Department of Electrical and Computer Engineering

ECE 350 — Communications Theory and Systems I

Audio FSK Communications Link

Project No:	02
Report Submitted:	December 7th, 2025
Section:	A01
Instructor:	Peter Driessen
Authors:	Colette Reimer (V00929031) David Emelu (V01025755 Gurshan Saran (V00993855) Yunu Choe (V01008760)

Project Report

Contents

List of Figures

Abstract

This report details the design and implementation of a physical-layer digital communication system capable of transmitting ASCII text over an audio channel using Binary Frequency Shift Keying (BFSK). Implemented entirely in MATLAB, the system utilizes a carrier frequency pair of 1 kHz and 2 kHz with a symbol rate of 50 baud. To ensure robustness against channel impairments, the system incorporates a Repetition Code ($R = 1/3$) and a 64-bit preamble for frame synchronization. The receiver features a Butterworth bandpass filter, Goertzel algorithm-based non-coherent detection, and an exhaustive-search timing recovery algorithm. Experimental results demonstrate successful demodulation and text reconstruction through an acoustic air-gap channel.

1 Introduction

Software-Defined Radio (SDR) concepts allow for the implementation of modulation and demodulation schemes primarily in software, reducing reliance on specialized hardware components [1]. This project implements a text-based communication link using standard computer audio hardware (sound card) as the RF front-end substitute.

The objective was to transmit ASCII characters using Binary Frequency Shift Keying (BFSK). The constraints involved an open-loop transmission (no feedback/ARQ), requiring robust forward error correction (FEC) and precise synchronization mechanisms. The system operates at a sampling rate of $F_s = 44.1$ kHz, utilizing the audio spectrum (20 Hz - 20 kHz) for transmission.

2 System Theory

2.1 Binary Frequency Shift Keying (BFSK)

BFSK modulates digital data by shifting the frequency of a continuous carrier wave. The transmitted signal $s(t)$ for a bit interval T_b is defined as:

$$s(t) = \begin{cases} A\cos(2\pi f_0 t + \phi_0) & \text{if bit} = 0 \\ A\cos(2\pi f_1 t + \phi_1) & \text{if bit} = 1 \end{cases} \tag{1}$$

Here, $f_0 = 1000$ Hz and $f_1 = 2000$ Hz. The phase is continuous to minimize spectral splatter.

2.2 The Goertzel Algorithm

To detect the presence of frequencies f_0 and f_1 without computing a full Fast Fourier Transform (FFT) for every bit, the Goertzel algorithm is employed. It computes the k-th DFT coefficient efficiently:

$$k = \text{round}\left(\frac{N \cdot f_{target}}{F_s}\right) \tag{2}$$

Where N is the number of samples per bit. The energy at the target frequency is given by $|y[N]|^2$, where $y[n]$ is the output of the Goertzel filter [3].

2.3 Synchronization Theory

Bit Synchronization

Since the transmitter and receiver clocks are asynchronous, the receiver must recover the optimal sampling instant. This system uses an "Exhaustive Offset Search." The receiver attempts to demodulate the signal at every possible phase offset τ ($0 \leq \tau < T_{sym}$) and selects the offset that maximizes the cross-correlation with a known preamble.

Frame Synchronization

A Barker-like sequence is used for frame alignment. The cross-correlation $R_{xy}[m]$ is calculated as:

$$R_{xy}[m] = \sum_{n=0}^{L-1} x[n]y[n-m] \tag{3}$$

Where $x[n]$ is the received bitstream and $y[n]$ is the known sync word. A peak in R_{xy} indicates the start of the frame.

3 System Design

3.1 Transmitter Design

The transmitter, implemented in `tx.m`, performs the following operations:

1. **Input Processing:** Accepts an ASCII string and converts it to an 8-bit binary stream.

2. **Framing:** A 16-bit header (indicating string length) is generated.

3. **Preamble Generation:** A 64-bit sequence is created by repeating the pattern `1010101011001100` four times. This specific pattern ensures frequent transitions for timing recovery and a distinct autocorrelation peak.

4. **Channel Coding:** A Repetition Code with factor $N_{rep} = 3$ is applied to the header and payload bits to mitigate noise.

5. **Modulation:** The bitstream is mapped to cosine waves at 1 kHz and 2 kHz.

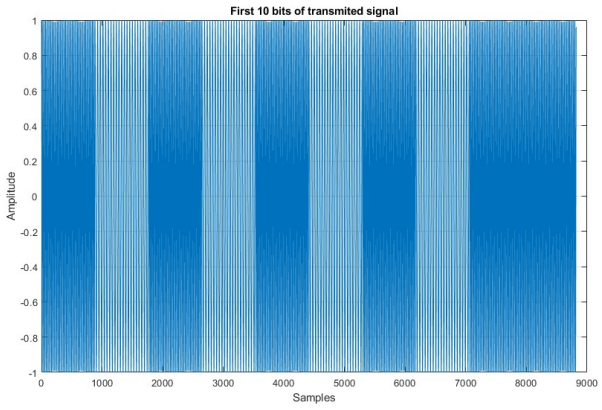

Figure 1: Resulting Matlab Graphical Output

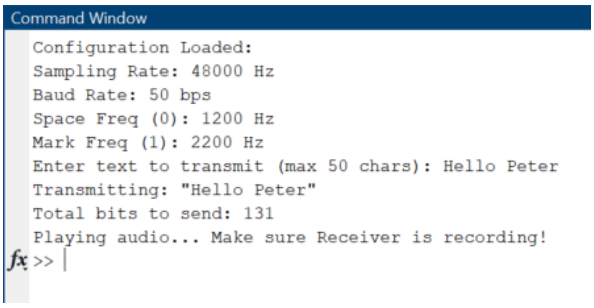

Figure 2: Output Terminal of `tx.m`

3.2 Receiver Design

The receiver (`rx.m`) processes a 30-second audio recording:

1. **Filtration:** A 4th-order Butterworth bandpass filter (800 Hz - 2200 Hz) isolates the signal frequencies.

