Program Documentation: CS 378 Lab 2

Atharva Bendale (22B0901), Nivesh Aggarwal (22B0912),
Dhvanil Gheewala (22B0923), Vishal Bysani (22B1061)

September 1, 2024

1 Included Files and Dependencies

Dependencies

- NumPy:

```
pip install numpy
```

- SciPy:

```
pip install scipy
```

- Pyaudio:

```
sudo apt-get install portaudio19-dev
pip install pyaudio
```

Files included

```
22b0901_22b0912_22b0923_22b01061_CS378_lab2.tar.gz/
├── main.py
├── receiver.py
├── sender.py
├── crc.py
└── Report/
    ├── 22b0901_22b0912_22b0923_22b01061_dd_CS378_lab2.pdf
    ├── 22b0901_22b0912_22b0923_22b01061_dd_CS378_lab2.tex
    ├── documentation.pdf
    └── documentation.tex
```

main.py

Main script for sending and receiving binary messages using CRC encoding, audio transmission, and error correction

Functions:

- **send :** Takes input message, number of errors and its position from the user , creates transmission message by appending special sequence, preamble and CRC bits
- **recv :** Calibrates the system for background noise, receives signals, decodes it to bits and corrects the error bits and prints the message

receiver.py

Receives and decodes audio signals into binary messages using Welch's method of frequency analysis
Functions:

- **open_audio_stream :** Opens the audio stream
- **receive_audio :** Receive an audio signal from the default audio input device for the specified duration
- **calibrate :** Calculates the ambient noise power for each frequency range
- **decode_audio_to_bits :** Decode an audio signal to a list of bits

sender.py

Generates and transmits audio signals representing binary messages using frequency modulation
Functions:

- **generate_tone :** Generate a sine wave tone of given frequency and duration
- **encode_bits_to_audio :** Encode a list of bits into an audio signal
- **send_audio :** Send an audio signal to the default audio output device

crc.py

Implements 2-bit error correction with CRC generating polynomials with length depending upon the length of the input message
Functions:

- **preamble :** returns the binary representation of preamble
- **polyDivision :** Performs polynomial division for CRC calculation
- **bruteCheck :** Iterates over all possible double/triple bit errors and checks whether the modified dividend is perfectly divisible by the polynomial for finding the correct message
- **encodeCrc :** Encodes the given message and adds redundancy using CRC for error detection and correction
- **decodeCrc :** Decodes the given transmission, detects and corrects errors

2 Usage

Sender's End

To initiate the transmission process, follow these steps:

```
python3 main.py --send
```

1. Enter the message you wish to transmit.
2. Specify the number of bit errors you want to introduce.
3. Provide the fractions at which you want to introduce the errors.
4. Press Enter to start the transmission.

Receiver's End

To receive and process the transmitted message, follow these steps:

```
python3 main.py --recv
```

1. Run the command above in the terminal.
2. Wait for the incoming transmission.
3. The received message will be displayed in the terminal.