**INTRODUCTION**

**1.1 INTRODUCTION**

In this fast-paced world, we need data at rates faster than ever. Challenges are not to develop such a system instead to handle the problem aroused due to attenuation and speed. However greater problem is the error in the information. LDPC codes were invented by Robert Gallager in 1960 at MIT. A great deal of research effort has been expended to the design construction, encoding, decoding, performance analysis and applications. Our project concentrates on a rather simplistic approach on coding and decoding LDPC codes. It aims to make LDPC codes less complex and how to implement is hardware using Raspberry Pi 3 Model B+.

Main features of LDPC codes are as follows:

* Near Shannon limit performance.
* Simple decoding algorithms.
* Low decoding complexity.
* Flexible in choice of parameter.
* Amenable to rigorous analysis.

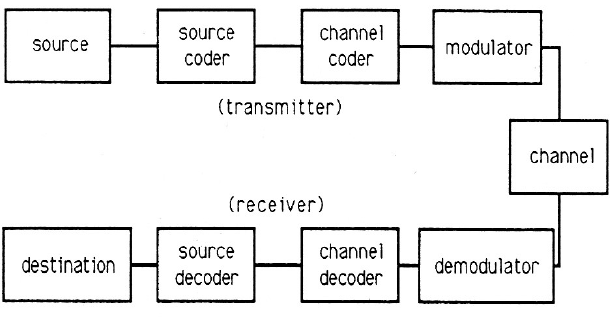
Earlier work on LDPC design concentrates on how to decode and encode LCPC codes in the most optimize manner to make them applicable for real time network use. This project explores the process as well as feasibility of implementation of the design of LDPC system using Raspberry Pi 3 Model B+.

**1.2 General Block Diagram of Digital Communication.**

Analog signals can be transmitted directly via carrier modulation over the communication channel and demodulated accordingly at the receiver. We call such a communication system an analog communication system. Alternatively, an analog source output may be converted in to a digital form and the message can be transmitted via digital modulation and demodulated as a digital signal at the receiver. There are some potential advantages to transmitting an analog signal by means of digital modulation. The most important reason is that signal fidelity is better controlled through digital transmission than analog transmission.

1

In particular, digital transmission allows us to regenerate the digital signal in long-distance transmission, thus eliminating effects of noise at each regeneration point. In contrast, the noise added in analog transmission is amplified along with the signal when amplifiers are used periodically to boost the signal level in long-distance transmission. Another reason for choosing digital transmission over analog is that the analog message signal may be highly redundant. With digital processing, redundancy may be removed prior to modulation, thus conserving channel bandwidth. Yet a third reason may be that digital communication systems are often cheaper to implement. In a digital communication system, the functional operations performed at the transmitter and receiver must be expanded to include message signal discretization at the transmitter and message signal synthesis or interpolation at the receiver. Additional functions include redundancy removal, and channel coding and decoding.



In a digital communication system, the messages produced by the source are usually converted into a sequence of binary digits. Ideally, we would like to represent the source output (message) by as few binary digits as possible. In other words, we seek an efficient representation of the source output that results in little or no redundancy. The process of efficiently converting the output of either an analog or a digital source into a sequence of binary digits is called source encoding or data compression.

2

The sequence of binary digits from the source encoder, which we call the information sequence, is passed to the channel encoder. The purpose of the channel encoder is to introduce, in a controlled manner, some redundancy in the binary information sequence which 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 effect, redundancy in the information sequence aids the receiver in decoding the desired information sequence. For example, a (trivial) form of encoding of the binary information sequence is simply to repeat each binary digit m times, where m is some positive integer. More sophisticated (nontrivial) encoding involves taking k information bits at a time and mapping each k-bit sequence into a unique n-bit sequence, called a codeword. The amount of redundancy introduced by encoding the data in this manner is measured by the ratio n/k. The reciprocal of this ratio, namely, k/n, is called the rate of the code or, simply, the code rate.

All the encoding and decoding part that we are going to do in our project is simply a channel coding.

**1.3 Fundamentals of LDPC Codes**

LDPC codes are the class of linear block codes. The name comes from the characteristics of their parity check matrix which contains only a few 1’s in comparison to the amount of 0’s. Their main advantage is that they provide performance which is very close to the capacity of a lot of different channels and their complex algorithms for decoding which was first introduced by Gallager in his PhD thesis in 1960. Due to the computational efforts of the implementation coder and encoder for such codes and the introduction of Reed-Solomon codes, mostly ignored until ten years ago.

