Atal Bihari Vajpayee
Indian Institute of Information Technology
and Management (ABV-IIITM), Gwalior
(An Institute of National Importance, Ministry of Education, Government of India)

Computer Networking

Project Report

Line Coding

Submitted to:
Dr.Binod Prasad

Team Members:
Aleesha 2021BCS-11
Narra Abhigna 2021BCS-48
Ravi Jwalana 2021BCS-56

Abstract:

Information that has been electronically encoded in a language or format that computers can interpret is referred to as digital data. To put it another way, digital data is information that has been transformed from a binary format to a digital format that is machine-readable. The advantage of digital data is that any analog input, such as results from genome sequencing and straightforward text files, may be represented using binary notation.

Despite the fact that digital signals are sometimes associated with the binary electronic digital systems used in current electronics and computing, digital systems are actually very old and need not be either binary or electronic.

The difficulty we confront with digital data is how to move it from one location to another as data is constantly needed for apparent reasons, including:

1) Decentralisation of data
2) Moving data over a great distance is more effective than moving our physical stores with a tonne more.

Digital transmission has many advantages over analog transmission because there is far less chance of signal corruption during transmission.

To address this, line coding and block coding, which are used to convert digital data into digital signals, are used; in this study, we will only discuss line coding.

Introduction:

Line coding is the process of converting digital data into a form that can be transmitted over a communication channel such as a wire or a fiber-optic cable. It involves representing the binary data as a sequence of electrical or optical signals that can be transmitted over the communication medium.
At the sender, digital data are encoded into a digital signal; at the receiver, the digital data are recreated by decoding the digital signal.

Line coding techniques can be broadly categorised into five types:

1. Unipolar
2. Polar
3. Bipolar
4. Multilevel
5. Multitransition

Line codes are primarily utilised in Ethernet networks, token ring networks, fibre optic data links, magnetic recording systems, satellite communication systems, and pulse code modulation systems.

1 and 0 are used to represent binary data. It can be sent using various pulse kinds. Line coding refers to selecting a certain pair of pulses to represent 1s and 0s. When deciding on the type of line coding, the choice should be made based on:

1) DC level is present or absent;

2) Power spectral density

3) Bandwidth

4) Performance of the BER

5) Transparency

Background Survey:

To achieve low latency, high-performance reliability, and high-speed data transfer, websites, and apps must be supplied in a high-quality manner to as many locations as possible.
Digital-to-digital encoding is the representation of digital information by a digital signal. The process of converting binary 1s and 0s from a computer into a series of voltage pulses that can go along a wire is known as "digital-to-digital line encoding."
Unipolar, polar, bipolar, and Manchester codes are typical line encodings that are used in communications to represent digital data conveyed via a communication channel, such as a transmission medium or data storage device.

Before the subject start, here are some terms to be aware of:
Difference Between a Signal and a Data Element:
A data element, often known as a bit, is the smallest unit that may represent a piece of information. In digital data transmissions, a signal element carries data elements. A signal element is the smallest (temporally) part of a digital signal. Signal elements serve as carriers and carry data components. The quantity of data carried by each signal element is designated as the ratio r.

Data Rate vs. Signal Rate:
How many bits of data are transmitted during an IS depends on the transmission rate. Bits per second (bps) are measured. The signal rate in IS is the total number of signal components transmitted. The unit is the baud.
One goal in data communications is to raise the data rate while lowering the signal rate. While the bandwidth requirement is reduced by lowering the signal rate, the transmission speed is raised by increasing the data rate.
The link between data rate and signal rate can be expressed as 1 S =c xNx 1/r baud where N is the data rate (bps), and c is the case factor, which varies for each case. The signal component amount is denoted by the letter S, and the predefined factor by the letter r.

Bandwidth:
Digital signals that contain information have a nonperiodic bandwidth. We also showed that the bandwidth and range of a nonperiodic signal are continuous. The vast majority of digital signals we encounter in daily life, however, have a limited bandwidth.

Objective:

This study will cover the definition and operation of a few of the line coding techniques used for data transmission. In each approach, we will look at how well it works, its drawbacks, real-world situations where it is used to transmit data, and some examples of data encoded using that technique. This aids in developing a thorough understanding of the line coding techniques we might use to categorize data transmission for various purposes.

The process of line coding converts digital data into digital signals. We assume that bits are stored as data in computer memory as text, numbers, graphic images, audio, or video. Line coding is used to convert a series of bits into a digital signal. At the sender, digital data are encoded into a digital signal, and the digital signal is decoded at the receiver to reproduce the digital data.

