

**University Of Dhaka  
Department of Computer Science & Engineering**

CSE-2213: Data and Telecommunication Lab

2nd Year 2nd Semester

Session: 2022-2023

**Exp No. 1:** Implementation of Different Encoding and Decoding Schemes

**Submitted by:**

Farhan Bin Rabbani

Roll: FH-35

**Submitted to:**

1. Dr. Md. Mustafizur Rahman, Professor, Dept. of CSE, DU
2. Mr. Palash Roy, Lecturer, Dept. of CSE, DU

**Lab Report 1**

**Lab Report Name:**  Implementation of Different Encoding and Decoding Schemes

**Introduction:** Data Encoding schemes are implemented for synchronization, error detection, and lowering signal bandwidth. Some encoding schemes are mentioned as follows:

**NRZ-L Encoding/Decoding:** Non-Return to Zero Level (NRZ-L) is an encoding scheme in which binary 1 is mapped to a logical high level and binary 0 is mapped to a logical low level.

**NRZ-I Encoding/Decoding:** Non-Return to Zero Inverted (NRZ-I) is an encoding scheme in which binary 1 is mapped to a change of state (toggle) and binary 0 keeps the state unchanged as the previous bit.

**Manchester Encoding/Decoding:** A bit's duration is divided into two halves (the voltage remains the same in the first half and changes in the next half). When the bit is ‘0’, the voltage starts from high and goes to low. The voltage goes from low to high when ‘1’ is encountered.

**AMI Encoding/Decoding:** Alternate Mark Inversion (AMI) is a synchronous clock encoding scheme in which binary 1 is represented using bipolar pulses (if the first 1 is mapped to ‘+1’, the next bit is mapped to ‘-1’). Binary 0 is mapped to ‘0’.

**Pseudoternary Encoding/Decoding:** Pseudoternary encoding scheme in which binary 0 is represented using alternating positive and negative voltages (if the first 1 is mapped to ‘+1’, the next bit is mapped to ‘-1’). Binary 1 is mapped to 0.

**MLT3 Encoding/Decoding:** Multi-Level Line encoding uses more than 3 voltage levels to represent binary data. In MLT3 encoding, no change occurs when ‘0’ is encountered. But when ‘1’ is encountered, the transition to the next level follows the sequence: 0→+1→0→-1→0...

**Objectives:**

1. To encode a given sequence of bits using different encoding schemes such as NRZ-L, NRZ-I, Manchester, AMI, Pseudoternary, and MLT3.
2. To decode the encoded signal to see whether the received data is the same as the input data.
3. To provide the waveforms of the input data, encoded signal, and output data
4. To understand the working principles of the aforementioned encoding schemes as well as their benefits and drawbacks.

**Algorithms/Pseudocode:**

**NRZ-L Encoding:**

* We initialize an empty list for the encoded signal.
* We iterate through the input binary sequence: a) If the bit is ‘0’, we append low voltage. b) If the bit is ‘1’, we append high voltage.
* Finally, we output the encoded signal.

**NRZ-L Decoding:**

* We initialize an empty list for the decoded binary sequence.
* We iterate through the encoded signal: a) If the voltage level is low, we append ‘0’ to the decoded sequence. b). If the voltage level is high, we append ‘1’.
* Finally, we output the decoded sequence.

**NRZ-I Encoding:**

* We initialize an empty list for the encoded signal.
* We set the initial signal level to be high (1).
* We iterate through the input binary sequence: a) If the bit is ‘1’, we invert the signal level and append it. b) If the bit is ‘0’, we retain and append the previous level.
* Finally, we output the encoded signal.

**NRZ-I Decoding:**

* We initialize an empty list for the decoded signal.
* We set the initial signal level to be high (as we assumed it to be 1, we will set it to 1).
* We iterate through the encoded signal: a) If a transition is detected from the previous level, we append ‘1’ to the decoded sequence. b) If no transition occurs, we append ‘0’.
* Finally, we output the decoded sequence.

**Manchester Encoding:**

* We initialize an empty list for the encoded signal.
* We iterate through the input binary sequence: a) If the bit is ‘0’, we append a transition from low to high. b) If the bit is ‘1’, we append a transition from high to low.
* Finally, we output the encoded signal.