The biggest difference between LDPC codes and classical block codes is how they are decoded. Classical block codes are generally decoded with ML like decoding algorithms and so are usually short and designed algebraically to make this task less complex. LDPC codes however are decoded iteratively using graphical representation of their parity check matrix which we are going to show in the upcoming chapters.

3

**1.4 Objective**

In communication system, forward error correction codes have been widely used to battle data errors caused by transmission through a noisy channel. By adding extra bits to the end of message bits, to the end of message bits, a certain number of bit errors can be detected and corrected without a frequent retransmission. Low Density Parity Check (LDPC) is a powerful FEC coding scheme which can achieve good error performance under very low signal to noise ratio. A communication system utilizing LDPC codes is able to get very close to the channel capacity. A communication system utilizing LDPC code is able to get very close to the channel capacity limit established by Claude Shannon in the 1940’s. In addition, LDPC codes have lower complexity in the decoding process compared to the FEC codes. With advances in the computing power, LDPC codes have been adopted in many high-speed communication standards such as digital video broadcasting.

**1.5 Methodology**

The project aims to explore the actual hardware implementation of the encoding and use of Raspberry Pi 3 Model B+ in the decoding using LDPC bit flipping scheme. Decoding algorithm requires large number of iterations to be performed in a short time. For this reason, Raspberry Pi 3 Model B+ is used.

**1.6 Project Description**

This project was initially broken down into four phases.

* In phase one, the goal was to gain the knowledge of different channel coding schemes and their implementation.
* In phase two, the goal was to gain the knowledge and good understanding of LDPC codes and decoding techniques through various research and studies.
* In phase three, the goal is to implement the actual circuit for encoding part using a random H matrix. The entire Graphical User Interface (GUI) software and encoding and decoding code was developed using Python on Spyder Integrated Development Environment (IDE).
* In phase four, the ultimate goal is to assemble circuit using Raspberry Pi 3 and RF Transmitter and Receiver and to integrate software and hardware.

4

**2. CHANNEL CODING**

**2.1 ERROR CONTROL METHODS**

The purpose of channel coding/decoding is to detect and correct errors in noisy channels. In a noisy channel errors may occur during the transmission of data from information source to destination, so we need a method to detect these errors and then correct them. Channel coding deals with error control techniques. If the data at the output of a communications system has errors that are too frequent for the desired use, the errors can often be reduced by the use of a number of techniques. Coding permits an increased rate of information transfer at a fixed error rate, or a reduced error rate for a fixed transfer rate.

The two main methods of error control are:

* **Automatic Repeat Request (ARQ)** when a receiver circuit detects errors in a block of data, it requests that the data is retransmitted.
* **Forward Error Correction (FEC)** the transmitted data is encoded so that the data can correct as well as detect errors caused by channel noise.

The choice of ARQ or FEC depends on the particular application. ARQ is often used where there is a full duplex (2-way) channel because it is relatively inexpensive to implement. FEC is used where the channel is not full duplex or where ARQ is not desirable because real time is required.

**2.2 CATEGORIES OF CHANNEL CODES**

The two main categories of channel codes are:

**2.2.1 Block codes:**

A block of k information bits is encoded to give a codeword of n bits (n>k). For each sequence of k information bits, there is a distinct codeword of n bits. Examples of block codes include Hamming Codes and Cyclic Codes. A Cyclic Redundancy Check (CRC) code can detect any error burst up to the length of the CRC code itself.

5

**2.2.2 Convolutional Codes**:

The coded sequence of n bits depends not only on the present k information bits but also on the previous information bits.

The primary objective of coding is that the decoder can determine if the received word is a valid codeword, or if it is a codeword which has been corrupted by noise (i.e. detect one or more errors). Ideally the decoder should be able to decide which codeword was sent even if the transmitted codeword was corrupted by noise (i.e. error correction).

The block coder input is a stream of information bits. The coder segments this bit stream into blocks of k information bits and for each block it calculates a number of r check bits, or it picks the r check bits from a tabulated set of values. It then transmits the entire block, or codeword of n = k + r channel bits. This is called an (n, k) block code.

If errors occur in suﬃciently few of these transmitted channel bits, the r check bits may provide the receiver with suﬃcient information to enable it to detect and/or correct the channel errors.

The code eﬃciency (or code rate) is k/n.