Some of the factors we will consider for each line encoding scheme covered in this report to determine how well it can handle these challenges in data transmission and whether or not it is an appropriate strategy to use to achieve our goals are as follows:

Baseline Wandering: When decoding a digital signal, the receiver computes a running average of the received signal power. This average is referred to as the baseline. The data element's value is determined by comparing the incoming signal power to this baseline. A long sequence of Os or 1s may result in baseline drifting, making accurate decoding difficult for the receiver. An effective line coding scheme must be used to prevent baseline wandering.

DC components: When the voltage level in a digital signal remains constant for an extended period of time, the spectrum produces extremely low frequencies (results of Fourier analysis). A system that is unable to pass low frequencies or one that relies on electrical coupling will have problems with these frequencies around zero, which are

referred to as DC (direct-current) components (via a transformer). A telephone connection, for example, cannot transmit frequencies lower than 200 Hz. A long-distance link may also include one or more transformers to electrically separate various sections of the line. For these systems, we require a scheme that does not include a DC component.

Self-Synchronization: In order for the signals received from the sender to be correctly decoded, the receiver's bit intervals must exactly match the sender's bit intervals. The bit intervals are not matched regardless of whether the receiver clock is faster or slower, which may cause the receiver to misinterpret the signals.
Time information is contained in the data communicated in a self-synchronizing digital transmission. This is possible if the signal contains transitions that tell the receiver when the pulse is beginning, halfway through, or finished. If the receiver's clock is out of sync, these sites can reset it.

Built-in Error Detection: It is preferable to have a built-in error-detecting capability in the created code to detect some or all of the errors that occurred during transmission. Some of the encoding techniques we'll discuss can accomplish this to some extent.

Immunity to Noise and Interference: Another desirable code feature is immunity to noise and other interferences. Some of the encoding techniques we'll discuss are capable of doing so.

Complexity: A complicated plan is more expensive to implement than a simple one. A system with four signal levels, for example, is more difficult to comprehend than one with only two levels.

Coding Schemes

Unipolar Schemes-

All signal levels in this scheme are either above or below the axis.

Non-return to zero (NRZ) - In a unipolar line coding scheme, positive voltage designates bit 1 and zero voltage designates bit 0. Because the signal does not return to zero in the middle of the bit, it is known as NRZ.

The benefits of unipolar NRZ include:
• ease of implementation; and
The benefits of unipolar NRZ include:
• ease of implementation; and
• low bandwidth requirements.

The disadvantages of unipolar NRZ are as follows:
• There is a significant DC component, indicating that power is lost due to the transmission line's conductors heating up. Furthermore, because AC-coupled links will reject the DC component of the signal, channel links must be DC-coupled.
• No existing mechanism can insert a clock signal into the line code. Long ones or zeros sequences may cause the receiver to lose synchronisation because there are no voltage transitions.

Polar schemes-

NRZ-L & NRZ-I - These have a slight resemblance to a unipolar NRZ system, but we use two levels of amplitude here (voltages). Binary 1 corresponds to logic-level high, while binary 0 corresponds to logic-level low. The level of the voltage determines the bit value for NRZ-L (NRZ-Level), and for NRZ-I (NRZ-Invert), two-level signals have a transition at a boundary if the next bit to be transmitted is a logical 1, and do not have a transition if the next bit to be transmitted is a logical 0.
Polar NRZ has the following advantages:
• It is relatively simple to implement; and
• It requires a small amount of bandwidth.

The disadvantages of polar NRZ are:
• Although the DC component for polar NRZ-L is negligible if the number of ones and zeros is roughly equal, neither polar NRZ-L nor polar NRZ-I are DC balanced.
• No existing mechanism can insert a clock signal into the line code. Because of the lack of voltage transitions, extended zero sequences in both polar NRZ-L and polar NRZ-I can cause synchronisation loss at the receiver, as can long one sequences in polar NRZ-L.
• When extended sequences of zeros occur, both polar NRZ-L and polar NRZ-I experience baseline wandering (see above). Long sequences of ones can also cause baseline wandering in the polar NRZ-L, where the effect occurs twice as frequently.

Return to zero (RZ) - One approach to solving the NRZ problem is the RZ scheme, which employs the three values positive, negative, and zero. In this system, the middle bit of each signal is set to 0.
Note that in this example, the signal for bit 1 is represented by +V and the other half by zero voltage, whereas the signal for bit 0 is represented by -V and the other half by zero voltage.

