University of Dhaka

Department of Computer Science and Engineering

CSE 2213 – Data and Telecommunication Laboratory   Credits: 0.75

Batch: 29/2nd Year 2nd Sem 2025

Instructors: Prof. Dr. Md. Mustafizur Rahman (MMR), Mr. Palash Roy (PR)

Lab Experiment # 1

Name of the Experiment: Implementing encoding and decoding scheme using NRZ-L, NRZ-I and Manchester

NRZ-I Encoding and Decoding Scheme

Non-Return To Zero (NRZ) line code is a binary code in which ones are represented by one significant condition, usually a positive voltage, while zeros are represented by some other significant condition, usually a negative voltage, with no other neutral or rest condition.

Non-return to Zero Inverted (NRZ-I) is a type of NRZ line coding. In NRZ-I usually binary 1 changes the state (toggle), and binary 0 keeps the state the same as the previous bit. Although binary 0 changes state and binary 1 keeps the state unchanged is also NRZ-I. We will use the first way that is binary 1 changes the logic level and binary 0 keeps unchanged the logic level. In brief

NRZ-I Encoding Rule:

- If the bit is ‘1’, invert the signal level from the previous bit.
- If the bit is ‘0’, retain the previous signal level.

NRZ-I Decoding Rule:

- If a transition occurs, decode it as ‘1’.
- If no transition occurs, decode it as ‘0’.

Here is an example of NRZ-I line coding for bits 10111001 shown in Figure 1.

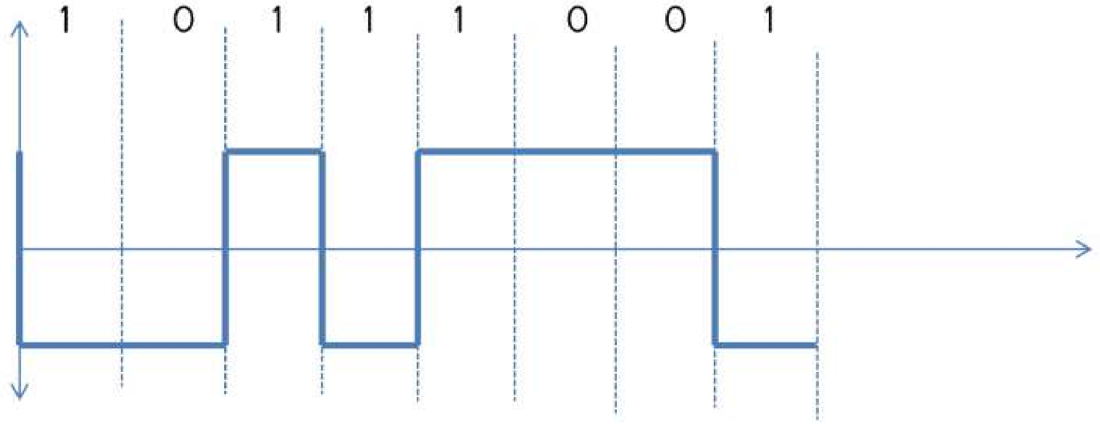

Figure 1: An example of NRZ-I line coding for bits 10111001

Algorithm for NRZ-I Encoding:

1. Initialize an empty list for the encoded signal.
2. Set the initial signal level (assume high or low).
3. Iterate through the input binary sequence: a) If the bit is '1', invert the signal level and append it. b) If the bit is '0', retain the previous level and append it.
4. Output the encoded signal.

Algorithm for NRZ-I Decoding:

1. Initialize an empty list for the decoded binary sequence.
2. Set the initial signal level.
3. Iterate through the encoded signal: a) If a transition is detected from the previous level, append '1' to the decoded sequence. b) If no transition occurs, append '0'.
4. Output the decoded sequence.

Experimental Procedure:

1. Write a program in Python or MATLAB to implement NRZ-I encoding.
2. Input a sample binary sequence (e.g., '10111001').
3. Compute the NRZ-I encoded signal
4. Display and plot the encoded waveform (Simulation software (e.g., MATLAB, Python with Matplotlib/Numpy libraries).
5. Implement the NRZ-I decoding algorithm.
6. Decode the encoded signal back to the original binary sequence.
7. Verify the correctness by comparing the decoded sequence with the original sequence.
8. Observe the effect of encoding and decoding and document the results.

NRZ-L Encoding and Decoding Scheme

Non Return To Zero Level (NRZ-L) is a type of NRZ line coding. In NRZ-L usually binary 1 maps to logic level high, and binary 0 maps to logic level low. Although binary 0 maps to logic level high, and binary 1 maps to logic level low is also NRZ-L. We will use the first way which is binary 1 maps to logic level high and binary 0 maps to logic level low.

Here is an example of NRZ-L line coding for bits 10111001 shown in Figure 2.

