ_l, α_l and τ_l are all constants for a range longer than the symbol time.
- It is assumed that all constants are generated so they have a negligible cross-correlation, which can be assumed as statistically independent.
- It is assumed that the noise disturbs a AWGN source, after passing through the canal.

3.5. Receiver

The next step is the reception at the receiver. A reciprocal process to the transmitter is perfume in order to recover the transmitted signals for each user. The received signal shifts the signal followed by a low pass filter to remove noise outside the band. The baseband signal is filtered by the matched filter receiver, powered by a synchronizer that provides the ideal moment to recover the signal. The last step is to recover the signal consists of a decimator to symbol frequency, followed by an estimate of the channel, passed through a detector and demodulator.

Note that the analog part has been omitted in this case since we are working on a digital environment, Matlab.

The first step is to pass the received signal to baseband, where $s_{in}[n]$ is a complex signal.

$$r_{in}[n] = Re\left\{(s_{in}[n] + w[n]) \cdot e^{j2\pi\left(-\frac{f_c}{f_s}\right)n}\right\}$$

A signal is then filtered to eliminate interference and noise that are outside the signal band, hence minimizing noise.

$$r[n] = LPF\{r_{in}[n]\}$$

Thus, the received signal is the sum of all users plus noise band associated with each user.

$$r[n] = \sum_{i=1}^{N} s_i[n]' + \overline{w_i[n]}$$

Where each user component breaks down the symbols that have been passed that have zeros between the displaced pulses.

$$s_i[n]' = \sum_{l=-\infty}^{\infty} g[l] \cdot \varphi_i'[n-l]$$

$$\overline{w[n]} = \sum_{i=1}^{N} \overline{w_i[n]} = Re\left\{w[n] \cdot e^{j2\pi\left(-\frac{f_c}{f_s}\right)n}\right\}$$

The signal is convolved with the reflected pulse in order to recover, thus obtaining.

$$dn[n] = r[n] * \varphi_i'[-n] = \sum_{n=\infty}^{n=-\infty} \left(\sum_{i=1}^{N} s_i[k]' + \overline{w_i[k]}\right) \cdot \varphi_i[k-n]$$

$$= \sum_{n=\infty}^{n=-\infty} \left(\sum_{i=1}^{N} s_i[k]' \cdot \varphi_i[k-n] + \overline{w_i[k]} \cdot \varphi_i[k-n]\right)$$

A key step is obtaining a good synchronization to correlate with a signal pattern. Then, a down-sampler is done obtaining the coefficients chip with its associated value:

$$y[n] = d_n[n]|_{downsampled} = \sum_{i=1}^{N} \left(\sum_{l=1}^{N} \rho_{il} \cdot \sigma_{ml}\,[n] + \beta_i[n]\right)$$

So, the values associated with each user are pilot sequence and information (σ_{ml}) by a factor ρ_{il} marked by the channel, plus an associated noise.

$$y_i[n] = \sum_{l=1}^{N} \rho_{il} \cdot \sigma_{ml}\,[n] + \beta_i[n]$$

This expression can be represented by a formulation as follows:

$$\begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix} = \begin{bmatrix} \rho_{11} & \cdots & \rho_{1N} \\ \vdots & \ddots & \vdots \\ \rho_{N1} & \cdots & \rho_{NN} \end{bmatrix} \cdot \begin{bmatrix} \sigma_1 \\ \vdots \\ \sigma_N \end{bmatrix} + \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_N \end{bmatrix}$$

$$\overline{\boldsymbol{y}} = \overline{\overline{\boldsymbol{R_s}}} \cdot \overline{\boldsymbol{s_m}} + \overline{\boldsymbol{n}}$$

Whereas it has been defined as:

$$\overline{\boldsymbol{y}} = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix}$$

$$\overline{\overline{\boldsymbol{R_s}}} = \begin{bmatrix} \rho_{11} & \cdots & \rho_{1N} \\ \vdots & \ddots & \vdots \\ \rho_{N1} & \cdots & \rho_{NN} \end{bmatrix}$$

$$\overline{\boldsymbol{s_m}} = \begin{bmatrix} \sigma_1 \\ \vdots \\ \sigma_N \end{bmatrix}$$

$$\overline{\boldsymbol{n}} = \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_N \end{bmatrix}$$

To minimize the impact of channel signal, the pilot signal is used to estimate the channel in a chip obtaining $\overline{\overline{R_s}}$. Then using this matrix, equalization is done to information data.

$$\bar{x} = \overline{\overline{R_s}}^{-1} \cdot \bar{y} = \overline{\overline{R_s}}^{-1} \cdot \left(\overline{\overline{R_s}} \cdot \overline{s_m} + \bar{n}\right) = \overline{s_m} + \overline{\overline{R_s}}^{-1} \cdot \bar{n}$$

The last step is to get the bits through a quantification of $\bar{x}$

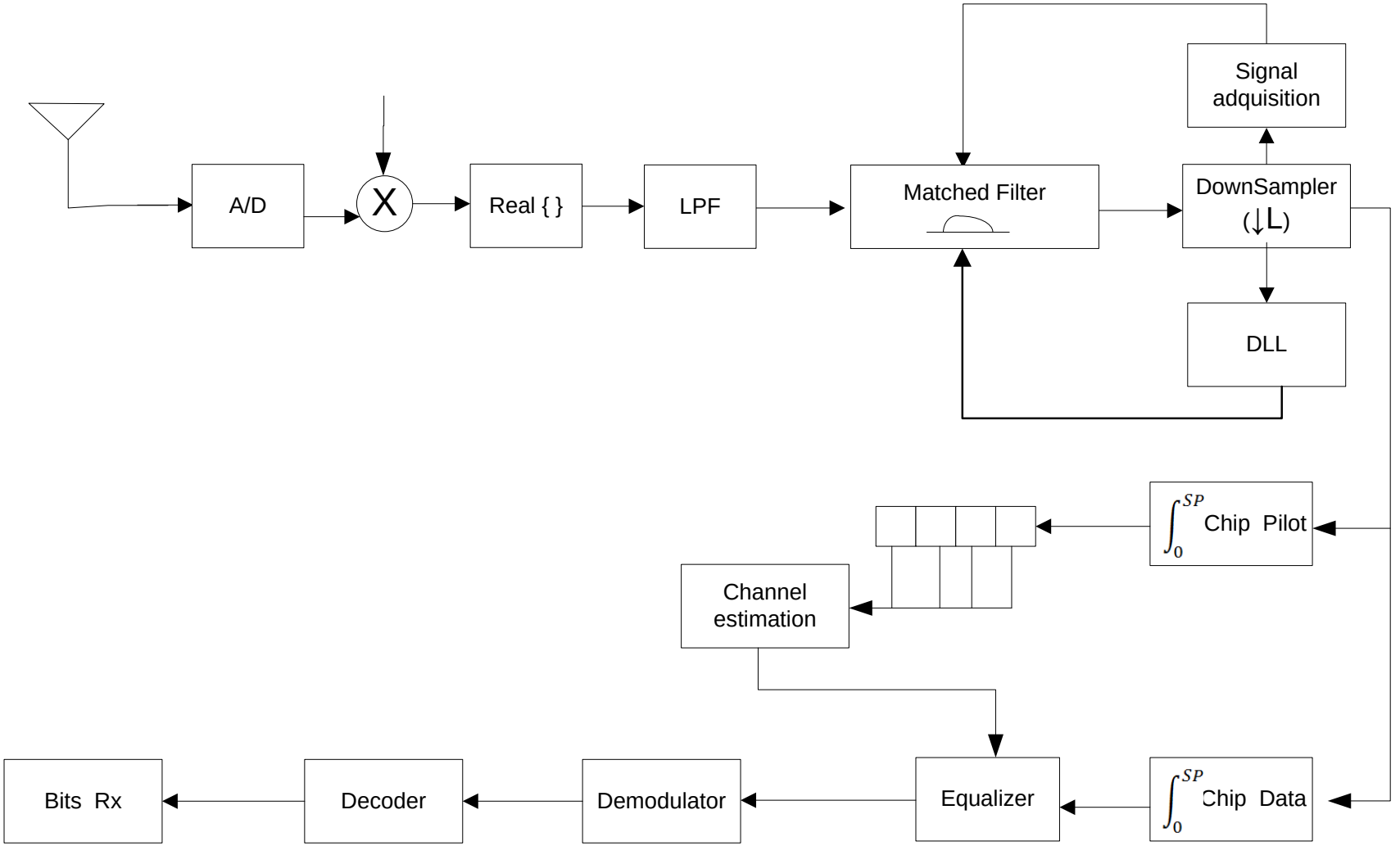


Figure 26: "Receptor chain for one user"

3.5.1. Low Pass Filter

3.5.1.1. Introduction

Low pass filter is a block that can pass frequencies with a low frequency signal while attenuating the higher frequencies from a cutoff frequency. The attenuation introduced by the filter depends on the type of filter used.

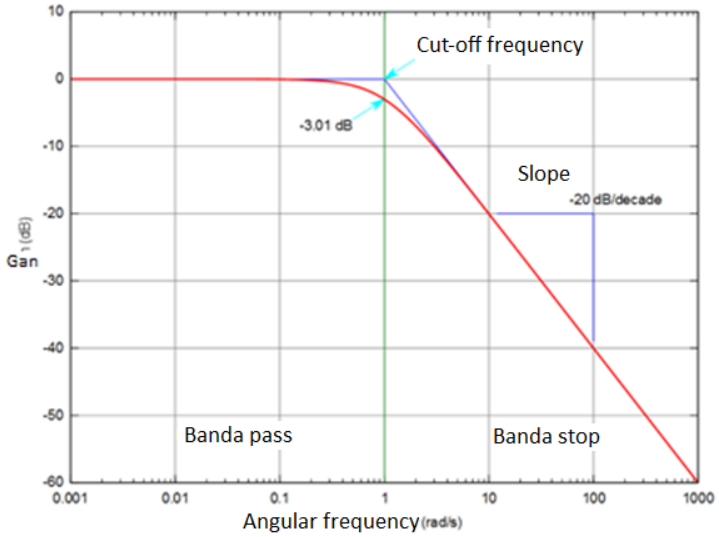


Figure 27: "Gain in a low pass filter with a first order approximation"

The ideal filter is a rectangular frequency response which is impossible to achieve because it requires infinite length signals, it is approximated (FIR). In addition to the filter that is a temporary sync requires all past and future values of the signal. This can be done by assuming that the digital signal is finite, so it has zero extensions in the past and the future or with the assumption that the signal is repetitive.

Real filters for real-time applications require an ideal filter approximated by truncation of the signal and delay to make causal filter.

3.5.1.2. Motivation choice: FIR

The addition of a low pass filter is to eliminate noise introduced by the channel, and removing pictures frequencies introduced by the upconverter and downconverter.

3.5.1.3. Mathematical formulation

To obtain a low pass filter has been approximated by a FIR filter. The impulse response has order N and last N +1, then all the values are zeros. Any signal x (n) through a filter is convoluted by it with the filter settings b_i, obtaining y (n)

$$\overline{y[n]} = \sum_{i=0}^{N} b_i \cdot x[n-i]$$

Using fir1, a Matlab function that implements a classic method of FIR filter with a Hamming window with specific cutoff frequency determined.

3.5.2. Downconverter

3.5.2.1. Introduction

This module is the analogue of the upconverter and aims to pass the band-pass signal to baseband.

This procedure also introduces a dual-frequency component of the carrier. This is the main reason why you need to filter out these images in a low-pass filter.

3.5.2.2. Motivation choice

The aim of the module is to shift the frequency modulation, as the actual communication devices are very noisy at low frequencies. Indeed, this allows to select a carrier frequency which transmits the communication.

3.5.2.3. Mathematical formulation

This module follows the following mathematical expression that makes a frequency shift:

$$y(t) = x(t) \cdot \cos(2\pi f_c t)$$

Where y (t) is the output signal

x (t) is the input signal

The equivalent sample to sample is

$$y[n] = x[n] \cdot cos[2\pi f_c \cdot n \cdot T]$$

3.5.3. Matched filter

3.5.3.1. Introduction

A matched filter is obtained by correlating a known signal with an unknown signal to detect signal presence known on the unknown signal.

The matched filter is the linear optimal filter to maximize signal to noise ratio (SNR) in the presence of additive stochastic noise.

The main uses of matched filters are in radar where a known signal is sent out, and the reflected signal is examined for common elements of the outgoing signal. They are also used in communications such as CDMA, because they are the basis of the receiver.

3.5.3.2. Motivation choice

The choice is minimizing the additive noise signal, thus maximizing the signal to noise ratio (SNR) at the point of entry to the decoder.

3.5.3.3. Mathematical formulation

Following there is going to be a reasoning for matched filter to minimize the additive noise.

$$y[n] = \sum_{k=-\infty}^{\infty} h[n-k] \cdot x[k]$$

Suppose the input signal is added an additive noise $x = s + v$, and$R_v = E\{vv^H\}$

$$y = \sum_{k=-\infty}^{\infty} h^*[k] \cdot x[k] = h^H x = h^H s + h^H v = y_s + y_v$$

Minimize the SNR, we obtain

$$SNR = \frac{|y_s|^2}{E\{|y_v|^2\}} = \frac{|h^H s|^2}{E\left\{|h^H v|^2\right\}} = \frac{|h^H s|^2}{E\left\{(h^H v)(h^H v)^H\right\}} = \frac{|h^H s|^2}{h^H E\{vv^H\} h} = \frac{|h^H s|^2}{h^H R_v h}$$

$$= \frac{\left(\left|\left(R_v^{\frac{1}{2}} h\right)^H \left(R_v^{-\frac{1}{2}} s\right)\right|^2\right)}{h^H R_v h}$$

So applying the Cauchy-Schwarz inequality ($|a^H b|^2 \leq (a^H a)(b^H b)$) it is obtained

$$\frac{\left(\left|\left(R_v^{\frac{1}{2}} h\right)^H \left(R_v^{-\frac{1}{2}} s\right)\right|^2\right)}{h^H R_v h} \leq \frac{\left(\left(\left(R_v^{\frac{1}{2}} h\right)^H \left(R_v^{\frac{1}{2}} h\right)\right)\left(\left(R_v^{-\frac{1}{2}} s\right)^H \left(R_v^{-\frac{1}{2}} s\right)\right)\right)}{\left(R_v^{\frac{1}{2}} h\right)^H \left(R_v^{\frac{1}{2}} h\right)}$$

$$= \frac{\left|\left(R_v^{\frac{1}{2}} h\right)^H \left(R_v^{-\frac{1}{2}} s\right)\right|^2}{\left(R_v^{\frac{1}{2}} h\right)^H \left(R_v^{\frac{1}{2}} h\right)}$$

According to the theorem, the limit will hold when$a =\propto b$

$$R_v{}^{\frac{1}{2}}h = \propto R_v{}^{-\frac{1}{2}}s$$

Thus the value of the filter is as follows

$$h = \propto R_v{}^{-1}s$$

Getting a SNR

$$SNR = \frac{\propto^2 \left|\left(R_v{}^{-\frac{1}{2}}s\right)^H \left(R_v{}^{-\frac{1}{2}}s\right)\right|^2}{\propto^2 \left(R_v{}^{-\frac{1}{2}}s\right)^H \left(R_v{}^{-\frac{1}{2}}s\right)} = \frac{\left|s^H R_v{}^{-1}s\right|^2}{s^H R_v{}^{-1}s} = s^H R_v{}^{-1}s$$

Assuming the additive noise power is 1

$$E\{|y_v|^2\} = E\left\{\left|h^H v\right|^2\right\} = E\left\{\left|\left(\propto R_v{}^{-1}s\right)^H v\right|^2\right\} = \propto^2 s^H R_v{}^{-1}s = 1$$

$$\propto = \frac{1}{\sqrt[2]{s^H R_v{}^{-1}s}}$$

This procedure is equivalent to the convolution of an unknown signal with the conjugate of the original signal, making the cross-correlation.

$$y = h^H x = \left(\frac{1}{\sqrt[2]{s^H R_v{}^{-1}s}} R_v{}^{-\frac{1}{2}}s\right)^H x$$

3.5.4. Synchronizer

3.5.4.1. Introduction

This module attacks the problem of determining the correct time to sample the signal using a sliding window [HEME 1997].