Polar RZ has the following advantages:
• It is relatively simple to apply; and
• It uses less electricity than polar NRZ-L or polar NRZ-I.

• Transitions allow the receiver to resynchronize itself during each bit period.

Polar RZ has the following disadvantages:
•It is more intricate than either the polar NRZ-L or the polar NRZ-I.
•Requires twice the bandwidth of either polar NRZ-L or polar NRZ-I.

Biphase Scheme

Manchester and Differential Manchester

The Manchester encoding system is a hybrid of the NRZ-L and RZ (transition at the middle of the bit) schemes. The bit's duration is split into two equal parts. During the first half, the voltage stays at one level while switching to another one during the second. Synchronisation is provided by the transition in the middle of the bit.

The RZ and NRZ-I methods are combined in some ways to create differential Manchester. The bit values are decided at the beginning of the bit, but there is usually a transition in the middle of the bit. There is no transition if the following bit is 1, and there is one if it is 0.

Note – 1. For bit 1, there is a transition from -V to +V volts in the centre of the bit, and for bit 0, there is a transition from +V to -V volts in the middle of the bit, according to the logic we are using to represent data using Manchester.

The advantages of Manchester correlation are:

· There is no baseline wandering, therefore the Manchester scheme solves several NRZ-L difficulties, while differential Manchester solves several NRZ-I problems.

- Since each bit contributes both positive and negative voltage, there is no DC component.

The disadvantages of Manchester correlation are:

· The only restriction on Manchester correlation is that its minimum bandwidth and differential bandwidth are both twice as large as NRZ's.

Bipolar Scheme:

There are three voltage levels in this system: positive, negative, and zero. One of the data elements has a voltage level of zero, while the other has a voltage level that alternates between positive and negative.

Alternate Mark Inversion (AMI)

Binary zero is represented by a neutral zero voltage. Positive and negative voltages alternate to represent binary 1's.

The advantages of bipolar AMI are:

- Simple to implement
- It Uses less power than polar NRZ line coding techniques while maintaining the same signalling rate as other NRZ schemes.
- There is no DC component in the signal.
- Baseline roaming poses no threat.
- Preventing polar ambiguity.

The disadvantages of bipolar AMI are:

- No clock signal is incorporated. Because there are no voltage changes, long zero sequences might make the receiver lose synchronisation.

Pseudoternary -

The opposite of the AMI system, bit 0 is encoded as alternating positive and negative voltages, whereas bit 1 is recorded as zero voltage.

With the sole exception that extended sequences of ones rather than long sequences of zeros can result in the receiver losing synchronisation, the benefits and drawbacks of pseudoternary line coding are identical to those of AMI.

MultiLevel Schemes

Many methods have been developed to increase data speed or reduce bandwidth requirements. The number of bits per baud can be increased by changing a pattern of m data elements into a pattern of n signal elements. A collection of m data items can combine to create 2m various data patterns because there are only two types of data elements (0s and 1s). We can have many signal element kinds by allowing for different signal levels.

If we have L distinct levels, we can combine Ln different signal pattern configurations and if 2m =Ln, then each data pattern becomes a single signal pattern.

If 2m <Ln, signal patterns only make up a subset of data patterns.

To prevent baseline drifting, provide synchronization, and locate transmission errors.

If 2m > Ln then data encoding is not possible because certain data patterns cannot be encoded.

A pattern of m data elements in mBnL schemes, where 2m = Ln, is saved as a pattern of n signal elements.

2B1Q:-

The first mBnL method we discuss, 2B1Q, uses 2-bit data patterns to encrypt 2-bit patterns as one signal component of a 4-level signal. M = 2, n = 1, and L = 4 (quaternary) are the encoding factors used in this process. The normal signal rate of 2B1Q is S = N/4.

As a result, we can deliver data twice as quickly using 2B1Q than we can use NRZ-L. Given that 2B1Q employs four different signal levels, the receiver must be able to distinguish between the four various thresholds. There are expenses related to the reduced bandwidth. Due to the fact that 22 = 41, this approach has no redundant signal patterns. 2B1Q is a part of the DSL (Digital Subscriber Line) technology, which provides high-speed Internet access through subscriber telephone lines.

8B6T:-

The eight binary, six ternary configurations (8B6T) is very interesting. This code is applied when 100BASE-4T cable is used. An 8-bit pattern is

designed to be encoded as a pattern of six ternary (three levels) signal elements. In this type of method, we can have 28 = 256 different data patterns and 36 = 478 different signal patterns. The redundant signal elements for synchronization and error detection are 478-256=222 Redundancy is another method used to balance DC. Each signal pattern's weight is either 0 or +1 DC values. This suggests that there is no pattern for the weight -1. To keep the entire stream DC balanced, the sender keeps an eye on the weight. The first group of weight 1 is provided exactly as it is when two successive groups of weight 1 are encountered, but the second group is totally inverted to generate a weight of -1.