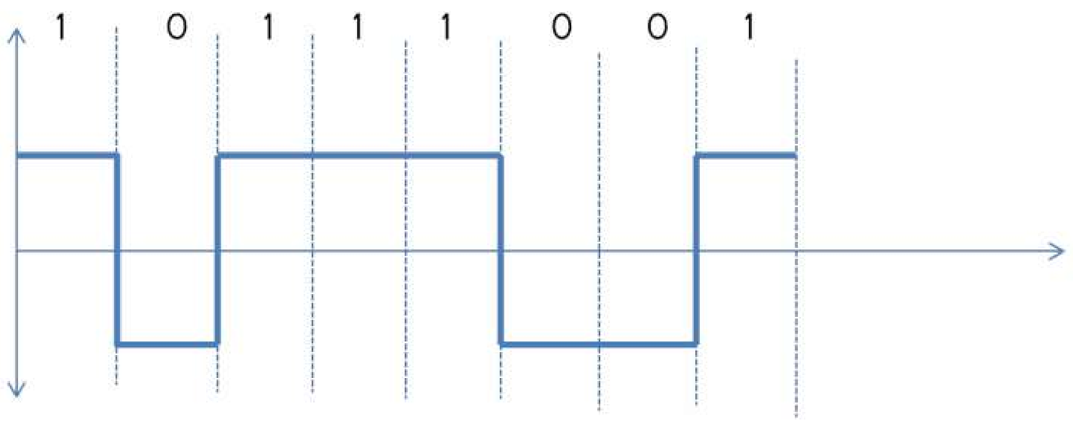

Figure 2: An example of NRZ-I line coding for bits 10111001

Algorithm for NRZ-L Encoding:

1. Initialize an empty list for the encoded signal.
2. Iterate through the input binary sequence: a) If the bit is '0', append a low voltage level. b) If the bit is '1', append a high voltage level.
3. Output the encoded signal.

Algorithm for NRZ-L Decoding:

1. Initialize an empty list for the decoded binary sequence.
2. Iterate through the encoded signal: a) If the voltage level is low, append '0' to the decoded sequence. b). If the voltage level is high, append '1'.
3. Output the decoded sequence.

Experimental Procedure:

1. Write a program in Python or MATLAB to implement NRZ-L encoding.
2. Input a sample binary sequence (e.g., '10111001').
3. Compute the NRZ-L encoded signal.
4. Display and plot the encoded waveform (Simulation software (e.g., MATLAB, Python with Matplotlib/Numpy libraries).
5. Implement the NRZ-L decoding algorithm.
6. Decode the encoded signal back to the original binary sequence.
7. Verify the correctness by comparing the decoded sequence with the original sequence.
8. Observe the effect of encoding and decoding and document the results.

<u>Manchester Encoding and Decoding Scheme</u>

In Manchester, the duration of a bit is divided into two halves. The voltage remains the same at one level during the first half & moves to the other level. Manchester code is a line code in which the encoding of each data bit is either low then high, or high then low, for equal time. Normally when the bit is 1 it starts from high and then goes to low, When the bit is 0 it starts from low and then goes to 1. However, the opposite coding is also Manchester coding.

Here is an example of Manchester line coding for bits 10111001 shown in Figure 3

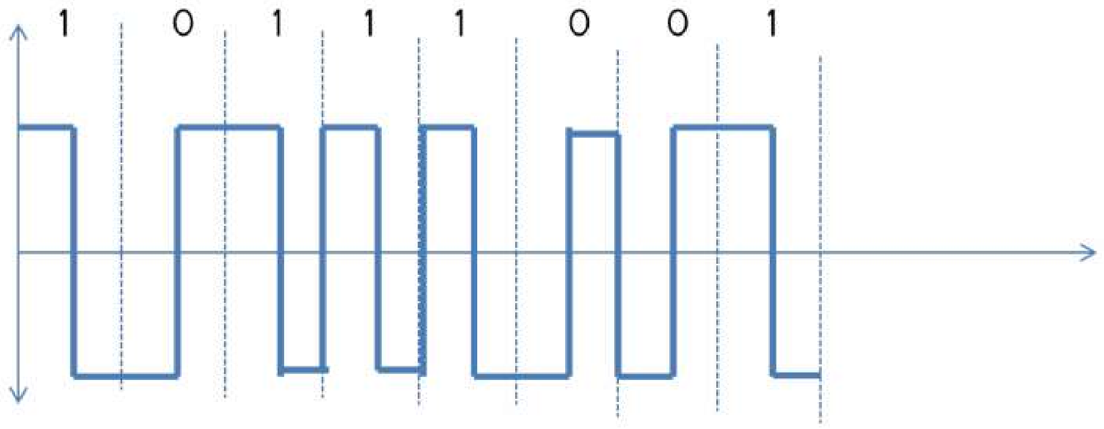

Figure 3: An example of Manchester coding for bits 10111001

Algorithm for Manchester Encoding:

1. Initialize an empty list for the encoded signal.
2. Iterate through the input binary sequence: a) If the bit is ‘0’, append a transition from low to high. b) If the bit is ‘1’, append a transition from high to low.
3. Output the encoded signal.

Algorithm for Manchester Decoding:

1. Initialize an empty list for the decoded binary sequence.
2. Iterate through the encoded signal: a) If the mid-bit transition is from low to high, append ‘1’. b) If the mid-bit transition is from high to low, append ‘0’.
3. Output the decoded sequence.

Experimental Procedure:

1. Write a program in Python or MATLAB to implement Manchester encoding.
2. Input a sample binary sequence (e.g., ‘1011001’).
3. Compute the Manchester encoded signal.
4. Display and plot the encoded waveform ((Simulation software (e.g., MATLAB, Python with Matplotlib/Numpy libraries).
5. Implement the Manchester decoding algorithm.
6. Decode the encoded signal back to the original binary sequence.
7. Verify the correctness by comparing the decoded sequence with the original sequence.
8. Observe the effect of encoding and decoding and document the results.