# EE451 PROJECT PROGRESS REPORT

**Students:** Harun Durmuş **Student ID's:** 270206025 **Date:** 22.11.2023

Mustafa Eren Çelebi 270206053

Necmettin Yiğit Dübek 260206049

## Abstract

The subject covered for the last two weeks are mentioned in this progress report. After a brief research made about what is equalizer filters and which methods will be applied specified channels. It is understood that the process is about processing audio signal as an example and observing the channel response. The main point for this project is mostly about understanding channel effect between the process in transmitter and receiver parts.

## Objectives

- Analyzing and determining which digital modulation will be upload to an audio signal considering efficiency.
- Searching channel response models from several resources and obtaining channel coefficients.

## 1. Modulation Type

After searching and testing two modulation types, it is observed that Phase Shift Keying modulation (PSK)[1] is more appropriate option to implement. Other modulation types such as PPM and PCM will not be applied for the process because of their primitiveness.

```
%% PSK (Phase Shift Keying) on audio signal
%% Reading Audio File
% Read the audio file

[x, fs] = audioread('tired.wav');

% Set the parameters for PSK modulation
M = 4; % 4-phase PSK (you can choose other values like 2, 8, 16, etc.)
symbolRate = 1000; % Symbol rate in symbols per second
Ts = 1/symbolRate; % Symbol duration

% Generate a time vector for the PSK modulation
t = 0:1/fs:length(x)/fs-1/fs;

% Generate a binary sequence based on the audio file
data = randi([0 M-1], 1, round(length(x)*symbolRate/fs));

% Apply PSK modulation
modulatedSignal = pskmod(data, M, pi/M); % Phase shift in radians
```

```
% Upsample the modulated signal to match the original audio file's length
upsampledSignal = upsample(modulatedSignal, round(length(x)*symbolRate/fs));
upsampledSignal = upsampledSignal(1:length(x)); % Trim to match the length of the
original signal


% Apply the modulated signal to the original audio signal
y = x + upsampledSignal(1:length(x));

% Save the modulated audio to a new file
outputFilename = 'modulated_tired.wav';
audiowrite(outputFilename, y, fs);
```

This is the MATLAB code of the PSK modulation which is failed to implement. We are expect to learn to process better after the laboratory experiment about PSK that will be applied next weeks. Then, it will be easier to take a step forward towards further processes. Here is an another modulation type Pulse Position Modulation (PPM)[2]:

```
%% PPM (Pulse Position Modulation) on audio signal
%% Reading Audio File
[audio,fs] = audioread('tired.wav');
audio = audio / max(abs(audio));
threshold = 0.5;
binary_sequence = (audio > threshold);
%% Applying PPM
time_interval = 0.01;
symbol_duration = 0.005;

ppm_signal = zeros(1, round(length(binary_sequence) * symbol_duration * fs));
symbol_index = 1;

for i = 1:length(binary_sequence)
    if binary_sequence(i) == 1
        pulse_position = round((symbol_index - 1) * symbol_duration * fs + 1);
        ppm_signal(pulse_position : pulse_position + round(symbol_duration * fs) -
1) = 1;
    end

    symbol_index = symbol_index + 1;
end
%% Demodulation
Pavg= sum(abs(ppm_signal).^2)/length(ppm_signal);
snr_db=[30 15 0];
snr_lin(1)=10^(0.1*snr_db(1));
snr_lin(2)=10^(0.1*snr_db(2));
snr_lin(3)=10^(0.1*snr_db(3));
var(1)=Pavg/snr_lin(1);
var(2)=Pavg/snr_lin(2);
var(3)=Pavg/snr_lin(3);
awgn30=sqrt(var(1))*randn(1,length(ppm_signal));
awgn15=sqrt(var(2))*randn(1,length(ppm_signal));
awgn0=sqrt(var(3))*randn(1,length(ppm_signal));
rt30 = ppm_signal + awgn30;
rt15 = ppm_signal + awgn15;
rt0 = ppm_signal + awgn0;
channel_coefficients = [1, 0.5, 0.2];  % Example channel coefficients
%% Zero Forcing Equalizer
```

```
equalizer_taps = fliplr(pinv(toeplitz(channel_coefficients)));
rt30 = conv(transpose(rt30), channel_coefficients);
rt30 = conv(rt30, equalizer_taps(:));

rt15 = conv(transpose(rt15), channel_coefficients);
rt15 = conv(rt15, equalizer_taps(:));

rt0 = conv(transpose(rt0), channel_coefficients);
rt0 = conv(rt0, equalizer_taps(:));

demodulated_signal30 = zeros(1, length(rt30));
demodulated_signal15 = zeros(1, length(rt15));
demodulated_signal0 = zeros(1, length(rt0));

for i = 1:length(demodulated_signal30)
    if rt30(i) > threshold
        demodulated_signal30(i) = 1;
    else
        demodulated_signal30(i) = 0;
    end
end

for i = 1:length(demodulated_signal15)
    if rt15(i) > threshold
        demodulated_signal15(i) = 1;
    else
        demodulated_signal15(i) = 0;
    end
end

for i = 1:length(demodulated_signal0)
    if rt0(i) > threshold
        demodulated_signal0(i) = 1;
    else
        demodulated_signal0(i) = 0;
    end
end
%% MMSE Equalizer
mmse_equalizer = inv(conv(channel_coefficients, conj(flip(channel_coefficients)))
+ var(3));
equalized_signal0 = zeros(size(rt0));
equalized_signal0 = conv(rt0, mmse_equalizer, 'full');
demodulated_signal0 = zeros(size(equalized_signal0));
demodulated_signal0 = equalized_signal0(i, :) > threshold;
figure(1)
plot(demodulated_signal0);
```