If the k information bits are transmitted unaltered first followed by the transmission of the r check bits it is called a systematic code.

A non-systematic block code is one which has the check bits interspersed between the information bits.

6

**2.3 HAMMING DISTANCE:**

The minimum number of positions in which any two code words in any particular block diﬀer from each other is called the Hamming distance, dmin. Consider the following set of code words:

C1 =0000,

C2 =0101,

C3 =1010,

C4 =1111.

The distance, d: – between C2 and C3 is4 – between C2 and C4 is 2 – between C3 and C2 is 2. The Hamming distance (the smallest distance between any pair) = dmin = 2. The distance between any code words and C1 is equal to its weight i.e. the number of 1s in the codeword. For example, the weight of C4 is 4 and the distance from C1 to C4 is 4.

The Hamming distance is important:

* If a received codeword has e errors, then provided that e ≤ dmin −1, it is possible to detect with certainty that errors have occurred.
* If a received codeword has e errors, then provided that 2e +1 ≤ dmin, it is possible to detect the errors and repair them to regenerate the original codeword. To repair a single error requires a Hamming distance of 3 or greater.

The 4-bit code in the example above, therefore, cannot correct the result of occurrence of a single error, (since dmin =2), but it can detect it.

7

Simplest types of block codes are:

**2.3.1 Repetition codes:**

One way to detect an error in an information block is to send the information twice. The two received blocks are compared bit by bit and if there is a diﬀerence an error has occurred. This method can be extended to send the information thrice. This type of block code is called as repetition code. This method is not efficient because it requires sending the information multiple times irrespective of the occurrence of error.

**2.3.2 Single Parity Check Codes:**

It appends a parity check bit to the end of the information bits. This check bit is the modulo – 2 sum (modulo – 2 addition is equivalent to the exclusive OR logical operation) of the codeword (n − 1) information bits. If the number of 1s in the information word is even, then the modulo – 2 sum of all the information bits will be equal to 0. If the number of 1s in the information word is odd their modulo – 2 sum will be equal to 1. The parity bit is calculated and appended to the information bits to form the codeword. Even parity means that the parity bit is set so that the total number of 1s in the codeword is even. Odd parity means that the total number of 1s in the codeword must be odd. This type of code can only detect and cannot correct errors.

**2.4 LINEAR BLOCK CODE:**

An (n, k) linear block code is a k – dimensional subspace of the n – dimensional vector space

Vn = {(c0, c1, …, cn−1)|∀cj, cj ∈ GF(2)};

n is called the length of the code, k the dimension.

Example: a (6, 3) code C = {000000, 100110, 010101, 001011, 110011, 101101, 011110, 111000}

8

**2.5 GENERATOR MATRIX:**

An (n, k) LBC can be specified by any set of k linear independent codewords c0, c1, ..., ck−1. If we arrange the k codewords into a k × n matrix G, G is called a generator matrix for C.

Let u = (u0, u1... uk−1), where uj ∈ GF (2). c = (c0, c1... cn−1) = uG.

A parity check for C is an equation of the form

a0c0 ⊕ a1c1 ⊕...⊕ an−1cn−1 =0,

which is satisfied for any c = (c0, c1... cn−1) ∈ C.

The collection of all vectors a = (a0, a1,..., an−1) forms a subspace of Vn.

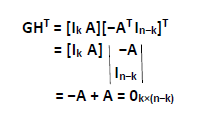
It is denoted by C⊥ and is called the dual code of C. The dimension of C⊥ is n−k and C⊥ is an (n, n−k) BLBC. Any generator matrix of C⊥ is parity check matrix for C and is denoted by H.

cHT = 01×(n−k) for any c ∈ C.

Let G = [ Ik A ].

Since cHT = uGHT = 0, GHT must be 0.

If H = [−AT In−k], then



Thus, the above H is parity check matrix.

Let c be the transmitted codeword and y is the binary received vector after quantization. The vector e = c ⊕ y is called an error pattern.

Let y = c ⊕ e. s = yHT = (c ⊕ e) HT = eHT which is called the syndrome of y.

9

**3. LDPC**

**3.1 Introduction**

In information theory, a low-density parity-check (LDPC) code is a linear error correcting code, a method of transmitting a message over a noisy transmission channel. An LDPC is constructed using a sparse bipartite graph. LDPC codes are capacity-approaching codes, which means that practical constructions exist that allow the noise threshold to be set very close to the theoretical maximum (the Shannon limit) for a symmetric memory less channel. The noise threshold defines an upper bound for the channel noise, up to which the probability of lost information can be made as small as desired.

