Practical Understanding of the Operating Principle of Digital Communication Systems

Gebrehiwet Gebrekrstos Lema*, Teklehaymanot Baweke, Dawit Hadush Hailu

School of Electrical and Computer Engineering, Mekelle University, Mekelle, Ethiopia.
Email*: g.jcool.com@gmail.com

Abstract

There are many students and researcher who doesn't really understand the practical operating principle of digital communication system. Hence in this paper, the digital communication system is studied and simulated. In the system kit development text and audio inputs are taken and encrypted with different encryption techniques including additive cipher, multiplicative cipher and affine ciphers. The encrypted data is converted in to an 8-bit binary, channel encoded with distinct channel coding styles like linear block encoder, cyclic encoder and convolutional encoder, line coded and band pass modulated by different digital modulation techniques. Finally, the developed software is tested with equivalent inputs of the current national TV broadcasting and the results found are correct according to the theoretical analysis of the discussion.

1 Introduction

This digital communication simulation software is a tool for practical learning applications in campus and it may also be used in practical transmission and reception of digital signals if modified somehow. In our country students are mostly worried about where they can practice what they have learnt in class. Because there is shortage of electronics components to practice in lab and see how it practically works. And due to this reason students have limited understanding of digital communication system and its application.

The growing of demand for the huge data transmission made the digital communication systems increasingly attractive. The fastest growing area in communication engineering is the design and

manufacturing of hardware and software for digital communication networks [1]. It is proven that digital signals can be coded to produce extremely low error rates and high fidelity [2].

Some bits can corrupt in the noisy channel and might get interference through transmission over the communication channel and the received signal will be demodulated into binary bits [3]. Hence, new kinds of code, and new decoding methods, have recently been developed and are starting to be applied [4]. Among the most sophisticated coding technique applied to the system can convert a noisy channel into an error-free channel [5]. Error-free transmission is done by applying a coding technique of a random nature [6].

Source encoder and decoder have been designed without knowledge about the statistical channel encoder and decoder [7]. The idea of jointly designing the source-channel coding, begun with the work of [8]. It has been reported that, in noisy channels the design of joint source-channel encoder and decoder can improve the performance dramatically. Digital communication has led to invent the social networking by three important applications of digital which represented by: the internet- World Wide Web, mobile communication, and satellite communication [9].

However, the practical simulation and testing the operational principles of the digital communication systems is not clearly verified in all the literatures. To the best of our knowledge there is no research that discusses the practical understanding of the principles of the digital communications systems.

This project on digital communication simulator almost completely solves the above problem. Obviously, electronics devices are more sensitive to damage during test in lab if the user makes a little mistake. This increases the cost of making and implementing the project. But this simulator kit takes no extra cost for design and development except we used a computer to develop it. This is a kind of fool proof project which you can't use it wrongly.

Digital communication simulator software simulates most of the basic blocks of digital communication systems. This simulator kit only processes text and audio files but if the size of

the audio file is large channel coding and channel decoding will take large time or probably the pc may stack.

The main objective of this project is to make sure that students have broad understanding of the basic blocks of digital communication systems and information theory and coding through practical simulations. And the specific objective of this project work includes:

1. To study the principles of different kinds of encryption and decryption methods
2. To examine the formatting technique
3. To see how the different types of channel coding work and see their performance related to band width.
4. To observe the different types of line coding and decoding techniques.
5. To study the behavior of base band modulation and demodulation techniques.
6. To see how error detected and corrected if any at the receiver using the following algorithms.

In developing this software, we use the c sharp programming language as a tool. In developing this software of digital communication simulation, we use the substitution method of encryption in the encryption block. For the channel coding and decoding we use the systematic linear block codes. In the band pass demodulation/detection also we include only the coherent detection method.

2. Fundamentals of digital communications

In this chapter, we will present the basic principles that underlie the analysis and design of digital communication systems. The subject of digital communication involves the transmission of information in digital form from source that generates the information to one or more destinations.

The sequences of binary digits from the source encoder are passed to the channel encoder. The purpose of channel encoder is to introduce, in a controlled manner, some redundancy in the binary information sequence that can be used at the receiver to overcome the effects of noise and interference encountered in the transmission of the signal through the channel. Thus, the added redundancy serves to increase the reliability of the received data and improves the fidelity of the received signal. In this case if k bits of message bits are channel encoded in to n bits the rate of the code will be k/n.

The binary sequence at the output of the channel encoder is passed to the digital modulator, which serves as the interface to the communication channel. Since nearly all of the communication channels encountered in practice are capable of transmitting electrical signals (wave forms), the primary purpose of the digital modulator is to map the binary information sequence in to signal wave forms. The digital modulator may simply map the binary digit 0 into a waveform s_0(t) and the binary digit 1 into a waveform s_1(t). In this manner each bit from the channel encoder is transmitted separately. We call this binary modulation. Alternatively, the modulator may transmit b coded information bits at a time by using M=2^b distinct waveforms s_i(t), $i = 0, 1, \dots, m-1$, one wave form for each of the 2^b possible b-bit sequences. This type of modulation we call M-array modulation.

The communication channel is the physical medium that is used to send the signal from the transmitter to the receiver. In wireless transmission, the channel may be the atmosphere (free space). On the other hand, telephone channels usually employ a variety of physical media, including wire lines, optical fiber cables, and wireless (microwave radio). Whatever the physical medium used for transmission of the information, the essential feature is that the transmitted

signal is corrupted in a random manner by a variety of possible mechanisms, such as additive thermal noise generated by electronic devices, man-made noise, and atmospheric noise.