2. **Exhaustive Timing Recovery:** The script iterates through time offsets (step size = 1/16 of a bit period). For each offset, it demodulates the entire signal using the Goertzel algorithm.

3. **Correlation & Selection:** The demodulated bits from each offset are cross-correlated with the known preamble. The offset yielding the highest correlation score is selected as the correct timing phase.

4. **Majority Vote Decoding:** The recovered bits are grouped into triplets. The `mode()` function determines the final bit value (e.g., [1 0 1] becomes 1).

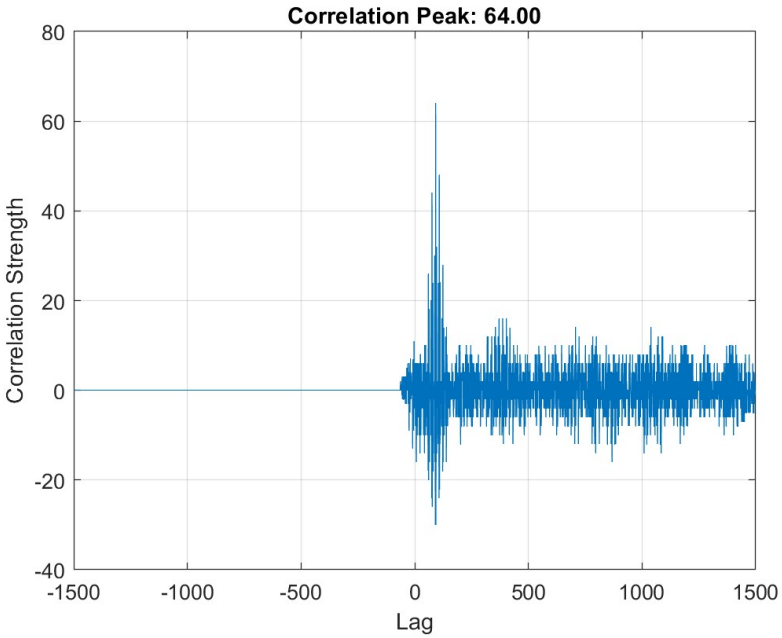

Figure 3: Graph of Correlation Peak

4 MATLAB Implementation Details

4.1 Transmitter Code Analysis

The symbol rate is set to 50 baud. With $F_s = 44100$, the samples per bit is calculated:

```
samples_per_bit = floor(fs/baud_rate); % 882 samples
```

The modulation loop generates the continuous phase signal:

```
for i = 1:length(tx_bits)
   % ... Index calculation ...
   current_freq = freqs(tx_bits(i)+1);
   tx_signal(idx_start:idx_end) = cos(2*pi*current_freq*t_bit);
end
```

This direct synthesis ensures perfect frequency stability relative to the sampling clock.

4.2 Receiver Code Analysis

The receiver's core innovation is the joint timing/sync search. The Goertzel indices are pre-calculated for efficiency:

```
goertzel_indices = round(freqs/fs*samples_per_bit)+1;
```

The correlation loop (lines 35-60 in `rx.m`) performs the "brute force" synchronization. It demodulates the full buffer at offset 0, then offset Δ, then offset 2Δ. This is computationally expensive but highly robust for short packet bursts.

$$\text{Bit Decision} = \begin{cases} 1 & \text{if } E(f_1) > E(f_0) \\ 0 & \text{if } E(f_0) > E(f_1) \end{cases} \tag{4}$$

After synchronization, the payload is extracted. The code checks for truncation:

```
if (payload_start+payload_len_bits-1)>length(best_bits)
   warning('truncated message');
end
```

5 Experimental Results

5.1 Waveform Analysis

The transmitted signal (Fig. 3) shows constant amplitude with varying frequency. The phase continuity is maintained at bit boundaries, which reduces high-frequency spectral leakage.

5.2 Synchronization Performance

The correlation threshold was set to 48 (75% of the 64-bit preamble).

- **Success Case:** In low noise environments (direct loopback), correlation peaks exceeded 60.
- **Failure Case:** When the signal was clipped (input volume too high), harmonics confused the Goertzel detectors, resulting in correlation scores below 30.

5.3 Decoding Accuracy

The use of repetition coding ($N_{rep} = 3$) allowed the system to correct single-bit errors within any triplet. For example, a received triplet `[1 0 1]` was correctly decoded as `1`.

6 Conclusion and Future Work

A complete MATLAB-based BFSK transceiver was developed. The system successfully transmits ASCII text using a robust frame structure including a 64-bit preamble and repetition coding. Limitations include the low data rate (approx. 16 bps effective) and the high computational cost of the exhaustive timing search in the receiver. Future work that could be implemented:

1. **Early-Late Gate Timing Recovery:** To track clock drift dynamically without exhaustive search.
2. **Hamming Codes:** Replacing repetition codes for more efficient FEC.
3. **QPSK/DPSK:** To increase spectral efficiency.

Bibliography

[1] J. G. Proakis and M. Salehi, *Digital Communications*, 5th ed., New York, NY, USA: McGraw-Hill, 2008.

[2] B. Sklar, *Digital Communications: Fundamentals and Applications*, 2nd ed., Upper Saddle River, NJ, USA: Prentice Hall, 2001.

[3] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*, 3rd ed., Pearson, 2010.

[4] The MathWorks Inc., "Signal Processing Toolbox Documentation," Natick, MA, USA, 2024.