Line Coding & Scrambling Visualization Application

By

1. Tirth A. Patel (18BCE243)
2. Tirth D. Hihoriya (18BCE244)
3. Shrey A. Viradiya (18BCE259)

Submitted to,

Prof. Parita Oza,
Assistant Professor,
Department of Computer Science & Engineering,
Institute of Technology Nirma University

Code Available at:
https://github.com/Shrey-Viradiya/LineCoding

Introduction:

In this digital world, most of the communicating devices are based on digital communication. In digital transmission, a long stream of constant voltage raises issues like baseline wandering, DC component and self-synchronization problem. To overcome these problems, line encoding schemes and scrambling techniques are used to encode data before transmitting. Here we implemented some line coding techniques and scrambling techniques and visually presented them. For this project, Python 3, matplotlib and tkinter are used.

Methodology:

Every character of the English language and special characters have a specific ASCII code representing it. So, we can convert each text message into ASCII. For example, ASCII code for 'H' is 72 and that is for 'I' is 73. So, ASCII representation of the message *'HI'* becomes 72 73.

HI → 72 73

Each number of ASCII is converted into binary numbers. Each ASCII character can be represented using a 7-bit binary code because the ASCII table has 127 characters. But most of the computer uses 1 byte of storage to store it i.e. 8-bits. Therefore,

(Decimal) 72 = (Binary) 0100 1000

(Decimal) 73 = (Binary) 0100 1001

Now converting this message into binary makes the message as *0100 1000 0100 1001.* First 8-bits represents 'H' and the last 8-bits represent 'I'. The last binary message which computer understands is transmitted. But before transmission encryption, header/trailer addition is completed in order to ensure reliable transmission. For this application encryptions and addition of headers or trailers are not considered for simplicity and visualization. Here we'll directly encode or scramble the text.

Message: *0100 1000 0100 1001*

To avoid, problems like baseline wandering and DC component which generated low frequency in transmission line coding schemes are used. Below are the different line coding schemes and scrambling techniques are used.

i) AMI
ii) Pseudo-ternary
iii) NRZ-I
iv) NRZ-L
v) Polar-RZ
vi) Manchester
vii) Differential Manchester
viii) 2B1Q
ix) MLT_3
x) B8ZS (scrambling technique)
xi) HDB3 (scrambling technique)

AMI:

Alternate Mark Inversion (AMI) is a bipolar encoding scheme. In this scheme, alternate 1 makes inversion. A binary zero is represented by neutral zero voltage.

Pseudo-ternary

Pseudo-ternary is a variation of AMI code in which binary 1 is neutral zero voltage and alternate 0 is encoded as alternate positive or negative voltages.

NRZ-L & NRZ-I

In Polar NRZ, two different voltage levels are used. In the NRZ-L scheme, the voltage level determines binary 1 and 0. For this application binary 0 is represented by positive voltage and binary 1 is represented by negative voltage. In the NRZ-I scheme, voltage level inversion takes place. If there is inversion in level it is binary 1 otherwise if voltage level is constant it is binary 0.

Polar-RZ

In NRZ schemes, self-synchronization is not achieved. The receiver does not know when one bit has ended and next started. In the Polar-RZ scheme, signal changes during the bit instead of between the bits.

Biphase Schemes

Manchester and Differential Manchester are the biphase encoding scheme. Manchester is the combined idea of NRZ-L and RZ. One bit is divided into two signals. In the first half of the bit duration voltage remains at the first level and in the other half voltage level changes. The transition between the bit provides synchronization in the transmission. Differential Manchester is the combined idea of NRZ-I and RZ. Binary 0 or 1 determines the value of voltage level.

2B1Q

2B1Q is a multilevel scheme. 2B1Q stands for *two binary one quaternary.* This scheme encodes two binary digits in one signal. For the scheme, four different levels are used.

	Previous level positive	Previous level negative
Next bits	Next Level	Next Level
00	1	-1
01	2	-2
10	-1	1
11	-2	2

MLT-3

MLT3 is the short form of *multiline transmission, three-level*. In this scheme 3 rules are followed:

i) No transition is next bit is 0.
ii) The next level is 0 if next bit is 1 and current level is not 0.
iii) If the next bit is 1 and current level is zero, next level is opposite of last non-zero level.

B8ZS

Bipolar with 8 zero substitution (B8ZS) is a scrambling technique mostly used in North America. In this scrambling, 8 zeroes are substituted with sequence 000VB0VB where V denotes violation and B denotes bipolar in accordance with AMI rules.