At the receiving end of a digital communications system, the digital demodulator processes the channel-corrupted transmitted waveform and reduces the waveforms to a sequence of numbers that estimates of the transmitted data symbols (binary or M-array). This sequence of numbers is passed to the channel decoder, which attempts to detect and correct errors and reconstruct the original information sequence from knowledge of the code used by the channel encoder and the redundancy contained in the received data.

A measure of how well the demodulator and decoder perform is the frequency with which errors occur in the decoded sequence. More precisely, the average probability of a bit-error at the output of the decoder is a measure of the performance of the demodulator-decoder combination.
As a final step, when non-digital output is desired, the source decoder accepts sequences of binary signals from the channel decoder and reconstructs the original encrypted signal. The decryption block accepts the encrypted signal and deciphered the signal to get the original transmitted message.
In the following subtitles we are going to see the detailed concepts and algorithms of each block.

Information source: Information source is the original message given to a digital communication system as an input. The message source can possibly be text, video, audio and digital data like computer output.

Encryption: Encryption is the process of hiding data to protect from external hackers. There are two types of encryption called binary and text ciphers. But in this article, we will see the substitution cipher or encryption which is text ciphering. Substitution cipher used to hide our data by replacing the characters in our plaintext by another characters. But the replacement is takes place systematically. Additive cipher, multiplicative cipher and affine cipher are the frequently used substitution method ciphers. Let’s see the algorithms one by one.

Additive cipher: An additive cipher is a substitution cipher. The letters of the plaintext stay in the correct position but are replaced by other letters.

We have the letters sequentially as follows.

a b c d e f g h i j k l m n o p q r s t u v w x y z

Algorithm for ciphering:

1. Translate letter into its corresponding array number (ex. b = 2).
2. Add with key number m (let m=5, b=2, 2+5=7).
3. Reduce result mod 26 and translate back into letter (7= g)

Algorithm for deciphering:

1. Translate letter into its corresponding number (ex. g = 7).
2. Add the number with additive inverse of **m**, $\mathbf{m^{-1}}$ ($\mathbf{m^{-1}}$=26-m, 26-5=21 and 7+21=28).
3. Reduce mod26 and translate back to letter (28mod26=2 which is = b)

Example: we have plaintext "mekelle". Then

1. Translate letter into its corresponding array number. Then the result will be "13, 5, 11, 5, 12, 12, and 5".
2. Add key number m=5 then the result will be "18, 10, 16, 10, 17, 17, and 10".
3. Reduce result mod 26 and translate back into letter. And the final ciphered text will be "rjpjqqj".

This takes place at the transmitter side and the ciphered text "rjpjqqj" is transmitted to the receiver and the decryption algorithms takes place to get the original data at the receiver side.

1. Translate letter into its corresponding number ", 10, 16, 10, 17, 17, and 10".
2. Add the number with additive inverse of **m**, $\mathbf{m^{-1}}$ ($\mathbf{m^{-1}}$=26-m, 26-5=21) and add $\mathbf{m^{-1}}$ with the received data to get "13, 5, 11, 5, 12, 12, and 5".
3. Reduce mod26 and translate back to letter. The final deciphered text will be "mekelle".

Multiplicative cipher: It is substitution ciphers, but letters are replaced by multiplication. Modular arithmetic becomes more obviously necessary here. The trickiest part is finding the multiplicative inverse in the modular system. For example, in common arithmetic, the

multiplicative inverse of 9 is 1/9 because 9*1/9=1. But in multiplication mod26, we can't multiply by 1/9. Instead we must find the number, in this case 3, which when multiplied by 9 still yields 1.

Example: 9*3=27=1(mod26).

Algorithm for encoding:

1. Translate letter into its corresponding number (ex. b = 2).
2. Multiply number by **t** (let t=5, b=2, 2*5=10).
3. Reduce result mod 26 and translate back into letter (10 = j).

Algorithm for decoding:

1. Translate letter into its corresponding number (ex. j = 10)
2. Multiply number by the multiplicative inverse of **t**, $\mathbf{t^{-1}}$ ($\mathbf{t^{-1}}$=21, 10*21=210)
3. Reduce mod26 and translate back to letter (210 =2mod26; 2 = b)
4.

Example: we have plaintext "abebe". Then at the transmitter the ciphering process will be as follows:

1. Translate letter into its corresponding number to get "1, 2, 5, 2, 5"
2. Multiply number by **t** (let t=5) then the result will be "5, 10, 25, 10, 25".
3. Reduce result mod 26 and translate back into letter to get ciphered text which is "ejyjy".

The ciphered text "ejyjy" is transmitted to the receiver and at the receiver side the decryption algorithm is used to get the original data as follows:

1. Translate letter into its corresponding number to get "5, 10, 25, 10, 25".
2. Find multiplicative inverse of **t**, $\mathbf{t^{-1}}$ so that ($\mathbf{t^{-1}}$*t) 2mod26=1 and multiply with the number. In this case the question ($\mathbf{t^{-1}}$*5) 2mod26=1 to be true $\mathbf{t^{-1}}$ should equal to 21. The result after multiplication will be "105, 210, 525, 210, 525".
3. Reduce mod26 (105, 210, 525, 210, 525)2mod26 = (1, 2, 5, 2, 5) and translate back to letters to get the original data which is "abebe".

Affine cipher: Affine ciphers are another variation of substitution ciphers. This time, we combine both additive and multiplicative steps. When encoding, add first, and then multiply. When decoding, multiply first by the multiplicative inverse (t^{-1}), then add the additive inverse ($\mathbf{m^{-1}}$).