**3.1.1 Error Correcting Code**

In telecommunication, information theory, and coding theory, forward error correction (FEC) or channel coding is a technique used for controlling errors in data transmission over unreliable or noisy communication channels. The central idea is the sender encodes the message in a redundant way by using an error-correcting code (ECC). The redundancy allows the receiver to detect a limited number of errors that may occur anywhere in the message, and often to correct these errors without retransmission. FEC gives the receiver the ability to correct errors without needing a reverse channel to request retransmission of data, but at the cost of a fixed, higher forward channel bandwidth. FEC is therefore applied in situations where retransmissions are costly or impossible, such as one-way communication links and when transmitting to multiple receivers in multicast. FEC information is usually added to mass storage devices to enable recovery of corrupted data, and is widely used in modems.

10

**3.2 Representation of LDPC Codes:**

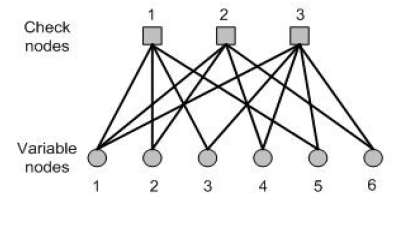
**3.2.1 Matrix Representation:**

A low-density parity-check code is a linear block code given by the null space of an m X n parity check matrix H that has a low density of 1s. A regular LDPC code is a linear block code whose parity check matrix H has a column weight g and row weight r, where r = g(n/m) and g<<m. If H is low density, but its row and column weight are not both constant, then the code is an irregular LDPC code. An additional structural property on H: no two rows (or two columns) have more than one position in common that contains a non-zero element. This property is called the row-column constraint, or simply, RC constraint. The code rate R for a regular LDPC code is bounded as

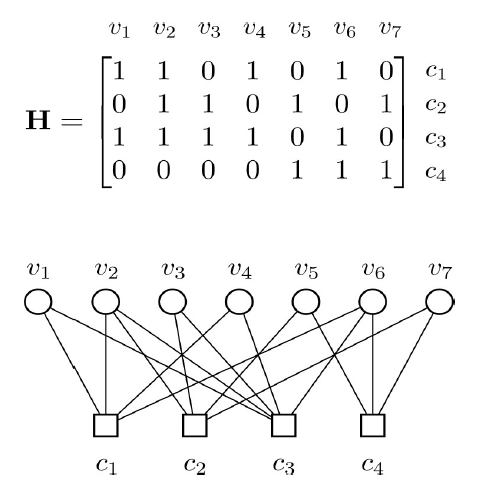
R ≥ 1 – (m/n) = 1 – (g/r)

**3.2.2 Graphical Representation:**

The Tanner graph of an LDPC code is analogous to the trellis of a convolutional code in that it provides a complete representation of the code and it aids in the description of decoding algorithms. A Tanner graph is a bipartite graph, that is, a graph whose nodes may be separated into two types, with edges connecting only nodes of different types. The two types of nodes on Tanner graph are the variable nodes and the check nodes. The Tanner graph of a code is drawn as follows: CN I is connected to VN j whenever element hij in H is 1. There are m CNs in a Tanner graph, one for each equation, and n columns of H specify the n VN connections.



11



**3.3 CLASSIFICATION OF LDPC CODES:**

The original LDPC codes are random in the sense that their parity- check matrices possess little structure. This is problematic in that both encoding and decoding become quite complex when the code possesses no structure beyond a linear code. More recently, LDPC codes with structure have been constructed.

12

**3.3.1 Cyclic Codes:**

The encoder of a cyclic code consists of a single length (n - k) shift register, some binary adders, and a gate. The nominal parity-check matrix H of a cyclic code is an n × n circulant; that is, each row is a cyclic-shift of the one above it, with the first row a cyclic-shift of the last row. The implication of a sparse circulant matrix H for LDPC decoder complexity is substantial: because each check equation is closely related to its predecessor and its successor, implementation can be vastly simplified compared with the case of random sparse H matrices for which wires are randomly routed. However, besides being regular, a drawback of cyclic LDPC codes is that the nominal H matrix is n × n, independently of the code rate, implying a more complex decoder. Another drawback is that the known cyclic LDPC codes tend to have large row weights, which makes decoder implementation tricky.

