# **Implementation of Different Encoding and Decoding Schemes**

## **1. Introduction**

In the field of data communication, encoding and decoding schemes are essential for transmitting digital data reliably and efficiently over physical media. These schemes convert binary data into signals suitable for transmission and then back into binary form at the receiving end. Different encoding techniques provide varying levels of synchronization, error detection, and signal efficiency.

The necessity for different schemes arises from the limitations of various transmission media and the requirement to maintain signal integrity, clock synchronization, and bandwidth efficiency. In this lab, we explored six commonly used encoding techniques:

* **NRZ-L (Non-Return to Zero-Level)**
* **NRZ-I (Non-Return to Zero-Inverted)**
* **Manchester Encoding**
* **AMI (Alternate Mark Inversion)**
* **Pseudo Ternary**
* **MLT-3 (Multi-Level Transmit – 3 Levels)**

Each technique has unique characteristics that make them suitable for specific communication scenarios.

## **2. Objectives**

* To understand the purpose and function of different digital encoding and decoding techniques.
* To implement the algorithms for six encoding schemes.
* To verify the correctness of encoding by decoding the signal back to the original data.
* To visualize the waveform of each encoding scheme for given input bits.
* To compare the efficiency and complexity of different schemes.

## **3. Algorithms / Pseudocode**

### **NRZ-L**

**Encoding Algorithm**

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

**Decoding Algorithm**

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

### **NRZ-I**

**Encoding Algorithm**

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.

1. Output the encoded signal.

**Decoding Algorithm**

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’.

1. Output the decoded sequence.

### **Manchester**

**Encoding Algorithm**

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.

1. Output the encoded signal.

**Decoding Algorithm**

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’.

1. Output the decoded sequence.

### 

### **AMI**

**Encoding Algorithm**

1. Initialize last = -1 (this tracks polarity of last '1')

2. For each bit in the binary string:

* o If bit is '0', append 0 (neutral)
* o If bit is '1':

▪ Flip last to its opposite sign

▪ Append last

**Decoding Algorithm**

1. For each voltage in the list:

* If value is 0, append '0'
* If value is +V or -V, append '1'

### **Pseudo Ternary**

**Encoding Algorithm**

1. Initialize last = -1 (polarity tracker)

2. For each bit:

* If bit is '1', append 0
* If bit is '0':

▪ Flip polarity (last \*= -1)

▪ Append last

**Decoding Algorithm**

1. For each voltage in the list:

* If value is 0, append '1'
* If value is +V or -V, append '0'

### **MLT-3**

**Encoding Algorithm**

1. Set the initial level to 0.

2. Define the transition sequence: [0, +1, 0, -1].

3. Initialize an index to track the current position in the transition cycle.

4. For each bit:

* If bit is '0': repeat last level.
* If bit is '1': move to the next level in the transition sequence.

**Decoding Algorithm**

1. Compare each signal level with the previous one.

* If it changes: append '1'
* If it remains the same: append '0'

## **4. Sample Input and Output**

### **Input Bit:**

11001010

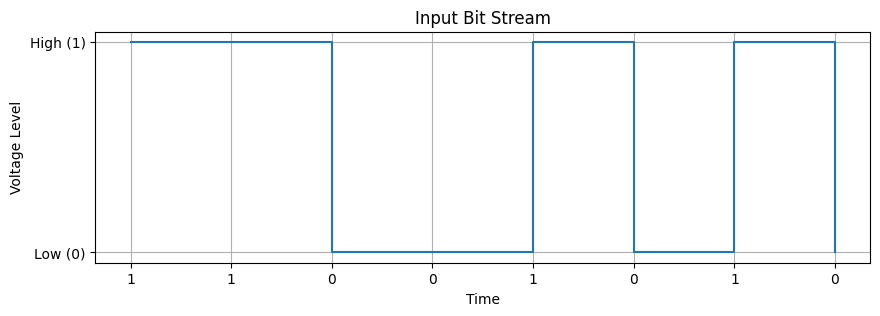
| **Encoding Scheme** | **Encoded Signal** | **Decoded Bit** |
| --- | --- | --- |
| NRZ-L | [1, 1, -1, -1, 1, -1, 1, -1] | 11001010 |
| NRZ-I | [0, 1, 1, 1, 0, 0, 1, 1] | 11001010 |
| Manchester | [1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1] | 11001010 |
| AMI | [1, -1, 0, 0, 1, 0, -1, 0] | 11001010 |
| Pseudo Ternary | [0, 0, -1, 1, 0, -1, 0, 1] | 11001010 |
| MLT-3 | [1, 0, 0, 0, -1, -1, 0, 0] | 11001010 |