Algorithm for ciphering:

1. Convert letter to number.
2. Add additive encoding key (**m**) and reduce mod26.
3. Multiply by multiplicative encoding key (t) and reduce mod26.
4. Convert number to letter.

Algorithm for deciphering:

1. Convert letter to number.
2. Multiply by the multiplicative inverse, t^{-1}, and reduce mod26. ($t*t^{-1}=1(\text{mod}26)$)
3. Add the additive inverse, $\mathbf{m}^{-1}$, and reduce mod26. ($\mathbf{m}+(\mathbf{m}^{-1})=0(\text{mod}26)$)
4. Convert number to letter.

Source encoder and decoder: Source encoding is the process of converting asci code of the array of characters in to binary equivalent. The source encoder block accepts the encrypted data from the encryption block and converts each character to asci 8 or asci 7. The source decoding is the process of converting the binary bits into the asci equivalent. The conversion process takes place by the following algorithm. Let we have an 8 bit array $a = [a_1\ a_2\ a_3\ a_4\ a_5\ a_6\ a_7\ a_8]$. Then, the asci equivalent will be

$$asci = a_1 + a_2 * 2 + a_3 * 4 + a_4 * 8 + a_5 * 16 + a_6 * 32 + a_7 * 64 + a_8 * 128$$

Channel encoder and Channel decoding**:** Channel coding is often used in digital communication systems to protect the digital information from noise and interference and reduce the number of bit errors. Channel coding is mostly accomplished by selectively introducing redundant bits in to the transmitted information stream. These additional bits will allow detection and correction of bit errors in the received data stream and provide more reliable information transmission. The cost of using channel coding to protect the information is a reduction in data rate or expansion in bandwidth. There are two main types of channel codes, namely block codes and convolutional codes. The block codes accept a block of k information bits and produce a block of n coded bits. By predetermined rule, n-k redundant bits are added to the k information bits to form the n coded bits. Commonly these codes are referred to us (n, k) block codes. Some of the commonly used

block codes are linear block codes and cyclic block codes. The convolutional coding encodes the sequence of message bits bit by bit instead of taking blocks of bits.

Channel decoding is the process of recovering the transmitted bits from the channel encoder block. Channel decoding involves receiving the channel codeword with its possible channel noise, error detection location and correcting if any. If no errors are found, the received block is taken as the errorless channel codes. Depending on the transmitter encoder, there are adopted corresponding decoding schemes.

In the following subtitles we will see the details of the following channel coding and decoding types.

1. Linear block code
2. Cyclic block code
3. Convolutional code

Any arithmetic addition in this channel coding is modulo 2 additions. That is

1+1=0,

1+0=1

0+0=0

Linear block coding and decoding

Linear block coding: Linear block encoder accepts binary data from the source encoder and segments it in to message blocks of length k bits denoted by **U**. the message blocks are then encoded by adding n-k parity bits to produce an n bit length codeword denoted by **V**. there are a total of 2^k distinct messages. The encoder according to certain rules transforms each input message u in to n bit **V** with $n > k$. Corresponding to the 2^k possible messages, there are 2^k code words called code word. An encoder which takes m bits and encode them to n bits is called (n, k) encoder.

The sequence of message and code word bits is given as follows:

$$U = [u_1, u_{2,} u_{3,---,} u_k]$$

$$V = [v_{1,} v_{2,} v_{3,---,} v_n]$$

The codeword **V** is generated by matrix multiplication of a generator matrix with the input vector **U**. Generator matrix is an important parameter and it has two parts called the parity bit and information bit parts.

$$G = [p_{kxn-k} : I_{kxk}]$$

The parity bit is represented by p_{kxn-k} and it is found by dividing x^{n-k+i} to the generator polynomial $g(x)$ for $i = 0,1, \dots, k-1$ and taking the remainder coefficients. The information part is represented by I_{kxk} and is $k\ by\ k$ identity matrix. The generator polynomial should by a primitive with degree $n-k$.

Then the resulting generator matrix will be:

$$G = \begin{bmatrix} p_{11} & p_{11} & \cdots & p_{1(n-k)} & : & 1 & 0 & \dots & 0 & 0 \\ p_{21} & p_{22} & \cdots & p_{2(n-k)} & : & 0 & 1 & \dots & 0 & 0 \\ . & . & . & . & : & . & . & \dots & . & . \\ . & . & . & . & : & . & . & \dots & . & . \\ p_{(k-1)1} & p_{(k-1)2} & \cdots & p_{(k-1)(n-k)} & : & 0 & 0 & \dots & 1 & 0 \\ p_{(k)1} & p_{(k)2} & \cdots & p_{k(n-k)} & : & 0 & 0 & \dots & 0 & 1 \end{bmatrix}$$

The encoding process takes place by multiplying each k message bits with the generator matrix as follows.

$$\boldsymbol{V = U * G}$$

Let $u = [u_1, u_{2,} \dots, u_k]$ be the message to be encoded and let $v = [v_{1,} v_{2,} v_{3,---,} v_n]$ be the encoded codeword. Then:

$$\boldsymbol{V} = [u_1, u_{2,} \dots, u_k] \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1(n-k)} & : & 1 & 0 & \dots & 0 & 0 \\ p_{21} & p_{22} & \cdots & p_{2(n-k)} & : & 0 & 1 & \dots & 0 & 0 \\ . & . & . & . & : & . & . & \dots & . & . \\ . & . & . & . & : & . & . & \dots & . & . \\ p_{(k-1)1} & p_{(k-1)2} & \cdots & p_{(k-1)(n-k)} & : & 0 & 0 & \dots & 1 & 0 \\ p_{(k)1} & p_{(k)2} & \cdots & p_{k(n-k)} & : & 0 & 0 & \dots & 0 & 1 \end{bmatrix}$$

