ROYAL INSTITUTE OF TECHNOLOGY

PROJECT IN WIRELESS COMMUNICATIONS
EQ2440

Project Report

Authors:

Alan ANTER

Daniel BREDSTEDT

Sergio A. CHÁVEZ CÁRDENAS

Henrik FORSELL

Johan OTTERSTEN

Francisco ROSÁRIO

Jue ZHANG

May 14, 2014

Abstract

The usage of mobile phones connected to a base station is strictly forbidden inside airplanes because of the inherent risk of interference with the navigation or communication system of the aircraft. For this reason, there is a flight mode in most devices, which turns off all RF signals. However, passengers often have the necessity to transmit files with their phones during the flight.

This report describes a way to design a communication method to transfer files between two smartphones using a 3.5 mm audio cable. The objective is to program an application that is able to transmit and receive an electronic file using the phone's embedded sound card. The system was applied in real time on two Samsung Galaxy S3 samrtphones running Android 4.2.1 operating system.

ADD: maximum speed of filetransfer... And what simulations have been done for the modulation

Contents

List of Figures

List of Tables

Acknowledgements

Thanks to the reader, if you are reading this line after the others, you at least read one page of this long report. Thank you.

Chapter 1

Introduction

ADD: A description of the document and its sections.

Chapter 2

Background

Before describing the implementation, it is important to study the possible coding and modulation schemes in order to determine the best combination for the given problem. In this section, the fundamental principles of each technique are analysed.

2.1 Modulation schemes

2.1.1 ASK

Amplitude shift keying (ASK) uses the variations in amplitude of a carrier wave to represent the digital data. The mapping or allocation of k information bits to the $M = 2^k$ possible signal amplitudes can be done in different ways. The preferred mapping is the one in which the adjacent signal amplitudes differ by one binary digit (Gray coding). The simplest case for the transmitted signal, when only two amplitudes coexist, are shown in figure 2.1.
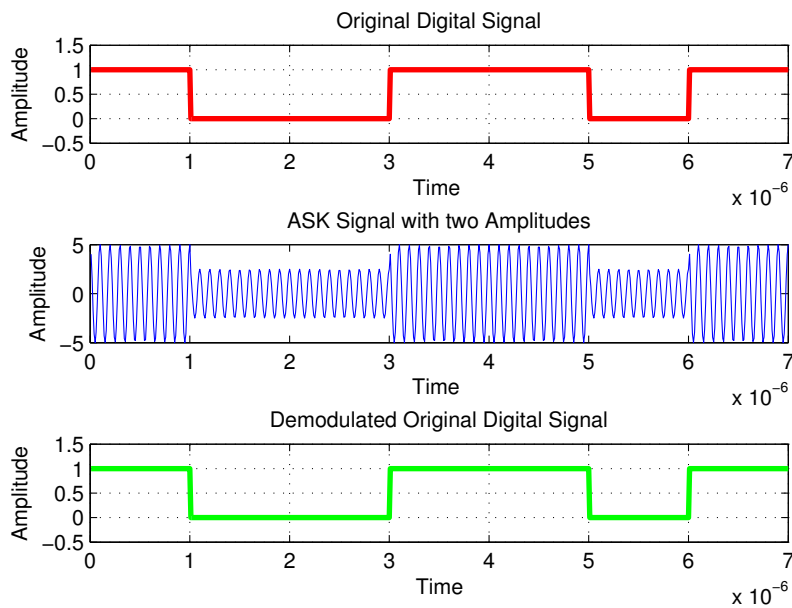
Figure 2.1: ASK modulation example in the absence of noise for amplitudes $a_1 = 2.5$ and $a_2 = 5$.

It is important to have an high signal-to-noise ratio (SNR), especially when the amplitudes are close so that the demodulator is able to identify them. The ASK scheme is very sensitive to atmospheric noise, distortions and propagation conditions. A possible demodulator is shown in figure 2.2.
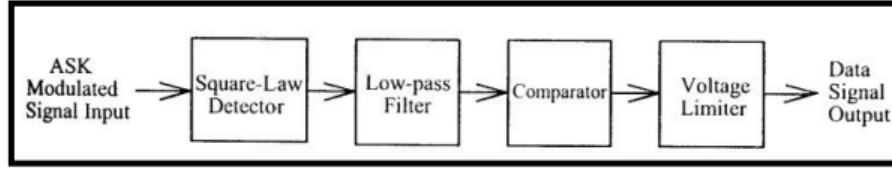
Figure 2.2: Possible demodulator for ASK[3].

2.1.2 PSK

Phase-shift keying (PSK) is a digital modulation scheme that conveys data by changing, or modulating, the phase of a carrier wave. A finite number of phases are assigned to an unique pattern of binary digits. Usually, each phase encodes an even number of bits. The demodulator, which is designed specifically for the symbol-set used by the modulator, determines the phase of the received signal and maps it back to the corresponding signal. In digital phase modulation, the number of signal waveforms M is represented as:

$$
s_i(t) = A\cos(\omega_c t + \phi_i(t)) = \overbrace{A\cos(\phi_i(t))}^{in-phase \text{ symbol } x_i(t)} \cdot \cos(\omega_c t) \underbrace{-A\sin(\phi_i(t))}_{quadrature \text{ symbol } x_q(t)} \cdot \sin(\omega_c t)
$$
$$
= x_i(t)\cos(\omega_c t) + x_q(t)\sin(\omega_c t) ,
$$

(2.1)

where A is the amplitude of the signal and ω_c is the carrier frequency in radians per second. The decision regions for two different M-PSK schemes are depicted in figure 2.3.

Alternatively, instead of using the bit allocation to set the carrier's phase it can be used to change the phase by a given angle. Therefore, the demodulator determines the changes in the phase of the received signal, rather than the phase. Since this scheme depends on the difference between successive phases it is called differential phase-shift keying (D-PSK).
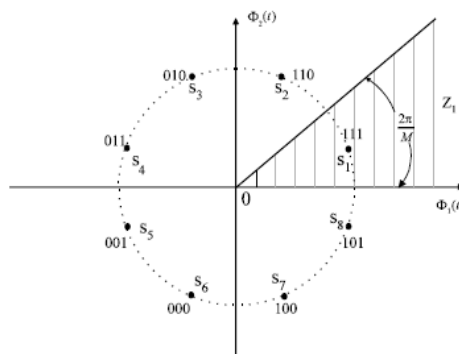
Figure 2.3: Decision regions for 8-PSK. The minimum distance between symbols is given by $d_{min} = A\sin\frac{2\pi}{M}$ [6].

One of the advantages of PSK/D-PSK is that it is more bandwidth efficient compared to the FSK scheme. However, the signal detection/recovery is much harder compared to ASK and FSK since the demodulator must determine the phase of received sinusoid with respect to some reference phase. Additionally, if a higher rate PSK scheme is used, the equipment must be capable of distinguishing small differences in phase, making it unusable in time-varying channels.

2.1.3 FSK

Frequency shift keying (FSK) is a modulation technique in which data is transmitted through discrete frequency changes of a carrier wave with fixed amplitude A. In figure 2.4 the simplest form of FSK (e.g. binary FSK or BFSK) is illustraded. A sinusoid carrier $s_i(t)$ is transmitted for each bit, zero or one, using the carrier frequencies f_0 and f_1, respectively.
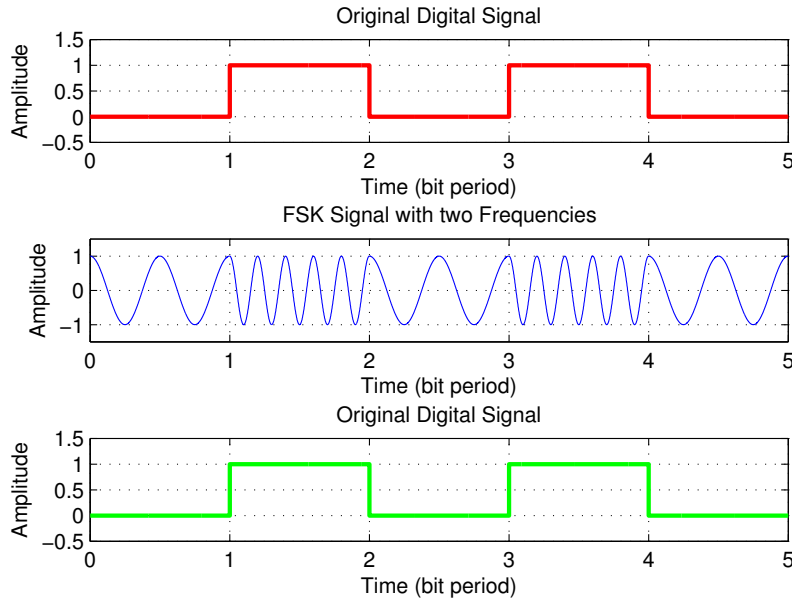
Figure 2.4: BFSK example for 2 low frequency carriers.

We can use multiple carriers to transmit $n > 1$ bits. In order to guarantee that the symbols are orthogonal there should be a $\Delta f = \frac{k}{2T}$ separation among the $M = 2^n$ carrier frequencies, where k is an integer and T is the bit period. The complex baseband signalling waveforms for M-ary FSK [1] are given in the following equation:

$$s_i(t) = e^{j2\pi f_i t} I_{[0,T]}, i = 1, 2, ..., M \tag{2.2}$$

Coherent and non-coherent demodulation methods exist and depend on whether the phase of the sinusoids ϕ are equal or not. Coherent demodulators consist of correlators (matched filters) and samples, but they require synchronisation of the reference signals. The demodulator for BFSK can be implemented with two correlators as shown in figure 2.5a. This receiver is optimum in the sense that it minimizes the error probability for equally-likely binary signals. If for example a sinusoid carrier with frequency f_1 is transmitted, the upper correlator yields a signal l_1 with a positive signal

component while l_2 has only a noise component. This is due to orthogonality of the signals. In case there is no phase reference, additional squarers or envelope detectors must be used for the non-coherent detection. In the last case, a minimum tone spacing of $\frac{k}{T}$ is required instead. The correlator implementation for non-coherent BFSK is illustrated in figure (2.5b).

(a) Coherent.
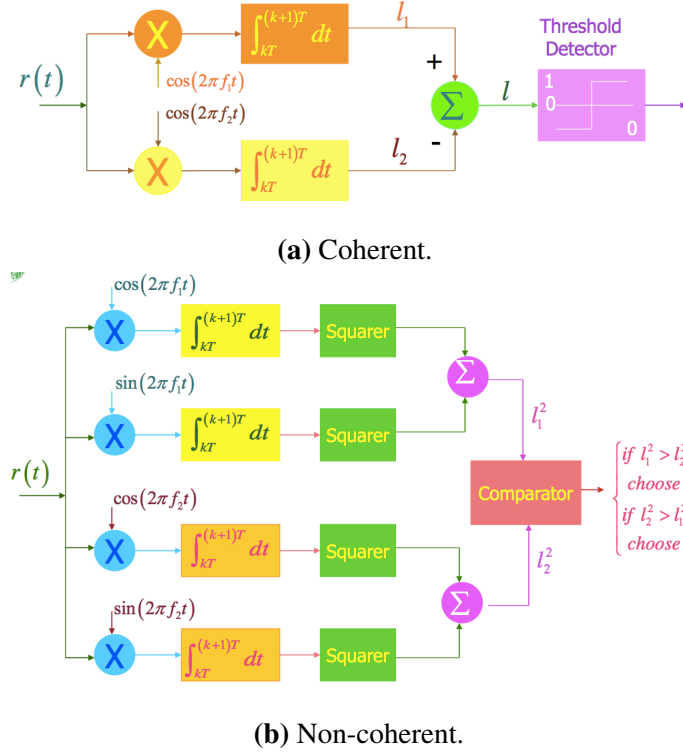
(b) Non-coherent.

Figure 2.5: Possible Demodulators for FSK [4].

It is expected that the error performance for the non-coherent receivers is greater than the coherent ones. However, the degradation is only a fraction of a decibel. Other options for demodulation are available such as the use of FFT schemes for peak picking or bandpass filtering at the working frequencies.

Performance

FSK systems have the highest power efficiencies among the available modulation schemes but they suffer from the low bandwidth (BW) efficiency. The main advantages of this modulating scheme are: the in-sensitiveness to amplitude variations in the channel, its compatibility with non-linear transmitters and receivers, and that it is not required to have an absolute frequency accuracy for correct demodulation (making it tolerant to local oscillators, drifts and Doppler shifts). One of the drawbacks is the fact that it is less bandwidth efficient when compared with other schemes such as ASK or PSK. To avoid undesirable spectral characteristics (which could cause adjacent channel interference) continuous phase FSK is required (see figure 2.6 for comparison).

(a) Non continuous phase FSK.

(b) Continuous phase FSK.
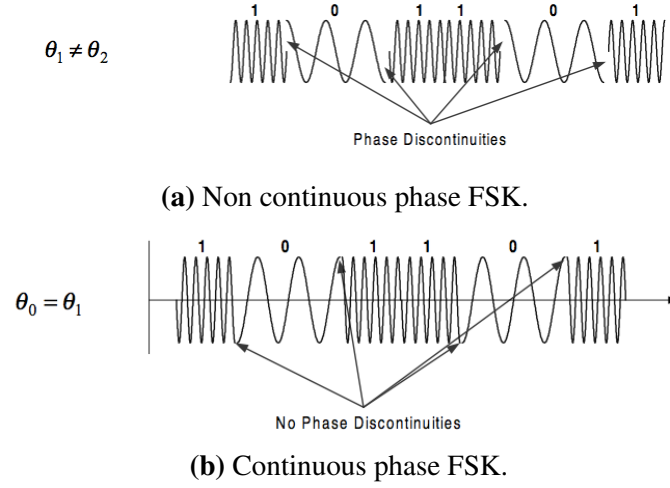
Figure 2.6: FSK modulated signal, with and without phase discontinuity[5].

The M-ary FSK modulated signals can be written as follows:

$$s_m(t) = A\varphi_m(t), \quad 0 \le t \le T_s \tag{2.3}$$

We can read T_s, from equation 2.3, as the period of the input symbol stream, and A as the amplitude of the basis function φ_m. In addition, the basis functions can be described by:

$$\varphi_m(t) = \sqrt{\frac{2}{T}} \cos\left(2\pi(f_c + \alpha_m \Delta f_c) + \theta_m\right), \quad 0 \le t \le T \tag{2.4}$$

where

$$\alpha_m \in \{(2m - 1 - M)|m = 1, 2, ..., M\}$$

In order to satisfy orthogonality the inner product between the basis functions must be 1 for $m = n$ and 0 for $m \ne n$:

$$\langle \varphi_m, \varphi_n \rangle = \int \varphi_m(t)\varphi_n(t)dt = \delta_{mn} = \begin{cases} 1, m = n \\ 0, m \ne n \end{cases} \tag{2.5}$$

Individual frequencies corresponding to different symbols are given by equation 2.6,

$$f_m = f_c + \alpha_m \Delta f_c \tag{2.6}$$

where f_c is the central frequency and Δf_c is the frequency separation.

In order to satisfy orthogonality [7] must apply:

$$\Delta f = 2\Delta f_c = \begin{cases} \frac{1}{2T}, & \theta_m = \theta_n, \quad \forall m, n \\ \frac{1}{T}, & \theta_m \ne \theta_n \end{cases} \tag{2.7}$$

The overall bandwidth occupied by the FSK signal depends on the modulation index h described in equation 2.8.

$$h = \frac{2\Delta f_c}{R} = 2\Delta f_c T_s \tag{2.8}$$

15

For continuous phase BFSK the minimum value for the signals to not interfere with each other is $h = 0.5$. An FSK system using continuous phase transitions has much lower side-lobe energy than the discontinuous case. All these parameters will cause inter-symbol-interference (ISI) and limit the rate of the system. For this reason it is important to further increase the bandwidth efficiency of the system by applying pulse shaping to the transmitted signal. This technique is described in section 2.2.