For this scrambling, the message '@0' is used as an example. Binary for '@0' is *0100000000110000.*

HDB3

High-density bipolar 3-zero (HDB3) is a more conservative scrambling technique than B8ZS mostly used outside of North America.

In this technique, four consecutive zeros are substituted with sequence 000V or B00V. If the number of non-zero levels after last substitution is even, then B00V is substituted and if it is odd, then 000V is substituted.

For this scrambling, the message '@0' is used as an example.

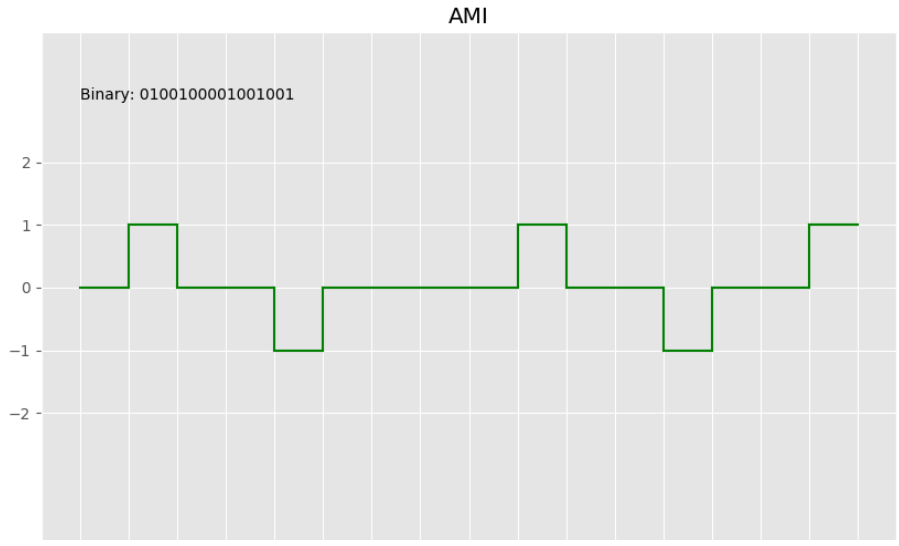
Conclusion

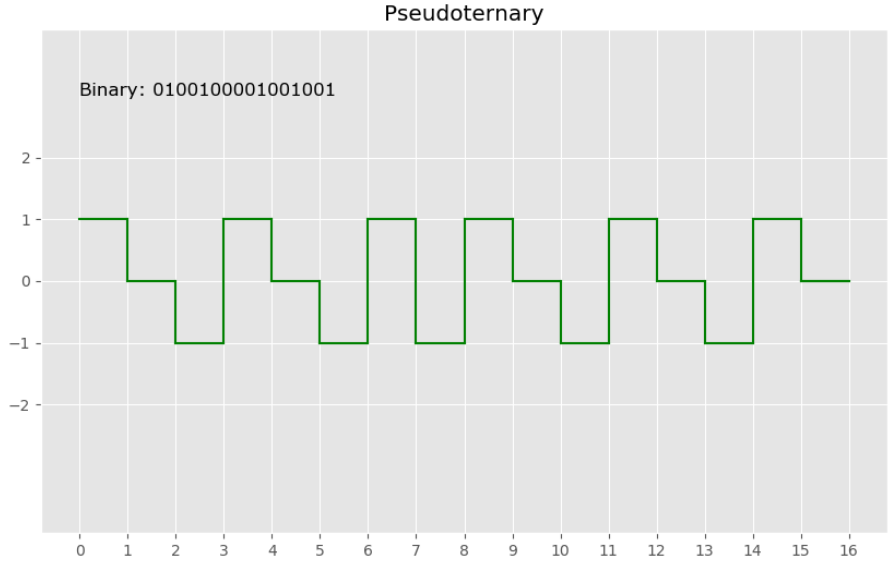
In this project work, we discussed different line-coding and scrambling techniques and developed an application with converts text into the binary code using ASCII and visually represents them in different line-coding / scrambling.