This block needs a training sequence which should be previously known before starting transmitting. The system input is the output of matched filter by choosing one of each F samples to compensate for the oversampled.

3.5.4.2. Motivation choice

It has been used the normalized cross-correlation as a measure of similarity between two signals in terms of a time lag is applied to one of them.

The normalized correlation gives a value between 0 and 1, not dependent on the strength of the signal used.

3.5.4.3. Mathematical formulation

The algorithm implemented to find the highest value of the normalized correlation between the received signal and the pilot signal previously known, namely:

$$n_{samp} = arg\left\{\max_{t_{samp} \in (n_1, n_2)} |\rho(t_{samp})|\right\}$$

Where n_1 is the first sign of the analyzed signal.

n_2 is the first sign of the analyzed signal.

n_{samp} is the sampling instant.

L is the length of training sequence.

The normalized correlation coefficient is calculated following expression:

$$\rho(n_i) = \frac{\sum_{k=0}^{L-1}(s(kF) - \mu_s)(c(k) - \mu_c)}{\sigma^2}$$

Where s is the input signal

c is the training sequence.

μ_s is the average of the input signal (in our case is zero)

μ_c is the average of the training sequence (in our case is zero)

F is the oversampling factor.

σ is the deviation of the two signals.

3.5.5. Delay-Locked Loop

3.5.5.1. Introduction

This module is done once the receiver has been synchronized within a time chip. This block synchronizes the system with more precision. It has also to be correct temporary deviations of the signal can be induced by a difference in clock frequency transmitter and transmitter or the channel itself.

3.5.5.2. Motivation choice

DLL is a robust method that keeps the system synchronous , protected in front of delays such as a delay-induced cumulative and have a different clock frequency between the transmitter and receiver or by the type of channel.

3.5.5.3. Mathematical formulation

This block is based on comparing the value of a symbol that is closest to the theoretical. In this case we have implemented a monitoring tool that compares three symbols that let you know if you move, stop or reverse the signal.

This task is part of the 6 samples captured on a chip that reach the matched filter. There are three samples processed for the pilot sequence:

- One in case you do not have displacement (t = t)
- Another in the case had been delayed signal (t = t-1)
- Another in the event that the signal had been advanced (t = t +1)

Once these signals have been processed compared to the theoretical value and select the one that comes closest to this value. Updating the time for the next symbol.

Note that a symbol is used to check whether to advance, hold or delay the signal as this compensates for possible deviations.

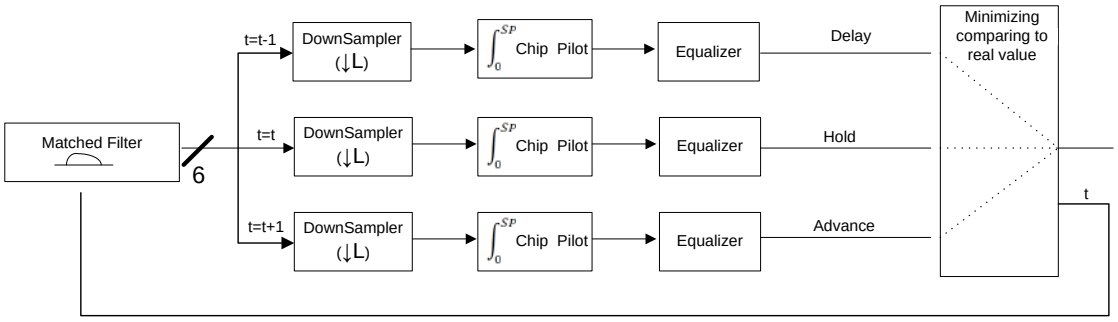


Figure 28: "DLL structure"

3.5.6. Downsampler

3.5.6.1. Introduction

The downsampling is the process of reducing the sampling frequency of a signal. This procedure is normally used to reduce the size of information processing.

This process is done as the analogue of the upsampler in the receiver chaining.

3.5.6.2. Motivation choice

The addition of this module is to recover the signal for further processing, this module is compensated upsampler which aims to reduce inter-symbolic noise component.

3.5.6.3. Mathematical formulation

Downsampler is to keep one of each L data, as the following mathematical expression as we are always real numbers.

$$g[k] = f[k \cdot L]$$

Where: $f[k]$ is the input signal

$g[k]$ is the output signal

L is the downsampling factor

3.5.7. Equalizer

3.5.7.1. Introduction

An equalizer is a filter, usually adjustable, designed to compensate the frequency response of a system.

An equalizer can be designed with a series of chained filters; filters are band pass or high pass and low pass filters.

3.5.7.2. Motivation choice: FIR

The addition of this module is to offset the effects introduced by the channel. In our case we used an FIR filter to estimate the effect of this channel.

3.5.7.3. Mathematical formulation

The impulse response has order N and last N +1, and then all the values are zero. Any signal x(n) through a filter is by convolving h_i filter settings, obtaining y (n)

$$\overline{y[n]} = \sum_{i=0}^{N} h_i \cdot x[n-i]$$

Suppose we have a set of input values X and output Y:

$$Y = \begin{pmatrix} y[0] \\ y[1] \\ \vdots \\ y[N] \end{pmatrix}$$

$$X = \begin{pmatrix} x[0] \\ x[1] \\ \vdots \\ x[N] \end{pmatrix}$$

Whereas H is:

$$H = \begin{pmatrix} h[0] & h[1] & \dots & h[N] \\ 0 & h[0] & \dots & h[N-1] \\ 0 & 0 & \dots & h[1] \\ 0 & 0 & \dots & h[0] \end{pmatrix}$$

Thus the above expression is in and in the following expression:

$$\bar{Y} = H^T X$$

To obtain the most similar will minimize the MSE

$$MSE(Y - \bar{Y}) = E\{(Y - H^T X)^2\}$$

$$\frac{\delta}{\delta(h(k))} MSE(Y - \bar{Y}) = 2\, E\{(H^T X - Y)Y(k)\} = 0 \quad para\ k = 0..N$$

$$E\{(H^T X - Y)Y\} = 0$$

$$E\{H^T XY\} - E\{YY\} = 0$$

If$r_{xy} = E\{XY\}$ y $R_Y = E\{YY^T\}$, then the previous expression is as follows

$$R_Y H = r_{xy}$$

So

$$\boldsymbol{H_{opt} = R_Y^{-1} * r_{xy}}$$

3.5.8. Despreader

3.5.8.1. Introduction

The despreader is the analogue module of the speader. It consists on retrieving user information from a signal where there are multiple users. This process involves using the pseudo-random sequence multiplied by the signal to obtain the original user information.

The transmission and reception of streams must be synchronized for the despreader to work properly. This requires that the receiver is synchronized to its sequence with the sequence of the transmitter through some process of search time.

The spreader and despreader introduce an increase in SNR related to the length of the sequence used.

3.5.8.2. Motivation choice

The addition of this module are the same as the spreader, the despreader is responsible to compensate the spread module for each user and pseudo random sequence.

3.5.8.3. Mathematical formulation

Despreader is based on a very basic pseudo-random sequences have very high autocorrelation while its cross-correlation is very low. Thus the expression that determines its value is:

$$b_j[n] = \sum_{i=0}^{2^m+1} s[(n-1)\cdot(2^m+1)+i]\cdot g_j[i]$$

Where: $b_j[n]$ is the bit sequence of user # j

gj [1 .. i.. 2m +1] is the sequence of length 2^m +1gold by user j

s [m] is the sequence after the spreader

3.5.9. Demodulator

3.5.9.1. Introduction

This module is to identify which symbols have been received at the receiver after being distorted by the channel. In order to make such assignment is appropriate to divide all possible values in regions assigned to each symbol.

The following illustrates the division of regions marked by the red line for equiprobable symbols and Gaussian noise distribution in Figure 29: "Separation regions if AWGN distribution for BPSK", Figure 30: "Separation regions if AWGN distribution for QPSK"and Figure 31: "Separation regions if AWGN distribution for 16-QAM".

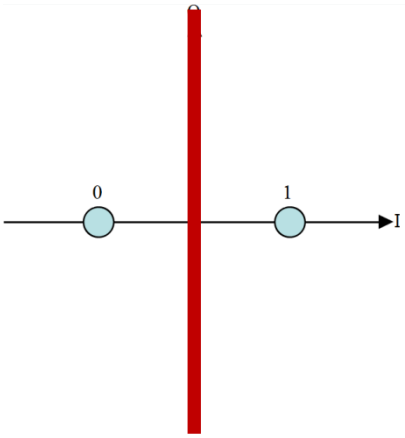


Figure 29: "Separation regions if AWGN distribution for BPSK"

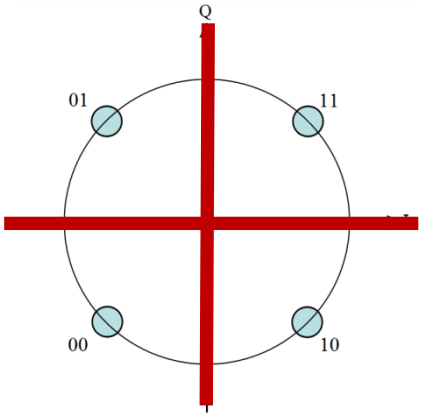


Figure 30: "Separation regions if AWGN distribution for QPSK"

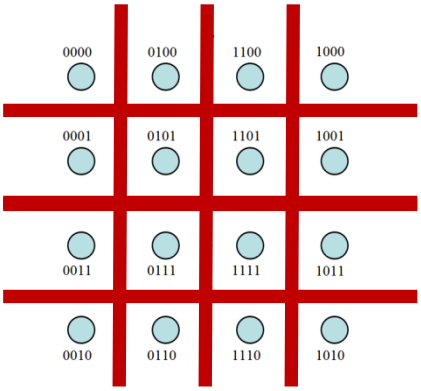


Figure 31: "Separation regions if AWGN distribution for 16-QAM"

3.5.9.2. Motivation choice

The reason for this module is assigning soft symbols or symbols that have not yet been assigned to a value for hard symbols once the receiver has a value for the soft symbols.

3.5.9.3. Mathematical formulation

The lines that minimize the global error need to be found by separating these regions. Following there are several assumptions that have been done:

- The bits are equally likely, which are also symbols.
- The additive noise is Gaussian.

As the previous modules do not affect the signal negatively, but prepare the received signal for this step, we can assume that at this point is the signal of the form:

$$y = x + n$$

The density function pdf of the additive Gaussian noise has the following mathematical expression.

$$f(x) = \frac{1}{\sigma^2\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

Where $\mu = 0\ and \sigma^2 = \frac{N_o}{2}$.is white noise of the form $N \sim \left(0, \frac{N_o}{2}\right)$

We now discuss the case for BPSK case can be extrapolated to all others, assuming that the energy is $E_b = 1$.

There are just two symbols with the expression to find the line that minimizes the error is easier to find a line in this case.

$$P_{ok1} = P_{ok2}$$

The probability that a symbol is correct will be the cumulative probability that the symbol is in the correct area $P_{s1}(X \leq \gamma)$ equal to the other symbol $P_{s2}(X \leq \gamma)$

$$P_{s1}(X \leq \gamma) = P_{s2}(X \leq \gamma)$$

So the density function for the symbol -1, associated with the bit 0 is

$$E\{y\} = E\{x + n\} = E\{x\} + E\{n\} = -1 + 0 = -1$$

$$f(x) = \frac{1}{\sigma\sqrt[2]{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} con \mu = -1\ y \sigma^2 = \frac{N_o}{2}$$

$$f_{-1}(x) = \frac{1}{\sqrt[2]{2\pi\frac{N_o}{2}}} e^{-\frac{1}{2}\left(\frac{x+1}{\sqrt[2]{\frac{N_o}{2}}}\right)^2}$$

Similarly the density function for the symbol 1, bit 1 is associated with:

$$f_1(x) = \frac{1}{\sqrt[2]{2\pi\frac{N_o}{2}}} e^{-\frac{1}{2}\left(\frac{x-1}{\sqrt[2]{\frac{N_o}{2}}}\right)^2}$$

So, the accumulated function is as follows:

$$P_{s1}(X \leq \gamma) = \Phi\left(\frac{x-1}{\sqrt[2]{\frac{N_o}{2}}}\right) = \frac{1}{2}\left(1 + erf\left(\frac{x-1}{\sqrt[2]{\frac{N_o}{2}}}\right)\right)$$

$$P_{s2}(X \leq \gamma) = \Phi\left(\frac{x+1}{\sqrt[2]{\frac{N_o}{2}}}\right) = \frac{1}{2}\left(1 + erf\left(\frac{x+1}{\sqrt[2]{\frac{N_o}{2}}}\right)\right)$$

This expression get solved when y=0.

3.5.10. Decoder

3.5.10.1. Introduction

A block decoder is similar to the encoder. This block processes a sequence of bits encoded and decoded depending on the selected algorithm.

3.5.10.2. Motivación elección : BCH code

BCH codes have been used in this project. They are used in communications systems since they need FEC error detection and correction of errors when the received signal can be susceptible to error or uncertainty.

The choice of such codes has been done for their properties of correctness of data, as well as easy implementation and low use of system resources.

3.5.10.3. Mathematical formulation

BCH decoders are implemented using a linear feedback shift register so as to retrieve the original values.

4. Simulation methodology

This project could be implemented using various tools. Two tools have been considered: a hardware level implementation (DSP) at a lower level and a more theoretical implementation using Matlab.

Each one have its pros and cons. Programming in a DSP board allows for a reliable prototype, in return requires a high programming time and material to perform the simulations [DAHNO 2000]. The Matlab programming is a very flexible tool with lots of features that are very useful for validation of theoretical concepts, their weakness is that it is a programming language inefficient for some purposes, but efficient if programmed using matrices structures.

The memory constraints on the computer where the simulations have been carried out are:

Description	Capacity
Maximum possible array	127 MB (1.333e+008 bytes)
Memory available for all arrays	582 MB (6.098e+008 bytes)
Memory used by MATLAB	1211 MB (1.270e+009 bytes)
Physical Memory (RAM)	3581 MB (3.755e+009 bytes)

Table2: "Restrictions on deployment platform"

It has chosen to use Matlab with memory constraints inherent in a DSP for implementing the entire system with DSP boards which requires a large infrastructure and programming time [AMTR 2010].

Under this concept, the project has been developed in three phases:

1. **Research on the field at different theoretical models**.
   I have studied and consulted several books and technical papers in several journals, some of them appear on reference section.
2. **Implementation of a first version to validate the concepts**.
   After selecting what type of system will be implemented and what kind of algorithms were going to be studied. It has proceeded to implement them without any memory restrictions or any kind of optimization.
3. **Implementation of a stable release with memory constraints inherent DSP.**
   The last step in this work was to study the restrictions of an implementation in hardware and simulate in Matlab, allowing a code that is easily translatable to DSP in

the future. In order to adapt the code to memory constraints to invariant individual modules using methods such as using overlap-add when convolving a sequence with an FIR filter is needed. The overlapped-add method is based on keeping the results of the convolution at the end of each cap and add them to the results of convolution in the beginning of the buffer is as shown in Figure 32: "Overlapp-add method".