It is decided that operation is not truly applied. Also it is observed that, simulation will be applied on a very short audio signals since the computational load on MATLAB.

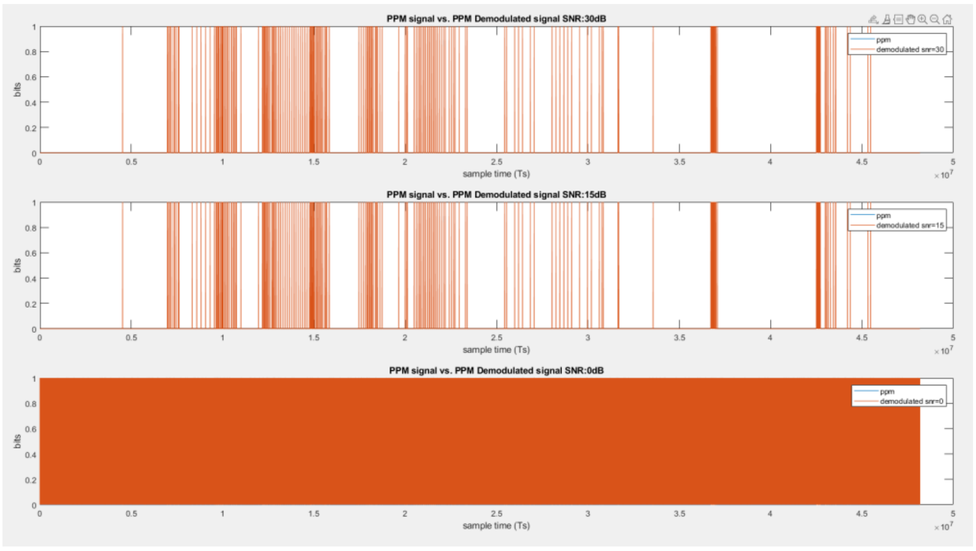

**Figure 1:** PPM signal vs. Demodulated signal over different SNR values

## 2. Channel Modeling

Here is the sample channel creation with MATLAB:

```
numSamples = 1000;  % Number of samples in the signal
SNR_dB = 20;       % Signal-to-Noise Ratio in decibels
t = linspace(0, 1, numSamples);
signal = sin(2 * pi * 5 * t);  % Example signal (5 Hz sinusoidal)
SNR_linear = 10^(SNR_dB / 10);
noisePower = var(signal) / SNR_linear;
noise = sqrt(noisePower) * randn(1, numSamples);
receivedSignal = signal + noise;
noiseCoefficients = sqrt(var(signal) / 10^(SNR_dB / 10)) * randn(1, numSamples);
```