From this

$$v_1 = u_1 p_{11} + u_2 p_{21} + \cdots + u_k p_{(k)1}$$

$$v_2 = u_1 p_{12} + u_2 p_{22} + \cdots + u_k p_{(k)2}$$

$$. \quad = \quad . \qquad\qquad .$$

$$. \quad = \quad . \qquad\qquad .$$

$$v_{n-k} = u_1 p_{1(n-k)} + u_2 p_{2(n-k)} + \cdots + u_k p_{(k)(n-k)}$$

$$v_{n-k+1} = u_1$$
$$v_{n-k+2} = u_2$$
$$. \quad = \quad . \quad .$$
$$v_{n-1} = u_{k-1}$$
$$v_n = u_k$$

Linear block decoding: In this decoding we have useful parity check matrix (**H** matrix) which helps to know if the received codes are generated from the generator matrix G. for each (n, k) generator matrix G, there exist an $(n-k) \times n$ matrix **H**, such that the rows of G are orthogonal to the rows of H; that is, $GH^T = \mathbf{0}$, where H^T is the transpose of H, and $\mathbf{0}$ is a $k \times (n-k)$ all-zeros matrix.

H^T is an $n \times (n-k)$ matrix whose rows are the columns of **H** and whose columns are the rows of **H**. To fulfill the orthogonality requirements for a systematic code, the components of the H matrix are written as:

$$G = [p_{kxn-k} : I_{kxk}]$$

Then the H matrix is

$$H = [I_{n-kxn-k} : p^T{}_{n-kxk}]$$

Hence, the H^T matrix is written as:

$$H^T = \begin{bmatrix} I_{n-kxn-k} \\ \dots\dots\dots\dots \\ p_{k\times n-k} \end{bmatrix}$$

If the received codeword v is generated from the generator matrix G, then $\boldsymbol{V} * \boldsymbol{H^T} = 0$ otherwise the received codeword is in error.

Syndrome calculation and error detection: Syndrome is the strength of a codeword against error or noise. Syndrome is calculated as follows:

Given that a codeword **v** generated from generated matrix G is transmitted and **r** with potential noise is received. Then

$$\boldsymbol{r} = \boldsymbol{v} + \boldsymbol{e}$$

Where $\boldsymbol{e} = e_1, e_2, \ldots, e_n$ is an error vector or error pattern introduced by the channel. There is a total of $2^n - 1$ potential nonzero error patterns in the space of 2^n n-tuples. The syndrome of r is defined as

$$\boldsymbol{S} = \boldsymbol{rH^T}$$

The syndrome is the result of a parity check performed on **r** to determine whether **r** is a valid member of the codeword set. If the received codeword **r** is generated from **G** then the syndrome **s** has a value **0**. If **r** contains detectable errors, the syndrome has some nonzero value. The syndrome of **r** is seen to be

$$\boldsymbol{S} = (\boldsymbol{v} + \boldsymbol{e})\boldsymbol{H^T} = \boldsymbol{vH^T} + \boldsymbol{eH^T}$$

However, $\boldsymbol{vH^T} = 0$ for all members of the code word set. Therefore

$$\boldsymbol{S} = \boldsymbol{eH^T}$$

From the above equations we can conclude that whether performed on either a corrupted code vector or on the error pattern that caused it, the syndrome is the same. An important property of linear block codes, fundamental to the decoding process, is that the mapping between correctable error patterns and syndromes is one to one.

Error correction: We have detected a single error and have shown that the syndrome test performed on either the corrupted codeword, or on the error pattern that caused it, yields the same syndrome. This should be a clue that we not only can detect the error, but since there is one-to-one correspondence between correctable error patterns and syndromes, we can correct such error patterns.

Let us arrange the 2^n n-tuples that represent possible received vectors in array, called the *standard array*, such that the first row contains all the **codewords**, starting with the all zeros codeword, and the first column contains all the correctable error patterns. From the basic properties of linear codes, the all-zeros vector must be a member of the codeword set. Each row, called a *coset,* consists of an error pattern in the first column, called the *coset leader,* followed by the codewords corrupted by that error pattern.

if we take (7, 4) encoder with generator polynomial $g(x) = 1 + x + x^3$ .then, we arrange the $2^7 = 128$ seven-tuples in a standard array. The valid codewords are the sixteen vectors in the

first row, and the correctable error patterns are the seven nonzero *coset leaders* in the first column. Notice that all 1-bit error patterns are correctable. So, decoding will be correct if, and only if, the error pattern caused by the channel is one of the coset leader.

The syndromes value listed in the table at the last column are determined from the correctable error sequence by computing

$$\boldsymbol{S = eH^T}$$

The procedure for error correction and decoding proceeds as follows:

1. Calculate the syndrome of **r** using $\boldsymbol{S = rH^T}$
2. Locate the coset leader (error pattern) e, whose syndrome equals $\boldsymbol{rH^T}$
3. This error patter is assumed to be the corruption caused by the channel.
4. The corrected received vector, or codeword, is identified as $\boldsymbol{v = r + e}$. we can say that we retrieve the valid codeword by subtracting out the identified error.