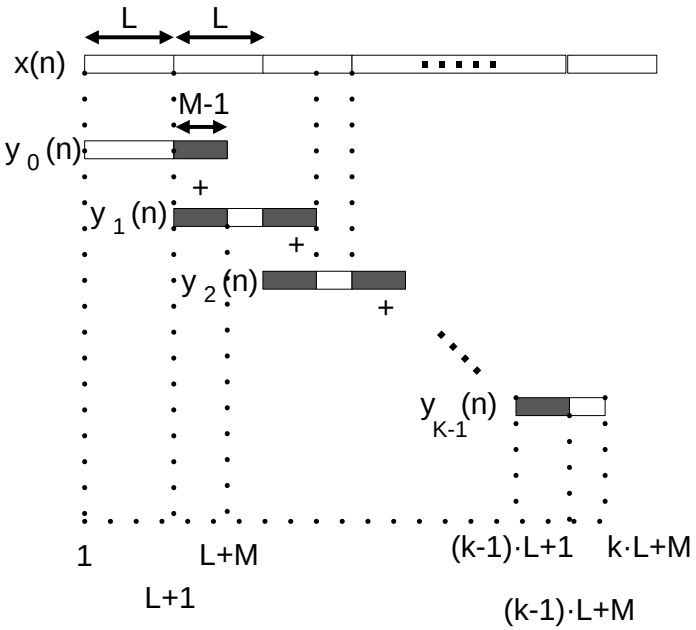


Figure 32: "Overlapp-add method"

4.1 Simulation programming

This simulation was done using a modular programming with infinite link allows simulations without memory constraints.

At this point we will explain how it has proceeded and the type of methodology followed.

4.1.1. Simulation structure

The simulation consists of a structure defined by levels.

- File "CallMain" that releases the memory used in previous simulations, the system includes all files in subdirectories and then calls the main function of the "Main"
- File "Main." Central to the simulation that is responsible for:
  - Assignment of the parameters.
  - Load parameters common to the different scenarios.
  - Scenario specific.
  - Calculate and display the results of the simulation

- Load File parameter, initialization of variables, system components and system performance.
    - Load parameters of the stage.
    - Call system components: transmitter, receiver and channel.

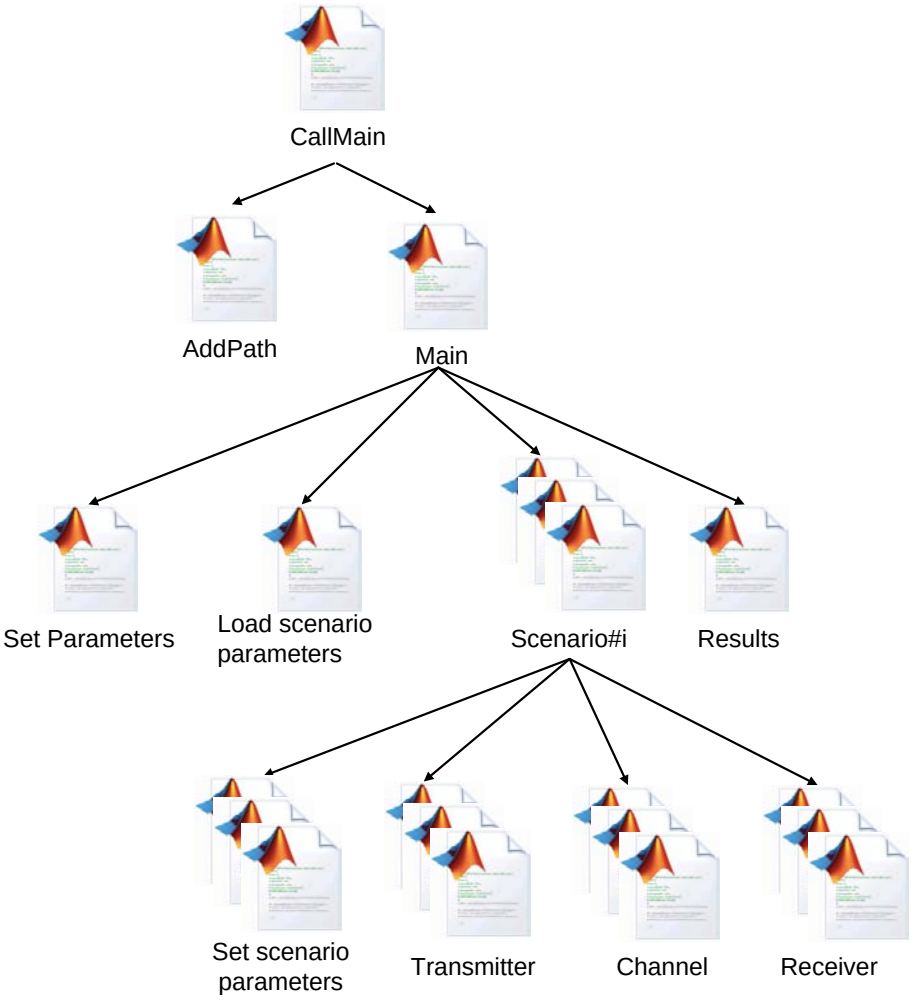
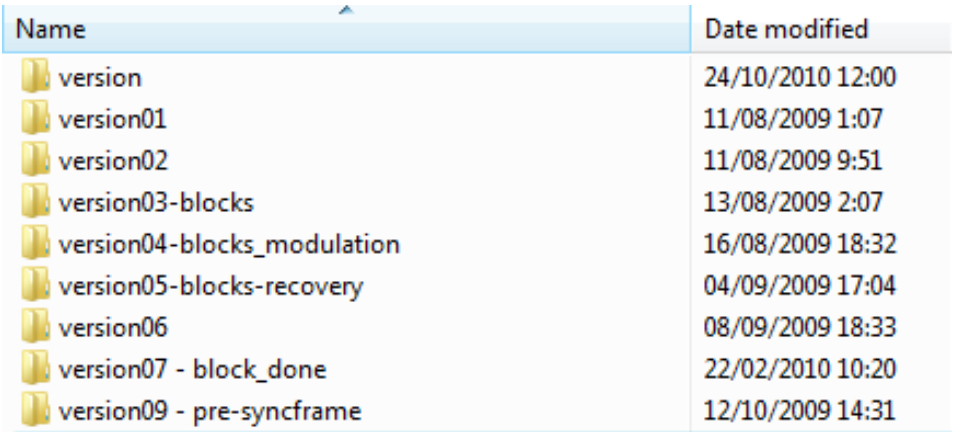


Figure 33: "Group structure functions and related functions"

4.1.2. Directory structure

The first level in the folder for the versioning system that follows the nomenclature "version" + "version number version "+"-"+" explanation if it were necessary." For example if version 12 and the main change is that the DLL has added his name will be "version12-DLL."

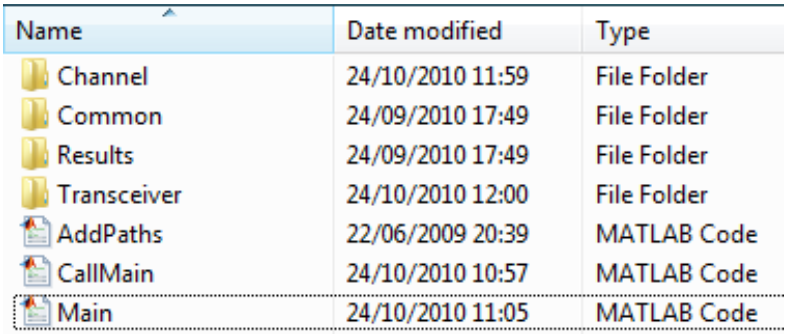


Name	Date modified
version	24/10/2010 12:00
version01	11/08/2009 1:07
version02	11/08/2009 9:51
version03-blocks	13/08/2009 2:07
version04-blocks_modulation	16/08/2009 18:32
version05-blocks-recovery	04/09/2009 17:04
version06	08/09/2009 18:33
version07 - block_done	22/02/2010 10:20
version09 - pre-syncframe	12/10/2009 14:31

Figure 34: "Versioning using multiple directories"

The next level is to separate each function according t o their purpose using folders:

- Functions in the directory are for general functions as the "Main" or "addpath."
- In each of the directories are the specific functions of each of them:
  - Channel functions are associated with the channel.
  - Common functions.
  - Results from the simulations.
  - Transceiver functions are receiver and transmitter's functions.



Name	Date modified	Type
Channel	24/10/2010 11:59	File Folder
Common	24/09/2010 17:49	File Folder
Results	24/09/2010 17:49	File Folder
Transceiver	24/10/2010 12:00	File Folder
AddPaths	22/06/2009 20:39	MATLAB Code
CallMain	24/10/2010 10:57	MATLAB Code
Main	24/10/2010 11:05	MATLAB Code

Figure 35: "General structure of the directory"

4.1.3. Functions' structure

The methodology followed consists of a generic explanation of the function implemented which can be accessed seeing the code or call using the help function in Matlab.

```
% Call:          [ParamOutput1,ParamOutput2]=NameFunction(ParamInput1, ParamInput2);
% Function:      Description about this function
% Input:         ParamOutput1 -> Description of "ParamOutput1"
%               ParamOutput2  -> Description of "ParamOutput2"
% Output:        ParamInput1    -> Description of "ParamInput1"
```

```
%                 ParamInput2     -> Description of “ParamInput2”
% Author:          Name of the author
% Created:         Date of creation
% Last modified: Date of last modification

function [ParamOutput1,ParamOutput2]=NameFunction(ParamInput1, ParamInput2);

<body>
```

Table3: “Explanation of the structure of a function "