**3.3.2 Quasi-cyclic codes**:

Quasi-cyclic (QC) codes also possess tremendous structure, leading to simplified encoder and decoder designs. Additionally, they permit more flexibility in code design, particularly irregularity, and, hence, lead to improved codes relative to cyclic LDPC codes.

**3.3.3 Random:**

This type of codes possesses no structure and the bits are arranged in random manner in the entirety of the matrix.

**3.4 LDPC ENCODER:**

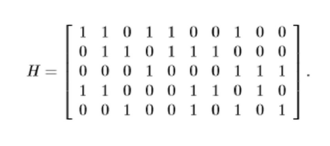
A generator matrix for a code with parity-check matrix H can be found by performing Gauss-Jordan elimination on H to obtain it in the form

H = [A, In−k]

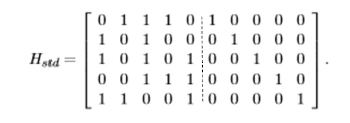
where A is a (n − k) × k binary matrix and In−k is the size n − k identity matrix. The generator matrix is then

G = [Ik, AT].

13



The procedure to get the H in the required format requires it to be put in row-echelon form. It needs to be done so that in any two successive rows that do not consist entirely of zeros, the leading 1 in the lower row occurs further to the right than the leading 1 in the higher row. The matrix H is put into this form by applying elementary row operations in GF (2), which are; interchanging two rows or adding one row to another modulo 2.



We can then derive the generator matrix G from the H matrix we obtained from the elementary row operations.



14

**3.5 LDPC DECODER:**

The class of decoding algorithms used to decode LDPC codes is collectively termed message-passing algorithms since their operation can be explained by the passing of messages along the edges of a Tanner graph. Each Tanner graph node works in isolation, only having access to the information contained in the messages on the edges connected to it. The message-passing algorithms are also known as iterative decoding algorithms as the messages pass back and forward between the bit and check nodes iteratively until a result is achieved (or the process halted). Different message-passing algorithms are named for the type of messages passed or for the type of operation performed at the nodes. In some algorithms, such as bit-flipping decoding, the messages are binary and in others, such as belief propagation decoding, the messages are probabilities which represent a level of belief about the value of the code word bits. It is often convenient to represent probability values as log likelihood ratios, and when this is done belief propagation decoding is often called sum-product decoding since the use of log likelihood ratios allows the calculations at the bit and check nodes to be computed using sum and product operations.

15

**3.5.1 Bit-flipping decoding:**

The bit-flipping algorithm is a hard-decision message-passing algorithm for LDPC codes. A binary (hard) decision about each received bit is made by the detector and this is passed to the decoder. For the bit-flipping algorithm the messages passed along the Tanner graph edges are also binary: a bit node sends a message declaring if it is a one or a zero, and each check node sends a message to each connected bit node, declaring what value the bit is based on the information available to the check node. The check node determines that its parity-check equation is satisfied if the modulo-2 sum of the incoming bit values is zero. If the majority of the messages received by a bit node are different from its received value the bit node changes (flips) its current value. This process is repeated until all of the parity-check equations are satisfied, or until some maximum number of decoder iterations has passed and the decoder gives up.

**3.5.2 Sum-product decoding:**

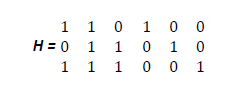
The sum-product algorithm is a soft decision message-passing algorithm. It is similar to the bit-flipping algorithm described in the previous section, but with the messages representing each decision are now probabilities. Whereas bit-flipping decoding accepts an initial hard decision on the received bits as input, the sum-product algorithm is a soft decision algorithm which accepts the probability of each received bit as input. The input bit probabilities are called the a priori probabilities for the received bits because they were known in advance before running the LDPC decoder. The bit probabilities returned by the decoder are called the posteriori probabilities. In the case of sum-product decoding these probabilities are expressed as log-likelihood ratios.