**Manchester Decoding:**

* We initialize an empty list for the decoded binary sequence.
* We iterate through the encoded signal: a) If the mid-bit transition is from low to high, we append ‘1’. b) If the mid-bit transition is from high to low, we append ‘0’.
* Finally, we output the decoded sequence.

**AMI Encoding:**

* We initialize an empty list for the encoded signal.
* We also initialize a ‘count’ variable and set it to 0.
* We iterate through the input binary sequence: a) If the bit is ‘0’, we append ‘0’ to the string. b) If the bit is ‘1’, we check the variable ‘count’. If it’s even, we append +1 to the list, or else we append -1. We increase the count by 1 in the end.
* Finally, we output the encoded signal.

**AMI Decoding:**

* We initialize an empty list for the decoded binary sequence.
* We iterate through the encoded signal: a) If ‘-1’ or ‘+1’ is encountered, we append 1. b) If ‘0’ is encountered, we append 0.
* Finally, we output the decoded sequence.

**Pseudoternary Encoding:**

* We initialize an empty list for the encoded signal.
* We also initialize a ‘count’ variable and set it to 0.
* We iterate through the input binary sequence: a) If the bit is ‘1’, we append ‘0’ to the string. b) If the bit is ‘0’, we check the variable ‘count’. If it’s even, we append +1 to the list. We append -1 if the count is odd. We increase the count by 1 in the end.
* Finally, we output the encoded signal.

**Pseudoternary Decoding:**

* We initialize an empty list for the decoded binary sequence.
* We iterate through the encoded signal: a) If ‘-1’ or ‘+1’ is encountered, we append 0 to the sequence. b) If ‘0’ is encountered, we append 1 to the sequence.
* Finally, we output the decoded sequence.

**MLT3 Encoding:**

* We initialize an empty list for the encoded signal.
* We also initialize two variables, ‘prev\_state’ and ‘level’, to 0 and 1, respectively.
* We iterate through the encoded signal: a) If ‘0’ is encountered, we keep ‘prev\_state’ unchanged. b) If ‘1’ is encountered, we check the level: i) For level 1, we set prev\_state to +1 and level to 2. ii) For level 2, we set prev\_state to 0 and level to 3. i) For level 3, we set prev\_state to -1 and level to 4. i) For level 4, we set prev\_state to 0 and level to 1. Then, we append prev\_state to the list.
* Finally, we output the encoded signal.

**MLT3 Decoding:**

* We initialize an empty list for the decoded binary sequence and append 1 at the beginning.
* We iterate through the encoded signal: a) If the previous bit is the same as the next bit, we append 0 to the sequence. b) If there is a change in state between the previous and next bit, we append 1 to the sequence.
* Finally, we output the decoded sequence.

**Sample Input/Output:**

**NRZ-L Encoding/Decoding:**







**NRZ-I Encoding/Decoding:**



The previous state is assumed to be 1.