4D-PAMS:-

The third communication method we'll discuss in this section is called (4D-PAM5), which stands for four-dimensional, five-level pulse amplitude modulation. Simultaneous data transmission across four wires is denoted by the acronym 4D.

There are five voltage levels: -2, -1, 0, 1, and 2. In contrast, Level 0 is only used for forward error detection. If the code is simply one dimension, then the four layers result in something that looks like 8B4Q. In other words, an 8-bit word is converted into a signal element with four levels. The worst signal rate for this fictitious one-dimensional form is N X 4/8, or N/2.

MultiLine Schemes:-

MLT-3:-

The multiline transmission, three-level (MLT-3) scheme switches between the levels using three levels (+V, 0, and -V) and three transition rules.

1. If the following bit is 0, there is no transition.
2. If the next bit is 1 and the current level is not 0, the next level is 0.
3. If the next bit is 1 and the current level is 0, the next level is the inverse of the previous non-zero level.

Example: Data=010010110110

The advantages of MLT-3 are:

- It turns out that the signal's structure in this technique aids in lowering the necessary bandwidth.
- Because MLT-3 requires four transitions to complete one full cycle, the maximum fundamental frequency that must be sustained by the transmission medium is just one-fourth of the baud rate, making MLT-3 appropriate for transmission over twisted-pair copper wire connections.

The disadvantages of MLT-3 are:

- The signal complexity is higher, but the signal rate is the same as that for NRZ-I. (three levels and complex transition rules).
- For long strings of zeros ('0'), self-synchronisation is not supported.