16

**4. TRANSMITTING CIRCUIT HARDWARE**

**4.1 Introduction**

The encoding method of LDPC codes is just similar to that of linear block coding. Thus, the encoding method which we have implemented here is similar to that if linear block encoding. Considering hardware requirements in order to generate a codeword only XOR gates are required. The H matrix which we have used in order to generate a codeword is given as



The matrix H is called as parity check matrix. Each row of H corresponds to a parity-check equation and each column of H corresponds to a bit in the codeword. Thus, for a binary code with m parity-check constraints and length n codeword the parity-check matrix is an m x n binary matrix. The received codeword Y = [c1 c2 c3 c4 c5 c6] is said to be an error free codeword if it satisfies the following equation

HYT= 0

In this example the codeword bits c1, c2, and c3 contain the three-bit message, c1, c2, and c3, while the codeword bits c4, c5 and c6 contain the three parity-check bits. Written this way the codeword constraints show how to encode the message. From our knowledge of previous sections, we can easily generate G matrix from the given H matrix.

Codeword is given as [c1 c2 c3 c4 c5 c6] = [c1 c2 c3] • G

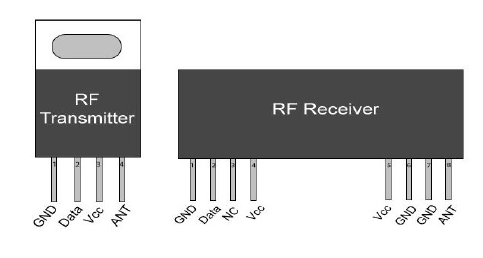
17

**4.3 RF MODULE:**

An RF module (radio frequency module) is a (usually) small electronic device used to transmit and/or receive radio signals between two devices. In an embedded system it is often desirable to communicate with another device wirelessly. This wireless communication may be accomplished through optical communication or through radio frequency (RF) communication. For many applications the medium of choice is RF since it does not require line of sight. RF communications incorporate a transmitter or receiver.

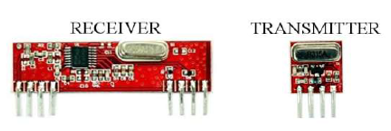
RF modules are widely used in electronic design owing to the difficulty of designing radio circuitry. Good electronic radio design is notoriously complex because of the sensitivity of radio circuits and the accuracy of components and layouts required achieving operation on a specific frequency. In addition, reliable RF communication circuit requires careful monitoring of the manufacturing process to ensure that the RF performance is not adversely affected. Finally, radio circuits are usually subject to limits on radiated emissions, and require Conformance testing and certification by a standardization organization such as ETSI or the U.S. Federal Communications Commission (FCC).

RF modules are most often used in medium and low volume products for consumer applications such as garage door openers, wireless alarm systems, industrial remote controls, smart sensor applications, and wireless home automation systems. They are sometimes used to replace older infrared communication designs as they have the advantage of not requiring line-of-sight operation. Several carrier frequencies are commonly used in commercially available RF modules, including those in the industrial, scientific and medical (ISM) radio bands such as 433.92 MHz, 915 MHz, and 2400 MHz.



18

In this project we have used two types of RF modules. RF transmitter to transmit data serially and RF receiver to receive data serially.



**4.3.1 RF Transmitter**

An RF transmitter module is a small PCB sub-assembly capable of transmitting a radio wave and modulating that wave to carry data. Transmitter modules are usually implemented alongside a micro controller which will provide data to the module which can be transmitted. RF transmitters are usually subject to regulatory requirements which dictate the maximum allowable transmitter power output, harmonics, and band edge requirements.

**Pin description:**

Following is the pin configuration which we have implemented in our project to transmit the data

**GND**: Connected to ground

**DATA**: Encoded data which is converted into a serial data by using parallel to serial converter IC.

**VCC**: Supply voltage of 5V.

**ANT**: Antenna output pin.



19

**4.3.2 RF receiver**

An RF receiver module receives the modulated RF signal, and demodulates it. There are two types of RF receiver modules: super heterodyne receivers and super-regenerative receivers. Super-regenerative modules are usually low cost and low power designs using a series of amplifiers to extract modulated data from a carrier wave. Super-regenerative modules are generally imprecise as their frequency of operation varies considerably with temperature and power supply voltage. Super heterodyne receivers have a performance advantage over super-regenerative; they offer increased accuracy and stability over a large voltage and temperature range. This stability comes from a fixed crystal design which in the past tended to mean a comparatively more expensive product. However, advances in receiver chip design now mean that currently there is little price difference between super heterodyne and super-regenerative receiver modules.

