Project Name: Hybrid Communication System with Multi-Mode Transmission and Synchronization

Prepared By:
1. Arafat Hossain Rupok
ID:20210205021
2. Ayon Rahman
ID: 20210205008
3. Labiba Farazi Sara
ID: 20210205009
4. Tabah Maimuna Momo
ID: 20210105078

Project Overview:

This MATLAB App is a Hybrid Communication System that allows users to record speech/music, modulate it using AM, FM, or SSB, introduce noise and delay, demodulate the signal, and then play back the received audio.

IntroductiontoAmplitudeModulation

DefinitionofAM

Amplitude Modulation (AM) is a fundamental technique in electronic communication where the amplitude of a carrier wave is varied in proportion to the amplitude of a message signal. This process encodes information onto the carrier wave, allowing it to be transmitted over long distances.
The key advantage of AM is its simplicity and ease of implementation, but it is more susceptible to noise and interference compared to other modulation techniques, such as Frequency Modulation (FM).

Types of AM Implemented:

1. AM with Large Carrier (AM-LC): Maintains the carrier component for easier demodulation.
2. Double Sideband Suppressed Carrier (AM-DSBSC): Suppresses the carrier for better power efficiency

Applications

Amplitude Modulation (AM) is widely used in various fields due to its simplicity and cost-effectiveness. It is commonly employed in AM radio broadcasting, which transmits audio signals over long distances. In aviation communication, AM is used for air-to-ground and ground-to-air communication because of its reliability and ability to handle multiple channels without interference.

Additionally, AM is utilized in two-way radios, such as walkie-talkies and CB radios, for effective short-range communication.

Project Objectives

Objective of this project is to develop a MATLAB-based Hybrid Communication System App that allows users to record, modulate, transmit, and demodulate audio signals using AM, FM, and SSB techniques. It aims to provide an interactive and educational platform for understanding signal processing concepts, including noise effects, time delay, and performance evaluation.

Methodology Overview

Signal Generation & Processing

Recording voice signals.
Recording Music Signal

MATLAB Code Development

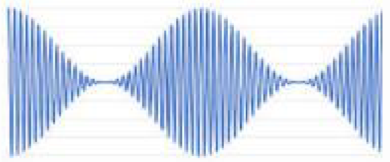
1.Writing scripts for AM FM and SSB Modulation and

.

demodulation.

.

2.Implementing visualization tools



Modulation & Demodulation

Implementing AM-LC and AM-DSBSC by multiplying the carrier with the filtered voice signal.
Using envelope detection for AM-LC and synchronous detection for AM-DSBSC.

Visualization

🎯 Hybrid Communication System App

🎤 Record Speech/Music

🎚 Modulate (AM/FM/SSB)

🎛 Add Noise & Delay -

📡 Transmit Signal

🔄 Demodulate

🎧 Play Received Audio

User Interface Elements

UI Element	Function
Record Speech/Music Buttons	Records audio input
Modulation Type (AM, FM, SSB) Buttons	Selects modulation method
Modulate Button	Modulates the signal
Noise & Delay Sliders	Adjusts noise level and delay
Demodulate Button	Recovers the original signal
Play Audio Button	Plays the received signal
Signal Plot	Displays modulated/demodulated signal

MATLAB Code Development

User Inputs & Initialization

- speech_signal & music_signal: Store recorded speech and music signals.
- modulated_signal & received_signal: Store the modulated and received signals after transmission.
- fs (22050 Hz): The sampling frequency for audio recording and processing.
- fc_am, fc_fm, fc_ssb: Carrier frequencies for Amplitude Modulation (AM) (1000 Hz), Frequency Modulation (FM) (2000 Hz), and Single Sideband (SSB) (3000 Hz).

```
properties (Access = private)
    speech_signal = []
    music_signal = []
    modulated_signal = []
    received_signal = []
    fs = 22050; % Sampling frequency
    fc_am = 1000; % AM Carrier Frequency
    fc_fm = 2000; % FM Carrier Frequency
    fc_ssb = 3000; % SSB Carrier Frequency
end
```

Recording Audio

This function records, normalizes, and stores speech audio, making it ready for modulation and transmission. 🎶🎤🔊🚀

```
methods (Access = private)

    function recordSpeech(app)
        duration = 5;
        recObj = audiorecorder(app.fs, 16, 1);
        recordblocking(recObj, duration);
        app.speech_signal = getaudiodata(recObj);
        app.speech_signal = app.speech_signal / max(abs(app.speech_signal)
        disp('Speech recorded successfully.');
    end
```

This function records, normalizes, and stores Music audio, making it ready for modulation and transmission. 🎶🎤🔊🚀

```
function recordMusic(app)
    duration = 5;
    recObj = audiorecorder(app.fs, 16, 1);
    recordblocking(recObj, duration);
    app.music_signal = getaudiodata(recObj);
    app.music_signal = app.music_signal / max(abs(app.music_signal)
    disp('Music recorded successfully.');
end
```

Modulates a recorded speech or music signal based on the selected modulation type (AM, FM, or SSB)

Modulation Techniques

- AM (Amplitude Modulation) for speech using a carrier (fc_am).
- FM (Frequency Modulation) for music using(fm_mod.)
- SSB (Single Sideband Modulation) for music using the Hilbert transform.