*(LH = Low to High, HL = High to Low transitions)*

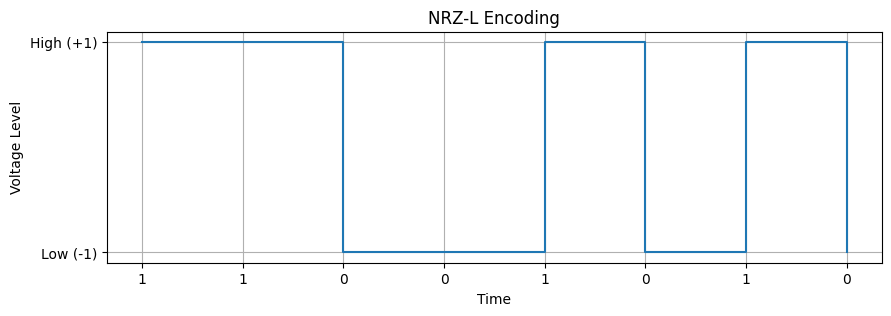
### **Waveforms:**

### **NRZ-L**

Input Stream:

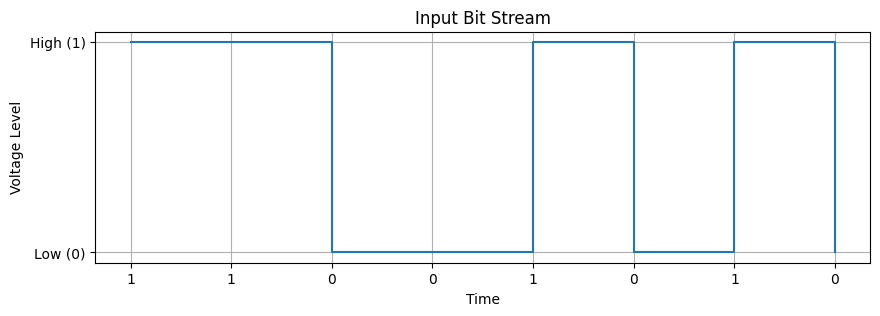


Output Stream:

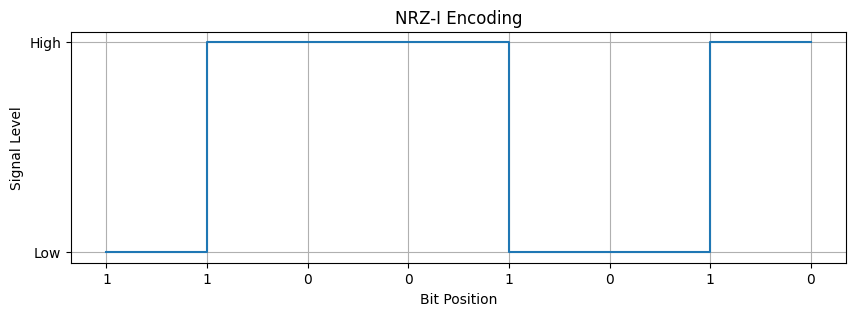


### **NRZ-I**

Input Stream:



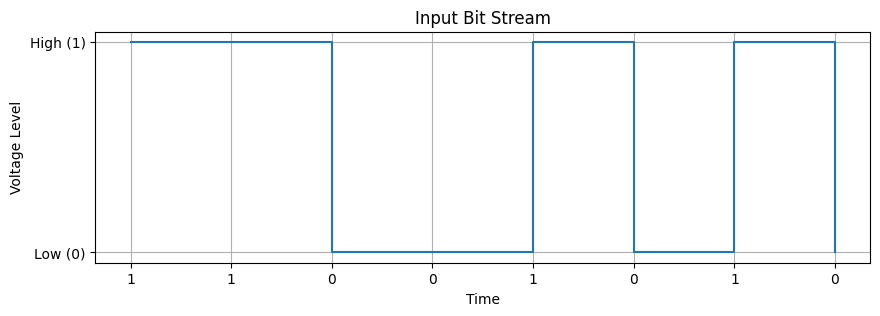
Output Stream:



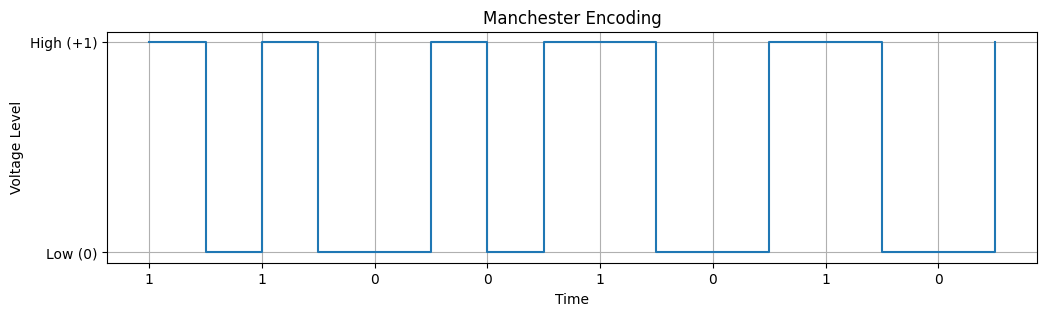
### 

### **Manchester**

Input Stream:



Output Stream:

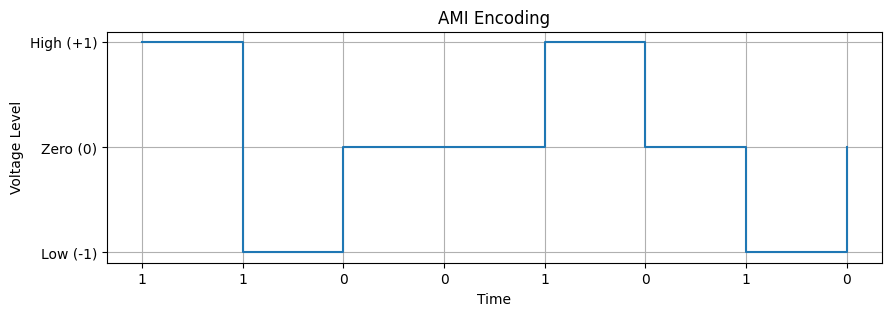


### **AMI**

Input Stream:

### 

Output Stream:

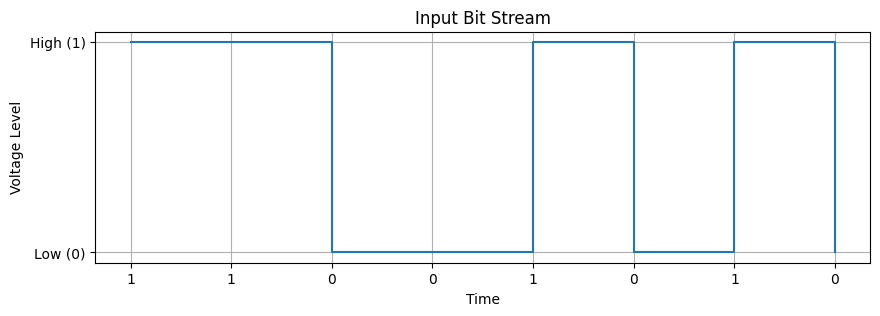


### 

### 

### **Pseudo Ternary**

Input Stream:

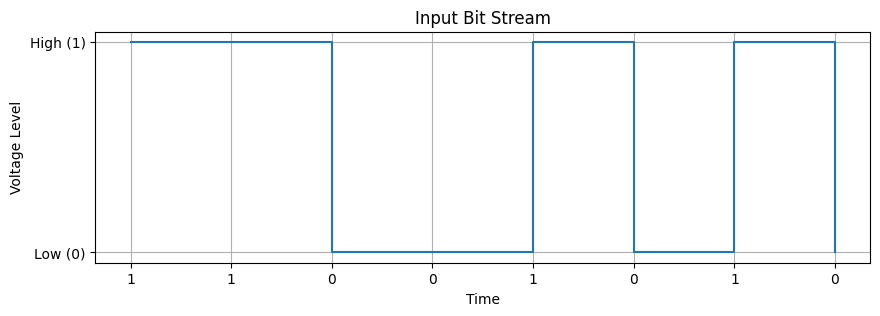


Output Stream:

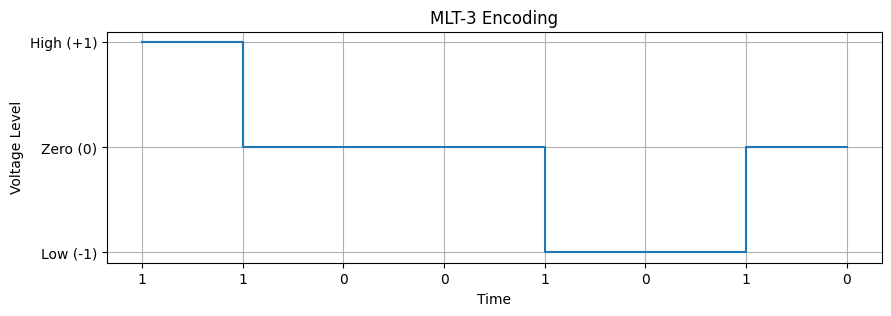
### 

### **MLT-3**

Input Stream:



Output Stream:



## **5. Learning and Difficulties**

**Learning Outcomes:**

* Understood the concept and use of various digital encoding techniques.
* Learned how to implement logic-based signal generation and decoding.
* Gained hands-on experience with waveform analysis.

**Difficulties Faced:**

* Differentiating between schemes like AMI and PseudoTernary initially due to similar appearance.
* Implementing MLT-3 required careful tracking of previous levels and transitions.
* Visualizing waveforms accurately, especially for transition-based encodings like Manchester and MLT-3.

## **6. Conclusion**

This lab enhanced our understanding of digital signal encoding and decoding. We observed the practical differences between various schemes and how they influence signal synchronization, bandwidth usage, and error detection. Each scheme offers trade-offs that make it suitable for specific applications in data communication systems. Future work can include analyzing performance in noisy environments and exploring advanced schemes like 4B/5B or 8B/10B encoding.