Decoder implementation: When the code is short as in case of the (7, 4) code described in the above section, the decoder can be implemented with simple circuitry. Consider the steps that the decoder must take:

(1) Calculate the syndrome, (2) locate the error pattern, and (3) perform modulo-2 addition of the error pattern and the received vector (which removes the error).

From the syndrome equation we can derive expression for each of the syndrome digits in terms of the received codeword digits as

$$\boldsymbol{S = rH^T}$$

$$\boldsymbol{S} = [\boldsymbol{r_1\ r_2\ r_3\ r_4\ r_5\ r_6\ r_7}] \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix}$$

Line coding (baseband modulation)

The term digital baseband modulation is synonymous to line codes. These are methods to transfer a digital bit stream over an analog baseband channel (low pass channel) using a pulse

train that is a discrete number of signal levels. The terminology line coding originated in telephony with the need to transmit digital information across a copper telephone *line.* more specifically, binary data over a digital repeated line. The most common line coding is NRZ-L, RZ, Manchester, Uni-polar RZ, Bi-polar RZ and alternative mark inversion (AMI).

NRZ-L (non return to zero): a 1 and 0 are represented by pulse of equal amplitude with positive and negative sign respectively.

RZ (on-off signaling): in this form, symbol '1' is represented by transmitting a pulse of constant amplitude for the entire duration of the symbol and symbol '0' is represented by switching off the pulse.

Uni-polar RZ: a one is represented by half bit wide pulse and a zero is represented by the absence of a pulse.

Bi-polar RZ: - the ones and zeros are represented by opposite level pulses that are half bit wide. That is a pulse is present in each interval.

AMI (alternative mark inversion): positive and negative pulses of equal amplitudes are used alternatively for symbol '1' and no pulse for symbol '0'.

Manchester: a symbol '1' is sent by transmitting positive volt for the first half of the bit interval and negative volt for the 2^{nd} half of bit interval. And the symbol '0' is sent with the inverse signal.

Bandpass modulation and demodulation/detection

Bandpass modulation: Digital modulation is the process by which digital symbols are transformed in to waveforms that are compatible with the characteristics of the channel. In the case of baseband modulation, the waveforms usually take the form of shaped pulses. But in the case of band pass modulation the shaped pulses modulate a sinusoidal wave form called a carrier wave, or simply a carrier.

In any event, the modulation process involves shifting or keying the amplitude, frequency or phase of the carrier in accordance with the incoming data. Thus, there are 3 basic digital modulation techniques.

1. Amplitude shift keying (ASK)
2. Frequency shift keying (FSK)

3. Phase shift keying (PSK)

Amplitude shift keying (ASK): In this method the amplitude of the carrier assumes one of the two amplitudes dependent on the logic states of the input stream. In ASK the modulated waveform may be written as:

$$v_{ASK}(t) = \begin{cases} A_c \cos w_{ct} = \sqrt{\frac{2Eb}{Tb}} \cos w_{ct}\,, & input = 1 \\ 0 \qquad\qquad\qquad\qquad\quad , & input = 0 \end{cases}$$

Where Eb = bit energy = $\frac{Ac^2 * Tb}{2}$

To generate binary ASK we have to represent the input binary signal in uni-polar form or on off signal.

Phase shift keying: The general analytic expression for PSK is

$$s_i(t) = \sqrt{\frac{2E}{T}} \cos[w_0 t + \phi_i(t)] \quad 0 \le t \le T \quad i = 1, \ldots, M$$

Where the phase $\phi_i(t)$, will have M discrete values, typically given by

$$\phi_i(t) = \frac{2\pi i}{M} \qquad i = 1, \ldots, M$$

For the binary PSK (BPSK) M is 2. The parameter E is symbol energy, T is symbol time duration, and $0 \le t \le T$. In BPSK modulations, the modulated data signal shifts the phase of the waveform $s_i(t)$ to one of the two states, either zero or π (180^0).

To generate a binary PSK signal, we have to represent the input binary sequence in NRZ with constant amplitude levels $+\sqrt{E_b}$ and $-\sqrt{E_b}$ for binary "1" and "0" respectively. The resulting NRZ signal and the sinusoidal carrier$\phi_1(t)$, whose frequency constant are applied to a product modulator.

Frequency shift keying: The general analytic expression for FSK modulation is

$$s_i(t) = \sqrt{\frac{2E}{T}} \cos(w_i t + \phi)\,, \qquad 0 \le t \le T, \quad i = 1, \ldots, M$$

Where the frequency term w_i has M discrete values, and the phase term ϕ is an arbitrary constant. For the binary FSK (BFSK) M is 2. The parameter E is symbol energy, T is symbol time duration, and 0 ≤ t ≤ T. In BFSK modulations, the modulated data signal shifts the frequency of the waveform $s_i(t)$ to one of the two frequency states; either$f_{c1} or\ f_{c2}$.

$$s_{BFSK}(x) = \begin{cases} \sqrt{\frac{2E}{T}} \cos w_{c1}t\,, \ for\ input =' 1' \ , & w_{c1} = 2\pi f_{c1} \\ \sqrt{\frac{2E}{T}} \cos w_{c2}t\,, \ for\ input =' 0' \ , & w_{c2} = 2\pi f_{c2} \end{cases}$$

To generate binary FSK we have to represent the input binary signal in ON-OFF signaling form. Then the signal is applied to product modulator with two frequency $f_{c1}$and f_{c2} and the input to the 2^{nd} product modulator is inverted.

$$s_{BFSK}(x) = d(t)cos\ w_{c1}(t) + d'(t) \cos w_{c2}(t)$$

$where\ d'^{(t)} is\ inverse$ of $d(t)$

Bandpass demodulation/detection**:** Band pass demodulation is the process of recovering what was transmitted to the channel considering any possible distortions or corruptions of the signal. When the carrier exploits knowledge of the carrier's phase to detect the signals, the process is called coherent detection; when the receiver does not utilize such phase reference information, the process is called noncoherent detection. In digital communications, the term demodulation and detection are often used interchangeably, although demodulation emphasizes waveform recovery, and detection entails the process of symbol decision. In this literature we are using the coherent detection. In the following subtitles we will see the demodulation/detection processes for the ASK, PSK and FSK modulation types.