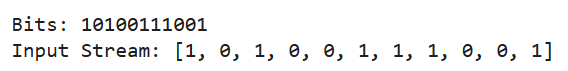
**Manchester Encoding/Decoding:**







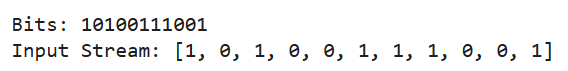
**AMI Encoding/Decoding:**







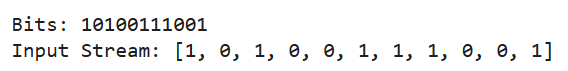
**Pseudoternary Encoding/Decoding:**





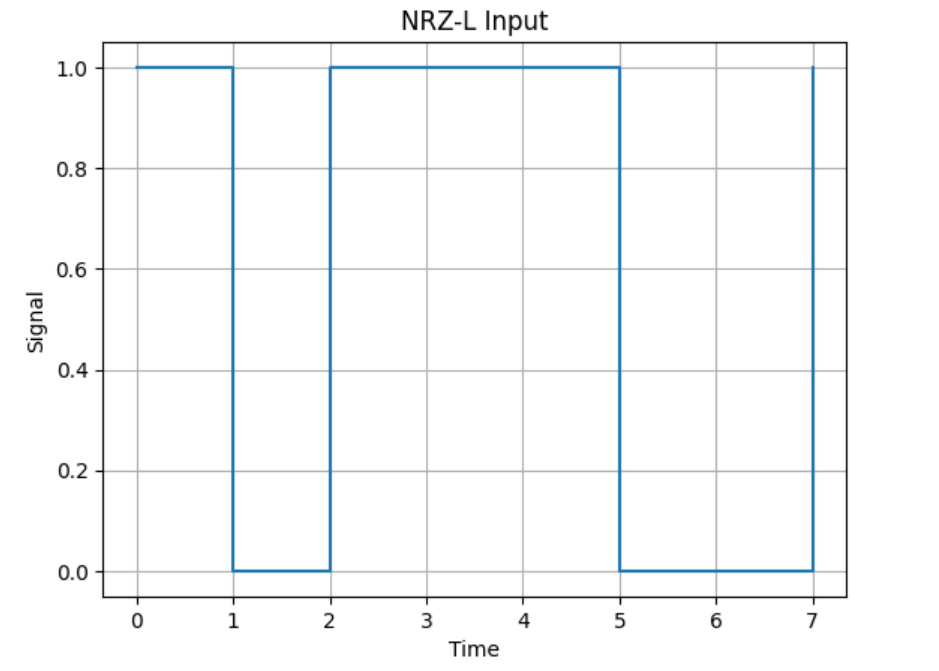


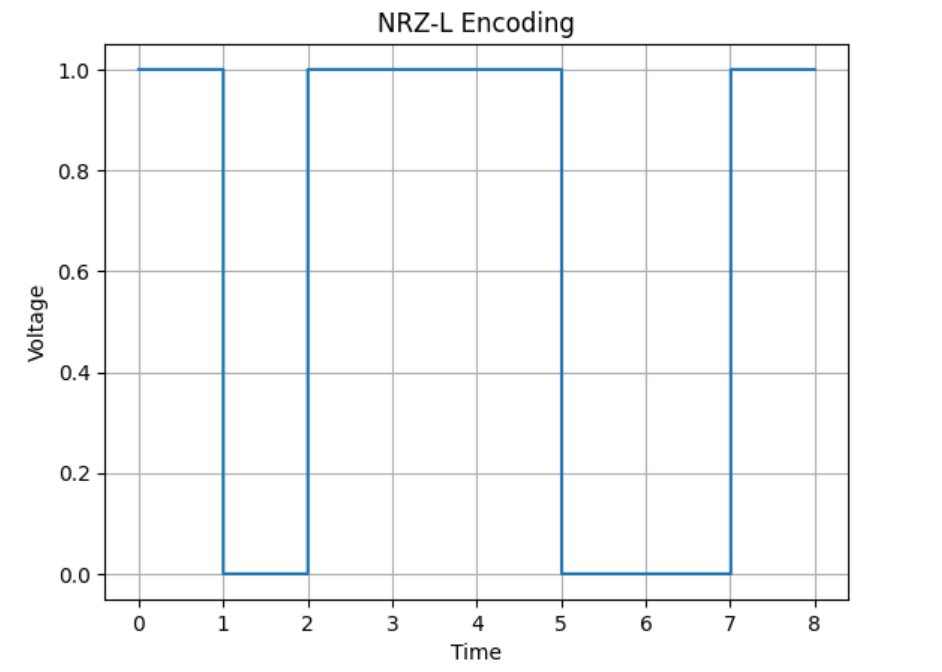
**MLT3 Encoding/Decoding:**

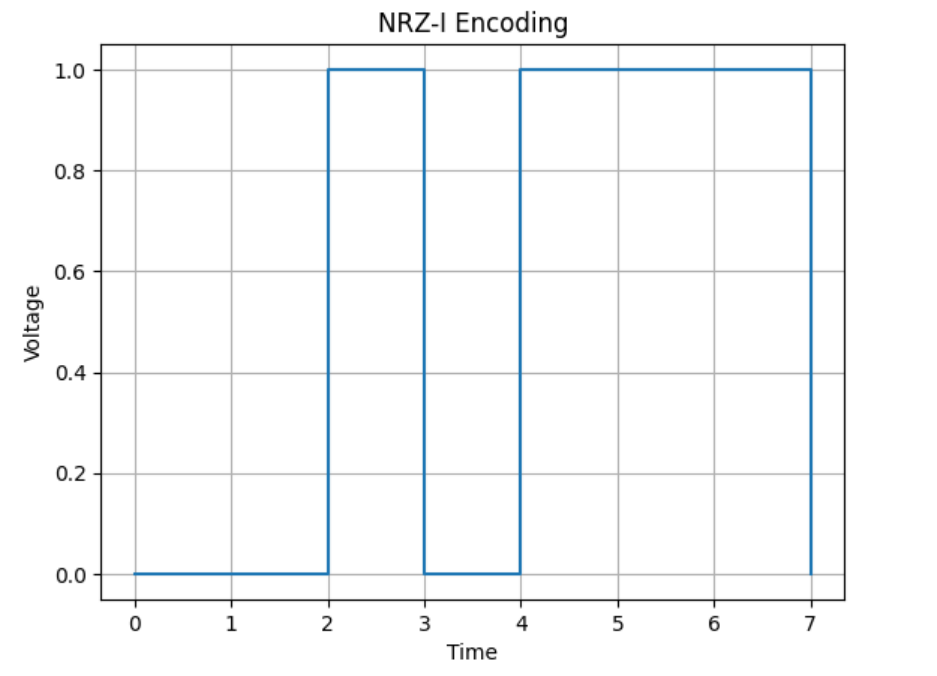
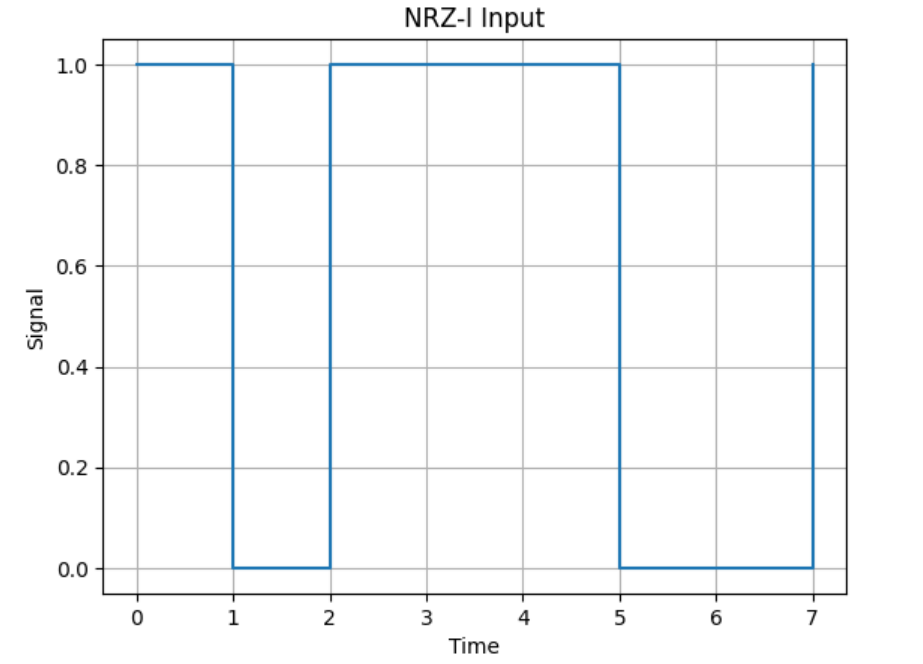