Note that the input parameters and output are used objects where each object has multiple values. Below an example of the implementation of a user transmitter.

```
% Call:             [ParamOut,InformationUser]=Transmitter(ParamGlobal,
InformationUser,user);
% Function:         Skeleton of transmitter
% Input:            ParamGlobal     -> Scenario's Specifications
%                   InformationUser -> Information about previous loop.
%                   user            -> User that is being proceced
% Output:           ParamOut        -> Transmitted Processed and pre-processed data
%                   InformationUser -> Information about this loop. This will be used
in the next loop
% Author:           Albert Armisen
% Created:          31-07-09
% Last modified:    04-09-10

function [ParamOut,InformationUser]=Transmitter(ParamGlobal,InformationUser,user)

% Initialize variables
ParamOut.DataTxUserInPhase=[];
ParamOut.DataTxUserQuadPhase=[];
InformationUser.PositionBits=1;

%% Running the transmitter
    % Init position var
    InformationUser.EncodedNumBits=1;
    InformationUser.MessageSymbolsPosition=1;
    % Calculate current oversampling position
    InformationUser.IndexFir =InformationUser.IndexFir-ParamGlobal.
PacketSizeUpsampled;
    % Calculate symbols to encode in this iteration
    InformationUser.FreeSymbols=floor((ParamGlobal.PacketSizeUpsampled-
InformationUser.IndexFir)/(ParamGlobal.OverSamplingFactor*ParamGlobal.SF)); %
Calculating how many symbols can be tx at the current buffer
    % Initialize symbol space
    Symbols=zeros(1,InformationUser.FreeSymbols);


    % Do until all symbols have been proceed.
    while(InformationUser.FreeSymbols>0)
        while (InformationUser.IndexModulatorSymbol== 0) %Check if some parameter
value is zero
            switch(InformationUser.ModeModulation) %Variable that controls the kind
of package (Sync or Data)as long as kind of information (Guard, Training or Data)
                case 0 % Sync symbols
                    InformationUser.IndexModulatorSymbol=ParamGlobal.
FrameDataGuardLength;
                    InformationUser.ModeModulation=1; % Change state to Guard Data
Symbols
                    InformationUser.ModulationType=ParamGlobal.BitsPerSymbolGuard;
                case 1 % Guard Data Symbols
                    InformationUser.IndexModulatorSymbol=ParamGlobal.
FrameDataInfoLength;
                    InformationUser.ModeModulation=3; % Change state to Info Data
Symbols
                    InformationUser.ModulationType=ParamGlobal.BitsPerSymbol;
                case 3 % Info Data Symbols
                    InformationUser.IndexModulatorSymbol=ParamGlobal.
```

```
FrameDataGuardLength;
                    InformationUser.ModeModulation=1; % Change state to Guard Data
Symbols
                    InformationUser.ModulationType=ParamGlobal.BitsPerSymbolGuard;
                end
            end

            switch(InformationUser.ModeModulation)%Variable that controls the kind of
package (Sync or Data)as long as kind of information (Guard, Training or Data)
                case 0, % Modulating sync bits
                    if (InformationUser.FreeSymbols*ParamGlobal.SF>=InformationUser.
IndexModulatorSymbol) % If syncronization bits can be modulate in this buffer
                    InformationUser.SymbolsLenOut=InformationUser.
IndexModulatorSymbol; % Update number to to modulate
                    InformationUser.BitsLenOut=InformationUser.
SymbolsLenOut*InformationUser.ModulationType;

                    % Modulate sync bits
                    Bits = ParamGlobal.SyncSeq(user,InformationUser.IndexModulator:
InformationUser.IndexModulator+InformationUser.BitsLenOut-1);
                    SymbolsAux=Modulator(InformationUser.ModulationType,Bits); %
Modulate bits
                    Symbols(InformationUser.EncodedNumBits:InformationUser.
EncodedNumBits+length(SymbolsAux)-1) = ParamGlobal.SyncAmplification * SymbolsAux;

                    % Change state to guard info data bits
                    InformationUser.IndexModulator=1; % Restart the inside position
                    InformationUser.NumPackets=0; %  Set to zero the packet counter.
A syncronization has been done.
                    InformationUser.IndexModulatorSymbol=ParamGlobal.
FrameDataGuardLength;
                    InformationUser.ModeModulation=1; % Change state to Guard Data
Symbols

                else
                    InformationUser.IndexModulatorSymbol=InformationUser.
IndexModulatorSymbol/ParamGlobal.SF-InformationUser.FreeSymbols;% Update remaining
information to be modulate in the current packet
                    InformationUser.SymbolsLenOut=InformationUser.FreeSymbols; %
Update number to to modulate
                    InformationUser.BitsLenOut=InformationUser.
SymbolsLenOut*InformationUser.ModulationType;

                    % Modulate info data bits
                    Bits = ParamGlobal.SyncSeq(user,InformationUser.IndexModulator:
InformationUser.IndexModulator+InformationUser.BitsLenOut-1);
                    SymbolsAux=Modulator(InformationUser.ModulationType,Bits); %
Modulate bits
                    Symbols(InformationUser.EncodedNumBits:InformationUser.
EncodedNumBits+length(SymbolsAux)-1) = ParamGlobal.SyncAmplification * SymbolsAux;

                    InformationUser.IndexModulator=InformationUser.
IndexModulator+InformationUser.FreeSymbols; % Update position inside the data packet
to modulate
                end
```

```
                %% Bypassing spreading
                MessageSymbols = SymbolsAux;
                ParamOut.MessageSymbols(InformationUser.MessageSymbolsPosition:
InformationUser.MessageSymbolsPosition+length(MessageSymbols )-1) = MessageSymbols;
                InformationUser.MessageSymbolsPosition = InformationUser.
MessageSymbolsPosition + length(MessageSymbols);
            case 1, % Modulating Guard Data Bits
                if (InformationUser.FreeSymbols>=InformationUser.
IndexModulatorSymbol) % If syncronization bits can be modulate in this buffer
                    InformationUser.SymbolsLenOut=InformationUser.
IndexModulatorSymbol; % Update number to to modulate
                    InformationUser.BitsLenOut=InformationUser.
SymbolsLenOut*InformationUser.ModulationType;

                    % Modulate Info Data Bits
                    SymbolsAux=zeros(1,InformationUser.BitsLenOut); % Guard symbols
                    Symbols(InformationUser.EncodedNumBits:InformationUser.
EncodedNumBits+length(SymbolsAux)-1) = SymbolsAux;

                    % Change state to info data bits
                    InformationUser.IndexModulator=1; % Restart the inside position
                    InformationUser.IndexModulatorSymbol=ParamGlobal.
FrameDataInfoLength;
                    InformationUser.ModeModulation=3; % Change state to Info Data
Symbols
                    InformationUser.ModulationType=ParamGlobal.BitsPerSymbol;
                else
                    InformationUser.IndexModulatorSymbol=InformationUser.
IndexModulatorSymbol-InformationUser.FreeSymbols;% Update remaining information to be
modulate in the current packet
                    InformationUser.SymbolsLenOut=InformationUser.FreeSymbols; %
Update number to to modulate
                    InformationUser.BitsLenOut=InformationUser.
SymbolsLenOut*InformationUser.ModulationType;

                    % Modulate Info Data Bits
                    SymbolsAux=zeros(1,InformationUser.BitsLenOut); % Guard symbols
                    Symbols(InformationUser.EncodedNumBits:InformationUser.
EncodedNumBits+length(SymbolsAux)-1) = SymbolsAux;

                    InformationUser.IndexModulator=InformationUser.
IndexModulator+InformationUser.FreeSymbols; % Update position inside the data packet
to modulate
                end
                %% Spread signal
                MessageSymbols = Spreader(InformationUser.Code_Data,SymbolsAux);
                ParamOut.MessageSymbols(InformationUser.MessageSymbolsPosition:
InformationUser.MessageSymbolsPosition+length(MessageSymbols )-1) = MessageSymbols;
                InformationUser.MessageSymbolsPosition = InformationUser.
MessageSymbolsPosition + length(MessageSymbols);
            case 3, % Modulating Info Data Bits
                if (InformationUser.FreeSymbols>=InformationUser.
IndexModulatorSymbol) % If syncronization bits can be modulate in this buffer
                    InformationUser.SymbolsLenOut=InformationUser.
IndexModulatorSymbol; % Update number to to modulate
```

```
                    SymbolsAux=zeros(1,InformationUser.SymbolsLenOut);
                    posInt=0;
                    % Processing pilot information
                    while(posInt<InformationUser.SymbolsLenOut)
                        if ((length(ParamGlobal.PilotDataSeqSymb)-InformationUser.
PilotSymbolsPos+1)<=(InformationUser.SymbolsLenOut-posInt))
                            aux=ParamGlobal.PilotDataSeqSymb(InformationUser.
PilotSymbolsPos:end);
                            InformationUser.PilotSymbolsPos=1;
                            SymbolsAux(posInt+1:posInt+length(aux))=aux;
                            posInt=posInt+length(aux);
                        else
                            aux=ParamGlobal.PilotDataSeqSymb(InformationUser.
PilotSymbolsPos:InformationUser.PilotSymbolsPos+InformationUser.SymbolsLenOut-posInt-
1);
                            InformationUser.PilotSymbolsPos=InformationUser.
PilotSymbolsPos+length(aux);
                            SymbolsAux(posInt+1:posInt+length(aux))=aux;
                            posInt=posInt+length(aux);
                        end
                    end
                    %% Spread signal
                    MessageSymbols = Spreader(InformationUser.Code_Pilot,SymbolsAux);
                    InformationUser.BitsLenOut=InformationUser.
SymbolsLenOut*InformationUser.ModulationType;

                    % Information data
                    Bits=[];
                    BitsCodedNum=min(length(InformationUser.Coder.Bits),
InformationUser.BitsLenOut);
                    BitsProtected=InformationUser.Coder.Bits(1:BitsCodedNum);
                    InformationUser.Coder.Bits=InformationUser.Coder.Bits
(BitsCodedNum+1:end);
                    while (InformationUser.BitsLenOut > length(BitsProtected))
                        BitsAux= randint(1,InformationUser.Coder.k); % Generation of
random data bits
                        Bits=[Bits,BitsAux]; % Use previous bits
                        BitsUnprotected = gf(logical(Bits)); % Set in the proper
format the bits to be encoded
                        BitsCoded = bchenc(BitsUnprotected, InformationUser.Coder.N,
InformationUser.Coder.k); % Encoding
                        InformationUser.Coder.Bits = double( BitsCoded.x );
                        BitsCodedNum=min(length(InformationUser.Coder.Bits),
InformationUser.BitsLenOut-length(BitsProtected));
                        BitsProtected=[BitsProtected,InformationUser.Coder.Bits(1:
BitsCodedNum)]; % Save encoded bits
                        InformationUser.Coder.Bits=InformationUser.Coder.Bits
(BitsCodedNum+1:end);
                    end

                    SymbolsAux=Modulator(InformationUser.ModulationType,
BitsProtected); % Modulate bits
                    Symbols(InformationUser.EncodedNumBits:InformationUser.
EncodedNumBits+length(SymbolsAux)-1) = ParamGlobal.InfoDataAmplification *
SymbolsAux;
```

```
                    % Change state to guard info data bits
                    InformationUser.IndexModulator=1; % Restart the inside position
                    InformationUser.IndexModulatorSymbol=ParamGlobal.
FrameDataGuardLength;
                    InformationUser.ModeModulation=1; % Change state to Guard Data
Symbols
                    InformationUser.ModulationType=ParamGlobal.BitsPerSymbolGuard;
                    %% Spread signal
                    MessageSymbols = Spreader(InformationUser.Code_Data,SymbolsAux)
+MessageSymbols;
                else % If syncronization bits cannot be modulate in this buffer
                    InformationUser.IndexModulatorSymbol=InformationUser.
IndexModulatorSymbol-InformationUser.FreeSymbols;% Update remaining information to be
modulate in the current packet
                    InformationUser.SymbolsLenOut=InformationUser.FreeSymbols; %
Update number to to modulate

                    SymbolsAux=zeros(1,InformationUser.SymbolsLenOut);
                    posInt=0;
                    while(posInt<InformationUser.SymbolsLenOut)
                        if ((length(ParamGlobal.PilotDataSeqSymb)-InformationUser.
PilotSymbolsPos+1)<=(InformationUser.SymbolsLenOut-posInt))
                            aux=ParamGlobal.PilotDataSeqSymb(InformationUser.
PilotSymbolsPos:end);
                            InformationUser.PilotSymbolsPos=1;
                            SymbolsAux(posInt+1:posInt+length(aux))=aux;
                            posInt=posInt+length(aux);
                        else
                            aux=ParamGlobal.PilotDataSeqSymb(InformationUser.
PilotSymbolsPos:InformationUser.PilotSymbolsPos+InformationUser.SymbolsLenOut-posInt-
1);
                            InformationUser.PilotSymbolsPos=InformationUser.
PilotSymbolsPos+length(aux);
                            SymbolsAux(posInt+1:posInt+length(aux))=aux;
                            posInt=posInt+length(aux);
                        end
                    end
                    %% Spread signal
                    MessageSymbols = Spreader(InformationUser.Code_Pilot,SymbolsAux);
                    InformationUser.BitsLenOut=InformationUser.
SymbolsLenOut*InformationUser.ModulationType;
                    % Modulate Info Data Bits
                    Bits=[];
                    BitsCodedNum=min(length(InformationUser.Coder.Bits),
InformationUser.BitsLenOut);
                    BitsProtected=InformationUser.Coder.Bits(1:BitsCodedNum);
                    InformationUser.Coder.Bits=InformationUser.Coder.Bits
(BitsCodedNum+1:end);
                    while (InformationUser.BitsLenOut > length(BitsProtected))
                        BitsAux= randint(1,InformationUser.Coder.k); % Generation of
random data bits
                        Bits=[Bits,BitsAux];% Use previous bits
                        BitsUnprotected = gf(logical(Bits));% Set in the proper
format the bits to be encoded
```

```
                        BitsCoded = bchenc(BitsUnprotected, InformationUser.Coder.N,
InformationUser.Coder.k);% Encoding
                        InformationUser.Coder.Bits = double( BitsCoded.x );
                        BitsCodedNum=min(length(InformationUser.Coder.Bits),
InformationUser.BitsLenOut-length(BitsProtected));
                        BitsProtected=[BitsProtected,InformationUser.Coder.Bits(1:
BitsCodedNum)];% Save encoded bits
                        InformationUser.Coder.Bits=InformationUser.Coder.Bits
(BitsCodedNum+1:end);
                    end

                    SymbolsAux=Modulator(InformationUser.ModulationType,
BitsProtected); % Modulate bits
                    Symbols(InformationUser.EncodedNumBits:InformationUser.
EncodedNumBits+length(SymbolsAux)-1) = ParamGlobal.InfoDataAmplification *
SymbolsAux;

                    InformationUser.IndexModulator=InformationUser.
IndexModulator+InformationUser.FreeSymbols; % Update position inside the data packet
to modulate
                    %% Spread signal
                    MessageSymbols = Spreader(InformationUser.Code_Data,SymbolsAux)
*MessageSymbols;
                end

                ParamOut.TxBits(InformationUser.PositionBits:InformationUser.
PositionBits+length(Bits)-1)=Bits;
                InformationUser.PositionBits=InformationUser.PositionBits+length
(Bits);
                ParamOut.MessageSymbols(InformationUser.MessageSymbolsPosition:
InformationUser.MessageSymbolsPosition+length(MessageSymbols )-1) = MessageSymbols;
                InformationUser.MessageSymbolsPosition = InformationUser.
MessageSymbolsPosition + length(MessageSymbols);

        end
        InformationUser.FreeSymbols=InformationUser.FreeSymbols-InformationUser.
SymbolsLenOut; % Update current free symbols for this buffer
        InformationUser.EncodedNumBits=InformationUser.
EncodedNumBits+InformationUser.SymbolsLenOut; % Update position in the buffer for
this buffer
    end
%% Upsample signal
ParamOut.UpSampledI=upsample(ParamOut.MessageSymbols,ParamGlobal.OverSamplingFactor);
InformationUser.IndexFir=InformationUser.IndexFir+ParamGlobal.PacketSizeUpsampled;

%% Pulse shape it
[ParamOut.UpSampledDataInPhase,InformationUser.ConvolutedSavedI]=convBlock
(ParamGlobal.PulseGen,real(ParamOut.UpSampledI),InformationUser.ConvolutedSavedI);
[ParamOut.UpSampledDataQuadPhase,InformationUser.ConvolutedSavedQ]=convBlock
(ParamGlobal.PulseGen,imag(ParamOut.UpSampledI),InformationUser.ConvolutedSavedQ); %
Upsample QuadraturePhase part

%% Upconvert the signal
[ParamOut.DataTxUserInPhase,ParamOut.DataTxUserQuadPhase,InformationUser.Phase]
=UpConversion(ParamOut.UpSampledDataInPhase,ParamOut.UpSampledDataQuadPhase,

  ParamGlobal.Fs,ParamGlobal.Fc,InformationUser.Phase);
```

5. Results

This section shows the results coming from the work done on this project. It starts by showing some SNR-BER plots, and then moving to show some outputs of each module where can be seen some partial results.

The following picture compares the different implementations of the chain of DSS-CDMA communications.

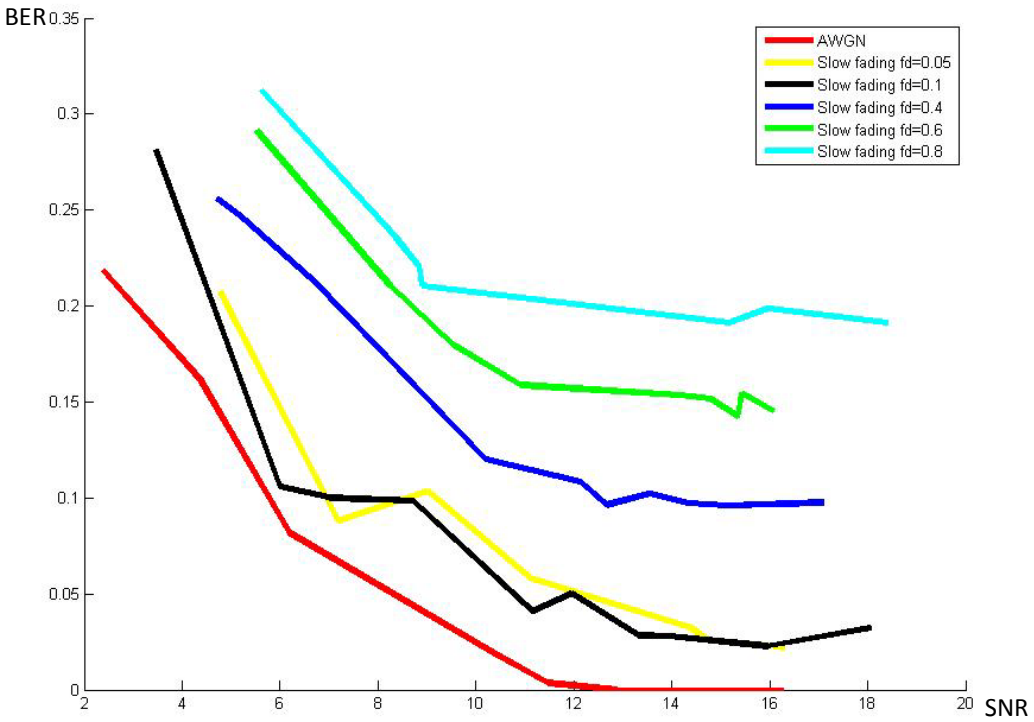


Figure 36: "SNR-BER plot of several implementation of the system in a 16-QAM constellation"

Several conclusions are drawn from this graph. On one hand it is observed that the system is robust against AWGN channel with fading and static. On the other hand it is appreciated that a slowly fading channel BER depends on how fast this channel change over time, the faster it does, the higher BER gets.

The below figure has opted for a 16-QAM (instead of QPSK) simulation in a higher SNR range, where the BER is lower for the same type of simulations in order to check some trends. This result is expected since QPSK space of each symbol is greater than 16-QAM. Indeed, the trend is the same as channel type is concerned in both cases, either in QPSK or in 16-QAM just differentiating from having a lower BER.

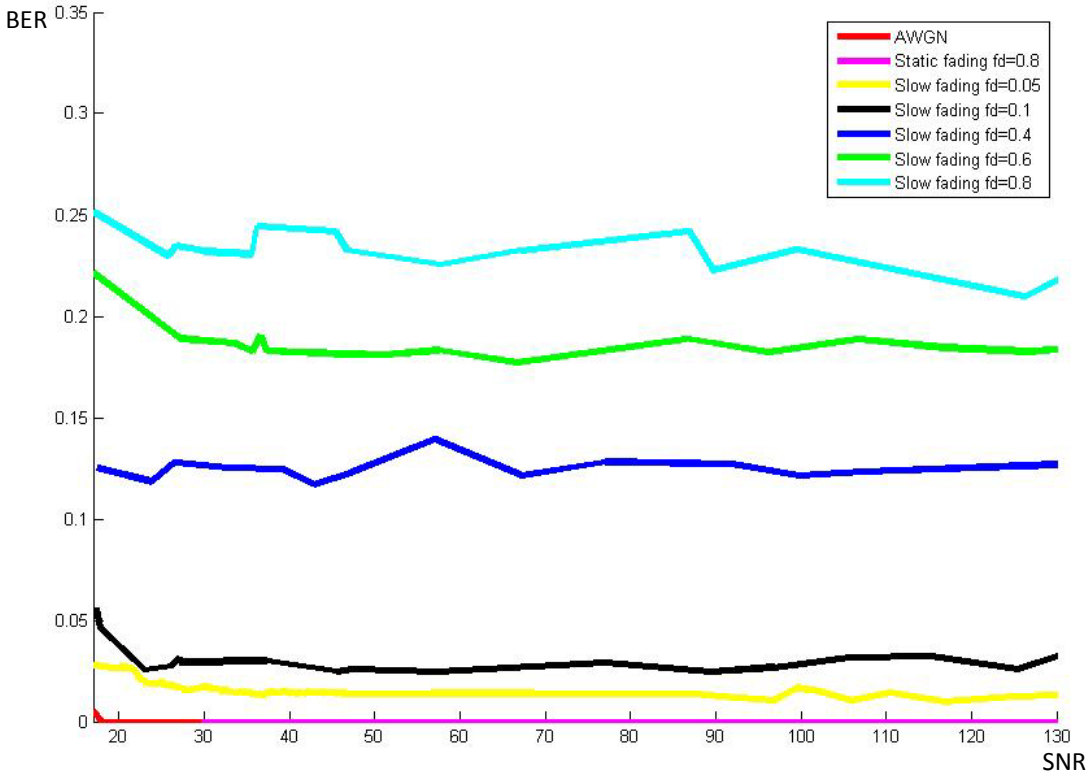


Figure 37: "SNR-BER plot of several implementation of the system in a 16-QAM constellation"

The next step is showing the results of each one of the key modules implemented in this system.

