**Computer Network**

**Name of the student: Swaraj Phand**

**Roll No. 7**

**PRN- 12211292**

**Class- TY – D -B1**

**Div: C Batch: 3**

**Date of performance: 24/01/2024**

**Github link**- <https://github.com/Swaraj7700/Computer_Networks/tree/master/linecoding>

**Code -**

App.py

from flask import Flask, render\_template, request, Response

import numpy as np

import matplotlib.pyplot as plt

import io

from matplotlib.backends.backend\_agg import FigureCanvasAgg as FigureCanvas

app = Flask(\_\_name\_\_)

def unipolar\_nrz(data):

    return data

def polar\_nrz\_l(data):

    return [1 if bit == 1 else -1 for bit in data]

def polar\_nrz\_i(data):

    output = []

    last = 1

    for bit in data:

        if bit == 1:

            last \*= -1

        output.append(last)

    return output

def rz(data):

    rz\_output = []

    for bit in data:

        if bit == 1:

            rz\_output.extend([1, 0])

        else:

            rz\_output.extend([-1, 0])

    return rz\_output

def manchester(data):

    manchester\_output = []

    for bit in data:

        if bit == 1:

            manchester\_output.extend([1, -1])

        else:

            manchester\_output.extend([-1, 1])

    return manchester\_output

def differential\_manchester(data):

    output = []

    last = 1

    for bit in data:

        if bit == 1:

            output.extend([-last, last])

        else:

            last \*= -1

            output.extend([last, -last])

    return output

def ami(data):

    last = -1

    output = []

    for bit in data:

        if bit == 1:

            last \*= -1

            output.append(last)

        else:

            output.append(0)

    return output

def pseudoternary(data):

    last = -1

    output = []

    for bit in data:

        if bit == 0:

            last \*= -1

            output.append(last)

        else:

            output.append(0)

    return output

def plot\_line\_code(data, encoding\_function, title):

    encoded\_data = encoding\_function(data)

    t = np.arange(0, len(encoded\_data))

    fig, ax = plt.subplots()

    ax.step(t, encoded\_data, where='post')

    ax.set\_ylim(-2, 2)

    ax.set\_title(title)

    ax.set\_xlabel('Time')

    ax.set\_ylabel('Amplitude')

    ax.grid(True)

    return fig

@app.route('/', methods=['GET', 'POST'])

def index():

    if request.method == 'POST':

        input\_data = request.form['data']

        try:

            data = [int(bit) for bit in input\_data]

            figures = []

            figures.append(plot\_line\_code(data, unipolar\_nrz, 'Unipolar NRZ'))

            figures.append(plot\_line\_code(data, polar\_nrz\_l, 'Polar NRZ-L'))

            figures.append(plot\_line\_code(data, polar\_nrz\_i, 'Polar NRZ-I'))

            figures.append(plot\_line\_code(data, rz, 'RZ'))

            figures.append(plot\_line\_code(data, manchester, 'Manchester'))

            figures.append(plot\_line\_code(data, differential\_manchester, 'Differential Manchester'))

            figures.append(plot\_line\_code(data, ami, 'AMI'))

            figures.append(plot\_line\_code(data, pseudoternary, 'Pseudoternary'))

            output = io.BytesIO()

            fig, axes = plt.subplots(len(figures), 1, figsize=(10, 20))

            for ax, figure in zip(axes, figures):

                canvas = FigureCanvas(figure)

                figure.tight\_layout()

                canvas.draw()

                buf = canvas.buffer\_rgba()

                image = np.asarray(buf)

                ax.imshow(image)

                ax.axis('off')

            fig.tight\_layout()

            FigureCanvas(fig).print\_png(output)

            return Response(output.getvalue(), mimetype='image/png')

        except ValueError:

            return "Invalid input. Please enter a string of 0s and 1s."

    return render\_template('index.html')

if \_\_name\_\_ == '\_\_main\_\_':

    app.run(debug=True)

Index.html -

<!doctype html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <title>Line Encoding Plots</title>

