Line Encoder Visualization Project Documentation

Group Members : Shashwat Saran, Pratham Kumar, Gaurav Kumar Soni

November 15, 2023

Overview

The Line Encoder Visualization project is a C++ program that utilizes the Simple DirectMedia Layer (SDL) library to create a graphical user interface for visualizing both digital and analog line encoding waveforms. The project aims to provide a user-friendly interface for inputting parameters such as frequency and amplitude and visualizing the corresponding waveforms.

Table of Contents

Libraries Used

The project primarily relies on the following libraries:

- **SDL (Simple DirectMedia Layer):** SDL is a cross-platform development library designed to provide low-level access to audio, keyboard, mouse, and display functions.
- **SDL_ttf:** This library is an extension to SDL that allows rendering TrueType fonts.

How It Works

The Line Encoder Visualization project follows a modular structure where different aspects of the application are encapsulated in functions and structures. The program initializes an SDL window and renderer, allowing users to interact with the application through a graphical user interface.

Users can choose between digital and analog modes through a radio buttons. In digital mode, the program provides functionalities related to digital line encoding, while in analog mode, it deals with continuous waveforms, and these functionalities can be selected using dropdown menu.

Building the Project from GitHub

To build the Line Encoder Visualization project from GitHub, follow these steps:

Prerequisites

Make sure you have the following tools installed on your system:

- **C++ Compiler:** Ensure you have a C++ compiler installed on your machine. Popular choices include GCC for Unix-like systems or MinGW for Windows.
- **CMake:** The project uses CMake for building. Install CMake on your system if you haven't already.

Clone the Repository

Clone the Line Encoder Visualization repository from GitHub:

```
https://github.com/GPS494712/DC_Assignment_lineEncoder
```

Build the Project

Navigate to the project directory:

```
cd DC_Assignment_lineEncoder
```

Run make to generate build files Compile the project:

```
make && ./main
```

Note

Adjust the commands based on your operating system and build environment. Ensure that the necessary dependencies, such as SDL and SDL_ttf, are available on your system.

Logic Implementation

Digital Mode Logic

- The program maintains a list of menu items for both digital and analog modes.
- The user can choose options from the menu, and the selected item appears on the dropdown title.
- The program includes logic for rendering the input fields and processing user input.
- These Encoding Techniques are being implemented : NRZ-L, NRZ-I, Manchester, diffrential Manchester, AMI and scramble techniques(B8ZS, HDB3).

Analog Mode Logic

- Analog mode involves drawing continuous waveforms based on user-specified parameters.
- The `draw_continuous_wave` function is responsible for rendering the continuous waveform using the SDL renderer.
- Input fields are provided for user input, such as frequency and amplitude..
- The waveform is generated based on a user-defined mathematical function, taking into account frequency, amplitude, and time.

PCM and Line Encoding

Pulse Code Modulation (PCM)

The Line Encoder Visualization project employs Pulse Code Modulation (PCM) to convert analog signals to digital form. PCM involves regular sampling, quantization of amplitude values, and encoding into a digital stream.

Integration with Line Encoding

PCM-encoded signals are integrated into the line encoding process for digital communication. Various line encoding algorithms are applied, and the results are visualized on the graphical interface.

Note

The PCM and line encoding logic allows users to visualize the conversion of analog signals to digital form and their subsequent encoding for transmission.

Graphics Implementation

- Graphics are implemented using the SDL library.
- The SDL renderer is used to draw various elements, including input fields, buttons, and waveform visualizations.
- The program handles user input events such as key presses and mouse clicks to enable user interaction with the graphical elements.