Once bits are encoded, they can be modulated in three types of modulation BPSK, QPSK and 16-QAM.

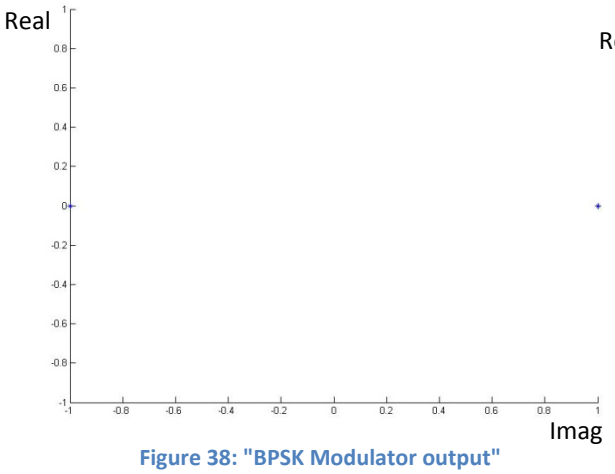


Figure 38: "BPSK Modulator output"

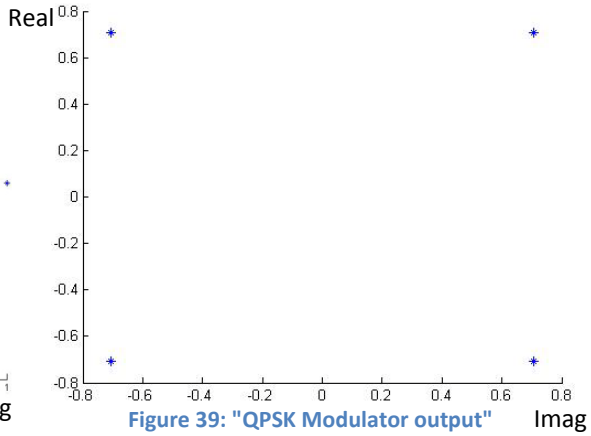


Figure 39: "QPSK Modulator output"

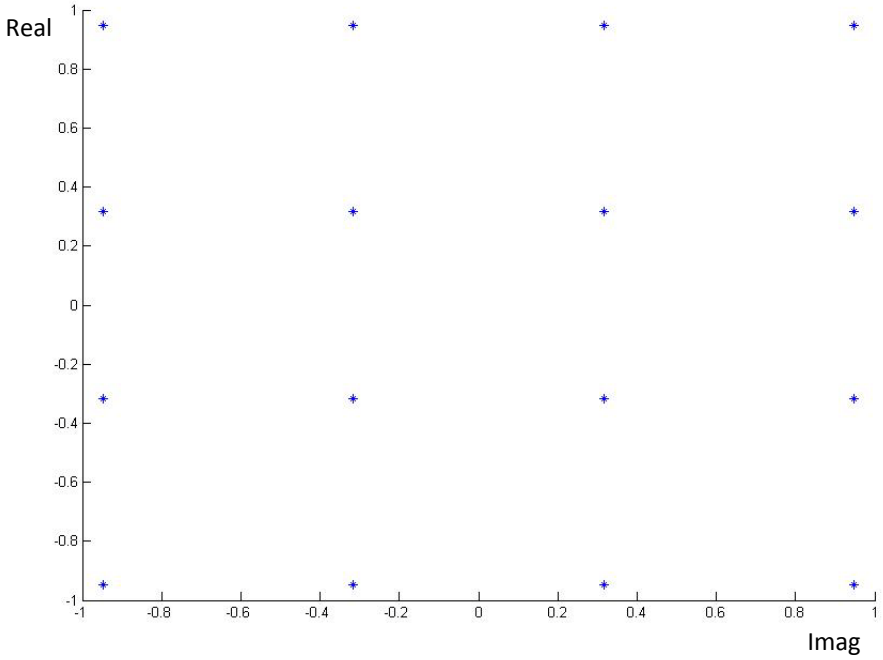


Figure 40: "16-QAM Modulator output"

These symbols are passed through the spreader in order to place them in a specific bandwidth associated to each user by a chip code. Below is the autocorrelation of Gold and Hadamard code chip. If correlation must be compared, this should be normalized by the maximum value that corresponds to the length of the sequence of chip.

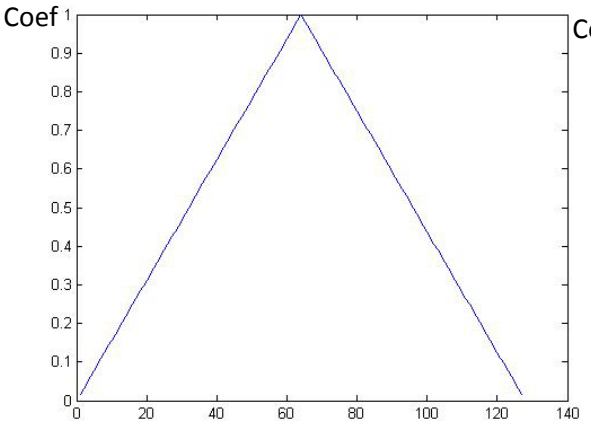


Figure 41: "First Hadamard sequence auto-correlation having a chip length sequence of 64"

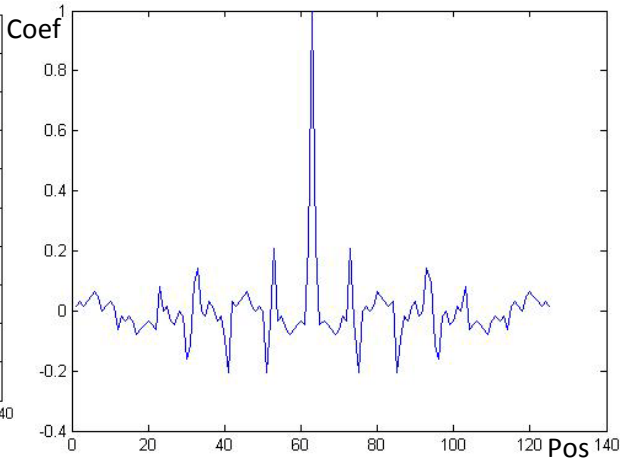


Figure 42: "First Gold sequence auto-correlation having a chip length sequence of 63"

First of all, note that the first Hadamard sequence does not have the properties of a spread sequence, while the Gold sequence is a spread sequence. The second one has a higher difference between the signal itself (central point equal to 1) and the same signal delayed; hence it has spread spectrum, which may randomize interferences

All cross-correlations have been analyzed for Hadamard and Gold chip sequences of chip length of 64 and 63 respectively. Below, two cross-correlations for each of them where it can be seen that Hadamard sequences are much more robust against MAI (with an order of magnitude of 100) than Gold sequences in an ideal case where all signals are synchronized. However, if synchronization is not perfect as in our case QS-CDMA Gold sequences are more robust against MAI interferences.

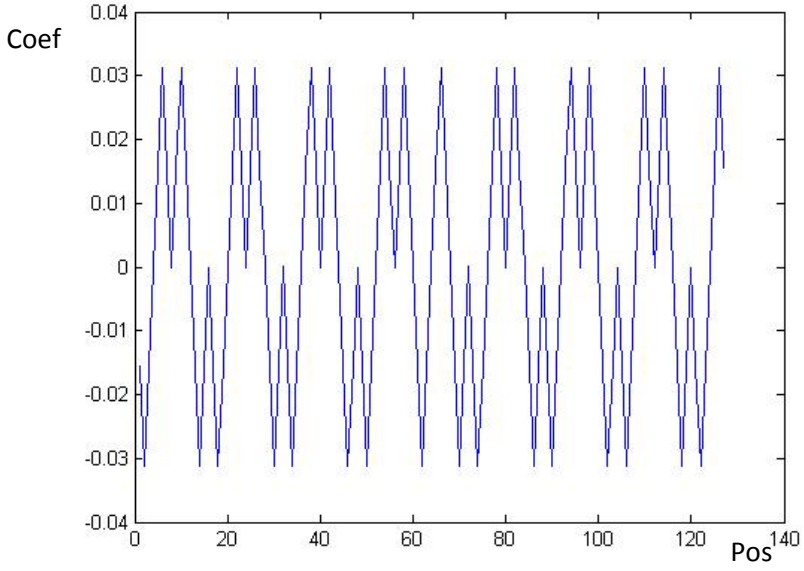


Figure 43: "Hadamard sequence cross-correlation having a chip length sequence of 64 chips from two different users, normalized by the maximum value of the auto-correlation of a user"

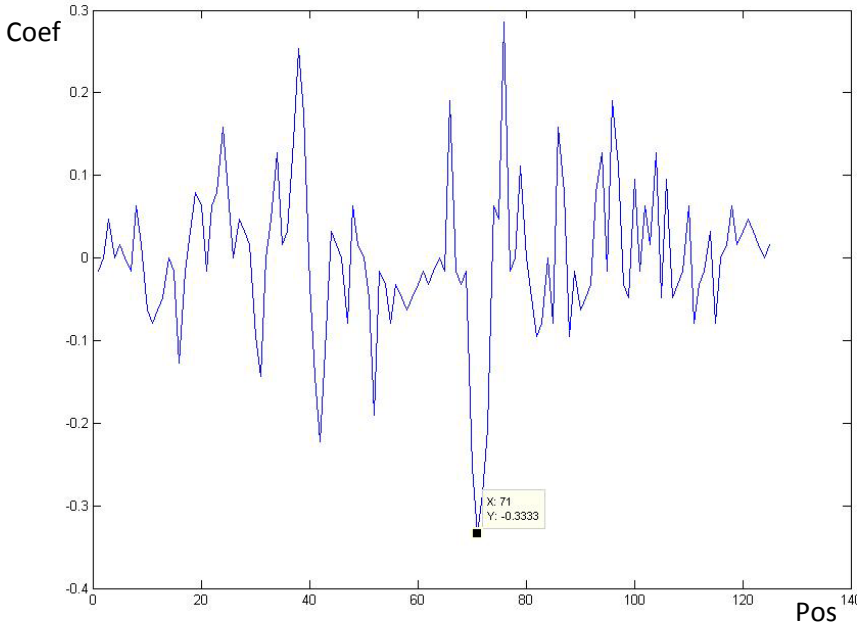


Figure 44: "Gold sequence cross-correlation having a chip length sequence of 63 chips from two different users, normalized by the maximum value of the auto-correlation of a user"

The next step is to pass through the zero padder to prepare the signal to convolve a filter. This step involves an increase on its frequency.

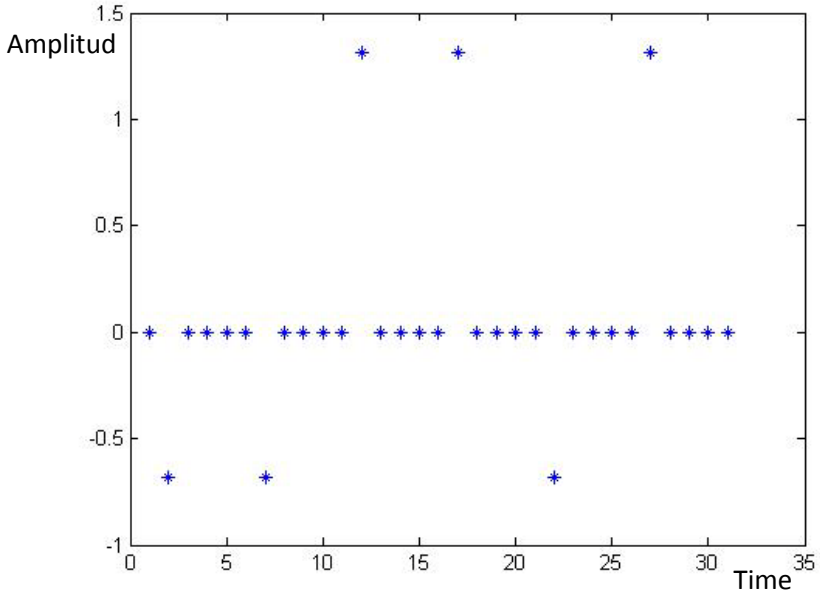


Figure 45: "Zero padder output"

The above signal is filtered by a pulse shaper filter to modify send signal's shape, its power spectrum is:

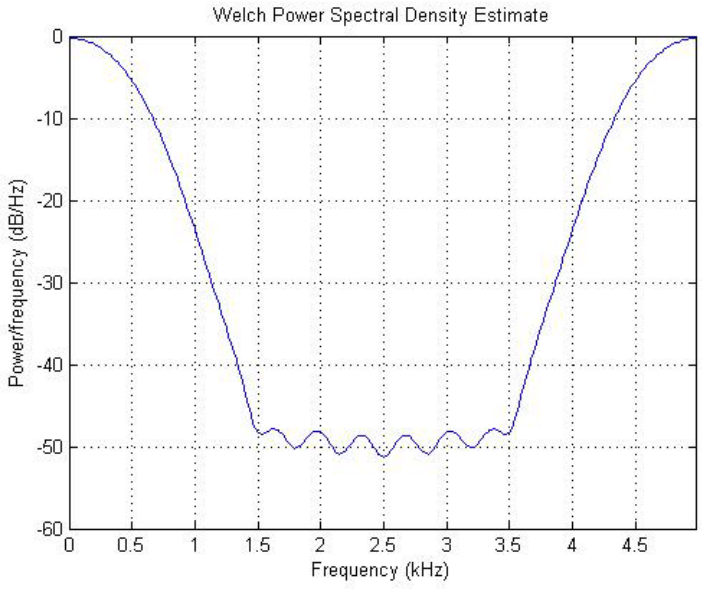


Figure 46: "Power spectral density estimation using Welch estimator of the base-band signal"

The last step is moving transmitter signal frequency to one determined by using upconverter to a carrier of 1.5kHz.

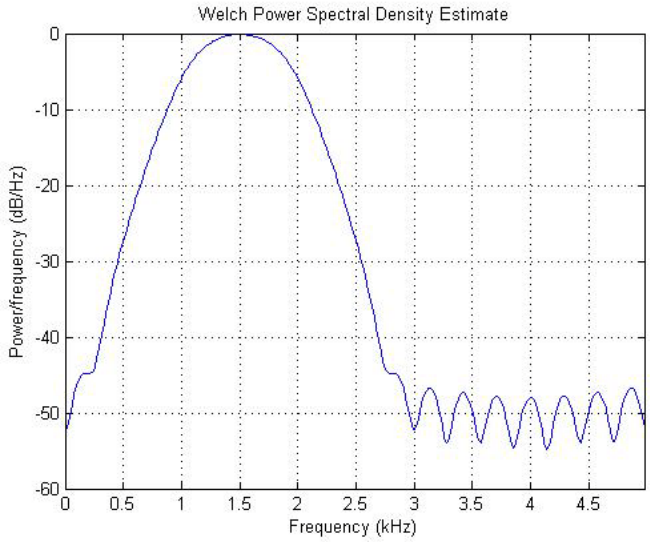


Figure 47: "Power spectral density estimation using Welch estimator of the band-pass signal"

Its in-phase and quad-phase components are illustrated in the figure below. These components tell that instant power is not constant over time because it is using 16-QAM, constellation which has not a constant envelope; while when using QPSK or BPSK envelope is constant, hence having a constant constellation symbol power.