CODE: (c++ using graphics.h)

```
#include <iostream>
#include <graphics.h>

#define BLUE 1
#define GREEN 2
#define CYAN 3
#define RED 4
#define MAGENTA 5
#define BROWN 6

using namespace std;

int gd = 5, gm;
int pos_x = 50, pos_y = 100;

string getSignal() {
    string signal;
    cout << "Enter signal to be encoded" << endl;
    cin >> signal;
    return signal;
}

int getEncoding() {
    int choice;
    cout
            << "Choose a type of Encoding\n1) Unipolar NRZ\n2) Polar
NRZ\n3) NRZ Inverted\n4) RZ\n5) Manchester\n6) Differential
Manchester\n7) AMI\n8) PSUEDOTERNARY\n10) MLT"
            << endl;
    cin >> choice;
    return choice;
}

void signalMove(int offset_x, int offset_y) {
    setlinestyle(0,1,5);
    line(pos_x, pos_y, pos_x + offset_x, pos_y + offset_y);
    pos_x = pos_x + offset_x;
    pos_y = pos_y + offset_y;
```

```
}

void signalShift(int new_x, int new_y) {
    setlinestyle(0,1,5);
    line(pos_x, pos_y, new_x, new_y);
    pos_x = new_x;
    pos_y = new_y;
}

void drawAxes() {
    line(0, pos_y, 640, pos_y);
    line(pos_x, 0, pos_x, 200);
}

void initGraph() {
    initgraph(&gd, &gm, nullptr);
    setbkcolor(WHITE);
    drawAxes();
    setcolor(RED);
}

class Encoding {
protected:
    string signal;
    int logic_high = -20;
    int logic_low = 20;
    int base = 0;
    int distance = 20;

public:
    explicit Encoding(string sgn) {
        signal = sgn;
    }
};

class Unipolar_NRZ : Encoding {
private:

    int logic_low = 0;

public:
    explicit Unipolar_NRZ(string sgn) : Encoding(sgn) {};
```

```
    void draw() {
        for (char c: signal) {
            if (c == '1')
                one();
            else
                zero();
        }
    }

    void zero() {
        signalShift(pos_x, 100 + logic_high);
        signalMove(distance, 0);
    }

    void one() {
        signalShift(pos_x, 100 + logic_low);
        signalMove(distance, 0);
    }
};

class Polar_NRZ : Encoding {
public:
    explicit Polar_NRZ(string sgn) : Encoding(sgn) {};

    void draw() {
        for (char c: signal) {
            if (c == '1')
                one();
            else
                zero();
        }
    }

    void zero() {
        signalShift(pos_x, 100 + logic_high);
        signalMove(distance, 0);
    }

    void one() {
        signalShift(pos_x, 100 + logic_low);
        signalMove(distance, 0);
    }
};
```

```
class NRZ_I : Encoding {
public:
    explicit NRZ_I(string sgn) : Encoding(sgn) {};

    void draw() {
        pos_y = 100 + logic_high;
        for (char c: signal) {
            if (c == '1')
                one();
            else
                zero();
        }
    }

    void zero() {
        signalMove(distance, 0);
    }

    void one() {
        if (pos_y == 100 + logic_low)
            signalShift(pos_x, 100 + logic_high);
        else if (pos_y == 100 + logic_high)
            signalShift(pos_x, 100 + logic_low);
        signalMove(distance, 0);
    }
};
class MLT : Encoding {
int PREV=1,PREVPREV=0;
public:
    explicit MLT(string sgn) : Encoding(sgn) {};

    void draw() {
        for (char c: signal) {
            if (c == '1'){
                one(PREV,PREVPREV);
                if(PREV==1)
                {PREV=0;
                PREVPREV=1;
                }
                else if(PREV==0&&PREVPREV==1)
                {PREV=-1;
                PREVPREV=0;
```

```
                }
                else if(PREV==0&&PREVPREV==-1)
                {PREV=1;
                PREVPREV=0;}
                else
                {PREV=0;
                PREVPREV=-1;}
            }
            else
                zero(PREV);
        }
    }

    void zero(int N) {
        if(N==1)
        {signalShift(pos_x, 100+logic_high);
        signalMove(distance, 0);}
        else if(N==-1)
        {signalShift(pos_x,100+logic_low);
        signalMove(distance, 0);
        }
        else
        {
            signalShift(pos_x,100);
            signalMove(distance,0);
        }
    }

    void one(int n,int a) {
        if(n==1&&a==0||n==-1&&a==0)
        {signalShift(pos_x, 100);
        signalMove(distance, 0);
        }
        else if(n==0&&a==1)
        {
        signalShift(pos_x, 100 + logic_low);
        signalMove(distance, 0);
        }
        else if(n==0&&a==-1)
        {
        signalShift(pos_x, 100 + logic_high);
        signalMove(distance, 0);
        }
```

```
    }
};
class AMI : Encoding {
int onesp=0;
public:
    explicit AMI(string sgn) : Encoding(sgn) {};

    void draw() {
        for (char c: signal) {
            if (c == '1'){
                one(onesp);
                onesp++;}
            else
                zero();
        }
    }

    void zero() {
        signalShift(pos_x, 100 );
        signalMove(distance, 0);
    }

    void one(int n) {
        if(n%2==0)
        {signalShift(pos_x, 100 + logic_low);
        signalMove(distance, 0);
        }
        else
        {
            signalShift(pos_x, 100 + logic_high);
        signalMove(distance, 0);
        }
    }
};
class PSUEDOTERNARY : Encoding {
int onesp=0;
public:
    explicit PSUEDOTERNARY(string sgn) : Encoding(sgn) {};

    void draw() {
        for (char c: signal) {
            if (c == '0'){
                one(onesp);
```

```
                onesp++;}
            else
                zero();
        }
    }

    void zero() {
        signalShift(pos_x, 100 );
        signalMove(distance, 0);
    }

    void one(int n) {
        if(n%2==0)
        {signalShift(pos_x, 100 + logic_low);
        signalMove(distance, 0);
        }
        else
        {
            signalShift(pos_x, 100 + logic_high);
        signalMove(distance, 0);
        }
    }
};
class RZ : Encoding {
public:
    explicit RZ(string sgn) : Encoding(sgn) {};

    void draw() {
        for (char c: signal) {
            if (c == '1')
                one();
            else
                zero();
        }
    }

    void zero() {
        signalShift(pos_x, 100 + logic_low);
        signalMove(distance, 0);
        signalShift(pos_x, 100 + base);
        signalMove(distance, 0);
    }
```

```
    void one() {
        signalShift(pos_x, 100 + logic_high);
        signalMove(distance, 0);
        signalShift(pos_x, 100 + base);
        signalMove(distance, 0);
    }
};


class Manchester : Encoding {
public:
    explicit Manchester(string sgn) : Encoding(sgn) {};

    void draw() {
        for (char c: signal) {
            if (c == '1')
                one();
            else
                zero();
        }
    }

    void zero() {
        signalShift(pos_x, 100 + logic_high);
        signalMove(distance, 0);
        signalShift(pos_x, 100 + logic_low);
        signalMove(distance, 0);
    }

    void one() {
        signalShift(pos_x, 100 + logic_low);
        signalMove(distance, 0);
        signalShift(pos_x, 100 + logic_high);
        signalMove(distance, 0);
    }
};

class Diff_Manchester : Encoding {
    int prev = 2;

public:
    explicit Diff_Manchester(string sgn) : Encoding(sgn) {};
```