Reference:

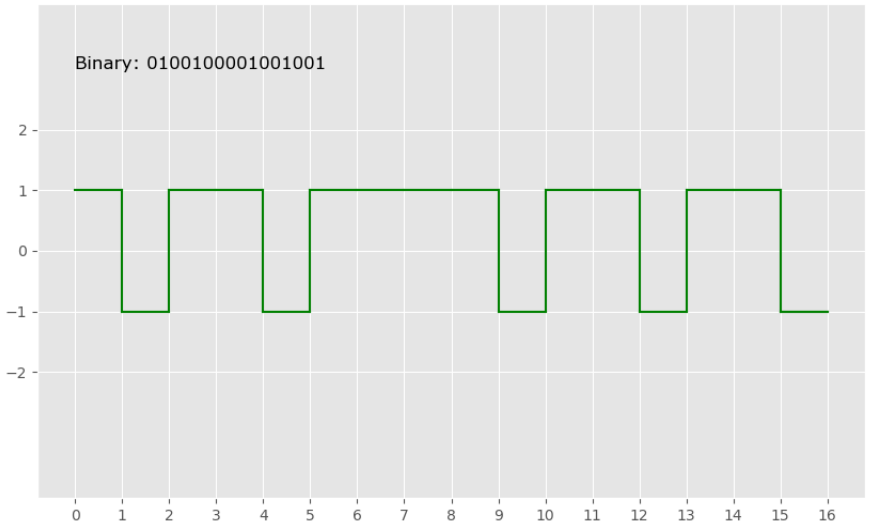
Data Communication and Networking by Behrouz A. Forouzan (4th.edition)

Screen Snips:

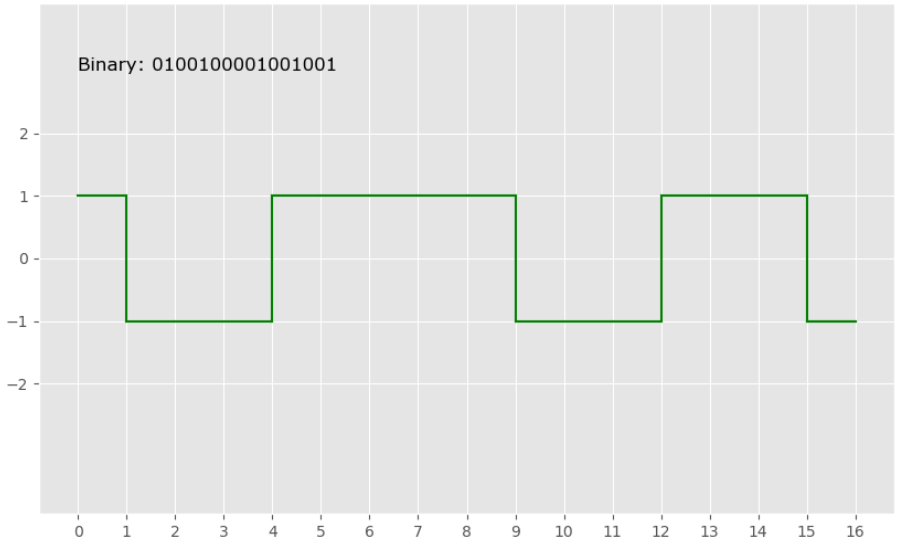




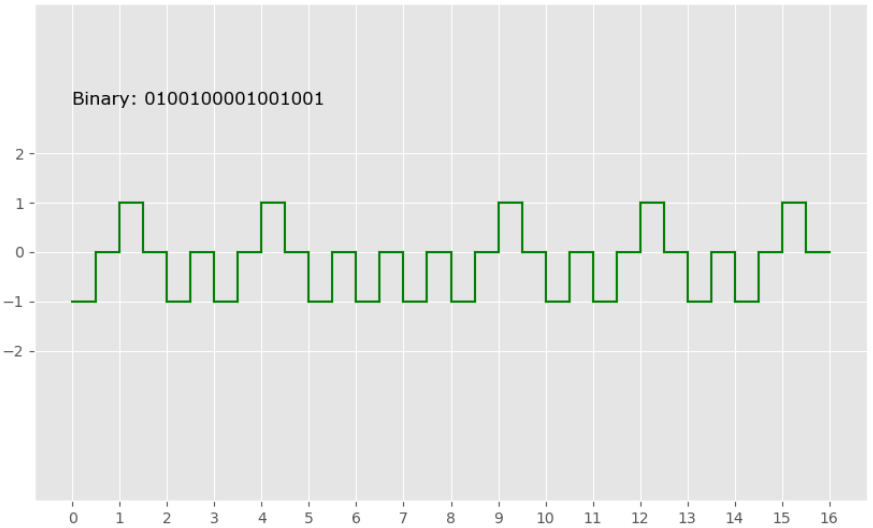
NRZ L

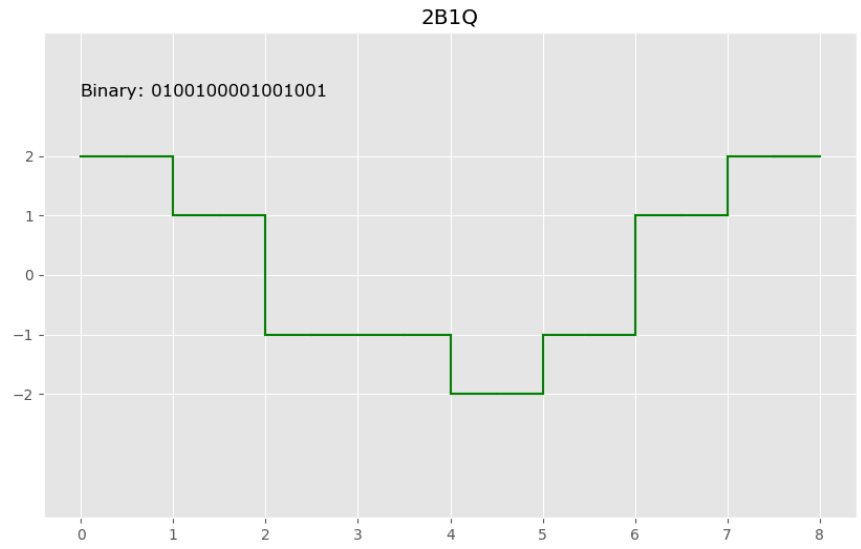


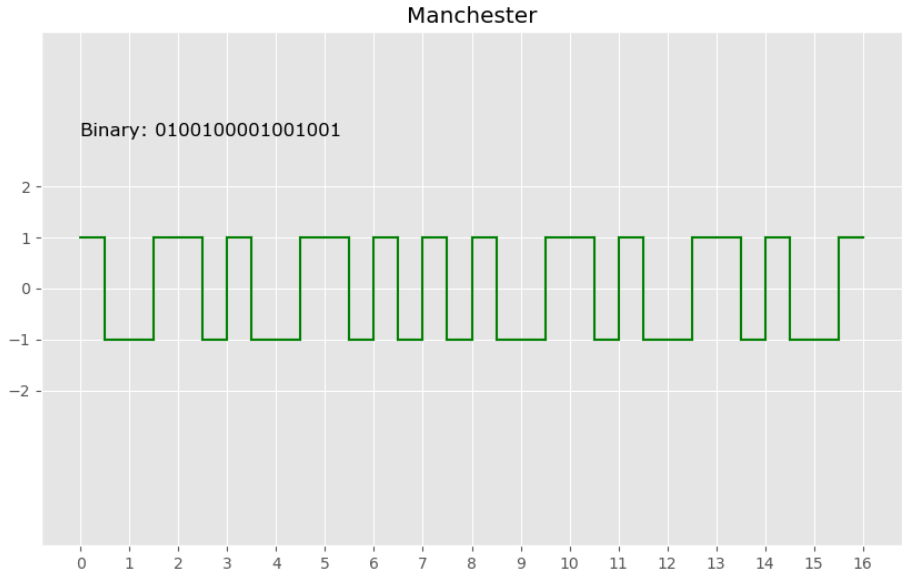
NRZ I

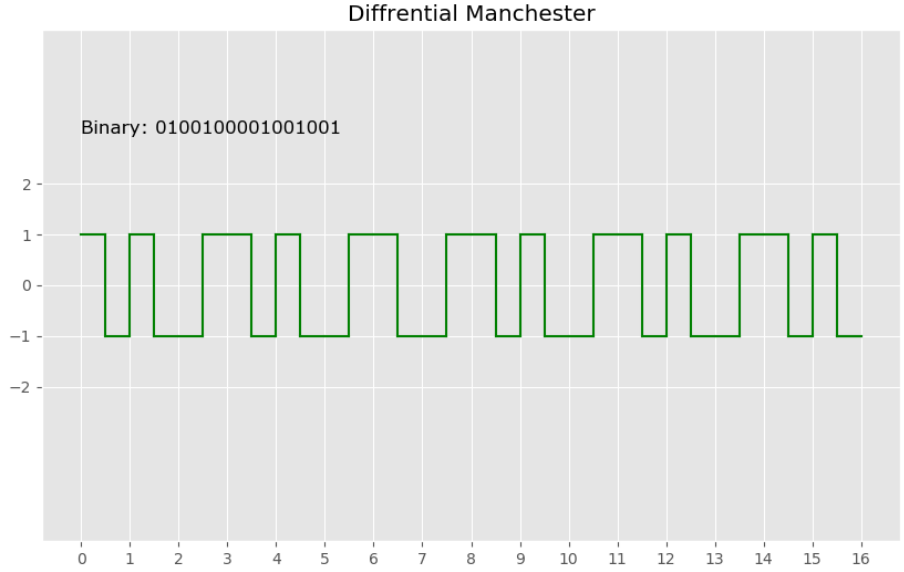


Polar RZ

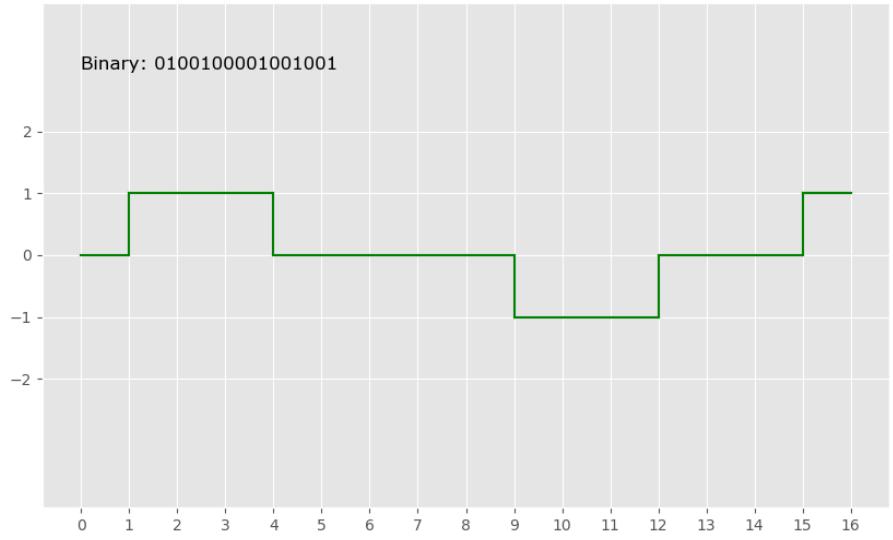



2B1Q
Binary: 0100100001001001
2
1
