Exploring Signal Generation and Line Encoding: User's Guide

Divyam Billowria[1]

November 12, 2023

[1]2021BITE088, Information Technology.

0.1 Project Overview

The Line Encoder project is implemented in C++ and utilizes the SFML library for graphics. The purpose of the project is to generate digital signals based on user input, offering options for analog or digital input. The program proceeds with line encoding or Pulse Code Modulation/Differential Modulation (PCM/DM) based on the user's choice. Various encoding techniques, such as NRZ-L, NRZ-I, Manchester, Differential Manchester, and AMI, are provided to the user.

For finding the longest palindrome in the data stream, the project uses **Kanacher's algorithm**, which has a time complexity of O(n). **For handling analog input, working with hardware like Arduino was deemed infeasible, and C++ lacks robust APIs for live sound processing, without any intervention from third-party libraries.Therefore, the project uses a pre-downloaded .wav file**. The WAV file is parsed, and the samples are quantized. *Each sample is 16 bits long, and instead of performing line encoding on the entire data stream, the project chooses to plot 16 bits for just a single sample.*

0.1.1 For more details on the exact implementation , Please refer : My Github

0.2 Installation

To install Line Encoder, follow these steps:

```
git clone https://github.com/Mayvid0/Line-encoder
```

Ensure the SFML library is installed:

- macOS: `brew install sfml`
- Ubuntu: `sudo apt-get install libsfml-dev`
- Windows: Install SFML SDK

Navigate to the project directory:

```
cd Line-encoder
```

Modify the `makefile` for necessary headers and libraries. For macOS, you may not need changes, but for Linux, find the include paths and libraries:

```
grep -r "sfml" /usr/include
grep -r "sfml" /usr/lib
```

Update the `SFML_DIR_LINUX`, `SFML_DIR_WINDOWS`, or `SFML_DIR_MACOS` variables accordingly. Finally, run:

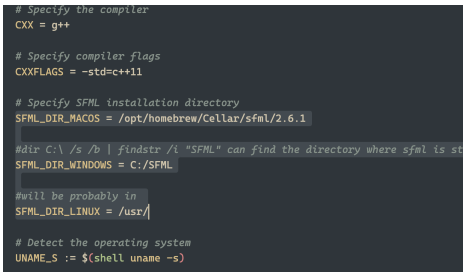

Figure 1: Edit these makefile variables accordingly.

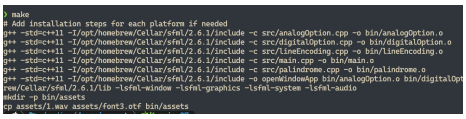

Figure 2: Build using make.

`make`

Execute the program:

`./openWindowApp`

0.3 Usage

The program prompts the user for digital signal generation preferences, offering various encoding techniques. It processes analog input for PCM/DM, generating digital data fed into line encoding techniques. The output includes the digital data stream, the longest palindrome in the data stream, the produced digital signal, and, in the case of scrambling, the scrambled signal.

0.4 Acknowledgments

- SFML Documentation
- Sound Files
- ChatGPT

```
❯ ./openWindowApp
Select Option:
1. Digital
2. Analog
Enter your choice (1 or 2): 1
Enter a binary data stream: 1100000000001010000

 What type of encoding would you like to have: (Enter 1-5)
1. NRZ-L
2. NRZ-I
3. Manchester
4. Differential Manchester
5. AMI
5

Would you like to scramble it:
1. B8ZS
2. HDB3
2
```

Figure 3: Run the executable for Digital data.

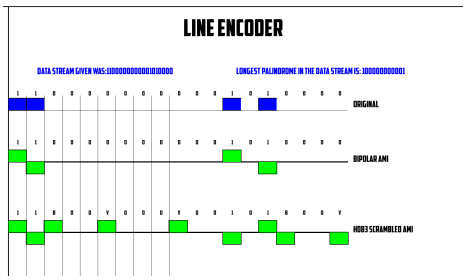

Figure 4: resulting window with Data stream, Longest Palindrome and selected Encoding with scrambling.

```
❯ ./openWindowApp
Select Option:
1. Digital
2. Analog
Enter your choice (1 or 2): 2
What Do you want to perform:
1. PCM
2. DM
2
```

Figure 5: Running for analog input.

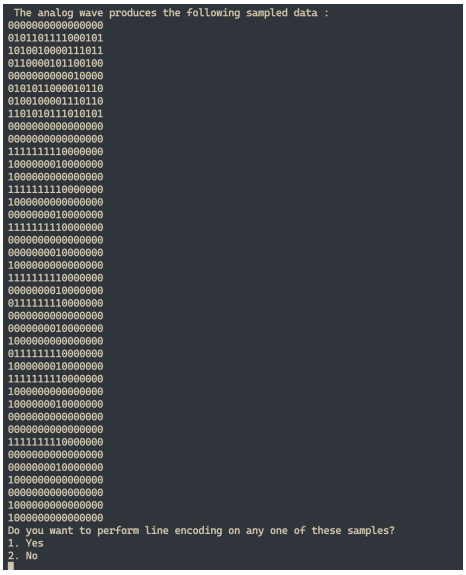

Figure 6: Samples from analog .wav file .

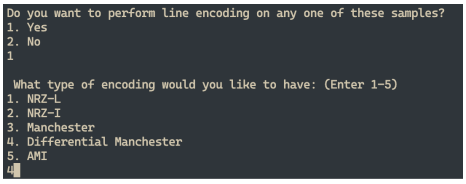

Figure 7: Encoding after selecting a random sample.

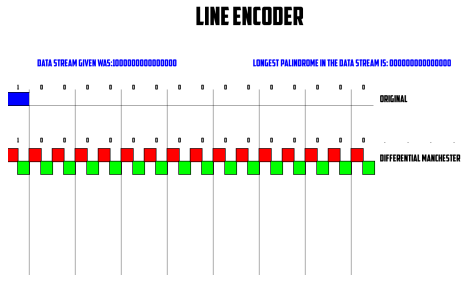

Figure 8: Digital stream and chosen encoding with longest palindrome.