Binary detection in ASK: To detect the original binary sequence of 1's and 0's we apply the noisy BASK signal $x(t)$ to a correlater, which is also supplied with a locally generated coherent reference signal $\phi_1(t)$. The correlator output x_1 is compared with a threshold of zero volt. If x_1 $>\sqrt{\frac{Eb}{2}}$, the receiver decides in favor of symbol "1". On the other hand, if $x_1 < \sqrt{\frac{Eb}{2}}$ it decides in favor of symbol "0".

Binary detection in PSK: To detect the original binary sequence of 1's and 0's we apply the noisy BPSK signal $x(t)$ to a correlater, which is also supplied with a locally generated coherent reference signal $\phi_1(t)$. The correlater output x_1 is compared with a threshold of zero volt. If x_1 >0, the receiver decides in favor of symbol "1". On the other hand, if x_1 < 0 it decides in favor of symbol "0".

Binary detection in FSK: To detect the original binary sequence given the noisy received signal $x(t)$. It consists of two correlater with common input, which are supplied with locally generated coherent reference signals $\phi_1(t)$ and $\phi_2(t)$. The correlater outputs are then subtracted one from the other, and the resulting deference y, is compared with a threshold of zero volt. If $y > 0$ then, receiver decides in favor of "1". On the other hand, if $y < 0$ it decides in favor of "0". If y is exactly zero, the receiver makes a random guess in favor of "1" or "0".

3 Results and discussions

In the digital communication simulation software, each block accepts input from the preceding block and also in some block some input parameters are required from the user. In this chapter we are going to discuss the input and output parameters for each block. In order to facilitate the input of these parameters and follow the flow of the simulation, the Graphical User Interface (GUI) is designed for convenience to the user. The input parameter can both be entered from the GUI or from prepared user file.

The GUI has 4 parts called the block diagram, transmitter, receiver and the help part. For the block diagram and help part no need of input. They are output of the GUI. When the user clicks at the block diagram tab the block diagram that we develop in our project are displayed in the GUI. And when the help tab is clicked all the algorithms and definitions of each block will be displayed so that the students can remember what they have learnt in class. Now we are going to see the input and output parameters for the transmitter and receiver. Have a look at the screen shot of the developed general block diagram of the graphical user interface shown below.