**Pin description:**

**GND**: There are 3 ground pins in receiver module. All of these pins are connected to the ground.

**DATA**: This is the pin where serial data which has been transmitted from the receiver is available

**VCC**: Supply voltage of 5V.

**ANT**: Antenna output pin.



20

**4.7 CIRCUIT DESIGN**

**4.8 ENCODER USING RASPBERRY PI 3 MODEL B+**

**5 DECODING HARDWARE**

**5.1 RF RECEIVER MODULE**

The pin description of the RF receiver module is already explained in the last chapter. The output is obtained at the data out pin of RF receiver module. The major problem corresponding with this is the value data output voltage even if it is not receiving anything. That value varies somewhere between 2V – 2.5V. But at the transmitter side we have used the voltage level nearly of 5V. Thus, by using a proper decoding technique this can also be compensated. Receiver code makes the use of first two bits of transmitted message for the synchronization. This helps in deciding a proper instant at which we can start the data receiving.

**6. RESULT AND LIMITATIONS**

**6.1 RESULT**

We intent to simplify the encoding and decoding process swift and simple. The system developed is highly reliable, fair and cost effective also saves time and easy to understand. The system we designed successfully encodes and decodes information using LDPC technique.

**Text to Binary**

def BreakingMessage(binmess, k, messlen):

i = 0

messarr = []

while messlen >= k :

messarr.append(binmess[i:i+k])

i = i+k

messlen = messlen-k

if(messlen > 0):

messarr.append(binmess[i:])

return messarr

**Basic GUI**

window = tkinter.Tk()

window.title("LDPC Transmitter")

window.geometry("900x700")

label = tkinter.Label(window, text = "Transmitter", font = ('arial', 25, 'bold'), pady=10).pack()

messlab = tkinter.Label(window, text = "Enter the message to be transmitted:", font = ('arial', 20, 'bold'), pady=5).pack()

mess = tkinter.Entry(window, font = ('arial', 25, 'bold'), width = 20, bg = "#ccc", bd = 3)

mess.pack()

tkinter.Button(window, text = "SEND IT", command = calhmat, bd=10, font = ('ubuntu', 15, 'bold'), pady=5).pack()

window.mainloop()

**Encoding**

def calhmat():

binmess = ''.join(format(ord(x), 'b') for x in mess.get())

temp = "\nMessage '"+str(mess.get())+"' in Binary format is: "

tkinter.Label(window, text = temp, font = ('ubuntu', 15, 'bold')).pack()

tkinter.Label(window, text = binmess, font = ('ubuntu', 15, 'bold')).pack()

num = 15 # Number of columns

dv = 4 # Number of ones per column, must be lower than d\_c (because H must have more rows than columns)

dc = 5 # Number of ones per row, must divide n (because if H has m rows: m\*d\_c = n\*d\_v (compute number of ones in H))

# H Matrix

H = pyldpc.RegularH(num,dv,dc)

# G Matrix

tG = pyldpc.CodingMatrix(H)

n,k = tG.shape # n = 15, k = 6

snr = 8

messlen = len(binmess)

messarr = BreakingMessage(binmess, k, messlen)

print("Messarr = ", messarr)

op = CodingMessage(messarr, tG, snr, k)

print(op)

temp2 = "Modulates Message to be sent is :\n "+str(op)

tkinter.Label(window, text = temp2, font = ('ubuntu', 15, 'bold')).pack()

**Code for Parity Check Matrix:**

def RegularH(n,d\_v,d\_c):

‘’’

Builds a regular Parity-Check Matrix H (n,d\_v,d\_c) following Callager's algorithm :

Paramaeters:

n: Number of columns (Same as number of coding bits)

d\_v: number of ones per column (number of parity-check equations including a certain variable)

d\_c: number of ones per row (number of variables participating in a certain parity-check equation);

Errors:

The number of ones in the matrix is the same no matter how we calculate it (rows or columns), therefore, if m is

the number of rows in the matrix:

m\*d\_c = n\*d\_v with m < n (because H is a decoding matrix) => Parameters must verify:

0 - all integer parameters

1 - d\_v < d\_v

2 - d\_c divides n

Returns: 2D-array (shape = (m,n))

‘’’

if n%d\_c:

raise ValueError('d\_c must divide n. Help(RegularH) for more info.')

if d\_c <= d\_v:

raise ValueError('d\_c must be greater than d\_v. Help(RegularH) for more info.')

m = (n\*d\_v)// d\_c

Set=np.zeros((m//d\_v,n),dtype=int)

a=m//d\_v

# Filling the first set with consecutive ones in each row of the set

for i in range(a):

for j in range(i\*d\_c,(i+1)\*d\_c):

Set[i,j]=1

#Create list of Sets and append the first reference set

Sets=[]

Sets.append(Set.tolist())

#Create remaining sets by permutations of the first set's columns:

i=1

for i in range(1,d\_v):

newSet = np.transpose(np.random.permutation(np.transpose(Set))).tolist()

Sets.append(newSet)

#Returns concatenated list of sest:

H = np.concatenate(Sets)

return H

**Code for CodingMatrix:**

def CodingMatrix(MATRIX,use\_sparse=1):

"""

CAUTION: RETURNS tG TRANSPOSED CODING MATRIX.