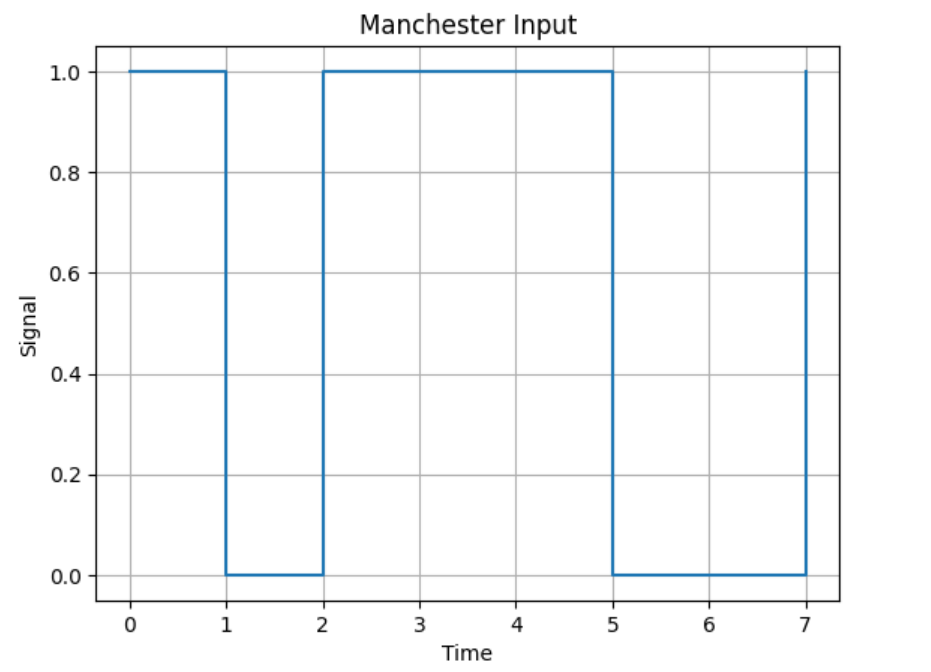


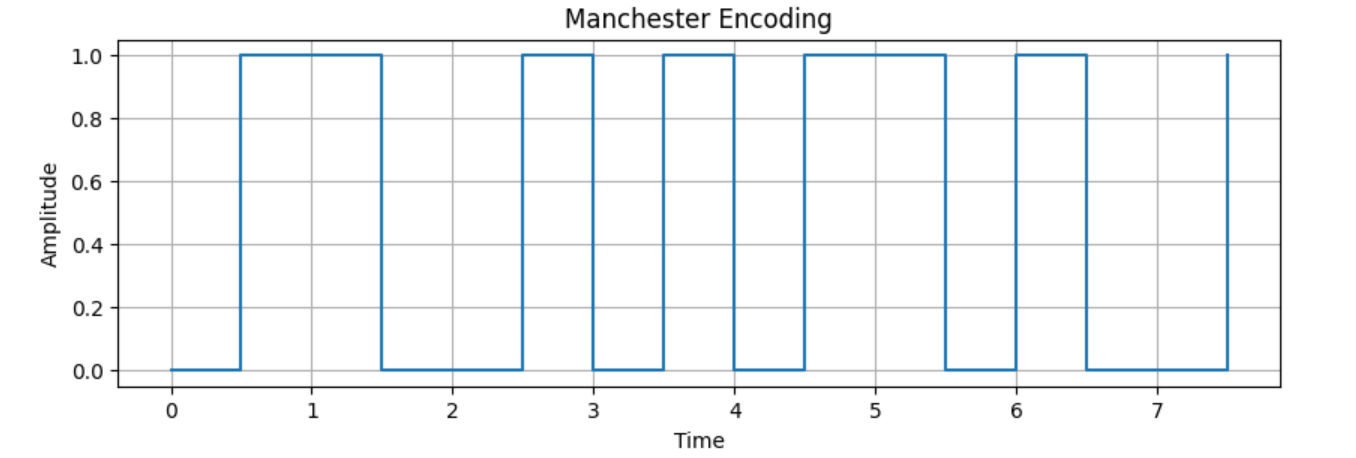


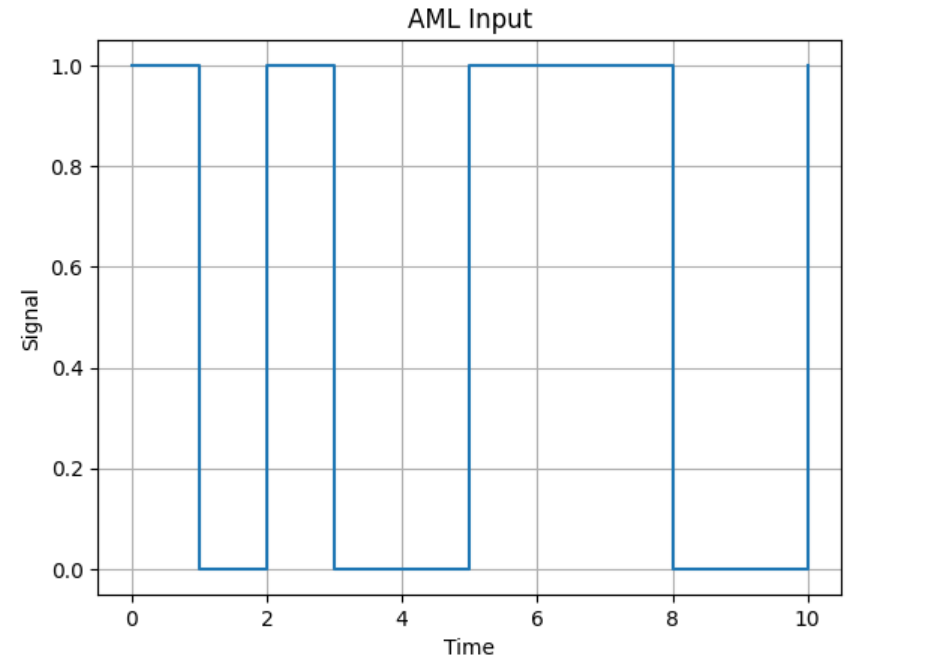


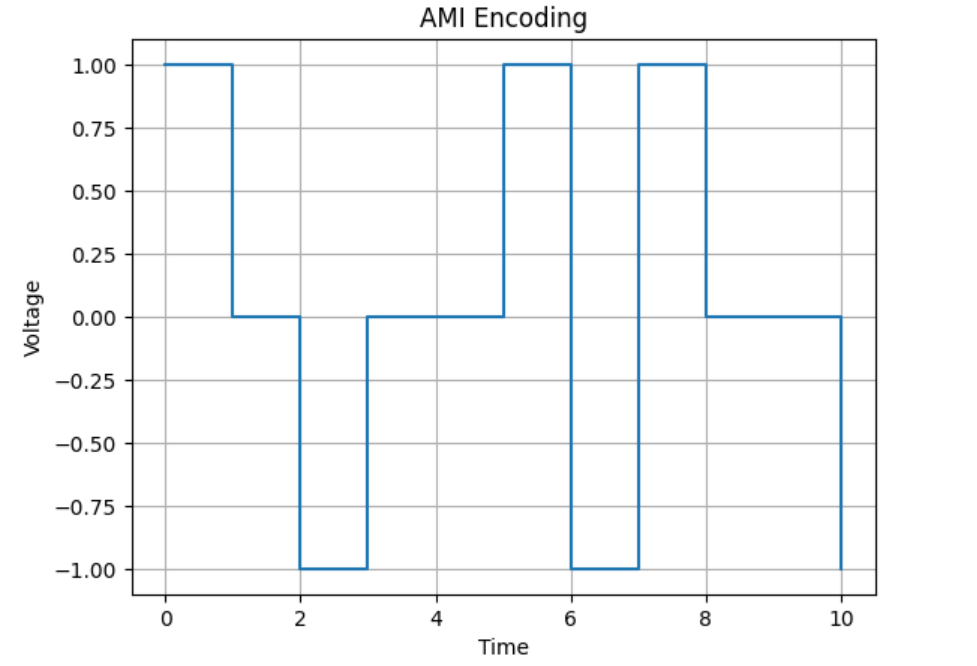


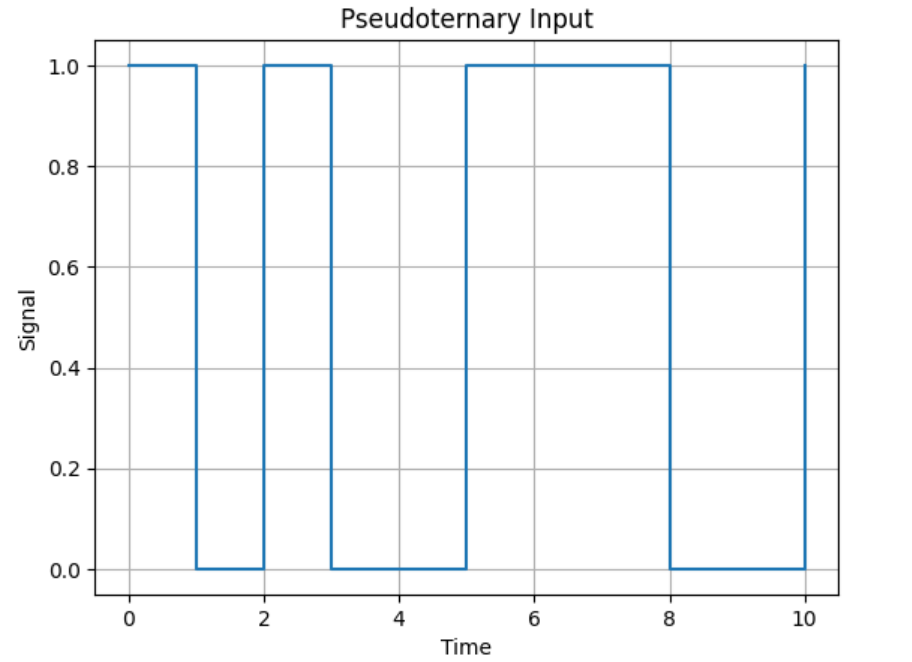


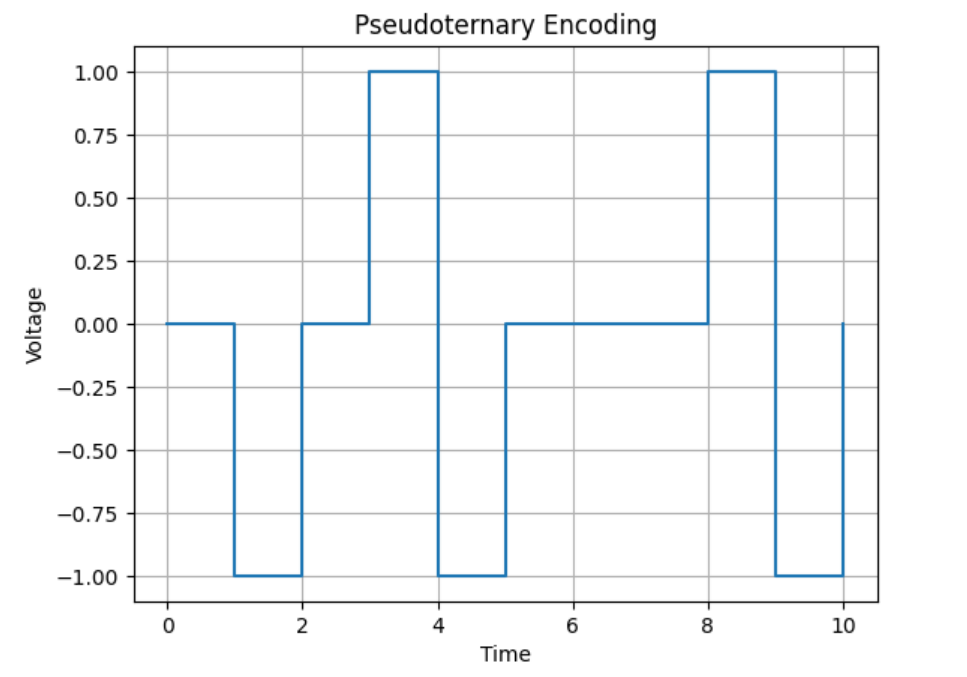


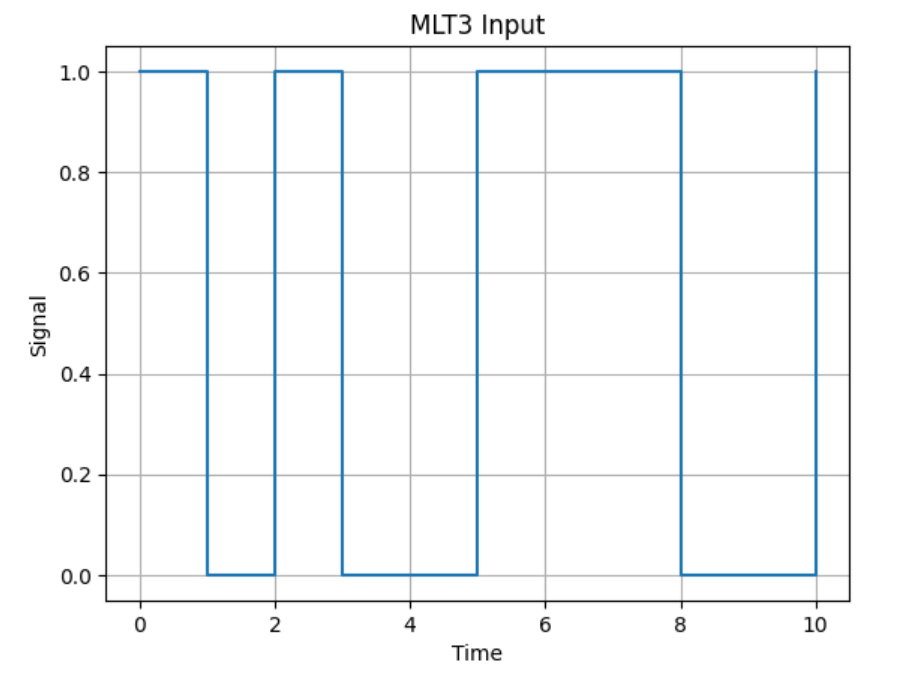