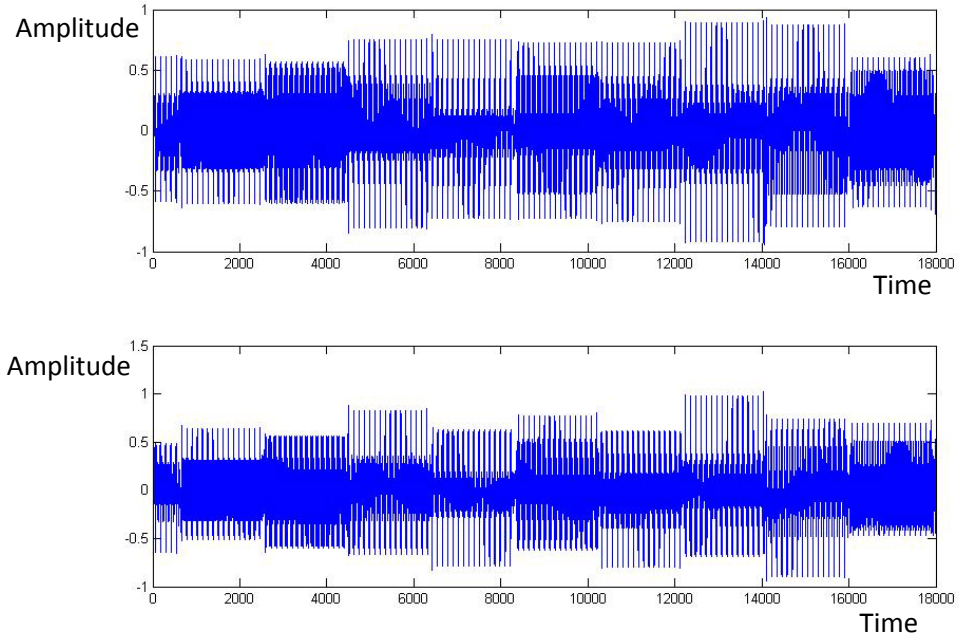


Figure 48: "Component In-phase and Quad-phase at the last point of the transmitter"

Once each signal has been generated for each transmitter, they are added in a common channel. Its spectral estimation is illustrated below. It shows how the sum of all user and channel interference produces a signal which is wider Gaussian than before.

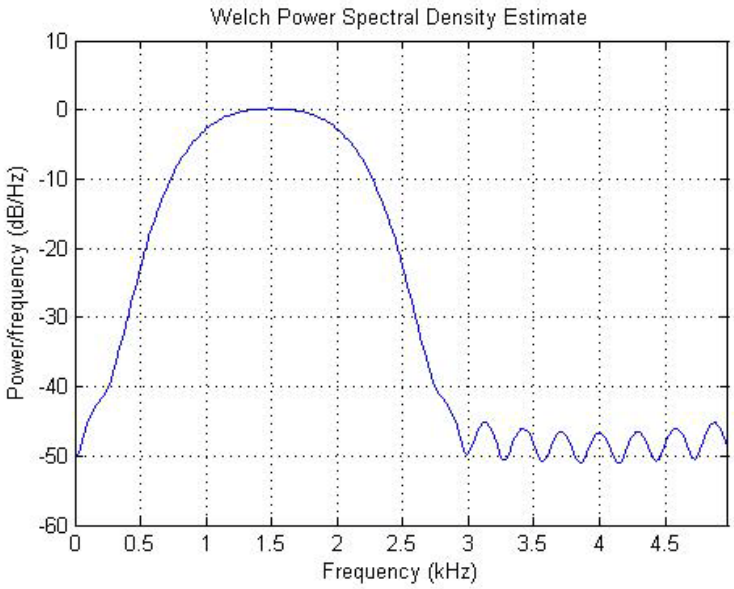


Figure 49: "Power spectral density estimation of all users summed (Fs is 5kHz)"

The next step is oversampling the signal since we want to introduce a random delay due to the channel. Note that there are several harmonics introduced to the signal for this reason. This is one of the consequences when extrapolating that some harmonics appear.

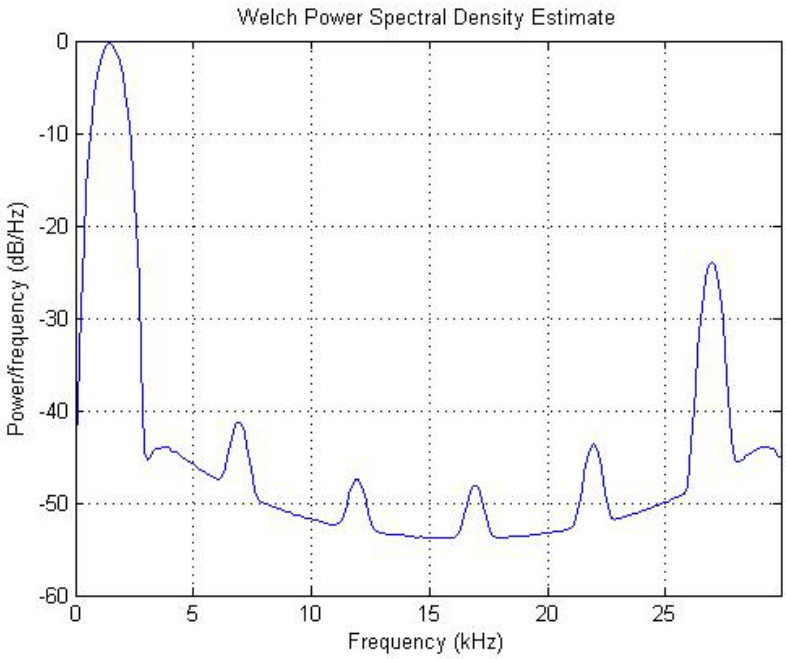


Figure 50: "Power spectral density estimation of all users summed, after oversampling"

This signal is transmitted through a channel that alters any of its properties; a channel which is shared by several users. It is seen that power difference between symbols is hidden by adding AWGN when plotting in-phase and quad-phase components.

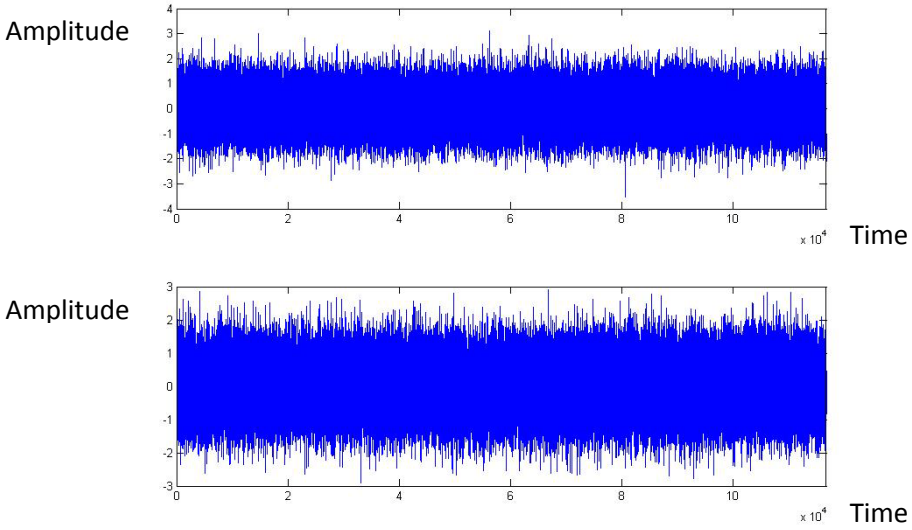


Figure 51: "In-phase and Quad-phase components after passing through channel which has been over-sampled at a ratio of 1 to 6 with an SNR of 9dB"

Below is the signal which has passed through a slow fading channel and AWGN. The effect introduced by oversampling and extrapolating the signal is hidden by the AWGN noise.

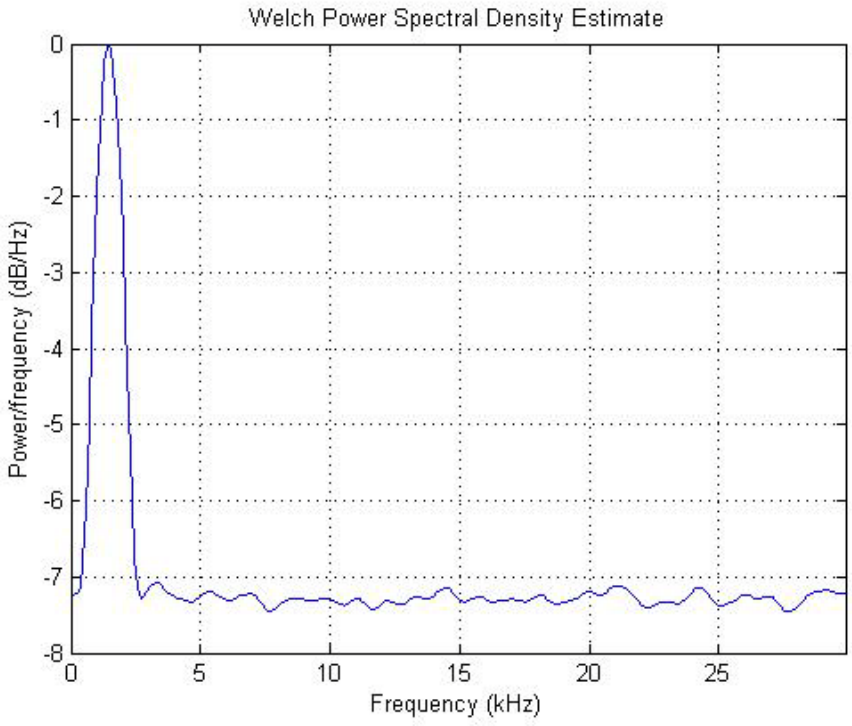


Figure 52: "Power Spectral Density estimation of a signal after passing through a slow fading channel with additive noise"

This signal is processed in a common receiver. The first step is moving the receiver band-pass signal to baseband, the output is shown on below figure.

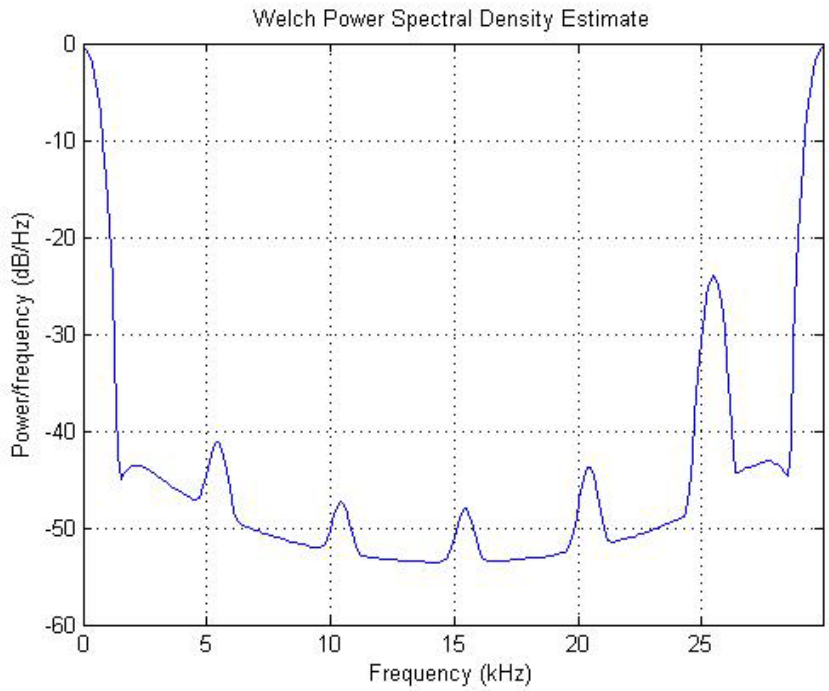


Figure 53: "Base-band signal received at downconverter output"

The signal is filtered by a low pass filter reducing interference, and hence obtaining a spectrum with a ripple at high frequencies introduced by this low pass filter. It can be seen as noise at high frequencies (frequencies above 1.5kHz) is reduced considerably.

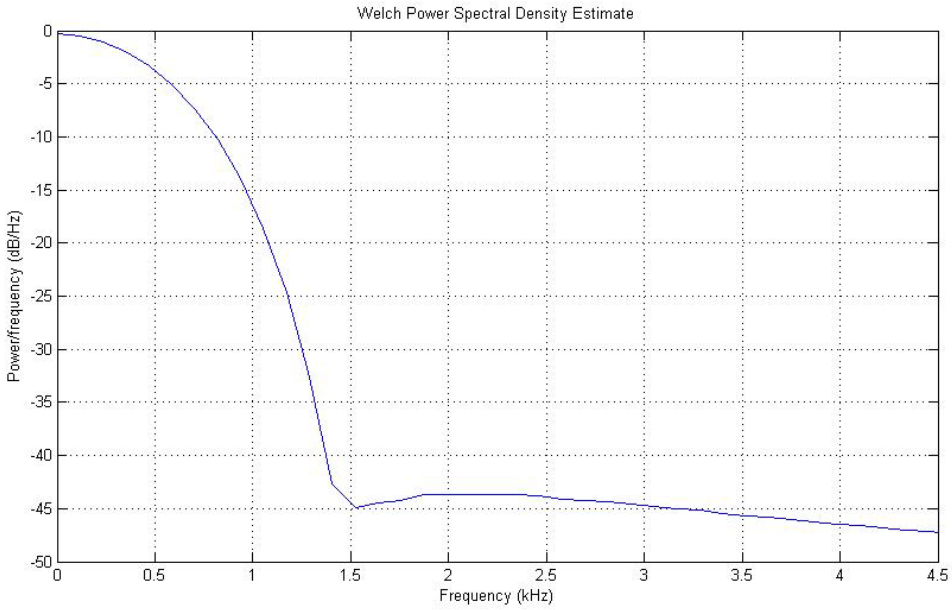


Figure 54: "Base-band signal filtered by a low pass filter"

This signal is used to calculate when transmission using a known pilot signal sent at the start of communication does begin to transmit using a known reference pilot signal sent at the beginning of communication. It consists on peak detection using cross-correlation, as explained above. Below are plotted the values which gives an intuitive idea where the signal starts by looking where there is an increase in power.