2.1.4 M-QAM

M-ary Quadrature-Amplitude modulation is a two-dimensional signalling scheme that combines both Amplitude and Phase Shift Keying. The transmitted signal for every symbol can be written as [2]:

$$s_i(t) = \sqrt{\frac{2E_0}{T}} a_i \cos(2\pi f_c t) - \sqrt{\frac{2E_0}{T}} b_i \sin(2\pi f_c t), \quad \begin{cases} 0 \le t \le T \\ i = 0, \pm 1, \pm 2, \ldots \end{cases} \tag{2.9}$$

The typical in phase (I) and quadrature (Q) known as IQ modulation and demodulation is shown in figure 2.7. We consider that the information to be transmitted is a complex signal $x = x_i + jx_q$. Ignoring the energy and period factors, the output of the IQ modulation transmitter is given by [1]:

$$y(t) = \Re\{x e^{j2\pi f_c t}\} = x_i \cos(2\pi f_c t) - x_q \sin(2\pi f_c t) \tag{2.10}$$
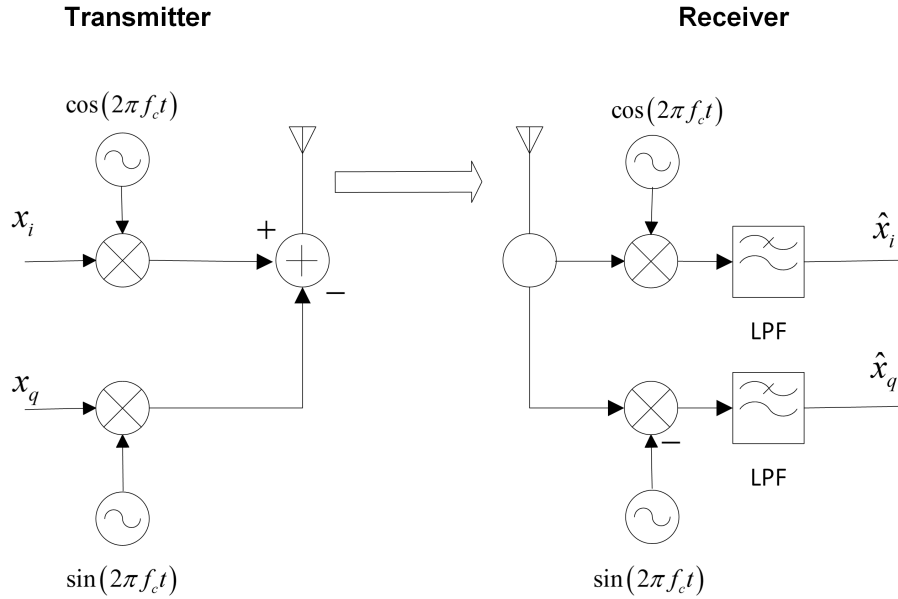
Figure 2.7: Schematic of M-QAM transmitter and receiver showing the in-phase (I) and quadrature (Q) components.

At the transmitter the information x_i is mapped to $\cos(2\pi f_c t)$ and x_q is mapped to $\sin(2\pi f_c t)$. At the receiver we multiply the signal with $\cos(2\pi f_c t)$ and $-\sin(2\pi f_c t)$ followed by low-pass filtering to extract $\hat{x}_i$ and $\hat{x}_q$. The mathematical expressions for the recovered information at the receiver, using the trigonometric identities, can be summarized as followed [1]:

$$\hat{x}_i = \int_0^T y \cdot \cos(2\pi f_c t)dt = \int_0^T \left(x_i \cos(2\pi f_c t) - x_q \sin(2\pi f_c t)\right) \cdot \cos(2\pi f_c t)dt \overset{LPF}{=} \frac{x_i}{2} \qquad (2.11)$$

$$\hat{x}_q = \int_0^T -y \cdot \sin(2\pi f_c t)dt = \int_0^T -\left(x_i \cos(2\pi f_c t) - x_q \sin(2\pi f_c t)\right) \cdot \sin(2\pi f_c t)dt \overset{LPF}{=} \frac{x_q}{2} \qquad (2.12)$$

By ignoring the scaling factor of $\frac{1}{2}$ and assuming that the channel is ideal, both x_i and x_q can be recovered successfully.

QAM symbol mapping

As stated previously the QAM scheme involves sending digital information by periodically adjusting the phase and amplitudes of a sinusoidal wave. For example 4-QAM consists of four unique combinations of phase and amplitude. These combinations, called symbols, are shown as the white dots on the constellation plot in figure 2.8. It is possible to send up to two bits per symbol when using 4-QAM modulation. It is also possible to send data at even higher rates by increasing the number of symbols in our symbol map. By convention the number of symbols in a symbol map is called the symbol map "M" and is considered the "M-ary" of the modulation scheme. The number of bits in an M-QAM scheme is given by $\log_2 M$.

Figure 2.8: Constellation plot of combinations of phase and amplitude in 4-QAM. The blue lines represent the phase and amplitude transitions from one symbol to another. Labelled in the constellation plot, there is the digital bit pattern that each symbol represents.

Impairments

Although QAM appears to increase the transmission efficiency for digital communication systems by utilising both amplitude and phase variati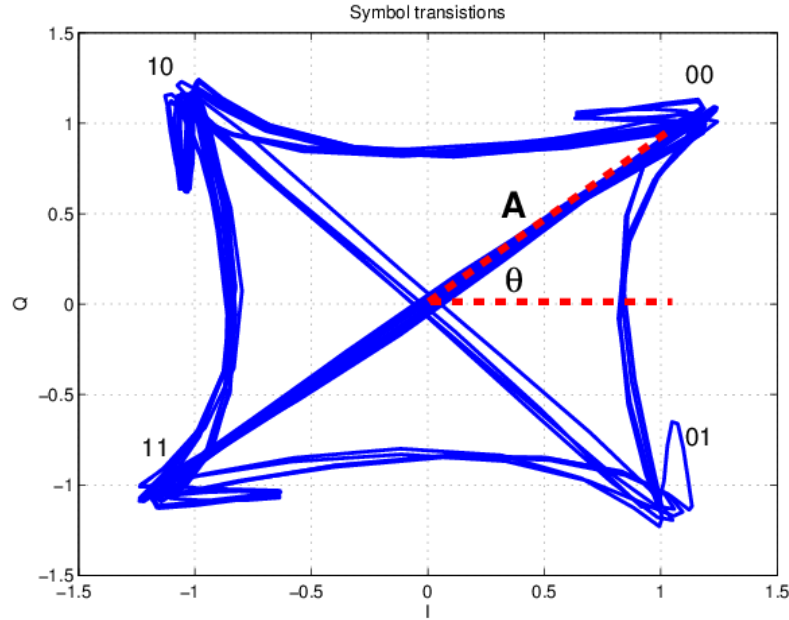ons, it has a number of drawbacks. It is more susceptible to noise because the states are closer together. This means a lower level of noise is needed to move the signal to a different decision point. The second limitation is that linearity must be maintained when the signal is passed through the channel and amplifiers. Both these limitations are associated with the amplitude component of the signal.

In an ideal I-Q modulator the phase difference between the signals used for modulating the I and Q arm is 90 degrees, resulting in using $\cos(2\pi f_c t)$ and $\sin(2\pi f_c t)$ for sending x_i and x_q. If there is phase imbalance, the phase difference might not be exactly 90 degrees. In that case, we can consider that $\cos(2\pi f_c t)$ and $\sin(2\pi f_c t + \varphi)$ carriers are used for transmitting instead. When there is amplitude imbalance there will be added small level variations in the sine and cosine carriers at the modulator. This can be modelled by using $\cos(2\pi f_c t)$ for the in-phase component x_i and $(1+\alpha)\sin(2\pi f_c t)$ to carry the quadrature component x_q.

The transmitted signal including the effect of phase and amplitude imbalance is then [1]:

$$y(t) = x_i \cos(2\pi f_c t) - x_q(1 + \alpha)\sin(2\pi f_c t + \varphi), \quad \text{where } 0 < \alpha < 1 \tag{2.13}$$

Assuming that we have an ideal IQ demodulator at the receiver we split and multiply the signal $y(t)$ with $\cos(2\pi f_c t)$ and $\sin(2\pi f_c t)$ carriers. The outcome of such multiplications is low-pass filtered, allowing to extract $\hat{x}_i$ and $\hat{x}_q$, respectively:

$$\hat{x}_i = \frac{1}{2}\Big[x_i - x_q(1 + \alpha)\sin(\varphi)\Big] \tag{2.14}$$

$$\hat{x}_q = \frac{1}{2}\Big[x_q(1 + \alpha)\cos(\varphi)\Big] \tag{2.15}$$

These non-idealities leads to a quadrature skew that causes a different mapping in both the transmitter and receiver. IQ gain imbalance equates to a rectangular stretch along the IQ axis. Likewise, quadrature error results in the IQ axis not being exactly 90° apart. These phenomena can be seen in figure 2.9.

(a) IQ gain imbalance.

(b) IQ quadrature error.
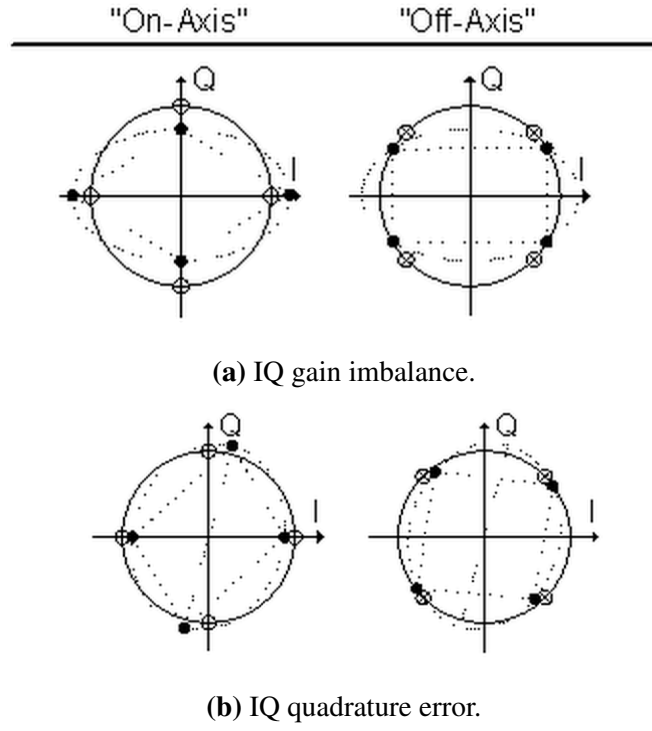
Figure 2.9: Two possible constellation mappings for 4-QAM and the effects of impairments on phase and amplitude[14].

Performance and Impact of Noise

When studying the impact of the noise in QAM systems we assume that Gray coding is used, all symbols are equiprobable, the noise is zero-mean additive white gaussian noise (AWGN) with variance $\frac{N_0}{2}$ and there are no errors from carrier recovery or symbol synchronisation. In a square M-QAM system the amplitudes of x_i and x_q can take $\log_2 M$ different values. Each level can be $-(\sqrt{M}-1)d, -(\sqrt{M}-3)d, \cdots, (\sqrt{M}-1)d$, where d is half of the minimum distance between two symbols. The value of d can be computed as [15]:

$$d = \sqrt{\frac{3\log_2(M) \cdot E_b}{2(M-1)}} \,, \tag{2.16}$$

where E_b is the energy per bit in the transmitter. Recall that at the demodulator, the received vector is given by [15]:

$$\mathbf{r} = \mathbf{s} + \mathbf{n} = \begin{bmatrix} I \\ Q \end{bmatrix} + \begin{bmatrix} n_i \\ n_q \end{bmatrix} \,, \tag{2.17}$$

The proof of the BER expression for the MQAM is out of the scope of this report. However the final generic result is given as:

$$P_b = \frac{1}{\log_2 \sqrt{M}} \sum_{k=1}^{\log_2 \sqrt{M}} P_b(k) \,, \tag{2.18}$$

The conditional BER, $P_b(k)$ in equation 2.18, corresponding to the k-th bits ($k = 1, 2, 3, \ldots \log_2 \sqrt{M}$)

on both I and Q components is defined as follows [15]:

$$P_b(k) = \frac{1}{\sqrt{M}} \sum_{j=0}^{(1-2^{-k})\sqrt{M}-1} \left[(-1)^{\left\lfloor \frac{j2^{k-1}}{\sqrt{M}} \right\rfloor} \left(2^{k-1} - \left\lfloor \frac{j2^{k-1}}{\sqrt{M}} + \frac{1}{2} \right\rfloor \right) \cdot \text{erfc} \left((2j+1) \sqrt{\frac{3\log_2(M) \cdot E_b}{2(M-1)N_0}} \right) \right] \quad (2.19)$$

The outcome of these expressions is displayed in figure 2.10.
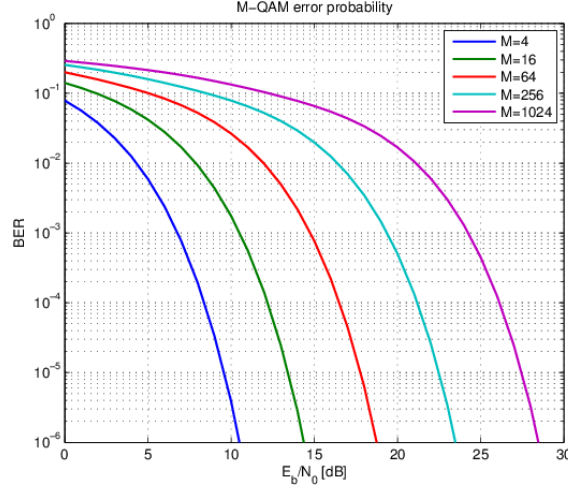
Figure 2.10: BER performances of M-ary QAM.

2.1.5   OFDM

OFDM or Orthogonal Frequency-Division Multiplexing is a multi-carrier modulation scheme that divides the available bandwidth W into N sub-bands with relatively narrow width, $\Delta f = \frac{W}{N}$. The signal in each sub-band can be independently encoded and modulated as a synchronous symbol with rate $T = \frac{1}{\Delta f}$. If Δf is small enough the channel frequency response $C(f)$ is essentially constant across each sub-band. Hence, the ISI is negligible and OFDM provides a solution that could yield transmission rates close to the channel capacity.