0
−1
−2
0
1
2
3
4
5
6
7
8

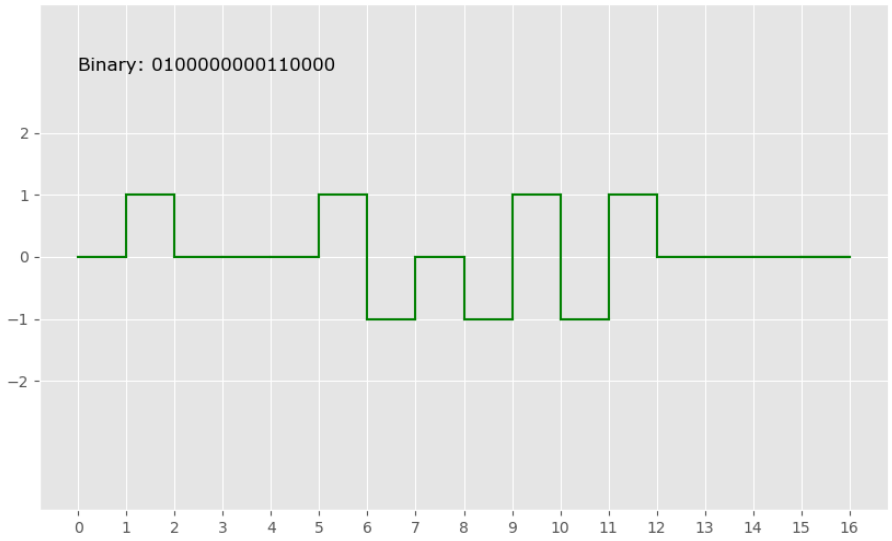

Manchester
Binary: 0100100001001001
2
1
0
−1
−2
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16


Diffrential Manchester
Binary: 0100100001001001
2
1
0
−1
−2
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16

MLT 3



B8ZS



HDB 3