Digital_Signal_Generator Project_Report

🧠 Project Report — Digital Signal Generator

Course: ITT036 — Programming Assignment 1

Submitted To: Department of Information Technology, NIT Srinagar

Submitted By:

Hamza Mehmood (Enrollment No: 2023BITE048)

Omar (Enrollment No: 2023BITE002)

Date of Submission: 23 October 2025

Language Used: Python 3.13

🎯 Objective

The objective of this project is to design and implement a **Digital Signal Generator** that performs both **Line Coding** and **Digital Modulation** techniques. The system allows the user to input either **digital** or **analog** data and generates corresponding digital signals using selected encoding or modulation schemes.

The project also implements:

Scrambling (B8ZS / HDB3) for AMI signals.

PCM (Pulse Code Modulation) and **Delta Modulation** for analog-to-digital conversion.

Longest Palindrome Detection in data stream.

Graphical Signal Representation using `matplotlib`.

Interactive Web Interface built with `Streamlit`.

Decoding (Bonus Feature) — extra credit implementation.

🧩 System Overview

Main Functional Blocks:

User Input:
The user can choose between *analog* or *digital* input.

Line Coding (Digital Input):

NRZ-L

NRZ-I

Manchester

Differential Manchester

AMI

Scrambling: B8ZS / HDB3

Modulation (Analog Input):

Pulse Code Modulation (PCM)

Delta Modulation (DM)

Decoding (Extra Credit):

NRZ-L Decode

NRZ-I Decode

Manchester Decode

Differential Manchester Decode

AMI Decode

- AMI Decode

Graphical Output:

Digital signals are plotted using `matplotlib`.

Web Interface (via Streamlit):

- Allows users to select schemes interactively.
- Provides a “Decode Signal” checkbox to recover data.
- Supports both digital and analog paths.

🧱 Folder Structure

```
Digital Signal Generator/
├── app.py                          # Streamlit web app interface
├── main.py                         # CLI (terminal) version
├── requirements.txt                 # List of required Python packages
│
├── utils/
│   ├── __init__.py
│   ├── helpers.py                  # Input and validation functions
│   ├── palindrome.py               # Longest palindrome finder
│   ├── plotting.py                 # Matplotlib-based signal plotting
│
├── encoding/
│   ├── __init__.py
│   ├── line_coding.py              # NRZ-L, NRZ-I, Manchester, AMI, etc.
│   ├── scrambling.py               # B8ZS and HDB3 scrambling
│
├── modulation/
│   ├── __init__.py
│   ├── pcm.py                      # Pulse Code Modulation
│   ├── delta_modulation.py         # Delta Modulation
│
├── decoding/
│   ├── __init__.py
│   ├── line_decoding.py            # Decoding logic for each scheme
│
└── venv/                           # Virtual environment (optional)
```

🧰 Libraries & Tools Used

Library	Purpose
numpy	Numeric computation and signal representation
matplotlib	Plotting digital signals
streamlit	Building an interactive web interface
os, sys	File and system handling
venv	Virtual environment setup

🧠 Key Functional Components

1. Digital Input Workflow

- User provides a binary string (e.g., `1011001`).
- System computes the **longest palindrome**.
- User chooses a **line coding scheme**.
- System generates and plots the encoded waveform.
- Optional **scrambling (B8ZS / HDB3)** for AMI signals.
- Optional **decoding** to retrieve original bits.

2. Analog Input Workflow

User enters analog values (e.g., `0.1 0.5 0.8 0.7`).

User chooses **PCM** or **Delta Modulation**.

Signal is converted to a digital bitstream.

NRZ-L line encoding applied for visualization.

User can decode the modulated signal (NRZ-L decode).

3. Longest Palindrome Finder

Efficient algorithm implemented in `utils/palindrome.py` to identify the longest palindromic substring in the input bitstream.

4. Signal Plotting

All signals are visualized using `matplotlib`, ensuring proper labeling, amplitude levels, and bit alignment.

5. Decoding Module

Implemented separately in `decoding/line_decoding.py` to reverse encoded signals, demonstrating full communication cycle.

🧪 How to Run the Project

Step 1 — Unzip the project

Extract the ZIP file

Step 2 — Create and activate virtual environment (recommended)

macOS/Linux:

```
python3 -m venv venv
source venv/bin/activate
```

Windows:

```
python -m venv venv
```

```
.\venv\Scripts\Activate.ps1
```

Step 3 — Install dependencies

```
pip install -r requirements.txt
```

Step 4 — Run Streamlit Web App

```
streamlit run app.py
```

Then open the displayed URL (e.g. http://localhost:8501) in your browser.
If Streamlit asks for an email, simply press **Enter** to skip.

To disable the welcome message permanently:

```
streamlit config set global.showWelcomeMessage false
```

Step 5 — (Optional) Run in Terminal

For CLI-based operation:

```
python main.py
```

Follow the prompts directly in the terminal.

Step 6 — Exit

Press **Ctrl + C** to stop the app and

```
deactivate
```

to exit the virtual environment.

⚙️ requirements.txt

Below is the content for your `requirements.txt` file:

```
streamlit==1.39.0
matplotlib==3.9.2
numpy==2.1.2
```

(These versions are stable as of October 2025.)

🧮 Sample Input / Output

Example 1 (Digital Input)

```
Input: 1011001
Scheme: Manchester
Output: Encoded waveform (plotted)
Longest Palindrome: 101
Decoded Output: 1011001
```

Example 2 (Analog Input)

```
Input: 0.1 0.4 0.9 0.7
Modulation: PCM
Output: Digital stream + NRZ-L encoded waveform
Decoded: Reconstructed bitstream
```

🌐 Streamlit Features

Dropdown and radio button selection for easy scheme switching.

Real-time plotting inside browser.

Checkbox for decoding last encoded signal.

Works offline once dependencies are installed.

🏁 Conclusion

This project successfully simulates and visualizes various **line coding and digital modulation techniques**, demonstrating the transformation of analog and digital signals into digital form.
It fulfills all requirements of the ITT036 Programming Assignment and implements the **extra credit decoding feature**.

The project is modular, extendable, and compatible with both CLI and GUI environments.

👨‍💻 Credits

Developed by:

Hamza Mehmood (2023BITE048)

Ommer (2023BITE002)

Deepanshu (2023BITE052)

Institution: National Institute of Technology, Srinagar
Course: B.Tech Information Technology (2023–2027)
Subject: ITT036 — Programming Assignment 1 (Autumn 2025)