By choosing the modulating waveforms to be *eigen-functions* of a linear-time invariant channel (LTI), it can be guaranteed that the waveforms will remain orthogonal after going through the channel. Therefore, the fact that complex exponentials $\exp(j2\pi f_k t)$ are orthogonal at different frequencies f_k can be exploited. Suppose that the N data to be transmitted are $X_k, k = 0, 1, ..., N-1$, where X_k is a complex number in a given constellation. If a carrier frequency f_k is assigned to each symbol X_k and each contribution is summed at the output, the following result is obtained:

$$x(t) = \sum_{k=0}^{N-1} X_k e^{j2\pi f_k t} \quad (2.20)$$

For the implementation of a digital system a transmitter will generate its output in a sampled-data fashion. By letting $t = nT_s$ and $f_k = k\Delta f$, where T_s is the sample interval and Δf the sub-carrier

spacing (assumed to be uniform), the output is given by:

$$x(nT_s) = \sum_{k=0}^{N-1} X_k e^{j2\pi k \Delta f n T_s} \tag{2.21}$$

The sub-carrier spacing should be computed such that the orthogonality of different sub-carriers holds over an interval of length T:

$$\int_0^T e^{j2\pi f_m t} e^{-j2\pi f_n t} dt = \frac{e^{j2\pi(f_n - f_m)t} - 1}{j2\pi(f_n - f_m)} = 0, \text{ for } (f_n - f_m)T \in \mathbb{Z}_{\neq 0} \tag{2.22}$$

Hence, $\Delta f = \frac{1}{NT_s} = \frac{1}{T}$ is the minimum separation to keep orthogonality among signals at different modulators, resulting in:

$$x(nT_s) = \sum_{k=0}^{N-1} X_k e^{j2\pi kn/N}, \, n = 0, 1, ..., N-1 \tag{2.23}$$

The above formula is the equation of an N-point IDFT for one OFDM symbol. If N is a power of two the samples $x(nT_s)$ can be efficiently generated (for example by using the FFT algorithm). A plot of the resulting signal in time and frequency domain is shown in figure 2.11.
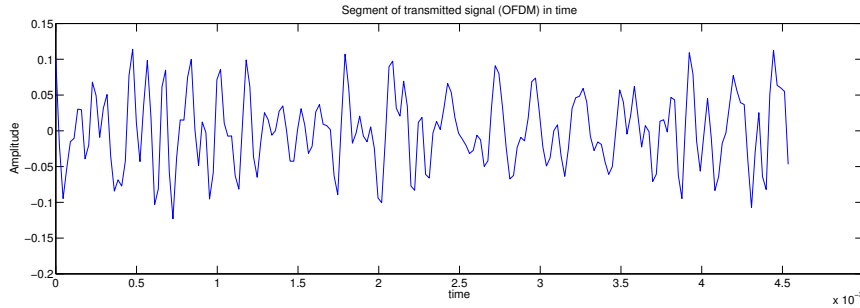
(a) OFDM signal segment.
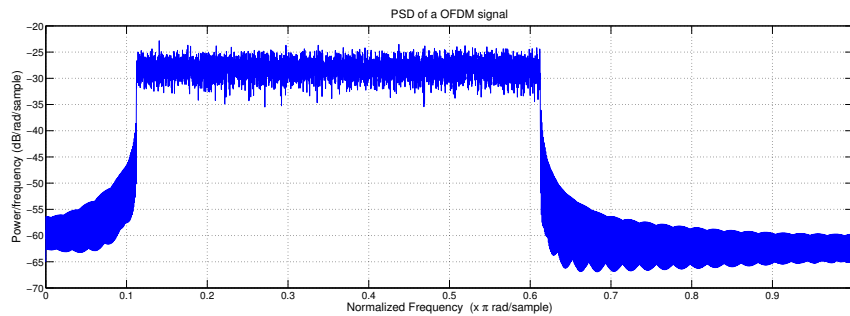
(b) PSD of a OFDM signal

Figure 2.11: Example of an OFDM signal centred at 8 kHz with a DFT size of $N = 2048$, the number of active subcarriers $N_c = 512$ are with a sampling frequency of $f_s = 44.1$ kHz.

Consider move this section after OFDM transceiver architecture In multi-path channels a receiver can receive several delayed replicas of the transmitted signal causing ISI. To eliminate the effect of inter symbol interference (ISI), a guard interval of N_g samples is usually inserted at the beginning of

each OFDM symbol. During the guard interval the transmitter can either:

- Zero pad (ZP) the transmission by sending a null waveform.

- Introduce a cyclic prefix (CP) which is an exact copy of a segment of the OFDM symbol located towards the symbol end. A cyclic suffix can be used in a similar fashion.

Since ZP introduces inter-carrier interference(ICI) cyclic prefixing transmission is preferred. To prevent significant leakage to adjacent bands, OFDM systems usually do not transmit any data on the sub-carriers near the two edges of the allocated band. These unused sub-carriers are known as guard sub-carriers or virtual sub-carriers. The collection of all the unused sub-carriers is called the guard band. As the PSD of the OFDM signal has quite high sidelobes, to reserve a guard band helps to minimise out-of-band emissions and thus eases the requirements on transmitter front-end filters. Nevertheless, adoption of a guard band misspend assigned bandwidth and decreases the spectral efficiency of the OFDM system. In addition to guard bands some sub-carriers around DC frequency (subcarrier index 0) should be nullified in order to evade unwanted DC and low-frequency components generated by the receiver's front-end. Other issues may arise when using of OFDM such as spectral shaping and peak-to-average power ratio. These issues are discussed in section 2.2.

OFDM transceiver architecture

A general OFDM transmitter integrates several functions including DFT processing, guard interval insertion and spectral shaping. In a receiver, besides DFT processing and guard interval removal, additional efforts are required to handle the channel-fading effect and synchronisation issues between the transmitter and receiver (discussed in section 2.3). As it will be discussed in part 2.3 the amplitude and phase maybe estimated via the use of a known training sequence. In an OFDM system these parameters should be estimated for each of the carriers frequencies. In the case of a time variant channel the new estimations can be done by inserting known reference symbols, sometimes referred to as pilot symbols, at regular intervals within the OFDM time-frequency grid. The effects of an amplitude varying channel and the previously mentioned grid are depicted in figure 2.12. Add references and footnotes from books if they fit better.
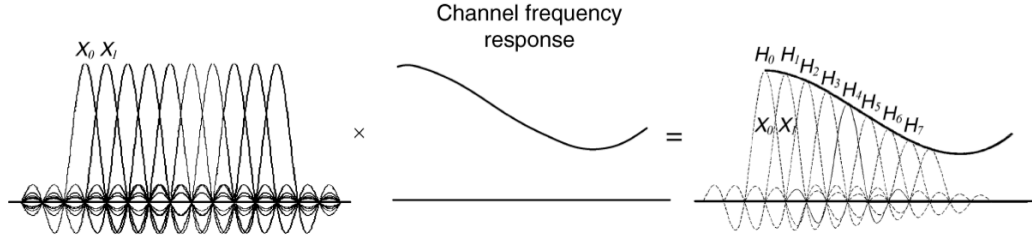
(a) The effects of a channel with non-flat frequency response to each subcarrier.
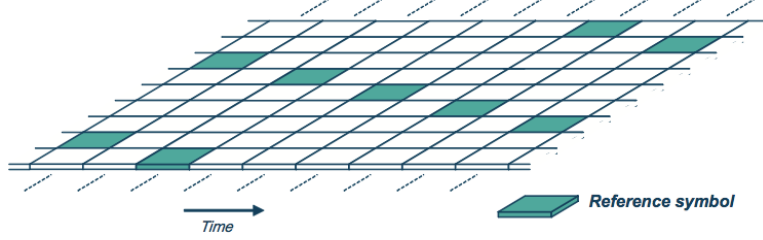
(b) Frequency-time grid containing the pilots. Reference symbols density should be chosen accordingly to the channel properties.

Figure 2.12: The effects of a non-flat channel and a grid containing the OFDM pilot symbols, in time and frequency domains for channel estimation.

In the implementation of an OFDM communication system, the following basic OFDM parameters need to be decided upon:

- The subcarrier spacing $\Delta f = \frac{1}{T_u}$. The OFDM subcarrier spacing should be as small as possible to minimise the relative cyclic-prefix overhead $\frac{T_{CP}+T_{CS}}{T_u+T_{CP}+T_{CS}}$. However, a too small sub-carrier spacing increases the sensitivity of the OFDM transmission to Doppler spread and different kinds of frequency inaccuracies.

- The total number of subcarriers, defined by the size of the DFT, and the number of active subcarriers N_c. This parameter will define the basic bandwidth occupied by the signal:

$$W = N_c \times \Delta f = \frac{N_c}{T_u} = \frac{N_c}{N_{FFT}} \times f_s \qquad (2.24)$$

- The cyclic prefix and suffix length $T_g = T_{CP} + T_{CS}$, which will determine the overall symbol time $T = T_u + T_g$. In principle, the cyclic-prefix length T_{CP}, should cover the maximum length of the time dispersion that is expected in the channel.

This parameters, along with others, will determine the performance of the system and, hence, must be chosen carefully.

On the receiver side the, after the synchronization is complete, the inverse operations are perfomed, like depicted in figure 2.13
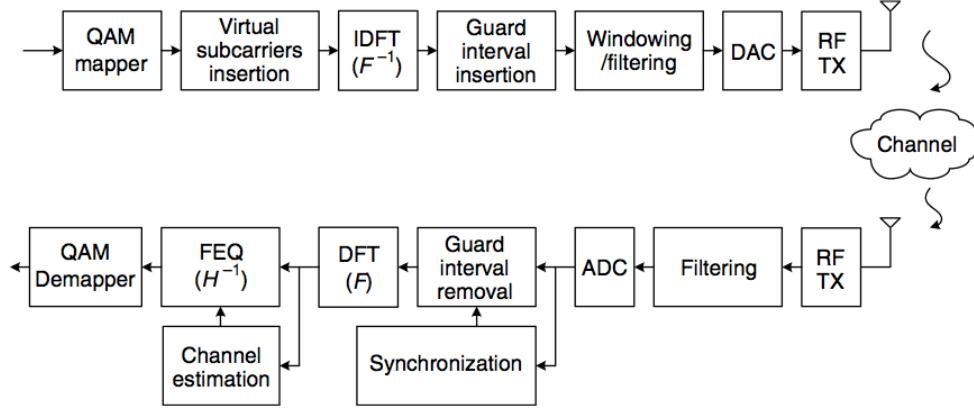
23

Figure 2.13: OFDM schematic for transceiver using a QAM constellation.

2.2 Pulse Shaping

So far, we have considered the case where the modulated signal is shaped by a rectangular function, inside the interval $0 \leq t \leq T_s$. If $g(t)$, the pulse shaping function, is a rectangular pulse of width of T_s, then the envelope of the signal is constant. However, a rectangular pulse has very high spectral sidelobes, which means that signals must use a larger bandwidth to eliminate some of the adjacent channel sidelobe energy. Pulse-shaping is a method to reduce the sidelobe energy relative to a rectangular pulse, however the shaping must be done in such a way that ISI between pulses in the received signals is not introduced (or, at least minimised). Assuming the channel model to be AWGN, the effective receive pulse is given by $p(t) = g(t) * c(t) * g^*(-t) = g(t) * g^*(-t)$, since the channel impulse response is given by $c(t) = \delta(t)$. This pulse must satisfy the Nyquist criterion, which requires the pulse to equal zero at the ideal sampling point associated with past or future symbols [8]:

$$p(kT_s) = \begin{cases} p_0 = p(0), k = 0 \\ 0, k \neq 0 \end{cases} \tag{2.25}$$

Some of the pulses (also called windows) that satisfy this criterion are the rectangular, raised cosine (Hanning) and the root-raised cosine. Each pulse has advantages and disadvantages, but the last two are aimed to improve the spectral efficiency. The most common window in FSK is the Gaussian pulse, defined as:

$$g(t) = \frac{\sqrt{\pi}}{\alpha} e^{-\pi^2 t^2 / \alpha^2} \tag{2.26}$$

The parameter α in the gaussian window is related to the 3dB bandwidth B_z of $g(t)$, and is defined as follows [8]:

$$\alpha = \frac{\sqrt{-\ln \sqrt{0.5}}}{B_z} \tag{2.27}$$

Hence, the spectrum of $g(t)$ is given by:

$$G(f) = e^{-\alpha^2 f^2} \tag{2.28}$$

The different windows characteristics can be seen in figure 2.14a and the effects when they are applied to a simple BFSK system are depicted in figure 2.14b.
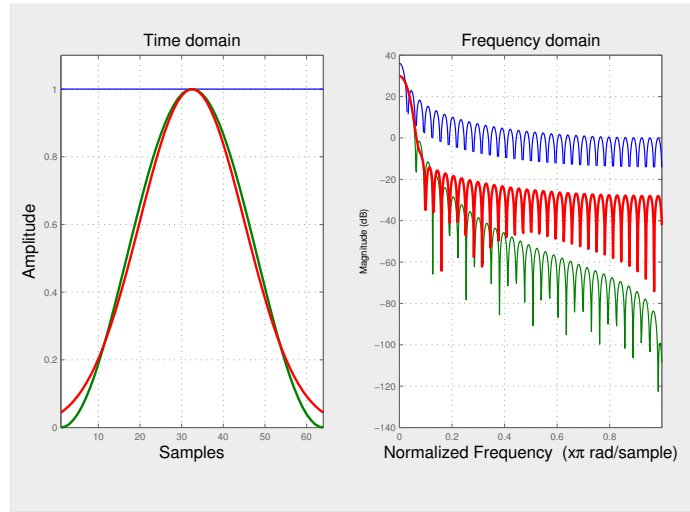
(a) Different windows in time and frequency domain. Blue, red and green lines correspond to the rectangular, Gaussian and Hanning window, respectively.
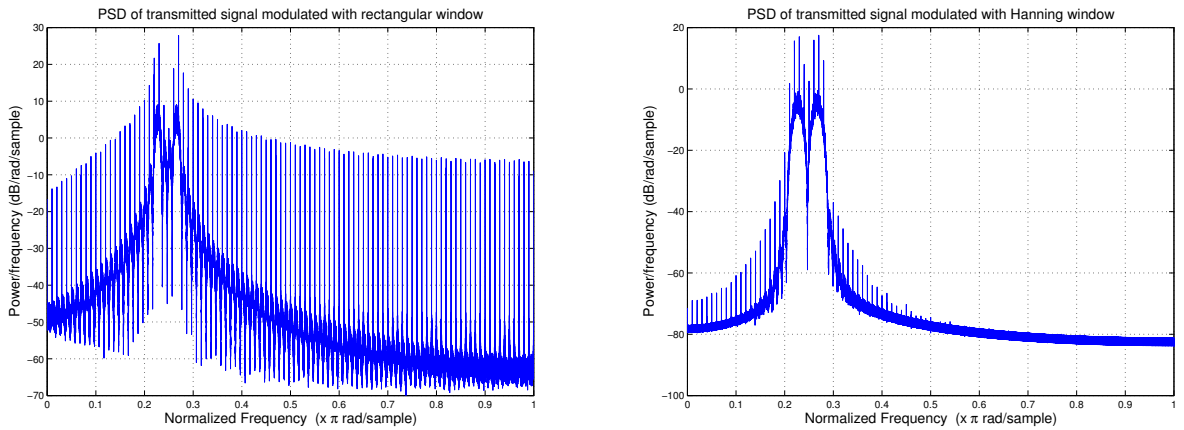
(b) PSD of the transmitted pulse using the rectangular and Hanning window for a BFSK system.

Figure 2.14: The importance of choosing the right window for pulse shaping.

We can see that if we make α large, it will result in a higher spectral efficiency. The Gaussian pulse does not satisfy the Nyquist criterion, and therefore the pulse shape introduces ISI, which increases as α gets larger. Therefore, improving spectral efficiency by increasing α leads to a higher ISI level, creating an irreducible error floor from this self-interference. The variable α should, hence, be chosen with care. The gaussian window effect can be seen in figure 2.15 and will be further developed in section 2.3.
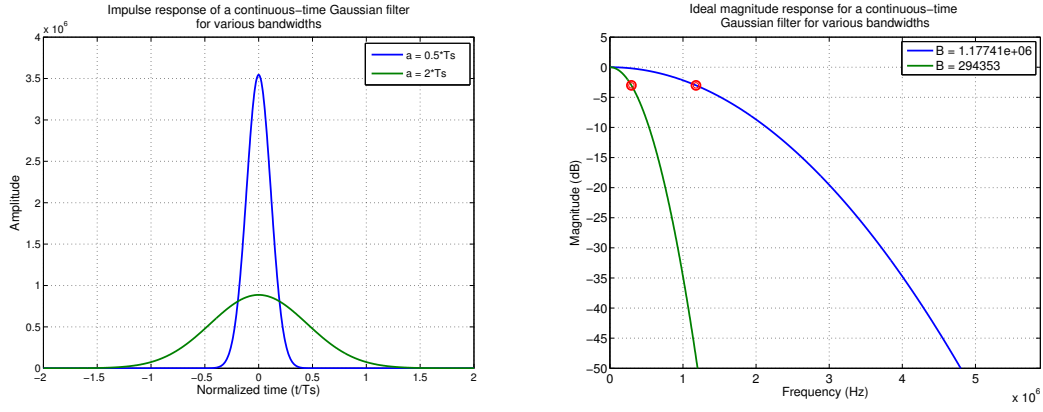
Figure 2.15: Ideal Gaussian filter with different values of α. The tradeoff between high spectral efficiency and reduced ISI is clear. A large α leads to an increased efficiency but the amplitude for $t/T_s > 1$ is not zero, leading to ISI.