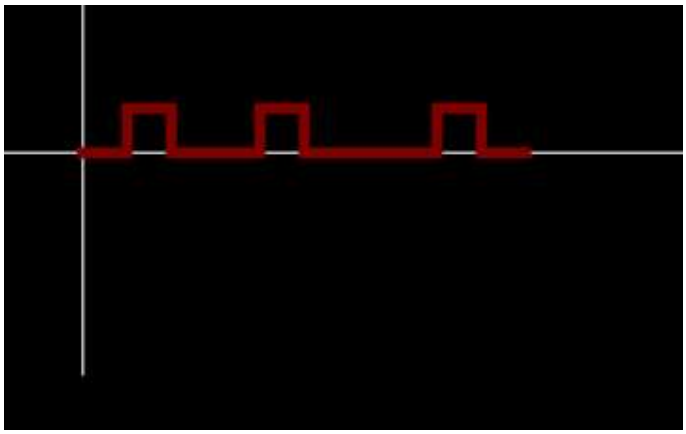
```
    void draw() {
        for (char c: signal) {
            if (c == '1') {
                int n = (prev == 2 ? 1 : 2);
                pattern(n);
                prev = n;
            } else {
                pattern(prev);
            }
        }
    }

    void pattern(int num) {
        if (num == 2) {
            signalShift(pos_x, 100 + logic_low);
            signalMove(distance, 0);
            signalShift(pos_x, 100 + logic_high);
            signalMove(distance, 0);
        } else if (num == 1) {
            signalShift(pos_x, 100 + logic_high);
            signalMove(distance, 0);
            signalShift(pos_x, 100 + logic_low);
            signalMove(distance, 0);
        }
    }
};

int main() {
    string sgn;
    int c;
    sgn = getSignal();
    c = getEncoding();

    initGraph();
    setbkcolor(YELLOW);
    switch (c) {
        case 1: {
            Unipolar_NRZ signal(sgn);
            signal.draw();
            break;
        }
        case 2: {
            Polar_NRZ signal(sgn);
```

```
            signal.draw();
            break;
        }
        case 3: {
            NRZ_I signal(sgn);
            signal.draw();
            break;
        }
        case 4: {
            RZ signal(sgn);
            signal.draw();
            break;
        }
        case 5: {
            Manchester signal(sgn);
            signal.draw();
            break;
        }
        case 6: {
            Diff_Manchester signal(sgn);
            signal.draw();
            break;
        }
        case 7: {
            AMI signal(sgn);
            signal.draw();
            break;
        }
        case 8: {
            PSUEDOTERNARY signal(sgn);
            signal.draw();
            break;
        }
        case 9: {
            MLT signal(sgn);
            signal.draw();
            break;
        }
    }
    getch();
    closegraph();
}
```

OUTPUT:-

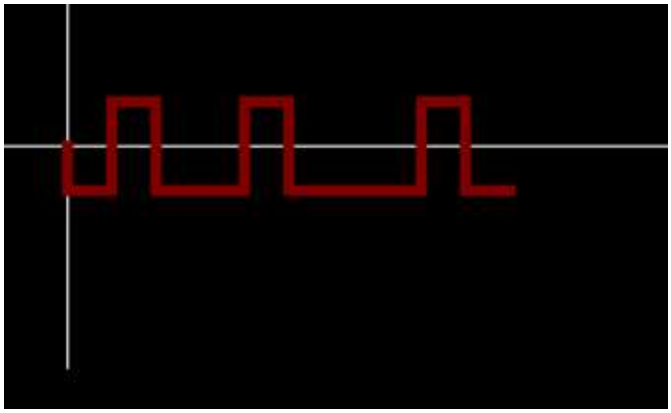
1) Unipolar NRZ

```
Enter signal to be encoded
1011011101
Choose a type of Encoding
1) Unipolar NRZ
2) Polar NRZ
3) NRZ Inverted
4) Bipolar
5) RZ
6) Manchester
7) Differential Manchester
8) AMI
9) PSUEDOTERNARY
10) MLT
1
```