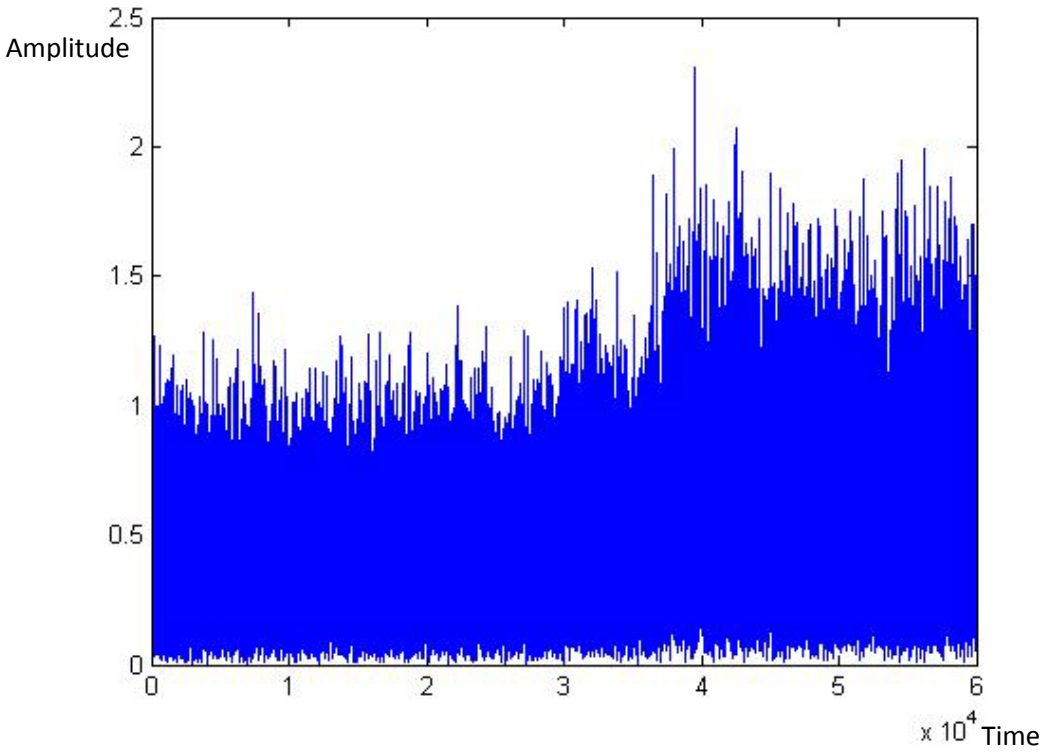


Figure 55: "Signal at receiver ready to start signal acquisition on synchronizer block"

In this particular scenario, a change of power was applied to better distinguis transmition stage from non-transmittion stage. The resulting synchronization signal is amplified thanks to a highly configurable system which allows, among other things, to configure the associated power with synchronization symbols, pilot symbols and data symbols.

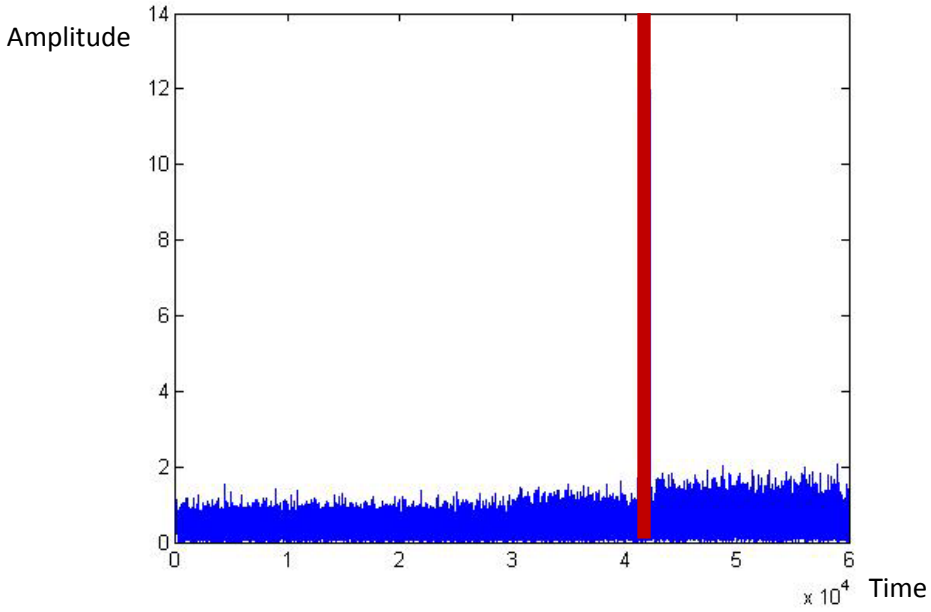


Figure 56: "Signal at receiver ready to start acquiring synchronization"

The above amplified signal is used to trigger the system; however this change in power does not affect the system since it uses a normalized correlation to trigger signal meaning that just affects a change in order between magnitudes. This mechanism does not depend on power but on the properties of the received signal. The values obtained are explained by:

1. If there is no signal and there is just white noise, no resemblance exists.
2. If a signal has been transmitted, synchronization resemblance is high except for some noise introduced by the channel.
3. The last case is when synchronization signal has been sent, and data is being transmitted. This data uses chip sequences which have some resemblances; hence, having a considerable normalized almost constant coefficient ratio.

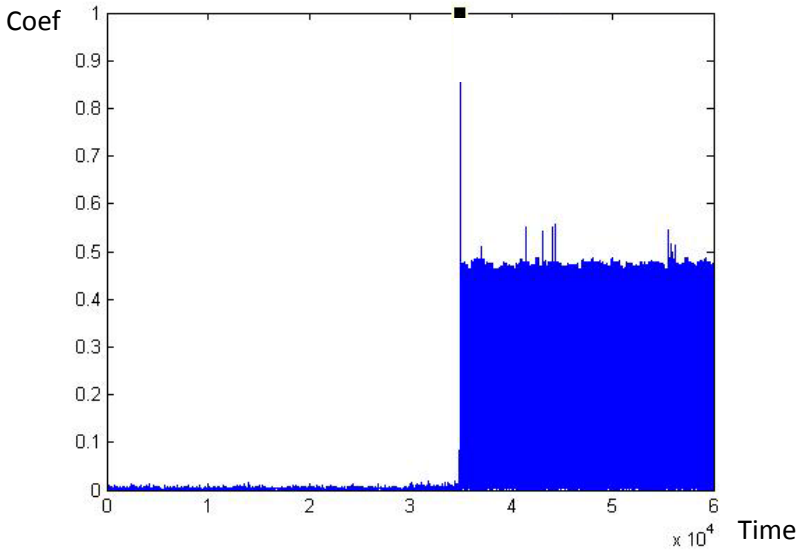


Figure 57: "Values of cross-correlation between the known pilot signal and received signal"

Zooming previous figure shows how this implementation is highly reliable, however one must note that there are several high values due to the use of a chip. These possible errors are compensated by the DLL.

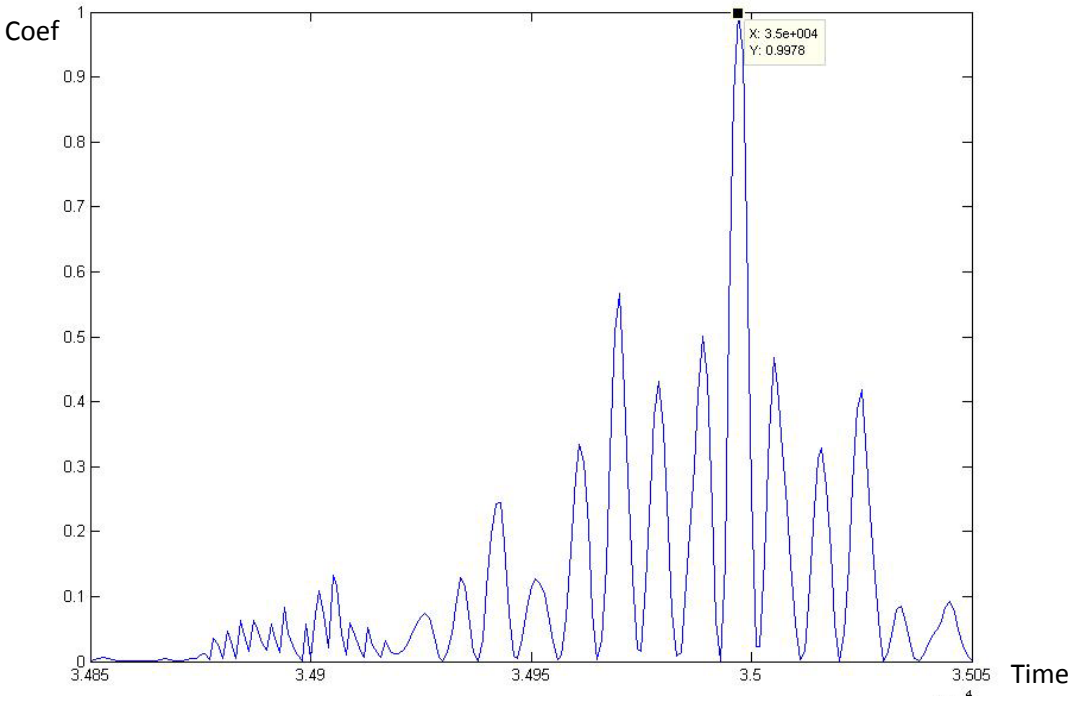


Figure 58: "Zoom to cross-correlation values between known pilot signal and received signal"

Once synchronization time is obtained, the signal is decimated giving each user a specific signal as shown in the illustrations. Note that the amplitude difference between the equalized and non-equalized symbols is introduced for system customization where data and pilot symbols have an amplification of 0.5.

To obtain an accurate SNR, the signal and noise are measured before entering the despreader module to consider only the noise affecting the symbols.

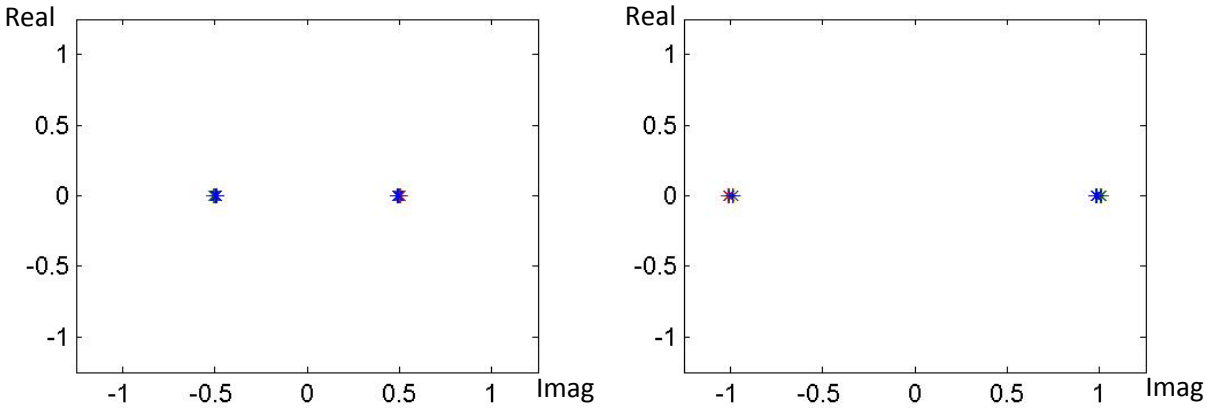


Figure 59: "Not equalized and equalized symbols in a 16-QAM modulation with 23 db SNR and 0 BER in a scenario with slow fading (fd=0.015) and additive noise"

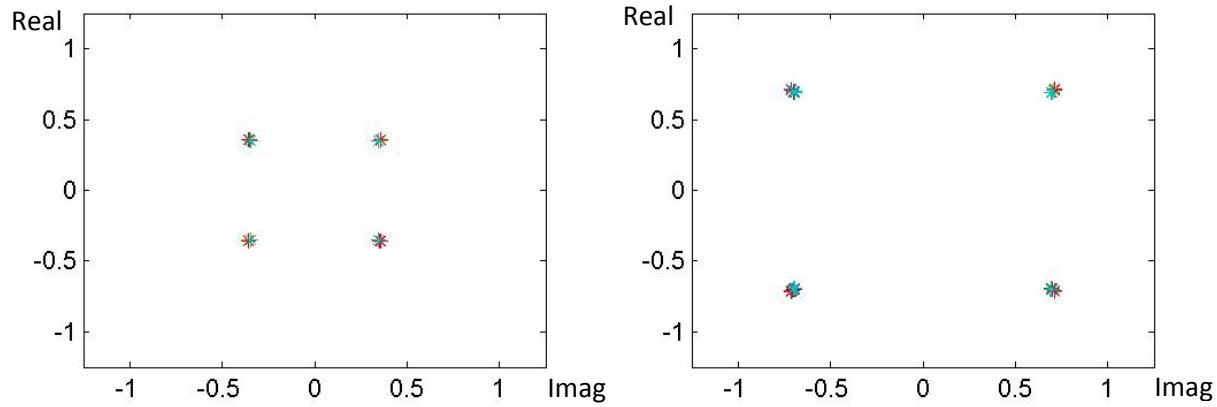


Figure 60: "Not equalized and equalized symbols in a QPSK modulation with 125 db SNR and 0 BER in a scenario without slow fading and additive noise"

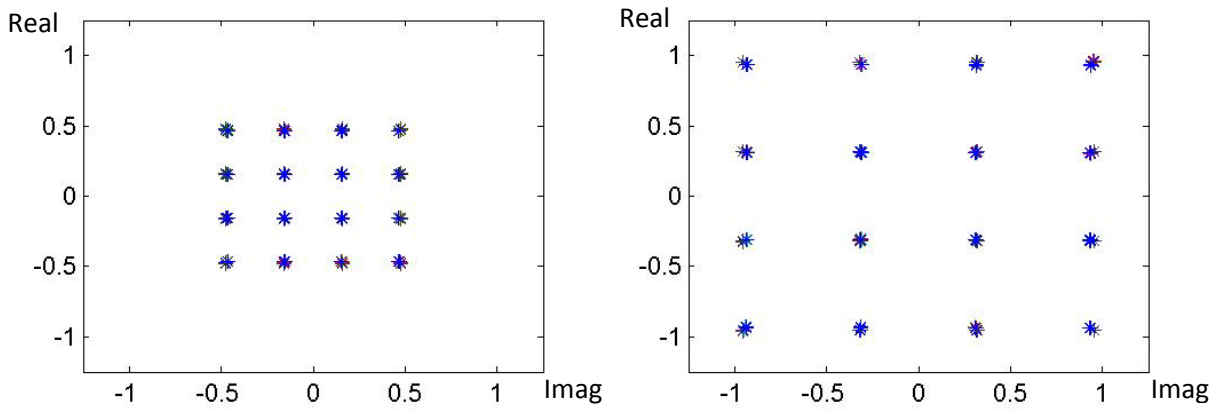


Figure 61: "Not equalized and equalized symbols in a 16-QAM modulation with 125 db SNR and 0 BER in a scenario without slow fading and additive noise"

Below are a few simulations in different scenarios.