2.3 Synchronisation

Carrier synchronisation is an extremely important parameter for the performance of the system. Up to this point we have assumed that the receiver is perfectly synchronised with the transmitter, and the only channel impairment is AWGN. In practice, however, it is often found that there is also uncertainty due to the randomness of some signal parameters, caused by distortion in the transmission medium and hardware constraints. One of the most common random signal parameter is the carrier phase, especially for narrowband signals. A mistiming error in sampling may lead to a series of ISI components that converge and affects the overall system performance.

There are two basic modes of synchronisation in the reception of a digitally modulated signal:

1. **Carrier Synchronisation.** When coherent detection is used, knowledge of both frequency and phase of the carrier are necessary. The process of estimating such parameters is called carrier recovery.

2. **Clock recovery.** To perform demodulation, the receiver has to know the instants of time that at which the modulation in the transmitter changes its state. In other words, the receiver has to know the starting and finishing point of the individual symbols, so that it may determine when to sample. The estimation of these times is called symbol synchronisation.

The synchronisation algorithm is crucial for the operation of the system. Its task is to find the best sampling time for the sampling device. Ideally, the MF should be sampled such that the SNR for the decision variable is maximised. For a rectangular pulse shape, the best sampling time t_{samp} is at the peaks coming out from the matched filter (see figure 2.16).
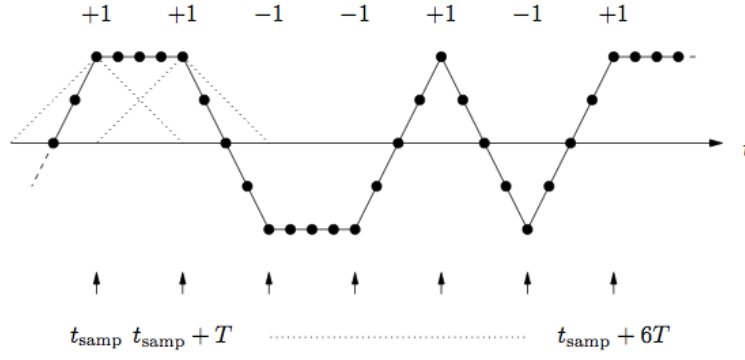
Figure 2.16: Output from the matched filter for successive signalling in absence of noise. The small arrows illustrate the preferred sampling instants[9].

One of the most common synchronisation techniques is based on the usage of training sequences. When the signal is received, the receiver is able to identify the training sequence and recover the symbol time, by cross-correlating the samples after the matched filter with a locally generated time-shifted replica of the training sequence. This algorithm can be either applied directly to the received samples and correlating with a modulated training sequence (resolution of 1 sample but very sensitive), or applied at the symbol level (resolution of T/Q, where Q is the number of samples per symbol). We will consider the last option to simplify. If $\{c(n)\}_{n=0}^{L-1}$ is the locally generated symbol-spaced replica of the training sequence of length L, $[t_{start}, t_{end}]$ represents the search window, and $r(n)$ denotes the output from the matched filter. The timing can be found as [11]:

$$t_{samp} = \arg \max_{t_{samp} \in [t_{start}, t_{end}]} \left| \sum_{k=0}^{L-1} r(kQ + t_{samp})^* c(k) \right| \tag{2.29}$$

The correlation properties of the training sequence are important as they affect the estimation accuracy. Ideally, the autocorrelation function for the training sequence should be equal to a delta pulse,which means it should have a zero correlation everywhere except at lag zero. Therefore, the training sequence should be carefully designed. If, for example, the training sequence is modulated with a different frequency or pulse shape than the rest of the signal, better results for the correlation will be obtained. Such effects are depicted in figure 2.17.
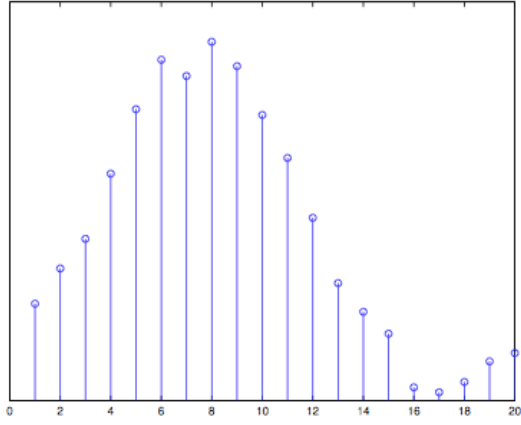
(a) Symbol spaced cross-correlation. In this example the delay is estimated to be 8 and therefore the system should be sampled at 8+kQ, k=0,1,2,..N.
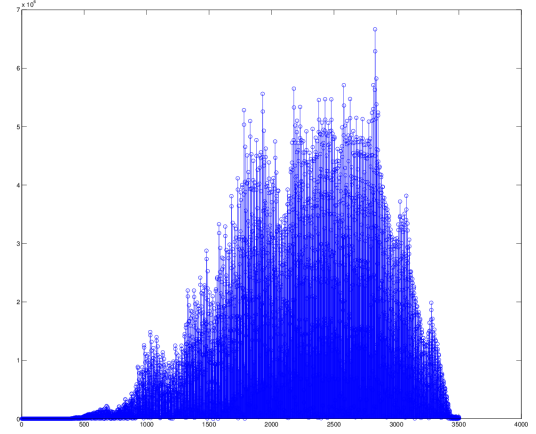
(b) One sample resolution cross correlation with training sequence of length 20 symbols.
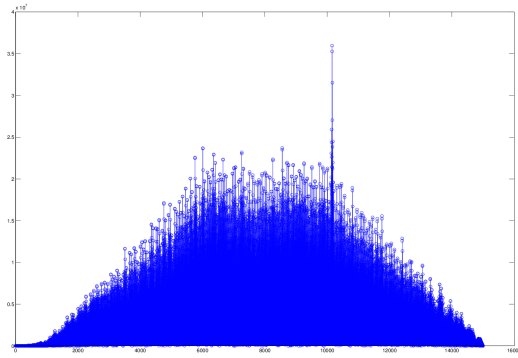
(c) One sample resolution cross correlation with training sequence of length 100 symbols and equal frequency to rest of transmission.
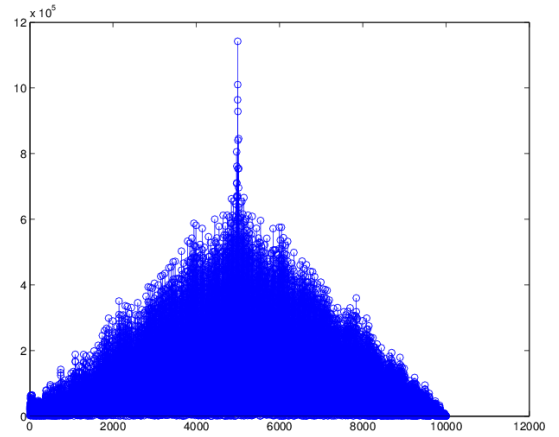
(d) One sample resolution cross correlation with training sequence of length 100 symbols and different frequency to rest of transmission.

Figure 2.17: Training sequence algorithm/length/modulation comparison.

2.3.1 Synchronisation in MQAM systems

The basic operations required by an all digital MQAM receiver are illustrated in figure 2.18. The quadrature down-conversion and matched filter operations produce estimates of the quadrature amplitudes that are the basis of the data decisions. The role of carrier phase synchronisation is to perform the quadrature down-conversion using phase coherent replicas of the quadrature carriers.
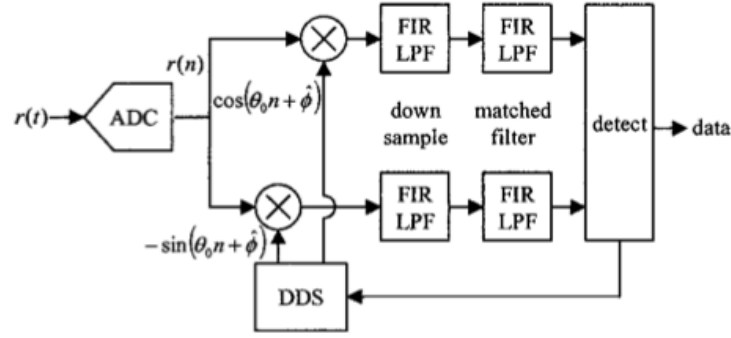
Figure 2.18: Block diagram of basic QPSK/QAM digital receiver. In the digital implementation, the VCO takes the form of a direct digital synthesizer (DDS).

There are many options for implementing carrier phase and frequency synchronisation in a digital communication system. At the heart of all synchronisers is the phase-locked loop (PLL). An all-digital receiver can be implemented with a digital phase-locked loop (DPLL) as shown in figure 2.19a. The phase detector is implemented using the arctangent suggested by the ATAN block. The complexity of the phase detector can be reduced by computing a signal proportional to the sine of the phase difference. The phase error is computed by comparing the phase difference between the received signal $x(n) + jy(n)$ and the closest constellation point $\hat{I}(n) + j\hat{Q}(n)$ as illustrated in figure 2.19b.
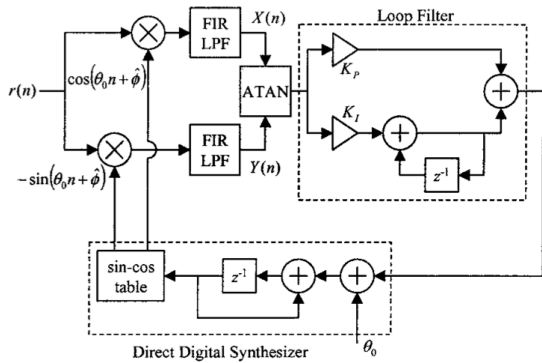
(a) Digital phase locked loop for carrier phase synchronisation.
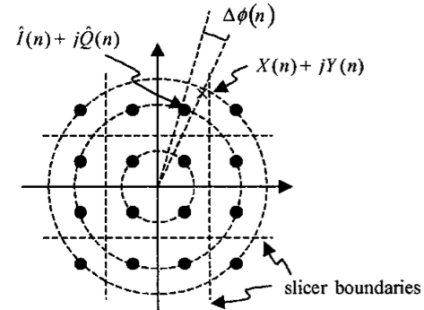
(b) 16-QAM constellation with decision boundaries and phase error.

Figure 2.19: Carrier phase estimation diagrams.

The initial phase estimation can be done using the procedure described above, taking into account the training sequence is known at transmitter and receiver. Let's assume the channel causes a rotation of φ to the symbol constellation. The value of φ can be computed using the element-wise multiplication of the received symbol with the complex conjugate of a modulated training sequence replica and then averaging over the sequence. In other words, if $\{\tilde{r}(n)\}_{n=0}^{L-1}$ denotes the L received symbols and $\{c(n)\}_{n=0}^{L-1}$ the local replica of the complex training sequence, an estimate of the phase offset can be

obtained as [9]:

$$\hat{\varphi} = \frac{1}{L} \sum_{k=0}^{L-1} \arg(\tilde{r}(k)c(k)^*) \tag{2.30}$$

The longer the training sequence, the better the phase estimate as the influence from the noise decreases. On the other hand, a long training sequence, reduces the amount of payload that can be transmitted during a given time.

2.3.2 Phase rotation

Another problem that may be faced in a real implementation is the Doppler shift introduced during the transmission, which causes the constellation to rotate further more (than the initial rotation). This frequency offset could be the result of relative motion between the transmitter and receiver, as would be the case with a mobile terminal travelling away or towards a receiver. It could also be attributed to small frequency variations of the various synthesisers in both the transmitter and receiver, which are functions of time and temperature. In order to overcome this problem, at least two approaches are possible. The first is to introduce a training sequence in the data so that a new phase estimation can be computed, resulting in a lower rate. The other solution, is to divide the decisions into various small blocks in which the offset does not influence the decision, estimate the phase rotation from the nearest points and rotate the constellation for future decisions. In this way, the rate is not affected but a correct decision in each block is assumed. The frequency offset effect can be seen in figure 2.20, where a 16-QAM signal of 5 seconds is transmitted.

(a) Transmitted constellation.

(b) Received constellation.

(c) Received constellation after phase correction.
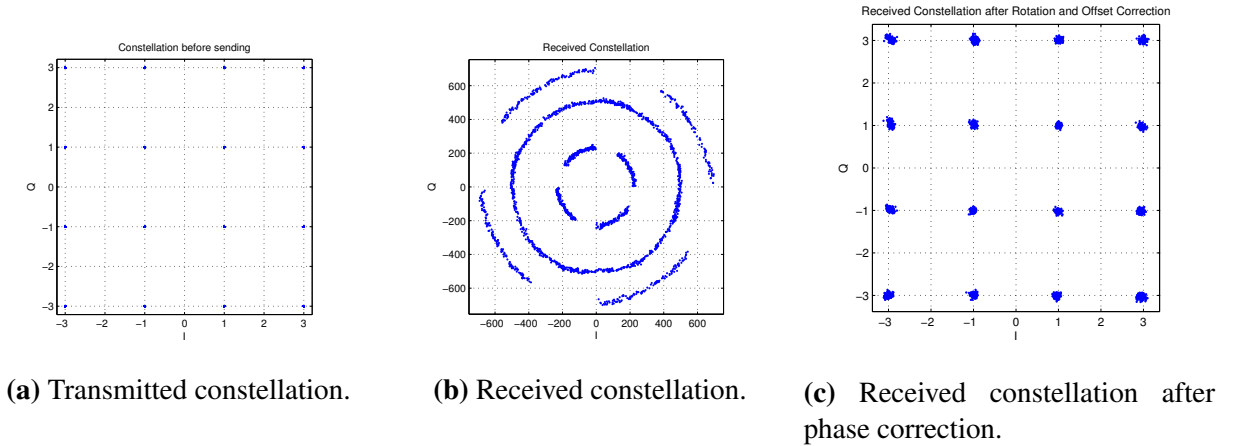
Figure 2.20: Example of frequency offset in a 16-QAM constellation with decision boundaries.