</head>

<body>

  <h1>Line Encoding Plots</h1>

  <form method="post">

    <label for="data">Enter data (string of 0s and 1s):</label>

    <input type="text" id="data" name="data" required pattern="[01]+">

    <button type="submit">Generate Plots</button>

  </form>

  {% if request.method == 'POST' %}

    <h2>Generated Plots</h2>

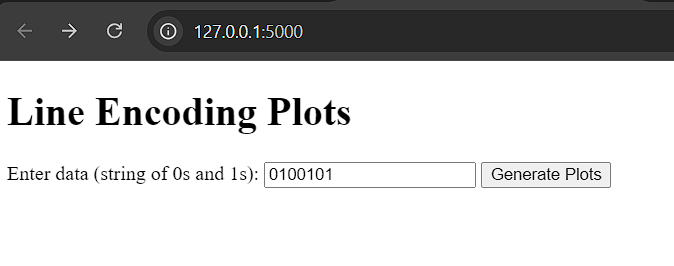
    <img src="{{ url\_for('index') }}">

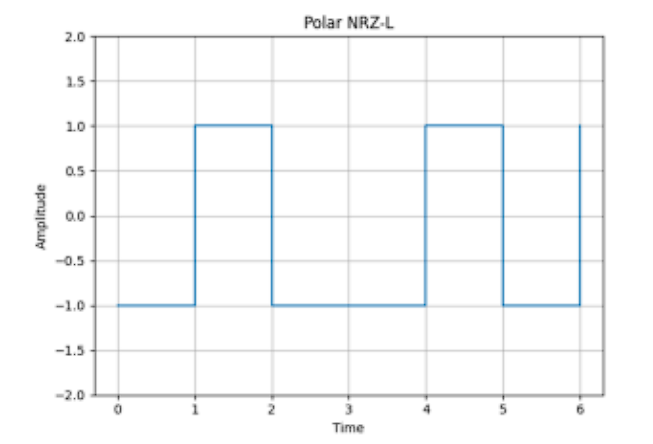
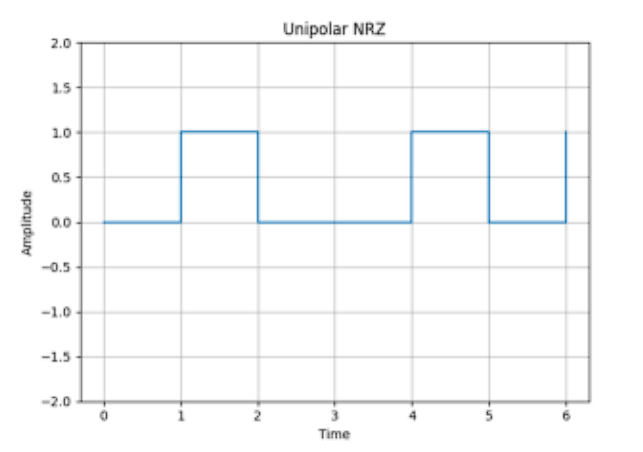
  {% endif %}

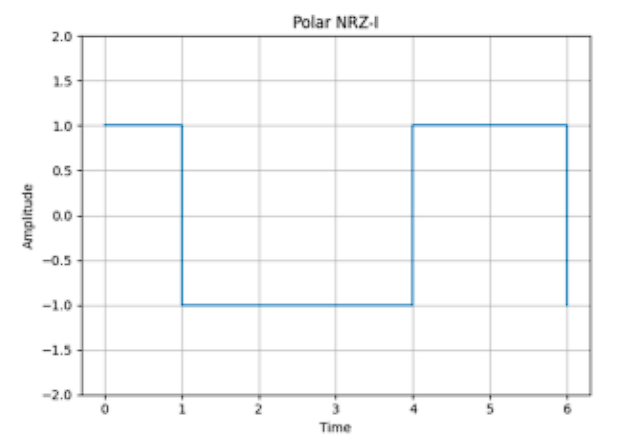
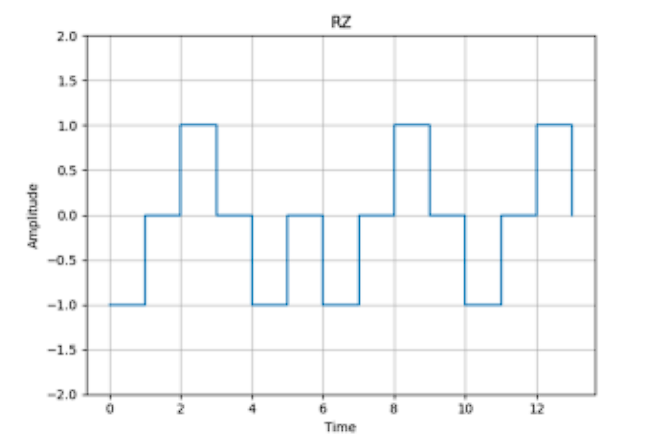
</body>