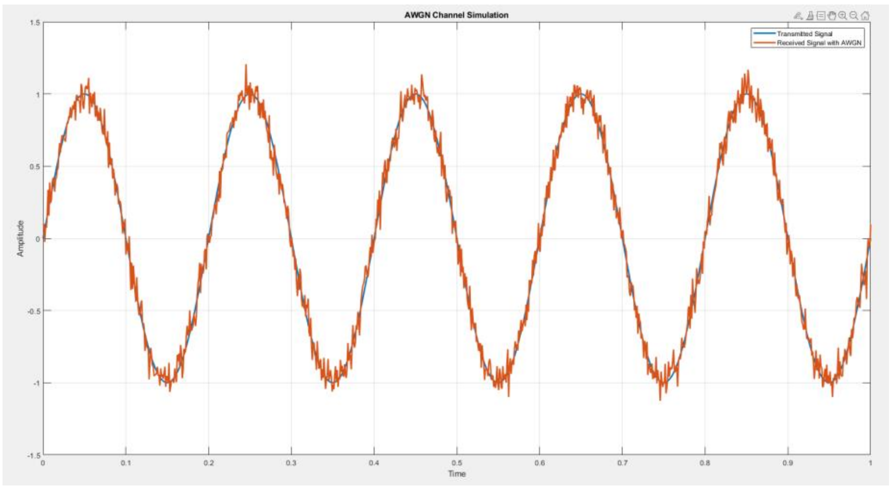

**Figure 2:** Signal through AWGN Channel

The chapter from “Fundamentals of Communication Systems”[3] introduces the mathematical model for an information source, including a measure of information and compression bounds based on source entropy. The focus now shifts to the communication channel, a critical component in communication systems, along with the concept of coding for error protection.

Communication channels, encompassing mediums like coaxial cable and fiber optic cables, exhibit complex input-output relations due to factors such as attenuation, noise, and bandwidth limitations. Practical channels are often waveform channels, treated as discrete-time channels, with input and output as discrete-time signals.

Discrete channels, like binary-input binary-output channels, may have memory, but all models discussed here are memoryless. An example is an additive white Gaussian noise channel with binary antipodal signaling.

The most notable continuous alphabet channel discussed is the discrete-time, additive white Gaussian noise channel with an input power constraint. This channel involves real-number inputs and outputs and is subject to a power constraint on the inputs.

## Conclusion

In conclusion, the progress report highlights the initial stages of the project, focusing on the modulation types, specifically Phase Shift Keying (PSK) and Pulse Position Modulation (PPM), applied to an audio signal. The attempted implementation of PSK modulation faced challenges and is slated for further refinement in upcoming laboratory experiments. PPM, on the other hand, was explored and simulated under different Signal-to-Noise Ratio (SNR) values, showcasing its behavior in noisy environments.

The report also delves into channel modeling, exemplified by a MATLAB-generated signal through an Additive White Gaussian Noise (AWGN) channel. The discussion broadens with insights from "Fundamentals of Communication Systems," emphasizing the importance of understanding channel behavior in communication systems.

It is acknowledged that the current operations were not executed as intended, and limitations, such as the computational load in MATLAB, were identified. The decision to conduct simulations on short audio signals was made in response to these constraints.

Moving forward, the project anticipates further experimentation and refinement, particularly in the modulation techniques, with the ultimate goal of gaining a comprehensive understanding of channel effects in the transmission and reception processes. The challenges encountered serve as valuable learning experiences, guiding future steps and contributing to the overall progress of the project.

## References

[1] Proakis, John G. *Digital Communications*. 4th ed. New York: McGraw Hill, 2001.

[2] Jigyansha Jeevan (2023). Pulse-position modulation (PPM), Retrieved November 22, 2023.

[3] Fundamentals of Communication Systems, John G. Proakis and Masoud Salehi, 2nd edition, 2005.