2.4 Channel Equalisation

The performance of a digital system is constrained by the type of channel in which the implementation is done. For our case, in theory, we have a nearly-ideal low-pass channel with low noise variance and limited up to about 20 kHz. However, the phone's audio card introduce non-linear distortion that affects the signal output and causes attenuation around 6kHz (see figure 2.21). This can

both cause ISI as a result of the channel impulse response, as well as distortion of the signal spectrum. Thus, to improve the performance of our system, we can use channel equalisation techniques to reduce the ISI, and allow a better recovery of the transmitted symbols. The equalisation method will depended on the type of modulation scheme, as well as the channel response. For the case of FSK, the amplitude distortion will not affect at a great extent, but for M-QAM we shall compensate such distortion without affecting the carrier phase. In addition, we can expect a non-varying channel that allows the usage of parametric digital filters.
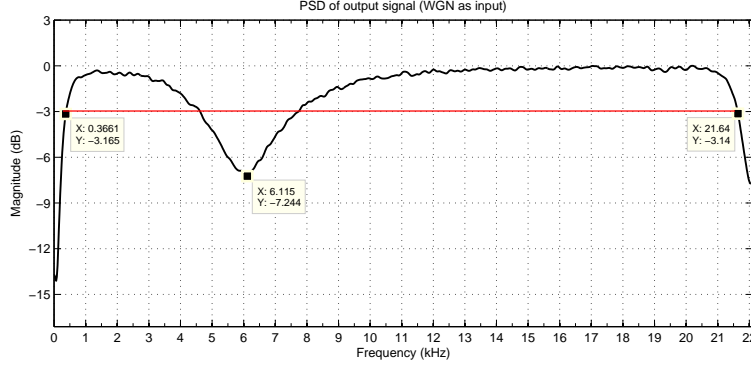
Figure 2.21: Estimated channel frequency response to WGN. The measured 3dB bandwidth is 21.27kHz.

2.4.1   IIR filters

Infinite Impulse Response (IIR) filters are recursive structures of a discrete LTI system that can be defined generically in the following form [16]:

$$H(z) = \frac{\sum\limits_{\ell=0}^{M} b_\ell z^{-\ell}}{1 + \sum\limits_{\ell=1}^{N} a_\ell z^{-\ell}}. \tag{2.31}$$

The rational transfer function described in equation 2.31 implies that the system will have an infinite response, when feeding it with a Dirac impulse. The coefficients from the numerator polynomial correspond to the feed-forward terms, and the ones from the denominator, correspond to the feedback terms (see figure 2.22). IIR filters can be represented in several forms or realisations, however, we generally have to base our design in a given set of initial parameters, such as pass-band, stop-band, ripple, or roll-off. Amongst the main advantages of these kind of filters, we have that they can efficiently meet a given specification with relative low complexity and efficiency. In general, this computational efficiency is rather large compared to FIR filters, described in the next section.
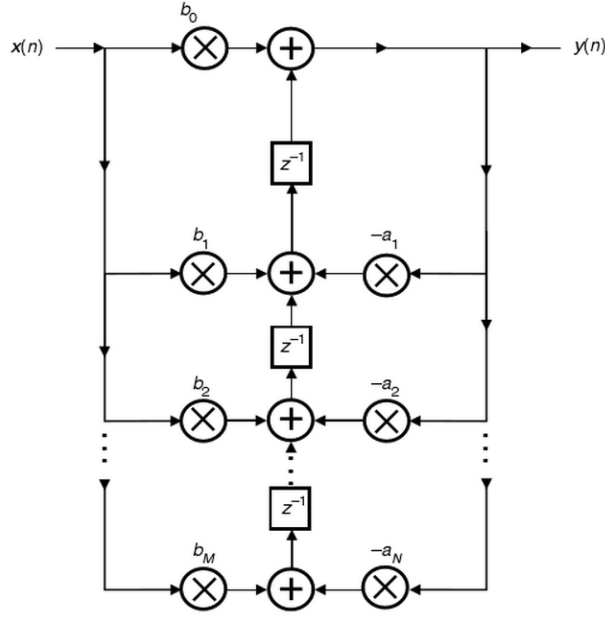
Figure 2.22: Shift register implementation of an IIR filter (Type 2 canonic direct form)[16].

2.4.2 FIR filters

Finite Impulse Response (FIR) filters are non-recursive linear systems that are also widely used in signal processing because of their stability and relative simple implementation. Their transfer function only contains feed-forward terms with coefficients b_ℓ directly related with the filter impulse response as follows[16]:

$$H(z) = \sum_{\ell=0}^{M} b_\ell z^{-\ell} = \sum_{\ell=0}^{M} h_\ell z^{-\ell} \tag{2.32}$$

As seen in equation 2.32, these kind of filters can be easily designed when we compute the coefficients that characterise the channel impulse response. FIR filters may also have good characteristics compared with the case of IIR filters, such as linear phase and stability. The biggest problem that we may face when using such designs, is that we need to have a relative high order compared to the case of a similar IIR filter. This adds additional delay to the signal processing and slows down the system. The shift register implementation of a typical FIR filter is depicted in figure 2.23.
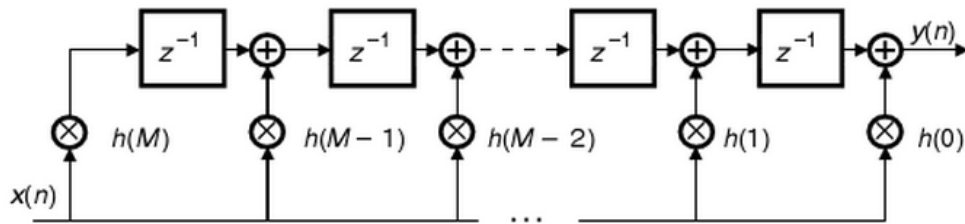
Figure 2.23: Shift register implementation for a FIR filter (Direct form)[16].

2.5   Channel Coding

The digital signal is subject to diverse perturbations that provoke distortion throughout a non-ideal channel. It is also likely that such distortion cannot be fully compensated with an Equaliser or filter, thus, error correcting codes may be implemented. Our channel is bandwidth and power limited, these two parameters together with the noise spectral density will determine the bit-error rate (BER)[11]. In our system, the channel impairments result in frequency distortion and ISI. An orthogonal modulation scheme may be able to mitigate such perturbations, however, it is still likely to have errors at the receiver that can be corrected by adding redundancy to the transmitted signal. This implies that the overall information rate will be decreased when using overheads such as training sequences or error correcting codes.

There are two basic schemes to implement error correction capabilities into our system: error detection/retransmission, and forward error correction (FEC)[17]. In the first scheme, the receiver terminal asks the transmitter to retransmit the data if it detects an error. When FEC is used, the receiver tries to recover the signal using the added redundancy. Hence, the detection and retransmission scheme requires a two-way communication contrary to the case of using forward error correction. We will focus on the last scheme since our system is only able to send data in one direction. In this fashion, the information message shall be encoded before it is modulated and sent trough the channel (figure 2.24a). The reverse process is done at the receiver, as shown in figure 2.24b. The most widely used codes are linear codes, which can be classified into two general forms: block codes and convolutional codes. One of the principal differences between the two groups is that the encoder for the case of convolutional codes has memory elements.

(a) Transmitter with channel encoder.

(b) Receiver with channel decoder.

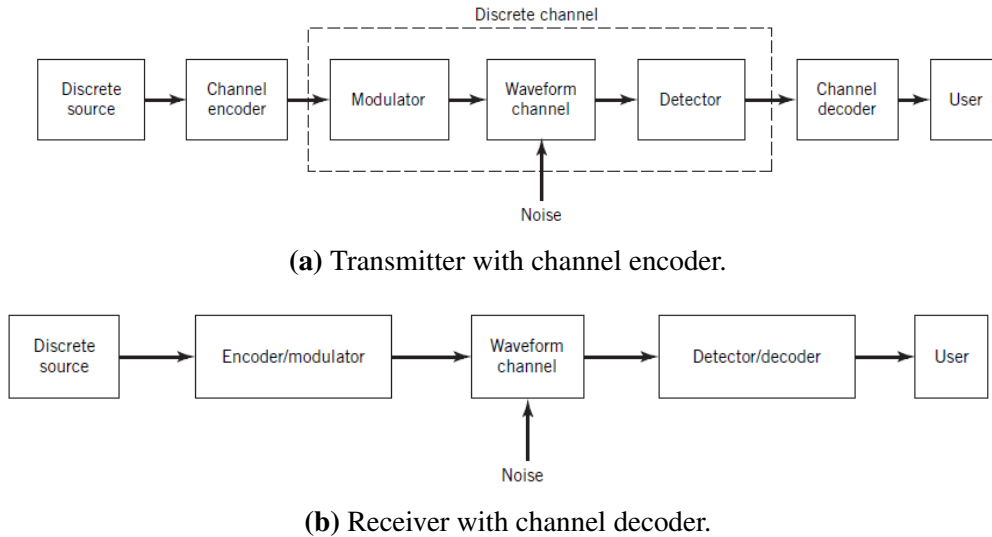**Figure 2.24:** Digital communication system with forward error correction[11].

2.5.1   Block Codes

In general, binary block codes are parity check codes that can be characterised by the parameters (n, k). In this notation, n represents the number of encoded bits, and k is the number of information bits. We can define the ratio k/n as the rate of the code. The mapping from the set of $M = 2^k$

information sequences of length k can be represented by a $k \times n$ generator matrix, conventionally denoted as $\mathbf{G}$. Therefore, we can write the encoded sequence $\mathbf{c}$ (of length n) in matrix form as follows:

$$\mathbf{c} = \mathbf{uG} \tag{2.33}$$

The vector $\mathbf{u}$ denotes the information sequence (of length k), and the row vectors of $\mathbf{G}$ determine the basis of a non-unique vector space. When $\mathbf{G}$ has the form $\mathbf{G} = [\mathbf{I}_k | \mathbf{P}]$, we could find a parity matrix $\mathbf{H} = \left[\mathbf{I}_{n-k} | \mathbf{P}^T \right]$ that would lead us to decode the original data[17]. These codes are mainly used for hard-decision decoding with a built-in algebraic structure based in the properties of finite fields. There exist a variety of block codes termed as Long-Density Parity-Check Codes (LDPC) that can achieve performance very close to the channel capacity.

2.5.2 Convolutional Codes

Convolutional codes are linear codes which are generated by passing the information sequence through a shift register. They can be described by some encoding function $G(\mathbf{u})$, such that $\mathbf{c} = G(\mathbf{u})$. The encoder outputs at any given time will only depend on the inputs. The number of output bits for each k-bit input is a sequence of length n. We also need to define the parameter K as the code constraint length, which represents the number of stages (of k-bits) in the encoder.

One possible method to describe a convolutional code is by giving its generator matrix. Nevertheless, such matrix would be semi-infinite since the input sequence can have any arbitrary length[2]. A different method to describe a convolutional code, comes from the fact that each of the output sequences, can be interpreted as the convolution of the input sequence with a binary filter, defined by some "impulse response" or, generator polynomial[1], denoting the links between the input, output and shift registers. A typical shift register implementation of such codes is shown in figure 2.25.
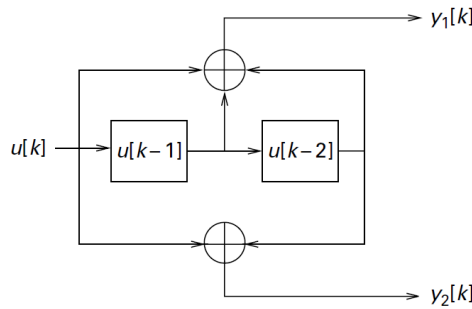
Figure 2.25: Shift register implementation of convolutional encoder with generator polynomials $[7, 5]$ [1].

Convolutional codes may be used as a building block for turbo-codes, which can perform very close to the channel capacity, as well as LDPC codes do. The major disadvantages for those two approaches are the relative complexity and hard processing that are required to achieve a good performance.

---

[1]Since we are using binary arithmetic operations, the polinomyals are typically given in octal form.

Chapter 3

Android Prototype

Our aim is to build a full digital communication system that uses a pair of Android phones as terminals and communicate them with an audio cable that will act as the channel together with the sound card of the device. The implementation consists in two modes: off-line and real-time. We define off-line and real-time as follows. Assume that the file size is F (bits) and the data-rate C (bps). The transmitter is allowed to transmit signals during F/C seconds. In off-line mode the receiver should be able to decode the signal within 100 seconds using the received signal downloaded from the phone and process it with MATLAB. In real-time the receiver should finish decoding within $0.5 \times F/C$ seconds after the transmitter has finished sending the data. The file should have size between 100 and 1000 bytes [10].

The advanced specifications requires a data-rate of 32 kbps in real-time and 128kbps in off-line mode. In real-time the system should be able to transmit file sizes up to at least 10kB.

3.1 Software and Hardware

Used Hardware

- Samsung Galaxy S3 (GT-i9300) phones with Android 4.2.1.

- 3.5mm audio cable [1].

- Dell PC's with Windows 7.

Used Software

- MATLAB R2013b

- Android Developer Tools IDE (Eclipse) with API for Android 4.1.2.

- Git source code manager software for Windows.

---

[1]The audio cable is modified so as to have the audio input in one end linked to the audio output in the other extreme.

3.2 Android FrameWork

Framework is an Android application developed by Martin Ohlsson and Per Zetterberg which has been designed as the groundwork for our project and to support our software implementation in real-time. The JAVA based implementation contains functions that enable the control and initiation of various sensors of the android phone (e.g. microphone, camera, speakers, among others.). In addition it has an embedded Guided User Interface (GUI) and a number of valuable built-in features allowing us to focus more in the communication system implementation instead of software designing.

The file StudentCode.java is included in the FrameWork which is the main file modified and worked in. [Some of the things that were added to StudentCode to make it work].

Figure 3.1 illustrates a diagram of the complete FrameWork and the connections between the main files. In the boxes some of the main function names that belong to that java file are shown. For example the file Complex.java [19] allows us to use and handle complex numbers in the program.
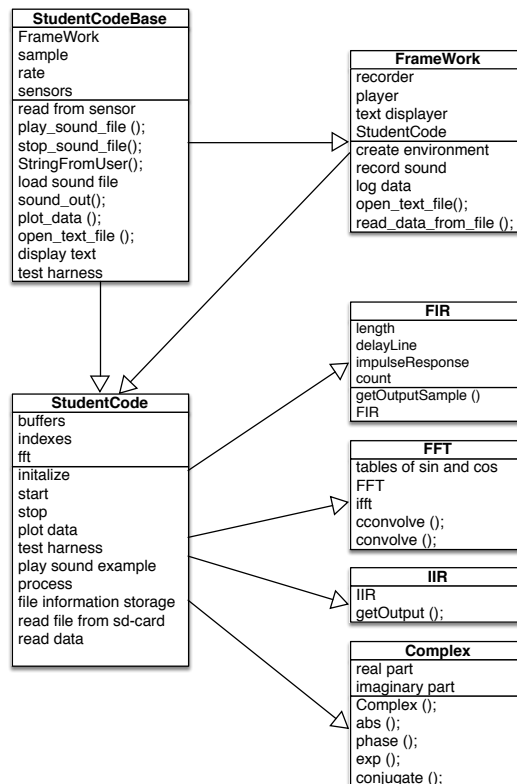
Figure 3.1: Description of FrameWork.