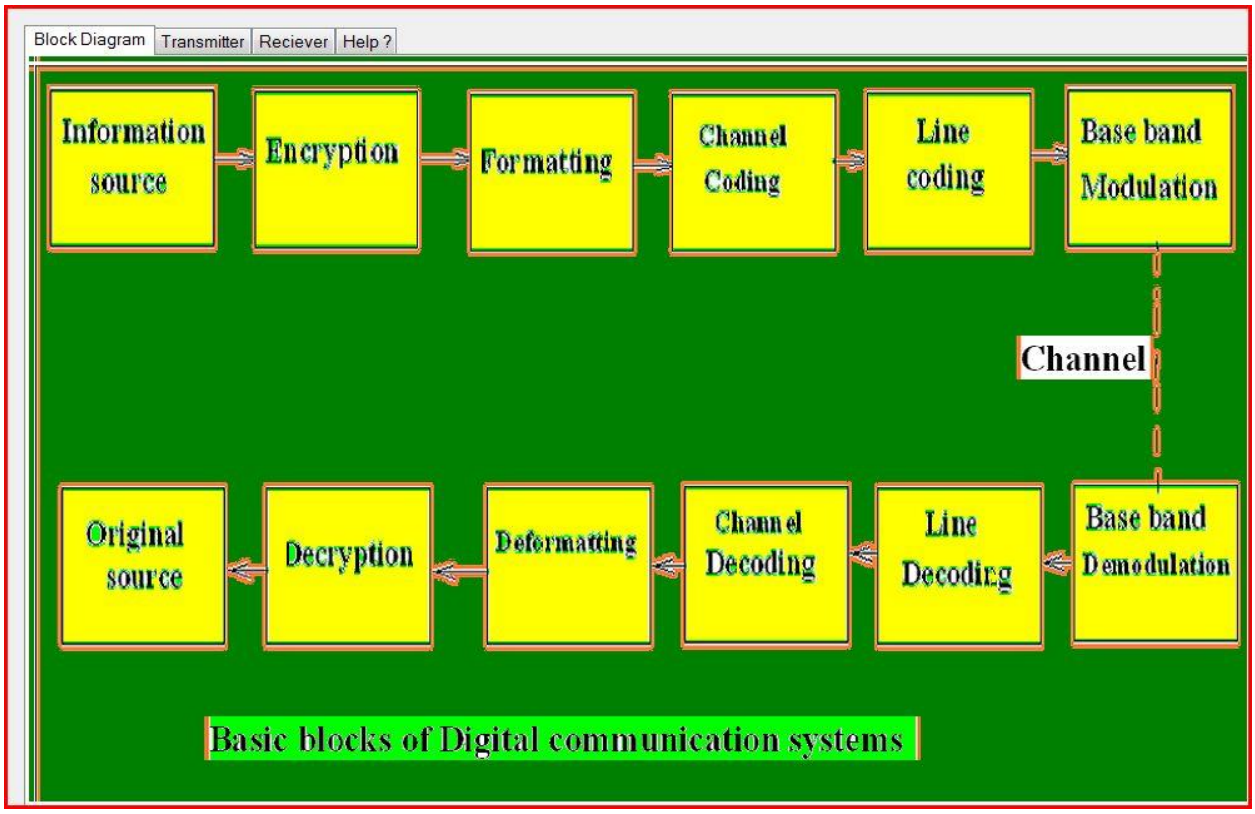

Fig. 1 block diagram part of the GUI

Transmitter side input and output parameters

In the transmitter side we have blocks of information source, encryption, source encoder, channel encoder, line encoder, and band pass modulation. In the information source block, we have combinational box used to select the input parameter. It includes as options like to write text directly, to take text data from stored file and to take audio file. And also, it has text box which help to write the direct input and to show as what we bring from the stored file.

The encryption block takes the plain text for textual data and array of integers for audio file from the source information block. Inside the block there is combinational box which helps the user to select among the different types of encryption techniques. There are also to text box which helps to enter the user the additive key and multiplicative key. The output of the encrypted data is displayed in the text box found inside the block prepared for that.

The source encoder block accepts its input from the encryption block. This block does not accept any input from the user. Only clicking the source encoder block is expecting from the user. The source encoded binary output is displayed in the text box in side that block. The channel encoding block consist of combinational box to select the different methods of channel encoding and one text box to display the channel encoded binary data. This block accepts input data from the source encoder block.

The line coding block accepts its input from the output of the channel encoder block. And it has one combinational box which helps to the user to select among the different types of line coding. The line coded output graph is displayed in the display box found in that block. The band pass block accepts some of its inputs from the user and the other from the channel encoder and converts to ON-Off signal before it uses as input to the modulator. This block has combinational box which helps to select the user one of the band pass modulation types and also it has two text boxes that helps the user to enter the frequencies $f_1, and\ f_2$. The modulated out put is displayed in the display box found in that block. This time the operation of the transmitter is completed and the data is sent to the receiver.

Fig. 2 transmitter part of the GUI

Receiver side input and output parameters

In the receiver side we have blocks which perform the reverse of the blocks in the transmitter. The bocks are the pass band demodulation, line decoding/ detection, channel decoder, source encoder and decryption. In the following paragraphs we are going to see the parameters in each block.

The first block in the receiver side is the pass band demodulator. This block does not accept any input from the user. When we develop the software, we make it to take the selected modulation type and frequency entered by the user at the transmitter side. Because the user may forget what he chooses and entered in the transmitter side and also the modulation type and frequency should be compatible.

The line decoder block consists only the display box to display the received line coded data so that to decode in to binary equivalent. The line decoded/ detected binary data are displayed in a block called detected data.

The channel decoder block accepts the detected data from the detector block and checks for error and if there is correctable error it corrects. The corresponding channel decoder type with the encoder that we use at the transmitter is linked internally when the software developed.
The channel decoder gives two outputs called the corrected code word and the corrected massage bits. Then the corrected message bits will be used as input to the source decoder.

The source decoder block accepts the corrected input from the channel decoder and converts it back to the corresponding asci. If the information source was textual it transforms the asci to corresponding characters but if it was audio file it simply keeps it as array of integers. The output of the source decoder is displayed in the text box found inside the source decoder block.

The decryption block accepts the encrypted data from the source encoder block. This block in order to perform the decryption process it needs the inverse keys for all types of encryption methods. But normally while we are developing the software we make an internal link between the transmitter and the receiver in order to take the encryption type used at the transmitter in the receiver directly. Also, the user should have to inter the encryption key that he/she use at the transmitter side and the system by itself will find the inverse key because it is difficult to calculate the inverse key by our hands. After decryption, if the information source were audio file it will be written back in to its form using the wav writer. But if the information source were textual it will be stored in one of storage places.