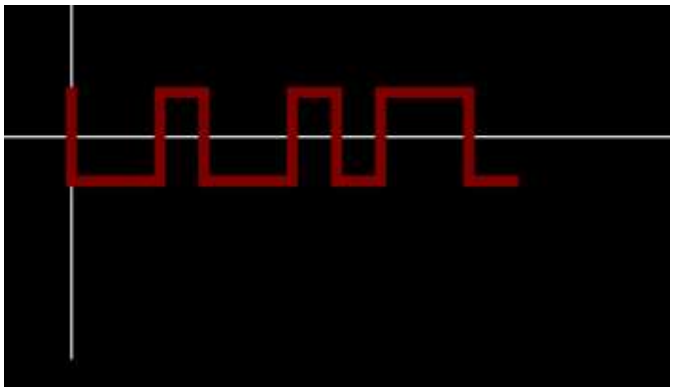
2) Polar NRZ

```
Enter signal to be encoded
1011011101
Choose a type of Encoding
1) Unipolar NRZ
2) Polar NRZ
3) NRZ Inverted
4) Bipolar
5) RZ
6) Manchester
7) Differential Manchester
8) AMI
9) PSUEDOTERNARY
2
```



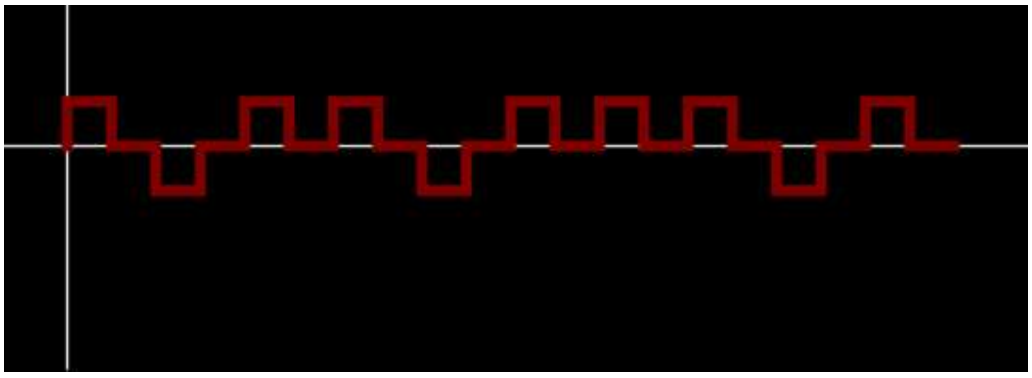
3) NRZ inverted

```
Enter signal to be encoded
1011011101
Choose a type of Encoding
1) Unipolar NRZ
2) Polar NRZ
3) NRZ Inverted
4) Bipolar
5) RZ
6) Manchester
7) Differential Manchester
8) AMI
9) PSUEDOTERNARY
3
```



4) RZ

```
Enter signal to be encoded
1011011101
Choose a type of Encoding
1) Unipolar NRZ
2) Polar NRZ
3) NRZ Inverted
4) Bipolar
5) RZ
6) Manchester
7) Differential Manchester
8) AMI
9) PSUEDOTERNARY
5
```

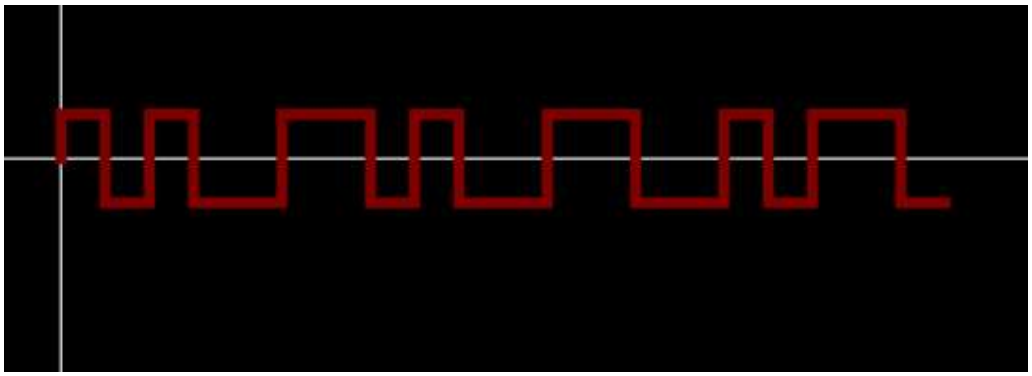


5) Manchester

```
Enter signal to be encoded
1011011101
Choose a type of Encoding
1) Unipolar NRZ
2) Polar NRZ
3) NRZ Inverted
4) Bipolar
5) RZ
6) Manchester
7) Differential Manchester
8) AMI
9) PSUEDOTERNARY
6
```