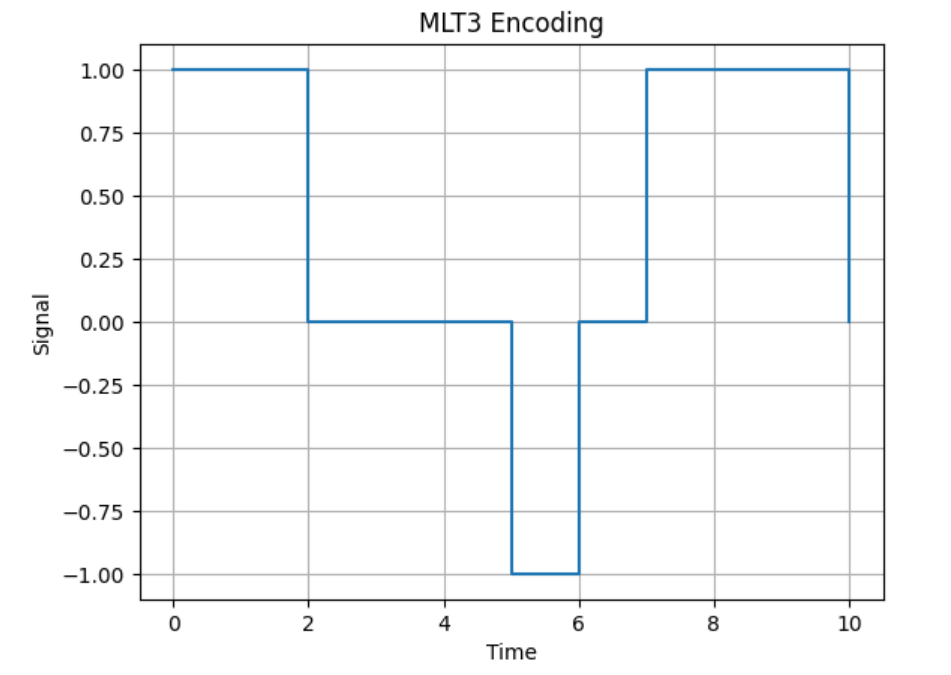












**Learning and Difficulties:**

* Accurate state transitions in NRZ-I and MLT3 encodings are to be ensured, as a small logic error could cause an incorrect decoded output.
* Confusion may occur between Manchester transitions and signal levels, especially when defining the exact point of mid-bit transition.
* Sign alternation in bipolar schemes like AMI and Pseudoternary must be managed with care, particularly when deciding how to toggle polarity consistently.
* Debugging mismatches between input and output sequences is often observed; hence, a detailed review of the encoding and decoding logic is required.

**Conclusion:**

We were successful in implementing different encoding schemes, such as NRZ-L, NRZ-I, Manchester, AMI, Pseudoternary, and MLT3, and learning about their benefits and drawbacks.