3.3 App implementation

Figure 3.2 illustrates the schematic of our developed file transfer system. The user has the option to choose a file to be transmitted. After the file is selected the phone pulls the file from the SD-Card and encodes it into binary form. This binary stream is then modulated and sent through the channel. When the transmission is done the transmitting phone changes to an idle state while the receiving phone listens to the process and starts the demodulation after receiving the data. The file is reconstructed and stored on the receiving phone's SD-Card. The user now has the option to open the received file, otherwise the state of the receiving phone changes to idle.
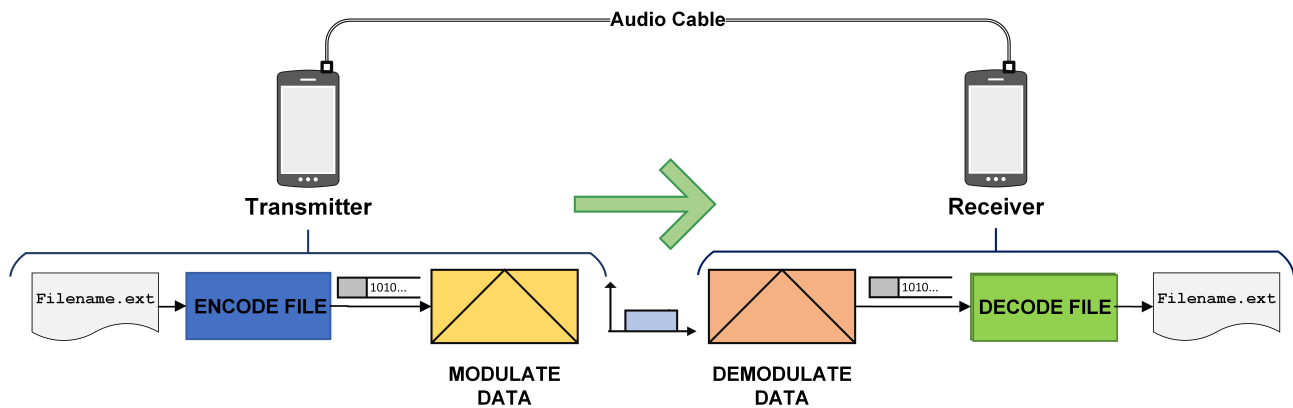
Figure 3.2: Schematic of the file transfer system.

3.3.1 Transmitter

In figure **??** a flow diagram of the transmitter functionalities are illustrated. The user chooses a file which is opened and read as an array of bytes. These bytes are then converted to a binary stream of ones and zeros, which are modulated with a given scheme along with a guard band and training sequence. The bits are then, modulated with a given scheme along with the guard-band and the training sequence. The total modulated signal (information data, guardband, training sequence and headers) is sent to the audio buffert, that plays the sound through the channel. The headers include the name, size and extension of the file along with a checksum value.
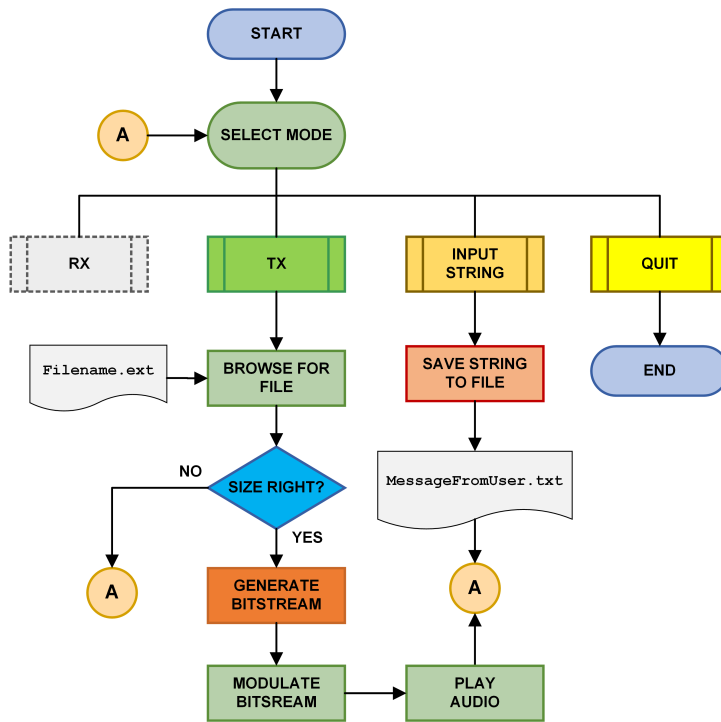
(a) Full transmitter system. (b) Generate Bit-stream diagram.
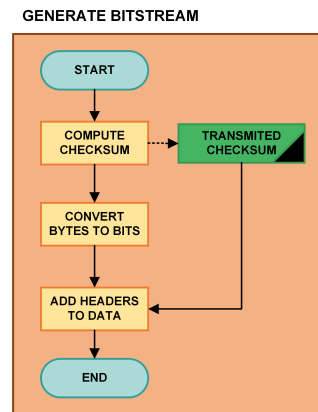
Figure 3.3: Transmitter client flow diagram.

3.3.2 Checksum

The checksum used in our application is the internet checksum which is widely used in standard internet protocols. It takes a vector of bytes as input and performs the following operations:

- Performs 1's complement sum of adjacent octets by combining them to one 16-bit integer

- Perform one last 1's complement on final checksum value . . .

The checksum value is computed for our information data and is then sent through the channel. This value is retreived at the receiving phone, in order to make sure that the file is not corrupt during the transmission.

3.3.3 Receiver

In figure **??** a flow diagram of the receiver functionalities is illustrated, along with a diamgram of how the received file is reconstructed. The application starts listening if it varifies that there are samples above the detection threshold. If samples are below the threshold it is only noise otherwise the application starts buffering the input sound until the transmission is completed or the buffer is full. If the buffer is full, a new one is created. The received data is then processed and synchronized by correlating with a training sequence. Then the application applies the tone detection algorithm retrieving the binary stream of ones and zeros. The stream is processed and the name, size, data and checksum value of the selected file are retrived. The application uses this information to reconstruct the original file before saving it to the phones SD-Card. The user now has the option to either open the received file or return to the idle state.
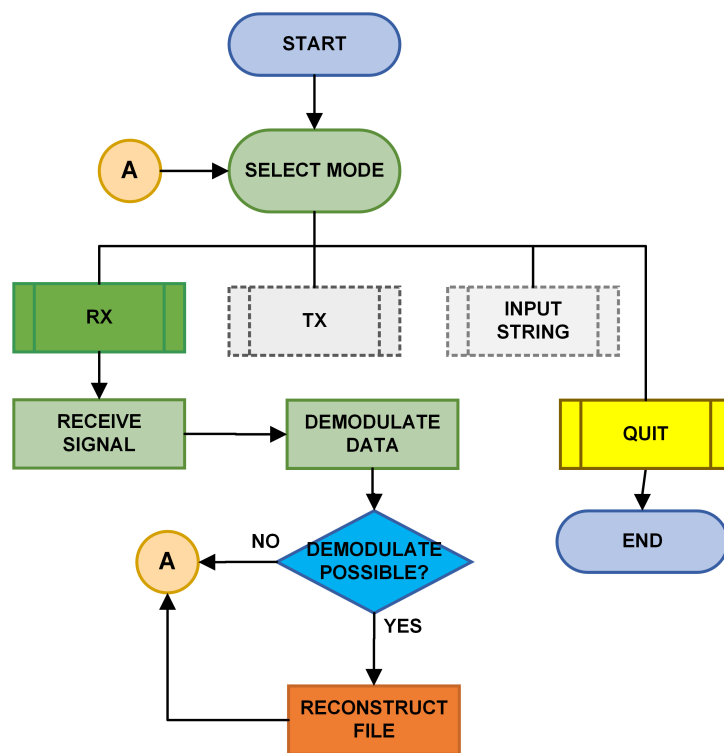
(a) Full receiver system.
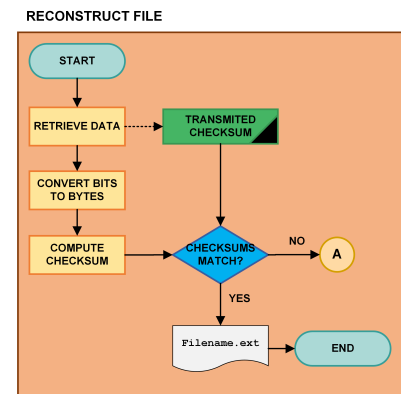
(b) Reconstruct file algorithm.

Figure 3.4: Receiver client flow diagram.

Chapter 4

Method

In chapter 2, the fundamentals of each modulation scheme were introduced. Now we will focus on developing the details and algorithms of our implementation in what refers to modulation, demodulation, channel adaptation and coding techniques.

4.1 M-FSK system

Our first design consists in a system based on FSK modulation with off-line and real-time implementations. The used system is a M-ary FSK system (2-FSK for syncronization and 4-FSK for data modulation) with orthogonal envelopes to improve the BW efficiency of the MFSK modulation.

4.1.1 M-FSK transmiter

With a 4 FSK system, it is possible to transmit $\log_2(4) = 2$ bits per symbol. It is then necessary to compute the frequencies that will accommodate the transmitted symbols. Denote n_{sym} as the number of samples per symbol. In order to increase the efficiency for our decoding scheme, the frequencies should be chosen such that:

$$f_k = f_s \times \frac{k}{n_{sym}} \tag{4.1}$$

These are the bins-center frequencies and they should have a sufficiently large minimum distance between them so that the peaks do not become indistinguishable. This requirement is of particular importance since the demodulation method is based in a DFT-based algorithm that will be described in section 4.1.2. Once the frequencies are picked, the transmitter merges the bits to be transmitted with the ones from the training sequence and guard bands and modulates them accordingly. The samples that correspond to the *n-th* symbol are given by:

$$s(k) = \cos\left(2\pi \frac{f_i}{f_s} k\right), \ k = n \times n_{sym} + 1, ..., n \times n_{sym} + n_{sym} \tag{4.2}$$

Where f_i is the frequency that matches with the symbol that is to be transmitted.

4.1.2 M-FSK receiver

In the receiver side, the first step is to synchronise and find the beginning of the training sequence. When the receiver knows where the transmission starts, the samples are then input for the decoding algorithm.

The demodulators presented in part 2.1.3 are interesting from a theory point of view. However, for real time, software-based applications there are other methods for efficient demodulation. One of these methods is the Discrete Fourier Transform (DFT) which evaluates the amplitude and phase for each frequency present in the received signal. Since we are using a M-FSK modulation scheme, a tone/frequency detector such as the DFT for a symbol period is a powerful tool for a correct and efficient demodulation. However, we are only interested in detecting one or more tones in an audio signal and at the same time we are constrained to the CPU horsepower provided by the smartphone. Taking this into account, we purpose a much faster method, when compared with FFT for instance, which is the Goertzel algorithm.

Goertzel Algorithm

There are various ways to detect the presence of a special known frequency in a received signal. The DFT algorithm is a simplistic way to check whether the desired frequency is present and a FFT produces the same numerical result for a single frequency of interest, making it a better choice for tone detection. The Goertzel algorithm is a DFT in disguise, with some numerical tricks to eliminate complex number arithmetic, increasing the efficiency over the FFT under some constraints.

The idea is to transform an ordinary N samples DFT into a Goertzel filter form. Defining $W_N^k = e^{-j2\pi k/N}$, and noting that $W_N^{-Nk} = 1$, we have for the DFT [8]:

$$X(k) = \sum_{n=0}^{N-1} x(n)W_N^{nk} = \sum_{n=0}^{N-1} x(n)W_N^{-Nk} \cdot W_N^{nk} = \sum_{n=0}^{N-1} x(n) \cdot W_N^{-(N-n)k} \qquad (4.3)$$

An efficient way to evaluate this polynomial is the nested form:

$$X(k) = \left\{ \left[\left(W_N^{-k}x(0) + x(1) \right) W_N^{-k} + x(2) \right] W_N^{-k} + \cdots + x(N-1) \right\} W_N^{-k} \qquad (4.4)$$

The last expression can be written in terms of a recursive difference equation:

$$y(n) = W_N^{-k} \cdot y(n-1) + x(n) \qquad (4.5)$$

Expressing the difference equation in the z-transform domain and by multiplying both numerator and denominator by $(1 - W_N^{-k}z^{-1})$, we get the transfer function:

$$\frac{Y(z)}{X(z)} = H(z) = \frac{1}{1 - W_N^{-k}z^{-1}} = \frac{1 - W_N^{-k}z^{-1}}{1 - (2\cos(\frac{2\pi k}{N})z^{-1} - z^{-2})} \qquad (4.6)$$

The Goertzel algorithm acts as an IIR filter that uses the feedback path to generate a very high Q

bandpass filter, where the coefficients are easily generated from the required center frequency. It is mostly implemented as the second order recursive IIR filter, as shown in figure 4.1.
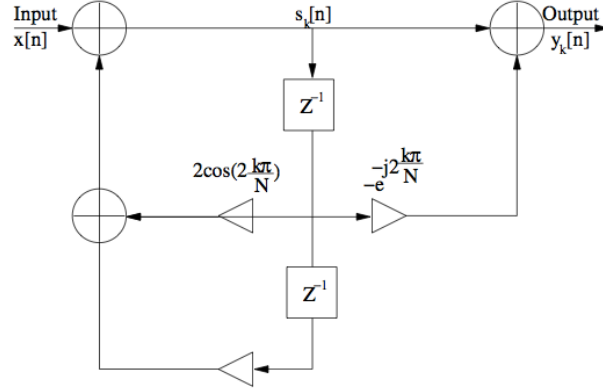

Figure 4.1: Direct-Form Realisation of the Goertzel Algorithm [11].

The algorithm computes the k-th DFT coefficient $X(k) = y(N)$ of the input signal $x(n)$ using the 2nd order filter:

$$
\begin{aligned}
y_k(n) &= s_k(n) - W_N^k s_k(n-1), \\
\text{with} \quad & s_k(n) = x(n) + 2\cos(\tfrac{2k\pi}{N})s_k(n-1) - s_k(n-2), \\
\text{and} \quad & s_k(-1) = s_k(-2) = 0
\end{aligned} \tag{4.7}
$$

The discrete frequencies in which the algorithm is able to compute the DFT are $f_k = \frac{k}{N}f_s$, for $f \le \frac{f_s}{2}$.

Computational and Memory complexities

The FFT algorithm used with N being a power of two has computational demands proportional to $O(N\log_2 N)$. The absolute number depends on the particular implementation. Usually, the number of real-number arithmetic operations found in the literature is approximately $6N\log_2 N$. If we analyse the number of operations of the Goertzel algorithm, we realize that for a real input signal, N real multiplications and $2N$ real additions are performed and, hence, $3N$ operations for a single frequency are required. Ignoring the computations of the cosine and the exponential constants in equation 4.7, and for N frequencies, we have that the Goertzel algorithm has quadratic complexity like the DFT. To clarify for how many frequencies (K), it is perfered to use the Goertzel algorithm insted of FFT [8], one can compare the complexity of both and conclude that:

$$
3NK < 6N\log_2 N \Leftrightarrow K < 2\log_2 N \tag{4.8}
$$

Such result holds only for N being a power of two; otherwise the inequality 4.8 can even be more favourable for the Goertzel algorithm.

Considering real inputs solely, the FFT algorithm requires a memory space of at least $2N$ samples. Also, the n values of the transformation kernel, *sin* and *cos*, are often precomputed and stored. The

FFT calculation itself can be performed with no values being moved in memory, but as the computation cannot be done until the last sample of a block of data is received, a buffer size of at least N samples must be used. Therefore, the overall FFT memory demand is $4N$ for real signals. For each considered frequency, the Goertzel algorithm requires: location for saving one real and one complex state variable, the complex final output and the last 2 real outputs $s_k(n)$ and $s_k(n-1)$. There is no need to implement input buffering, because the computation can be run as the new signal samples arrive. The total memory complexity of the Goertzel algorithm is, thus $7K$ positions. Combining all the above together, the Goertzel algorithm will be less memory-demanding than the FFT if [8]

$$7K < 4N \Leftrightarrow K < \frac{4}{7}N \tag{4.9}$$

As long as $N \geq 13$, equation 4.8 is decisive for choosing the best algorithm, since for these N it holds that $\frac{4}{7}N \leq 2 \log_2 N$.