Function Applies GaussJordan Algorithm on Columns and rows of MATRIX in order

to permute Basis Change matrix using Matrix Equivalence.

Let A be the treated Matrix. refAref the double row reduced echelon Matrix.

refAref has the form:

(e.g) : |1 0 0 0 0 0 ... 0 0 0 0|

|0 1 0 0 0 0 ... 0 0 0 0|

|0 0 0 0 0 0 ... 0 0 0 0|

|0 0 0 1 0 0 ... 0 0 0 0|

|0 0 0 0 0 0 ... 0 0 0 0|

|0 0 0 0 0 0 ... 0 0 0 0|

First, let P1 Q1 invertible matrices: P1.A.Q1 = refAref

We would like to calculate:

P,Q are the square invertible matrices of the appropriate size so that:

P.A.Q = J. Where J is the matrix of the form (having MATRIX's shape):

| I\_p O | where p is MATRIX's rank and I\_p Identity matrix of size p.

| 0 0 |

Therfore, we perform permuations of rows and columns in refAref (same changes

are applied to Q1 in order to get final Q matrix)

NOTE: P IS NOT RETURNED BECAUSE WE DO NOT NEED IT TO SOLVE H.G' = 0

P IS INVERTIBLE, WE GET SIMPLY RID OF IT.

Then

solves: inv(P).J.inv(Q).G' = 0 (1) where inv(P) = P^(-1) and

P.H.Q = J. Help(PJQ) for more info.

Let Y = inv(Q).G', equation becomes J.Y = 0 (2) whilst:

J = | I\_p O | where p is H's rank and I\_p Identity matrix of size p.

| 0 0 |

Knowing that G must have full rank, a solution of (2) is Y = | 0 | Where k = n-p.

| I-k |

Because of rank-nullity theorem.

-----------------

parameters:

H: Parity check matrix.

use\_sparse: (optional, default True): use scipy.sparse format to speed up calculations

---------------

returns:

tG: Transposed Coding Matrix.

"""

H = np.copy(MATRIX)

m,n = H.shape

if m > n:

raise ValueError('MATRIX must have more rows than columns (a parity check matrix)')

if n > 500 and use\_sparse:

sparse = 1

else:

sparse = 0

##### DOUBLE GAUSS-JORDAN:

Href\_colonnes,tQ = GaussJordan(np.transpose(H),1)

Href\_diag = GaussJordan(np.transpose(Href\_colonnes))

Q=np.transpose(tQ)

k = n - sum(Href\_diag.reshape(m\*n))

Y = np.zeros(shape=(n,k)).astype(int)

Y[n-k:,:] = np.identity(k)

if sparse:

Q = csr\_matrix(Q)

Y = csr\_matrix(Y)

tG = BinaryProduct(Q,Y)

return tG

6.2 ENCODER CODE

6.3 DECODER CODE

6.4 LIMITATION

High data rate can’t be achieved because of finite delay generated due for loops becomes significant at high data rate.

It may be possible that the received codeword may resemble to another valid codeword.

Complexity is increased as message size increases.

35

**References**

1. “Raspberry Pi based Wireless Transmission of Text Data using Low Density Parity Check (LDPC)”, IJRSI, Volume V, Issue VI, June 2018, ISSN 2321–2705.
2. Gallager, R.G, “Low Density Parity Check codes”, IRE Transactions on Information Theory Engineering, Vol. IT-8, January 1962.
3. “Low-density parity-check code”, https://en.wikipedia.org/wiki/Low-density-parity-check.
4. “Low Density Parity Check Code” Tutorial by Prof. Adrish Banerjee, IIT Kanpur, NPTEL.
5. “AN INTRODUCTION TO ERROR CORRECTING CODES”, Jack Keil Wolf, ECE 154 C Spring 2010.

36