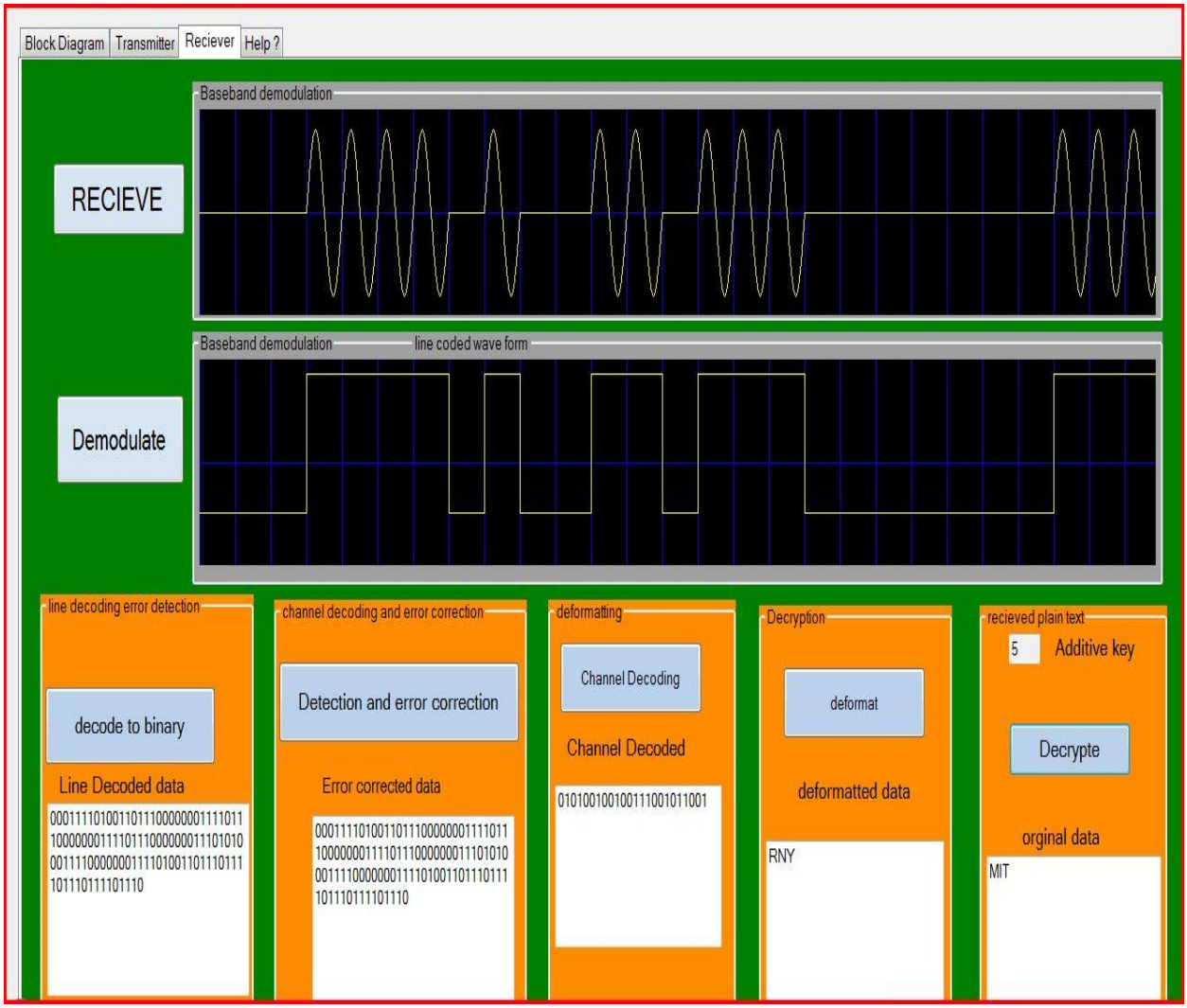

Fig. 3 receiver part of the GUI

5. Conclusion and recommendation

5.1 Conclusion

In this project many digital communication system applications are used to develop the entire kit. Text and audio inputs are taken, then this inputs are separately encrypted with different encryption techniques like additive cipher, multiplicative cipher and affine ciphers, encrypted data is converted in to an 8 bit binary, channel encoded with distinct channel coding styles like linear block encoder, cyclic encoder and convolutional encoder, line coded with RZ, RZ_L, unipolar RZ, AMI, bipolar RZ and band pass modulated by ASK, FSK and PSK at the transmitter side and at the receiver side, band pass demodulation, line decoding, channel decoding, reformatting and decryption corresponding to the transmitter side were taken. All the results found are correct according to the theoretical analysis.

5.2 Recommendation

Further modification of this project is possible by adding some features like video input may be taken. And it is also possible to use advanced encryption techniques to make the data transmission more secure than ever. It is also good to add spread spectrum to this project to make it less prone to noise. Another channel coding styles like **BCH** may also be used to see how the different channel coding methods differ and see which is more secure to use and which is spectral efficient. The (7, 4) linear block encoder and cyclic encoder may also be improved to correct more than one error by increasing the values of k and n compromising system complexity.

References

[1] Haykin, S. S., Moher, M., & Song, T. (1989). An introduction to analog and digital communications (Vol. 1). New York: Wiley.

[2] Kattoush, A. (2005) . Digital communication. Dar Al-Manahej for Pub. & Distributing, Amman.

[3] J. Li and K. Narayanan, “Rate-Compatible Low-Density Parity-Check Codes for Capacity-Approaching ARQ Scheme in Packet Data Communications," Int. Conf. on Comm., Internet, and Info. Tech. (CIIT), November 2002.

[4] Farahin Bt Tajul Arifin, N., Othman, N. S., & Ahmed, A. M. (2014, November). Overcomplete source expansion aided, soft-bit assisted speech transceiver. In Telecommunication Technologies (ISTT), 2014 IEEE 2nd International Symposium on (pp. 326- 329). IEEE.

[5] Nandan, S., & Deepthi, P. P. (2012, November). Joint Source Channel Coding Using LDPC Codes. In Computational Intelligence and Communication Networks (CICN), 2012 Fourth International Conference on(pp. 355-358). IEEE.

[6] Carlson, A. B., Crilly, P. B., & Rutledge, J. C. (1986). Communication systems: an introduction to signals and noise in electrical communication.Guía Académica, 129.

[7] Ali, K. (2005). An Enhanced Joint Source-Channel Decoder (Doctoral dissertation, McGill University). [8] Modestino, J. W., & Daut, D. G. (1979). Combined source-channel coding of images. Communications, IEEE Transactions on, 27(11), 1644-1659.

[9] Kaul, V. (2012). The Digital Communications Revolution. Online Journal of Communication and Media Technologies, 2(3), 113. [15] Kahn, R. E., & Cerf, V. G. (1999). What is the Internet (And What makes it Work)?