Pseudo-code for decoding algorithm

The pseudo-code of the implemented algorithm for the index k of the DFT spectral component is as stated in algorithm 1:

input : index k of the DFT spectral component; signal x of length n_{sym}

output: y, representing $X[k]$

% Precalculation of constants;

$A = 2\pi \frac{k}{N}$;

$B = 2\cos A$;

$C = \exp(-jA)$;

% State variables;

$s_0 = 0$;

$s_1 = 0$;

$s_2 = 0$;

% Main loop;

for $i = 0 : N - 1$ **do**

   % N multiplications, 2N additions;

   $s_0 = x[i] + B \times s_1 - s_2$;

   $s_2 = s_1$;

   $s_1 = s_0$;

end

% Finalizing calculations;

$s_0 = B \times s_1 - s_2$; *% 1 multiplication and 1 addition*;

$y = s_0 - s_1 \times C$; *% 4 multiplications and 3 additions*;

Algorithm 1: Goertzel Algorithm to estimate the k:th spectral component of signal x of length N.

After demodulation, the decisions are sent to the decoder and the BER is computed. Denoting n_{sym} as the number of samples per period, the rate for a 4-FSK system is given by:

$$R = \frac{log_2(4)}{n_{sym}} \times f_s = \frac{2}{n_{sym}} \times f_s \tag{4.10}$$

ADD: A general description of the used transmitter, pseudo-code, and major functions. IF NEC-ESSARY

4.2 M-QAM system

In order to meet the data throughput requirement, we considered to use a configurable M-QAM scheme to achieve the highest possible rate without compromising the performance and still with zero or a negligible amount of errors that could be corrected. In the case of the real-time implementation, we use a checksum comprobation to confirm the data integrity instead of FEC, however for the case of off-line test, we tried with different coding schemes. The results of such test are described in section 4.4.

4.2.1 M-QAM transmitter

There are a considerable set of parameters in the transmitter that will determine the performance of the system. We enlist the most important next:

N_b ; number of bits to be transmitted.

f_s ; sampling rate. The maximum value accepted by the phones is 44.1 kHz.

Window ; discrete window coefficients. Each window has different parameters which will influence the impulse and frequency responses (such as the roll-off factor in the RRC or the α in Gaussian window).

n_{sym} ; number of samples per symbol.

levels ; order of the constellation.

f_c ; carrier frequency.

The sampling frequency, the number of samples per period and the order of the constellation will determine the rate of the system according to:

$$R = \frac{2 \times levels}{n_{sym}} \times f_s \tag{4.11}$$

After these parameters are set, the algorithm comes into action. The first step is to merge the bits in the following order:

[guard_band | training seq. | bit_stream | guard_band]

Subsequently, the bits are mapped into the constellation (MQAM) according to the pseudo-code in algorithm 2.

input : bit stream; %data to be transmitted

output: z; %Complex modulated symbols $z = x + jy$

%Initialization of vectors;

mx=[];

my=[];

for $n = firstsymbol$ **to** $lastsymbol$ **do**

  %Initialize values of x_i and y_i;

  x_i=0;

  y_i=0;

  %Pick 2 bits of the symbol at each time;

  for $m = firstbit : 2 : lastbit$ **do**

    if *bit stream(n+m)==0* **then**

      $xi = xi + 2^{\frac{m-1}{2}}$;

    else

      $xi = xi - 2^{\frac{m-1}{2}}$

    end

    if *bit stream(n+m+1)==0* **then**

      $yi = yi + 2^{\frac{m-1}{2}}$;

    else

      $yi = yi - 2^{\frac{m-1}{2}}$

    end

  end

  mx=[mx xi];

  my=[my yi];

end

z = mx + $j$my

Algorithm 2: Modulation of MQAM symbols.

The discrete carriers to modulate the signal are then generated and have n_{sym} samples:

$$\begin{cases} \cos(2\pi f_c k) \\ -sin(2\pi f_c k) \end{cases} \text{, with } k = 0, \frac{1}{f_s}, \cdots, \frac{n_{sym}}{f_s} - \frac{1}{f_s} \tag{4.12}$$

The window is also generated in the initialization of the program:

$$W_{n_{sym}}(i) = w_i \text{ , with } i = 1, 2, \cdots, n_{sym} \tag{4.13}$$

The coefficients w_i in 4.13 are computed according to the previously chosen window. Then, the complex symbols, the window and the carriers are mixed together to generate the discrete transmitted

waveform. The generated samples for the *n-th* symbol have the following expression:

$$s(k) = \Re(z) \cdot W_{n_{sym}}(k\%n_{sym}) \cdot \cos(2\pi\frac{f_c}{f_s}k) - \Im(z) \cdot W_{n_{sym}}(k\%n_{sym}) \cdot \sin(2\pi\frac{f_c}{f_s}k)$$
$$\text{with } k = n \cdot n_{sym} + 1, \cdots, n \cdot n_{sym} + n_{sym} \text{ ; and with \% as the modulo operator.}$$

(4.14)

The signal $s(k)$ is then normalized to be in the range of $[-1, 1]$, and sent to the function to save it in short format.

4.2.2 M-QAM receiver

On the receiver side, the sound samples are extracted and processed to recover the sent data. A general schematic of the receiver can be seen in figure 4.2.
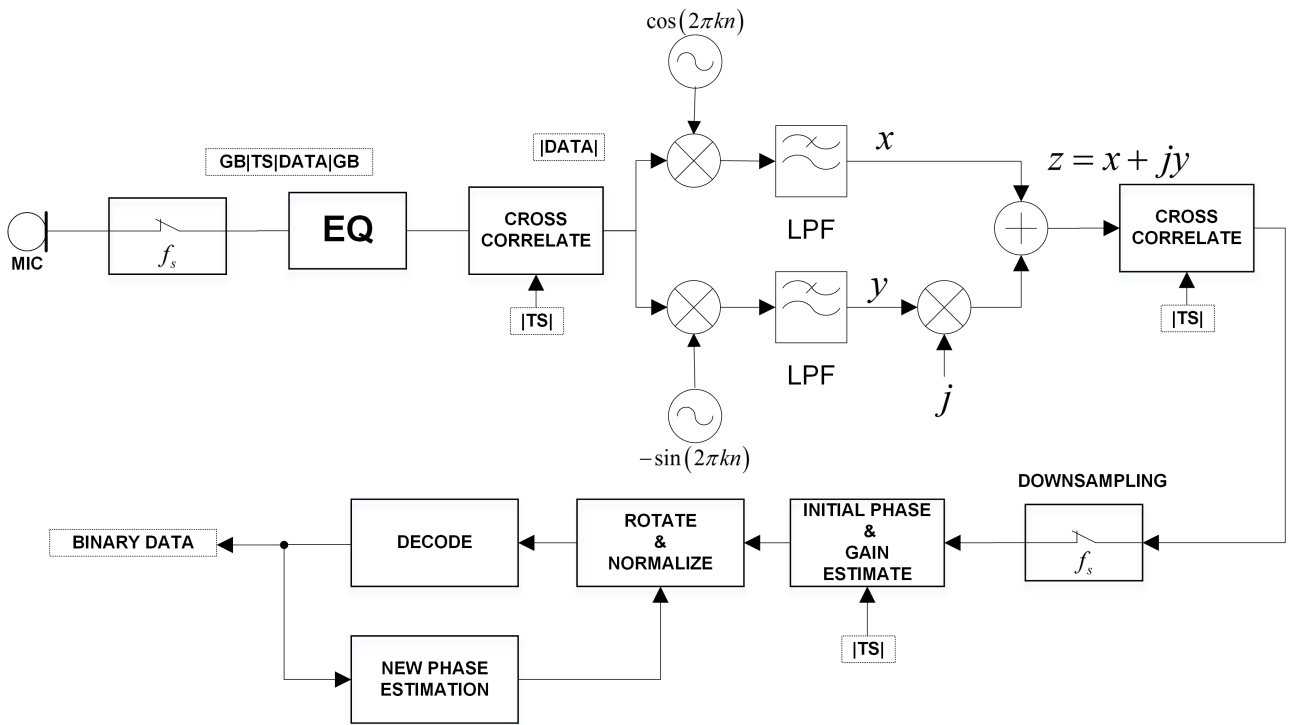

Figure 4.2: Schematic of M-QAM Receiver.

We will now describe the demodulation process. The received signal is filtered with the equaliser (EQ). The chosen equaliser is a parametric IIR filter which coefficients are computed according to the process that will be detailed in section 4.5.The received constellations before, and after applying the equaliser are depicted in 4.3.

47
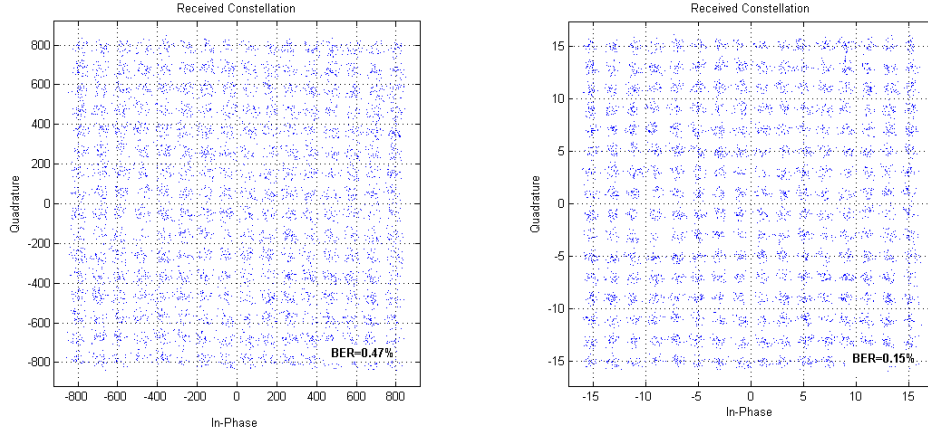
(a) Constellation after phase estimation without equaliser.

(b) Constellation after phase estimation with equaliser.

Figure 4.3: Received 64-QAM constellation and the effect of the equaliser. There is a clear improvement in the BER when using the filter.

After the equalisation process, the beginning of the transmission must be found and a sample-level cross-correlation between the modulated training sequence and the received sequence is computed. Then, the lag k, which leads to the maximum value in the cross-correlation, is computed and considered as the beginning of the training sequence. The samples are then multiplied by the carriers depicted in figure 4.2. The next step is low-pass filter the resulting samples. For this stage, we use an order $N = 14$ linear phase FIR filter modelled with the window method. The optimal parameters are obtained iteratively based on the resulting performance.

At this point, up sampled received symbols must be down sampled. The optimal choice for the sampling instants is estimated using a symbol-level cross-correlation, taking into account the received symbols and the training sequence symbols, as referred in section 2.3.1.

The consequent steps consist on the process blocks of phase and channel gain estimation. The latter parameter depend on the volume of the transmitter and is assumed to be constant during the whole transmission (even though there are implemented algorithms to keep track of it). The initial phase rotation is estimated according to the algorithm mentioned in section 2.3.1.

To make the system robust to longer transmissions(larger file sizes), the method mentioned in 2.3.2 is used to keep track of the Doppler shift. It consists on dividing the set of symbols into smaller subsets, to which the Maximum Likelihood (ML) decision is applied, subsequently the new phase rotation is estimated. This new phase estimation block is used to rotate the symbols in the following blocks. This algorithm assumes that the decoded symbol is correct and the added noise is zero-mean. It is important to choose a block with adequate length. It should not be too large neither too small[1]. Denoting t as the interval of time in which the Doppler shift does not majorly influence the decision, the size of the block in number of symbols is given by:

$$B_{sym} = \frac{t}{T_s} = \frac{t \times f_s}{n_{sym}} \tag{4.15}$$

---

[1]If the block is too large, there is a risk of error in the decoding due to the rotation, leading to a wrong decision feedback. If the block is too small, the assumption that the noise is zero-mean might not be accurate.

After processing all the blocks, the decisions from each one of those blocks can be merged together and compared with the transmitted data to compute the BER.

Optimization

There are several parameters that can be adjusted and will influence the performance of the system:

EQ. Equaliser. The gain, bandwidth and central frequency of the equaliser will have a major impact in the decoding.

LPF. Low-pass filter. The cut-off frequency, the order of the filter and the used window should be carefully chosen in order to suppress undesired spectral components.

t. Block length for decoding. The variable t denotes the estimated interval of time in which the Doppler shift does not majorly influence on the decision.

4.3   OFDM System

For the implementation of the OFDM system, the block schematic in figure 2.13 was used as a reference. On the transmitter side that are a set of parameters that need to be defined:

- N_{FFT}; The size of the FFT that is to be used. This parameter should not be neither too large or neither too small and it should be a power of two[2].

- f_c; The carrier frequency should be chosen so that the bandwidth of the signal remains inside the available bandwidth.

- P, S; Cyclic prefix and suffix lengths are set according to the channel response.

- N_c; The number of active subcarriers will define the utilized bandwidth according to equation 2.24, and it is a very important parameter for the performance of the system.

After these parameters are set and the size of the constellation is defined, the modulation algorithm takes place. Guard bits, training sequence and data to be transmitted are first merged and then mapped into the complex constellation. In the implementation, there is an option to insert pilots after each $N_{interval}$ transmitted bits. These pilots are used to reestimate the channel. However, apart from the frequency offset, the channel is time-invariant and hence, better performance is obtained when these reference symbols are not used. Then, the symbols are input into the OFDM modulation function. There are N_{FFT} subcarriers for each OFDM symbol, but only N_c are used (the number of virtual subcarriers is $N_{FFT} - N_c$). Each of these blocks of size N_c is split into two, and saved into the corners of a buffer of size N_{FFT}, as depicted in figure **??**. Then, the FFT of each of these blocks is computed and cyclic prefix and suffix are added, according to the mentioned in section **??** (this operation is shown

---

[2]A too small N_{FFT} will lead to more overhead (due to P and S) and will hence reduce the rate. A too large N_{FFT} will take more time to compute and the sub-carrier spacing will decrease which leads to increased ICI.

in figure **??**). The resulting signal is then upconverted using a complex exponential of frequency f_c and its real part is transmitted.

On the receiver side there is no need for equaliser, since OFDM is being used. An high pass filter might be used to filter the low components of the noise. The received signal is then cross-correlated with the training sequence and the guard bands and silent parts are cropped. Afterwards, down conversion is performed, using the carrier frequency exponential and, subsequently, the cyclic prefixes and suffixes removed. To retrieve the complex transmitted symbols, the resulting samples are divided into blocks of N_{FFT} size and FFT is applied. The received training sequence symbols are compared with the transmitted and the phase and gain estimations are computed for each of the subcarrier frequencies. An example of the outcome of these estimations is depicted in figure **??**. It is possible to verify the non-linearities of the channel in both amplitude and phases. Since it is known in which subcarrier the symbols are transmitted, the corresponding phase and amplitude corrections are applied accordingly. The frequency offset correction algorithm is the same as the one used in the complex baseband implementation, and is explained in section **??**. An example of the received constellations in OFDM before and after correction is depicted in figure **??**.

4.4 Channel Coding

Since our system is implemented purely in software, we have to use a coding scheme that does not introduce too much delay and heavy processing to the phone. On the other hand, our goal is to implement an application capable to transmit digital files, thereby we should guarantee a transmission with zero errors or being capable to correct all of them. Such task is not completely straightforward, since the BER will depend in many different factors that shall be analysed separately to measure the effect in the transmitted data. The chosen modulation scheme for our performance analysis test is 256-QAM. We selected such configuration, since it was not possible to get an error free transmission with this setup. The first step, is to compare the performance of our MATLAB implementation with the theoretical performance of uncoded 256-QAM. Using MATLAB's embedded function `berawgn`[3], and simulating an AWGN channel, we see in figure 4.4 that the achieved performance with our implementation approximates the theoretical one.