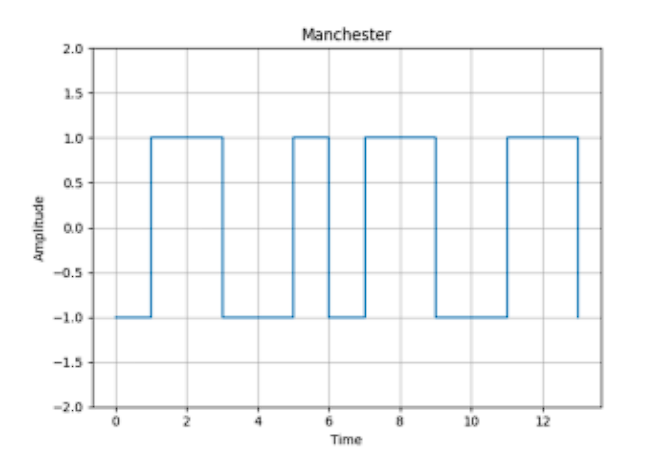
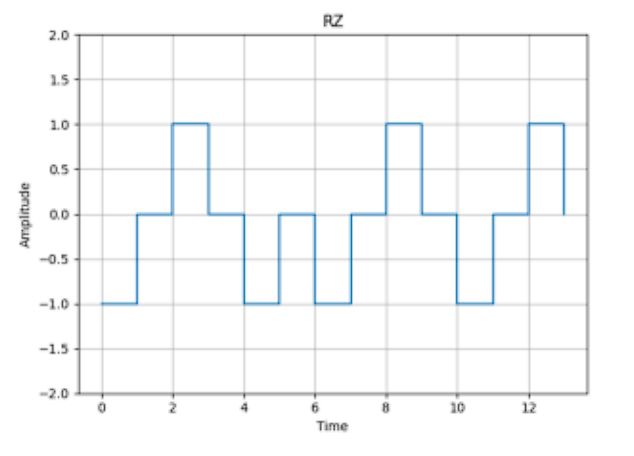
</html>

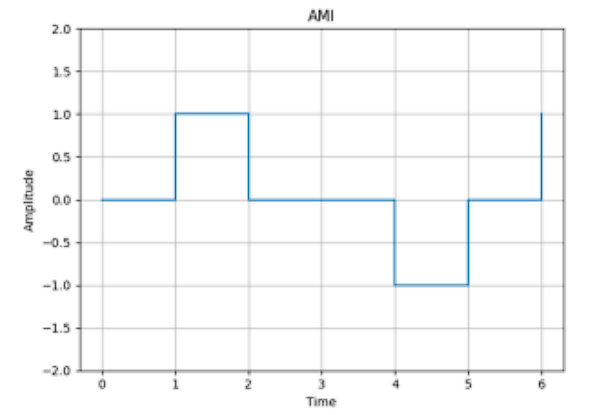
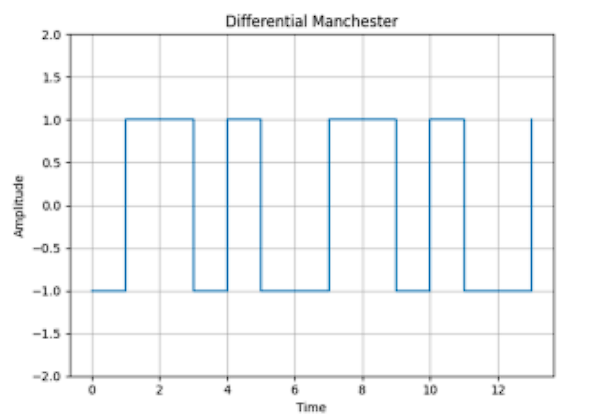
**Screenshots-**

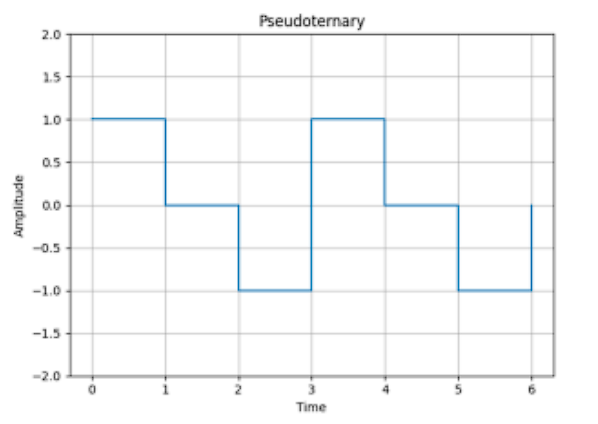












Conclusion –

Created a Flask web application to simulate various line encoding schemes, including Unipolar NRZ, Polar NRZ-L, Polar NRZ-I, RZ, Manchester, Differential Manchester, AMI, and Pseudoternary. The application takes a binary input string from the user, generates corresponding line encoding plots, and displays them in the browser. The application uses Matplotlib for plotting and handles user input validation to ensure only binary data is processed.