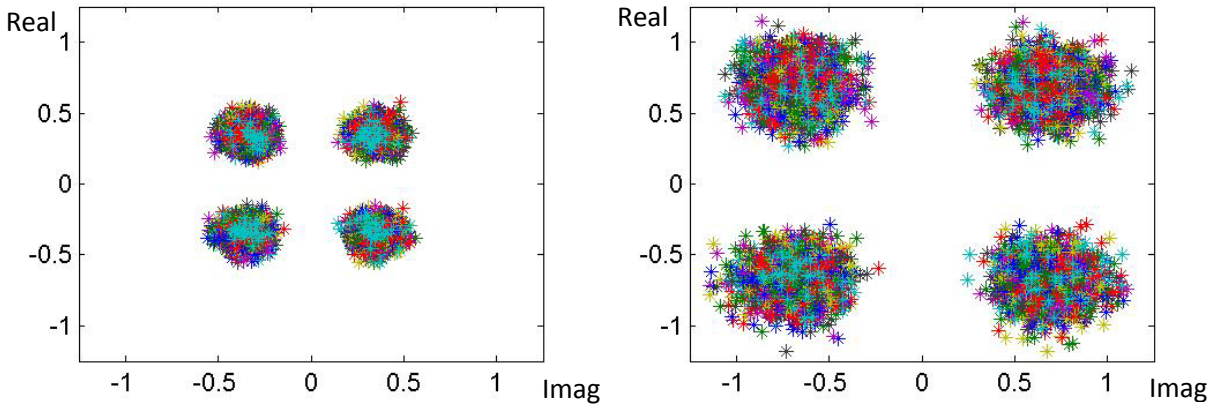


Figure 62: "Not equalized and equalized symbols in a QPSK modulation with 15 db SNR and 0 BER in a scenario without slow fading and additive noise"

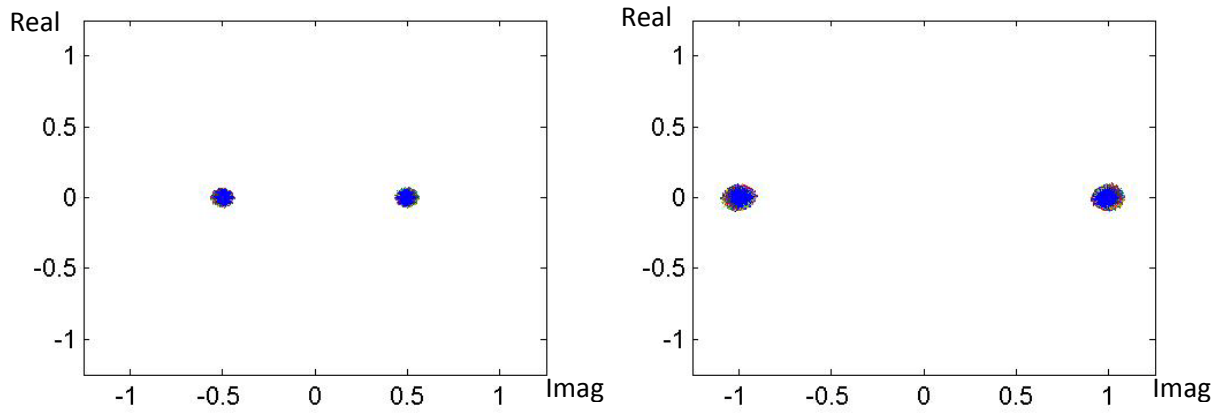


Figure 63: "Not equalized and equalized symbols in a BPSK modulation with 32 db SNR and 0 BER in a scenario without slow fading and additive noise"

Once slow fading channel is passed, soft symbols are obtained. These are scattered but are compensated properly with the equalizer. Note the factor fd is increase, its associated BER is increased over the same SNR. This is due to the system must adapt to the channel more quickly, hence having more error signal.

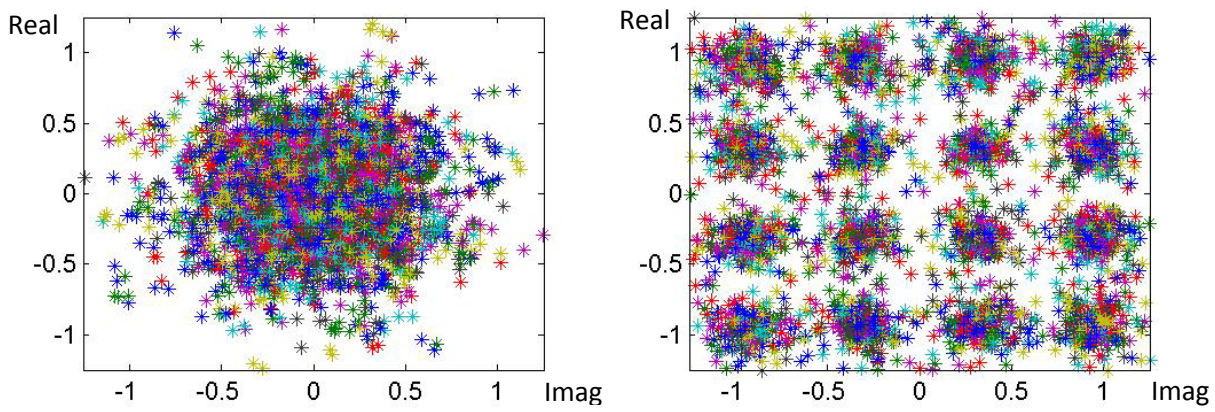


Figure 64: "Not equalized and equalized symbols in a 16-QAM modulation with 23 db SNR and 0.025 BER in a scenario with slow fading (fd=0.05) and additive noise"

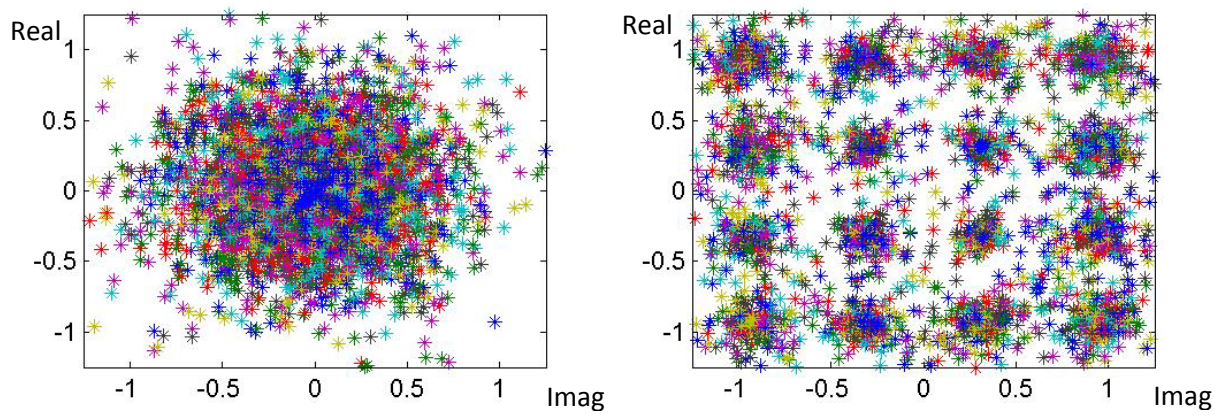


Figure 65: "Not equalized and equalized symbols in a 16-QAM modulation with 23 db SNR and 0.025 BER in a scenario with slow fading (fd=0.1) and additive noise"

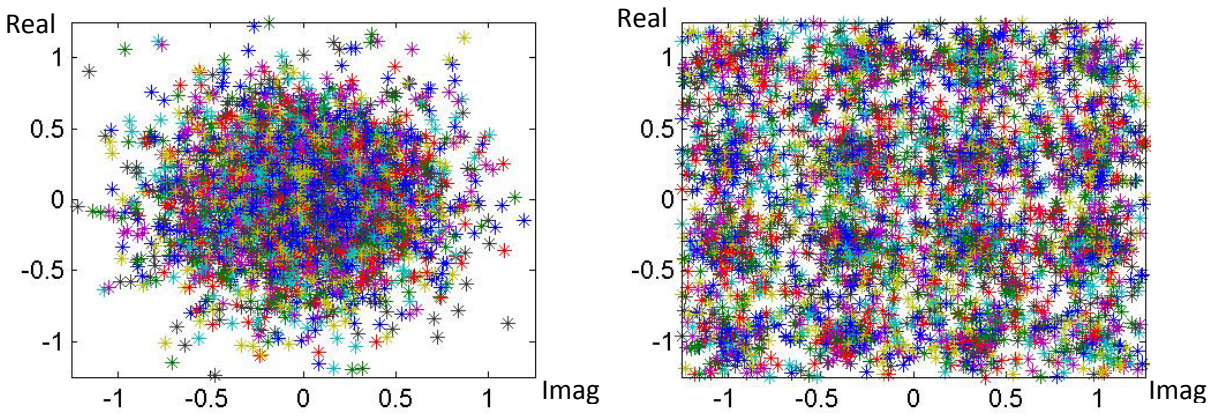


Figure 66: "Not equalized and equalized symbols in a 16-QAM modulation with 24 db SNR and 0.118 BER in a scenario with slow fading (fd=0.4) and additive noise"

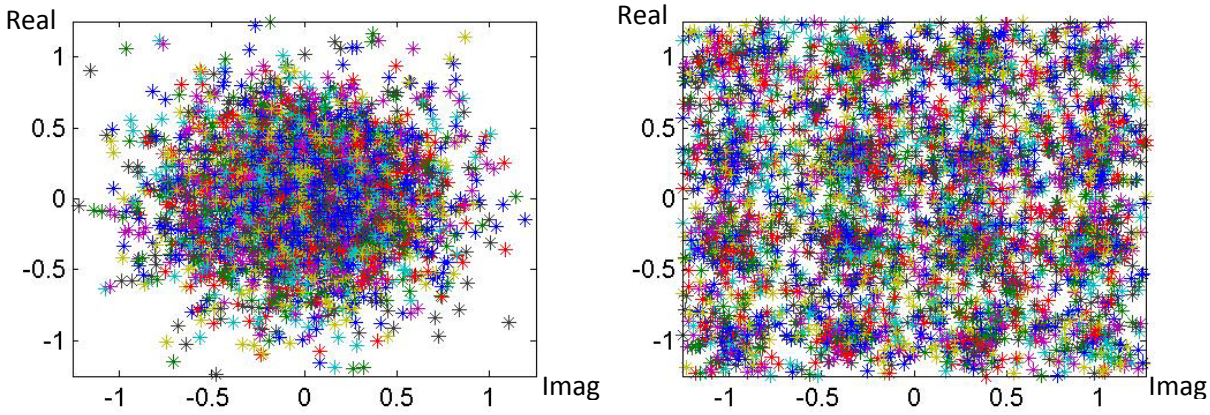


Figure 67: "Not equalized and equalized symbols in a 16-QAM modulation with 27 db SNR and 0.18 BER in a scenario with slow fading (fd=0.6) and additive noise"

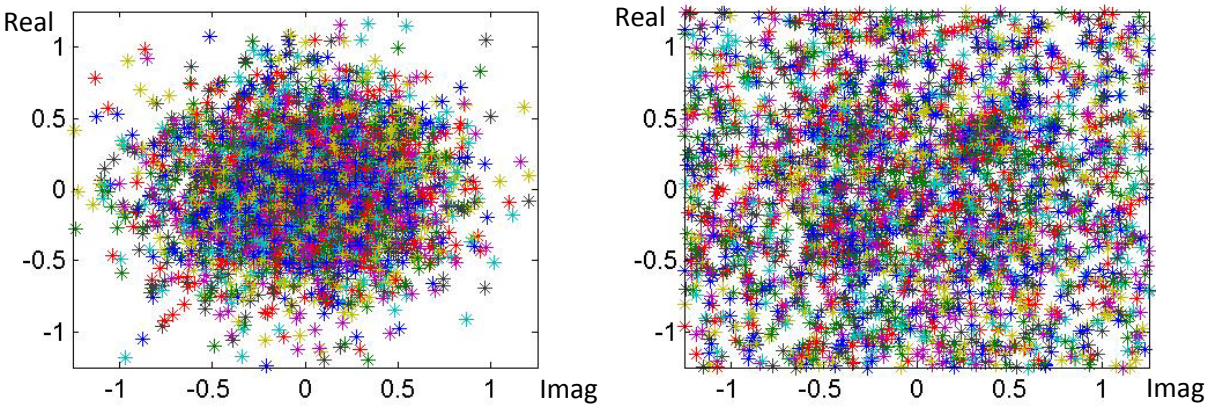


Figure 68: "Not equalized and equalized symbols in a 16-QAM modulation with 26 db SNR and 0.229 BER in a scenario with slow fading (fd=0.8) and additive noise"

Note that the gain obtained from the spreader average 14 dB.

The last step is to convert symbols to a sequence of bits and decode them.

6. Conclusion

Objectives initially set have been achieved, but initial estimated time for completion of this project has deviated from the estimate. I enjoyed my time doing this project, while learning a lot.

The main deviated factor from initial prediction has been the need to adapt the code to the restrictions of memory of a DSP board, as it has had to develop modules unchanged. This point has required great dedication and effort on my part.

The differences between the simulations are mainly due to two reasons:

- The first approach is introduced by using finite filters.
- The other is the random nature of each simulation.

The ideal channel implementations can recover the signal without any problem if inter-symbolic interference is bounded within a certain margin. However, when using a slowly fading channel is necessary to introduce an equalizer to compensate for this. This equalizer uses information from an adjacent channel frequency with a known sequence to estimate the effect of the channel.

Another key point in this project has been the acquisition of signal using a standard correlator allows a parameter completely independent of the power used in the receiver.

On a personal level this work has allowed me to develop into the theory and implementation of a modular system based on CDMA. In addition to the technical side, this project has consolidated much of the knowledge acquired during the career.

7. Future work lines

Although work on this project allows for a wide System Configuration, there are several future works that have not been pursued by economic or time constraints. Here are two main steps can be taken to continue the work on this project.

1. **System implementation in DSP hardware boards**

   The next step in continuing this project is to conduct a more practical implementation of the system analyzed in this project. This will require translating the C and assembler code when necessary.

   - Conversion of C code

     The system has been implemented in MATLAB but have been scheduled taking into account the memory constraints inherent in a DSP board. This system is unable to deal with problems arising from stacks since Matlab is a high-level language that allows the use of low-level programming pointers. The stack is an abstract data type and its structure is based on the principle of Last-In-First-Out (LIFO).

   - Bottleneck Optimization (loops)

     A key point is the optimization of the key points, bottlenecks; the code for the application can run in real time. Possible elements that must be optimized are the cosines, sinus and other intensive operations such as convolution. There are a variety of standard measures that developers can take to reduce the complexity of these operations in C and the use of intrinsic functions that are designed to optimize the resources of the DSP board, boards search, compiler options, and algorithm improvements.

     - Using intrinsic functions of the DSP board

       As mentioned previously there are features specifically designed and optimized for DSP, are called intrinsic functions. They have been programmed in assembly language and need much less time to be executed that implemented in C.

     - Conversion to assembly code

       The function should be optimized in assembler restrictive where there is an intrinsic function of a bottleneck.

- Software pipeline

  The C compiler pipeline software used to organize the statements inside the loops so that more than one instruction can be executed in parallel. Unfortunately, not all loops can be optimized by software pipeline. For example, loops that contain conditional statements (branches) may not be channeled.

2. Implementation of new algorithms in Matlab.

This system allows new algorithms to check each of the components that can range from a new coding, modulation, different pulse modulated, new signal acquisition algorithms or modeling of other channel types.

8. References

To carry out this project I have consulted a number of books, papers and other information of interest.

- [AMTR 2010] Amit Tripathi, M.S. Korde, “DSSS Based CDMA ModemUsing FPGA & Microcontroller”, 2010
- [DAHNO 2000] N. Dahnoun, “Digital Signal Processing: Implementation using the TMS320C6000 DSP platform”, Prentice Hall, 2000
- [HEME 1997] Heinrich Meyr, Marc Moeneclaey, and Stefan A. Fechtel, “Digital Communication Receivers: Synchronization, Channel Estimation, and Signal Processing”, Wiley-Blackwell; Nov 1997
- [JAWI 1974] Jakes, William C., ed. Microwave Mobile Communications, New York, IEEE Press, 1974.
- [JEMI 2000] Jeruchim, Michel C., Philip Balaban, and K. Sam Shanmugan, Simulation of Communication Systems, 2000
- [LEWI 1993] Lee, William C. Y., “Mobile Communications Design Fundamentals”, Second Edition<, New York, John Wiley & Sons, 1993.
- [MASI 2000] Marvin K. Simon and Mohamed- Slim Alovini, “Digital communication over fading channels” Second edition, John Wiley & Sons, Inc., 2000
- [TSVI 2005] D. Tse and P. Viswanath, "Fundamentals of Wireless Communication", Cambridge University Press, May 2005
- [TONO 1992] Tomás Novosad, "A binary sequences for CMA based on finite geometry", Spectrum Techniques and Applications (ISSTA), 1992
- [VITE 1995] Viterbi “Principles of Spread Spectrum Communication”, Addison-Wesley Publishing Company 1995