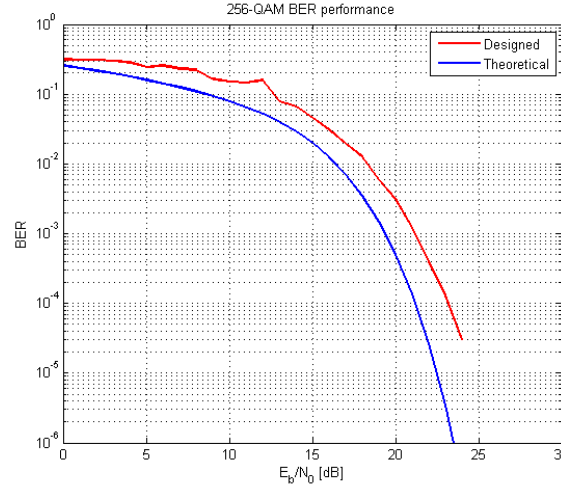[3]Computed according to equation 2.19 from section 2.1.4.

Figure 4.4: Performance of 256-QAM system compared with the theoretical.

From this point, we can introduce coding into our simulation and measure the performance for different code rates. The suggested FEC implementation is a variable rate convolutional encoder with maximum free distance (according to [2]) at the following code rates: 1/2, 2/3,3/4, and 4/5. The used generator polynomials are $[15, 17]$, $[27, 75, 72]$, $[13, 25, 61, 47]$, and $[237, 274, 156, 255, 337]$, respectively. The simulation shows that for the theoretical case, one can reduce the number of errors significantly. Despite of this, we could not correct all the errors in the real channel, even with using the lowest code rate (1/2). In addition, using 256-QAM with a coding scheme would end up getting us practically the same throughput compared to the case of uncoded 64-QAM. The results can be summarised in table 4.1.

Add performance results.

Scheme	Raw data rate[kbps]	Real data rate[kbps]	$\mathbf{E_b/N_0}$ [dB]	BER
256-QAM, NO FEC	44.10	35.46	21	1.397×10^{-4}
256-QAM, FEC:1/2	44.10	4	6	1.397×10^{-4}
256-QAM, FEC:3/4	44.10	4	4	1.397×10^{-4}
64-QAM, NO FEC	33.08	27.96	7	1.397×10^{-4}

Table 4.1: Performance of different modulation/coding schemes.

4.5   Channel Equalisation and filter design

In order to characterise the channel and determine the best possible way to deal with the distortion that is present, we estimated the channel transfer function with a non-parametric correlation method that got us $k = 45$ taps of a FIR model with order $N = k - 1 = 44$. Such process was done using MATLAB's function `impulseest`, which is based on a least-squares algorithm to obtain the coefficients. The resulting transfer function is plotted in figure 4.5.

An inverse optimal equaliser filter using this method is not stable, for this reason, the filter has to be constructed based on a parametrical IIR model. The function `fdesign.parameq` constructs a
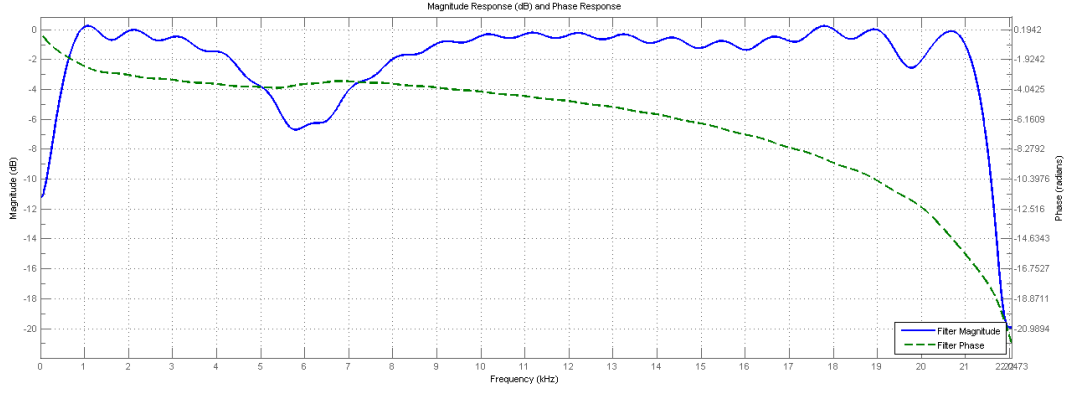
Figure 4.5: Estimated least-squares FIR model of the channel transfer function. The filter order is $N = 44$.

Chebyshev Type I peak filter (figure 4.6a) with the following transfer function:

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}} \tag{4.16}$$

The filter main design parameters are central frequency, passband bandwidth, reference gain, and passband gain. The reconstructed channel based on these approach is depicted in figure 4.6b. Once the parameters are defined we obtain the coefficients from MATLAB and run an iterative process to optimize the filter to perform well in all the phones.
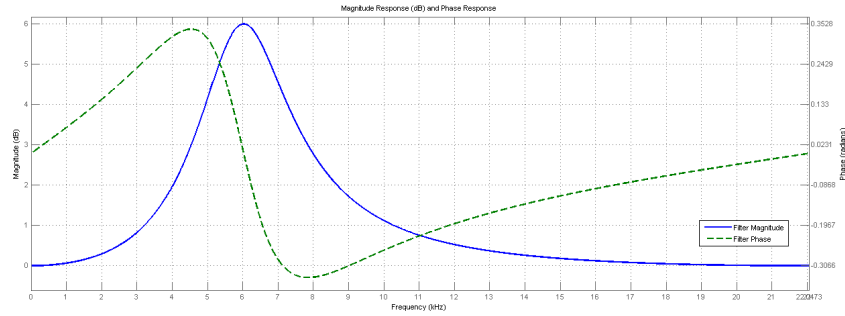
(a) IIR peak filter.
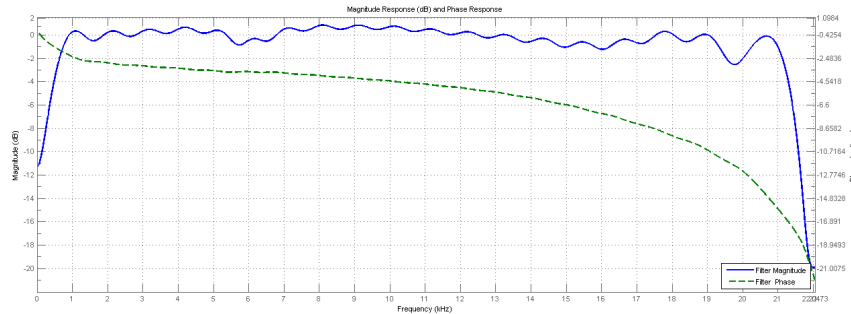
(b) Cascaded channel after filtering.

Figure 4.6: Channel reconstruction using the IIR peak filter.

Chapter 5

Results and discussion

We tested several modulations schemes modulating and demodulating random binary sequences, first with MATLAB (off-line mode), and then with the phones with the provided Android Framework (on-line mode).

In the offline mode, transmitter and receiver processing is done solely in MATLAB. The transmitted samples are sent to the phone in a file (`.dat`) of *shorts*, read by the application and delivered to the sound card buffer to be played. On the receiver side, there is a recording application which stores the audio samples in a `.log` file. The file is then transferred to the computer and the samples extracted to be processed.

MATLAB is a powerful tool for simulation due to its computation power and adaptability and was, hence, chosen to find the optimal algorithms/methods/parameters for the implementation of a robust system. Once found, these methods can, with a little effort, be translated into Java. If done correctly this translation should not cause the final results to be much different in both real-time and off-line systems.

Write about major differences in algorithm implementation.

5.1 Binary FSK

Our first and basic test was to generate a random sequence and transmit it with binary FSK throughout the channel. As discussed in section 4.1.2, we used the Goertzel algorithm to decode the received modulated signal with generally good results. The maximum bit rate was about 1.5 kbps. We end up using binary shift keying only to modulate the training sequence in the 4-FSK scheme that will be described next.

Figure 5.1: Add plots

5.2 4-FSK

In the implementation, n_{sym} = 35 samples were used, resulting in a rate of R = 2.52 kbps with no errors and without the use of any coding scheme. This value for the rate considers all the bits that were transmitted, including guard band and training sequence (of length 100 bits). Hence, the effective rate is lower than this value but still well above the basic requirements of 1 kbps. A section of the transmitted/received samples and corresponding spectrum is depicted in figure 5.2, using rectangular and Hanning window.

Figure 5.2: Add plots

5.3 4-FSK and FDM

In order to improve the rate and to exploit all the available bandwidth, a FDM (frequency division multiplexing) system is used. The idea is to replicate the 4-FSK transmission in the previous section N_r times until all available bandwidth is used. Replicating the system $N_r = 7$ times, it was still possible to obtain a reliable transmission. However, it was necessary to increase n_{sym} to 102 samples, resulting in a rate of:

$$R = \frac{N_r \times log_2(4)}{n_{sym}} \times f_s = 7 \times \frac{2}{102} \times f_s = 6052 \text{ bps}$$

This value more than doubles the value achieved with the single 4FSK system. A gaussian window was used in order to modulate the samples. The resulting signals and power spectrum densities are depicted in Fig. 5.3.


Figure 5.3: Spectrum plots of the FDM-FSK signal.

5.4 M-QAM

The 64 MQAM framework is tested by sending different types of files of different sizes and making sure the received file is not corruupt by using the check sum method described in section. In Fig. 5.4b a screenshot is provided of an example where a text file (more.txt) of size 10kB (9707 bytes) is sent. The functions at the receiver are timed in order to make sure that the data rate requirement is achieved. The checksum number is displayed in both the transmitter and receiver. If these numbers are the same the file transfer is successful.
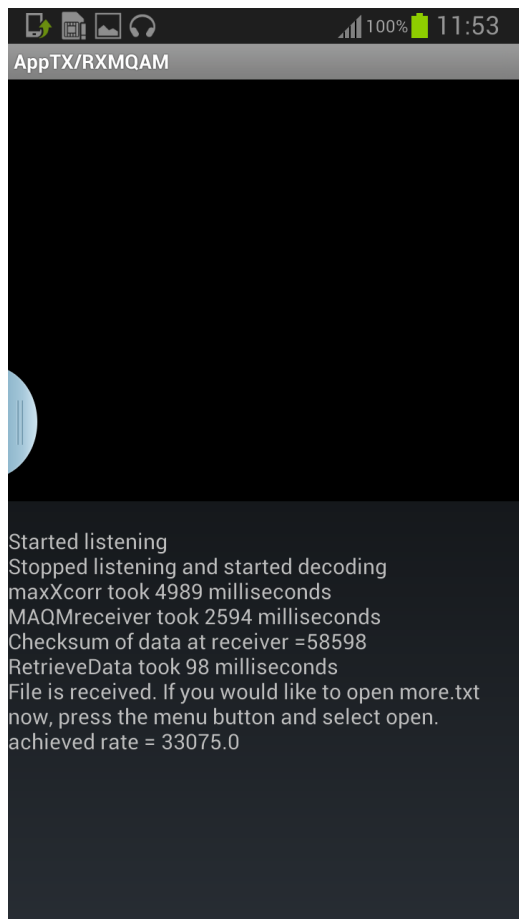
(a) Receiver
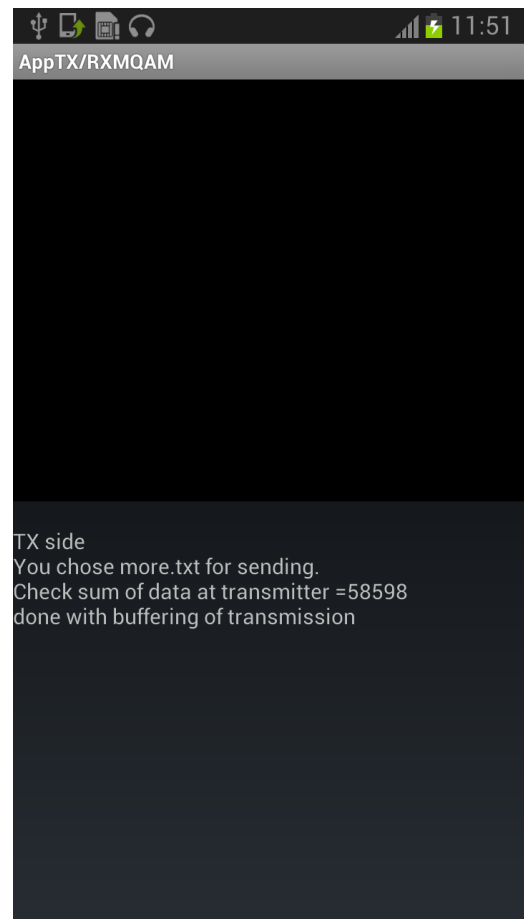
(b) Transmitter

Figure 5.4: Screenshots of receiver and transmitter in M-QAM application

5.5   OFDM

5.6   M-FSK

Chapter 6

Conclusion

Bibliography

[1] Madhow U., *Fundamentals of Digital Communication*, Cambridge , 2008.

[2] Proakis J. ,Salehi Masoud, *Digital Communications*, McGraw-Hill , 2008.

[3] Abu M., *ASK demodulator Lab Experiment*, Islamic University of Gaza, retrieved in April 2014.

[4] Sa-nguankotchakorn T. , *Frequency Shift Keying Lecture Slides*, Asian Institute of Technology, retrieved in April 2014.

[5] Okechukwu C. Ugweje, *Frequency Shift Keying, Lecture Slides*, University of Akron, retrieved in April 2014.

[6] Xiong F., *Digital Modulation Techniques*, Artech House, 2006.

[7] Shembil S., *M-Ary FSK with Orthogonal Pulse Envelopes and its Noncoherent Detection*, IJECS-IJENS, Vol. 10, No.06, Issued December 2010, pp. 103-106.

[8] Sysel and Rajmic, *Goertzel algorithm generalized to non-integer multiples of fundamental frequency*, EURASIP Journal on Advances in Signal Processing, 2012, 2012:56.

[9] Project Assignment EQ2310 Digital Communications, *Analysis and Simulation of a QPSK System*, KTH, 2013.

[10] Project Description EQ2320 PROJECT IN WIRELESS COMMUNICATION, *Project #1: In-flight file-transfer using 3-mm cable*, KTH, 2014.

[11] Haykin S., *Digital Communication Systems*, John Wiley & Sons, Inc., 2014.

[12] Dick D., Harris F., and Rice M., *FPGA Implementation of Carrier Synchronization for QAM Receivers*, Journal of VLSI Signal Processing 36, 57-71, 2004.

[13] National Instruments, (2012),*Quadrature Amplitude Modulation (QAM)[White paper]*, retrieved in May 2014.

[14] Agilent Technologies,(2013) *IQ Gain Imbalance and Quadrature Skew Error Data Interaction (Digital Demod)[9600 VSA Software - Product Documentation]*, retrieved in May 2014.

[15] Yoon D., Cho K., and Lee J., *Bit Error Probability of M-ary Quadrature Amplitude Modulation*, Vehicular Technology Conference, 2000. IEEE-VTS Fall VTC 2000. 52nd , vol.5, no., pp.2422,2427 vol.5, 2000.

[16] Diniz P., Da Silva E., Netto S., *Digital Signal Processing: System Analysis and Design*, Cambridge University Press, 2010.

[17] Sklar B., *Digital Communications Fundamentals and Applications*, Prentice Hall, 2001.

[18] Robert Sedgewick and Kevin Wayne, *FFT.java*, Princeton Copyright © 2000–2011

[19] Robert Sedgewick and Kevin Wayne, *Complex.java*, Princeton Copyright © 2000–2011

[20] D. Borman, Cray Research, C. Partridge, R. Braden, *Computing the Internet checksum*, BBN Laboratories, September 1988