6) Differential Manchester

```
Enter signal to be encoded
1011011101
Choose a type of Encoding
1) Unipolar NRZ
2) Polar NRZ
3) NRZ Inverted
4) Bipolar
5) RZ
6) Manchester
7) Differential Manchester
8) AMI
9) PSUEDOTERNARY
10) MLT
7
```

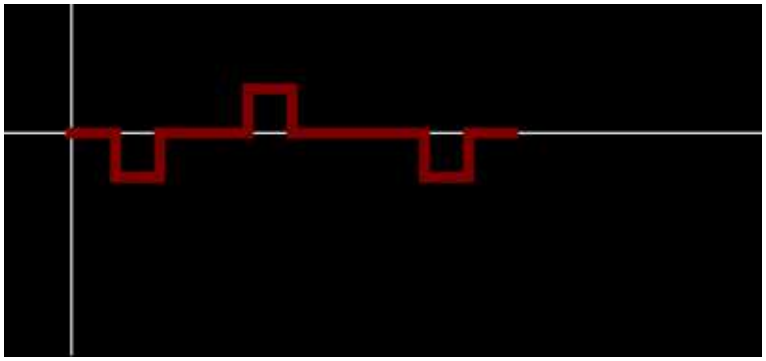


7) AMI

```
Enter signal to be encoded
1011011101
Choose a type of Encoding
1) Unipolar NRZ
2) Polar NRZ
3) NRZ Inverted
4) Bipolar
5) RZ
6) Manchester
7) Differential Manchester
8) AMI
9) PSUEDOTERNARY
10) MLT
8
```

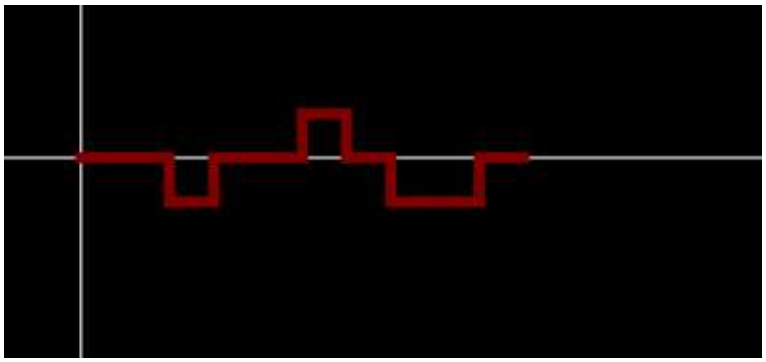
8) Psuedoternary

```
Enter signal to be encoded
1011011101
Choose a type of Encoding
1) Unipolar NRZ
2) Polar NRZ
3) NRZ Inverted
4) Bipolar
5) RZ
6) Manchester
7) Differential Manchester
8) AMI
9) PSUEDOTERNARY
10) MLT
9
```



9) MLT-3

```
Enter signal to be encoded
1011011101
Choose a type of Encoding
1) Unipolar NRZ
2) Polar NRZ
3) NRZ Inverted
4) Bipolar
5) RZ
6) Manchester
7) Differential Manchester
8) AMI
9) PSUEDOTERNARY
10) MLT
10
```



Conclusion

In this report, we've covered the main ways line coding is applied to binary data to enable transmission. We've also covered the benefits and drawbacks of line coding as well as how it can be represented as a signal. We've also covered the various types of line coding, how they are applied to binary data, and each type's benefits and drawbacks.

References

[1] Wikipedia: Line code (Section- Introduction, Transmission and storage, Common line codes)
https://en.wikipedia.org/wiki/Line_code

[2] Data Communications and Networking By Behrouz A. Forouzan (Behrouz A. Forouzan) [Chapter 4: Page 102-114] {FOURTH EDITION}

[3] JavaTPoint: Digital Transmission (Section- Digital to Digital Conversion)
Computer Network Digital Transmission - javatpoint

[4] tutorialspoint: Line Codes
Digital Communication - Line Codes (tutorialspoint.com)

[5] CDNetworks: Data Transmission: What is it? Everything You Need To Know
Data Transmission: What Is It? Everything You Need to Know - CDNetworks

[6] GeeksforGeeks: Difference between Unipolar, Polar and Bipolar Coding schemes (Diagrams + Article)
Difference between Unipolar, Polar and Bipolar Line Coding Schemes -GeeksforGeeks