Copy Edit

```
function modulateSignal(app)
    if isempty(app.speech_signal) && isempty(app.music_signal)
        uialert(app.UIFigure, 'No recorded signal available!', 'Error');
        return;
    end

    t = (0:max(length(app.speech_signal), length(app.music_signal)) - 1) / app.fs;

    if app.AMButton.Value && ~isempty(app.speech_signal)
        app.modulated_signal = (1 + app.speech_signal) .* cos(2 * pi * app.fc_am * t)
    elseif app.FMButton.Value && ~isempty(app.music_signal)
        app.modulated_signal = fmmod(app.music_signal, app.fc_fm, app.fs, 50);
    elseif app.SSBButton.Value && ~isempty(app.music_signal)
        hilbert_signal = hilbert(app.music_signal);
        app.modulated_signal = real(hilbert_signal .* exp(1j * 2 * pi * app.fc_ssb * t
    else
        uialert(app.UIFigure, 'Select a valid modulation type and ensure signal is re
        return;
    end
```

Generate noise

```
function addNoiseAndDelay(app)
    if isempty(app.modulated_signal)
        uialert(app.UIFigure, 'No modulated signal available!', 'Error');
        return;
    end

    noise_level = app.NoiseSlider.Value;
    delay_samples = round(app.DelaySlider.Value);

    noise = noise_level * randn(size(app.modulated_signal));
    app.received_signal = app.modulated_signal + noise;
```

Apply delay

```
% Add delay safely
if delay_samples > 0
    app.received_signal = [zeros(delay_samples, 1); app.received_signal
end
```

Demodulates a Received signal

Demodulation:

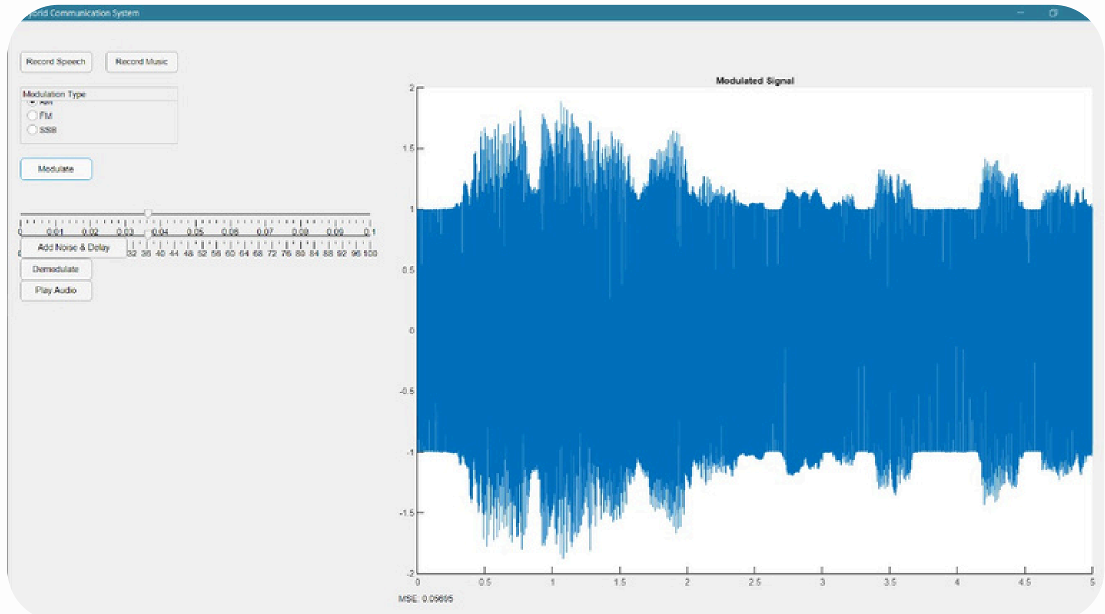
- AM: Uses the Hilbert transform to recover the signal's envelope (abs(hilbert)).
- FM: Applies FM demodulation using fmdemod with the given carrier frequency (fc_fm).
- SSB: Uses the Hilbert transform and complex exponential to demodulate the SSB signal.

```
function demodulateSignal(app)
    if isempty(app.received_signal)
        uialert(app.UIFigure, 'No received signal available!', 'Error');
        return;
    end

    t = (0:length(app.received_signal)-1) / app.fs;
    recovered = [];

    if app.AMButton.Value && ~isempty(app.speech_signal)
        recovered = abs(hilbert(app.received_signal));
    elseif app.FMButton.Value && ~isempty(app.music_signal)
        recovered = fmdemod(app.received_signal, app.fc_fm, app.fs, 50);
    elseif app.SSBButton.Value && ~isempty(app.music_signal)
        hilbert_received = hilbert(app.received_signal);
        recovered = real(hilbert_received .* exp(-1j * 2 * pi * app.fc_ssb * t)
    else
        uialert(app.UIFigure, 'Demodulation failed. Ensure correct settings!',
        return;
    end
```

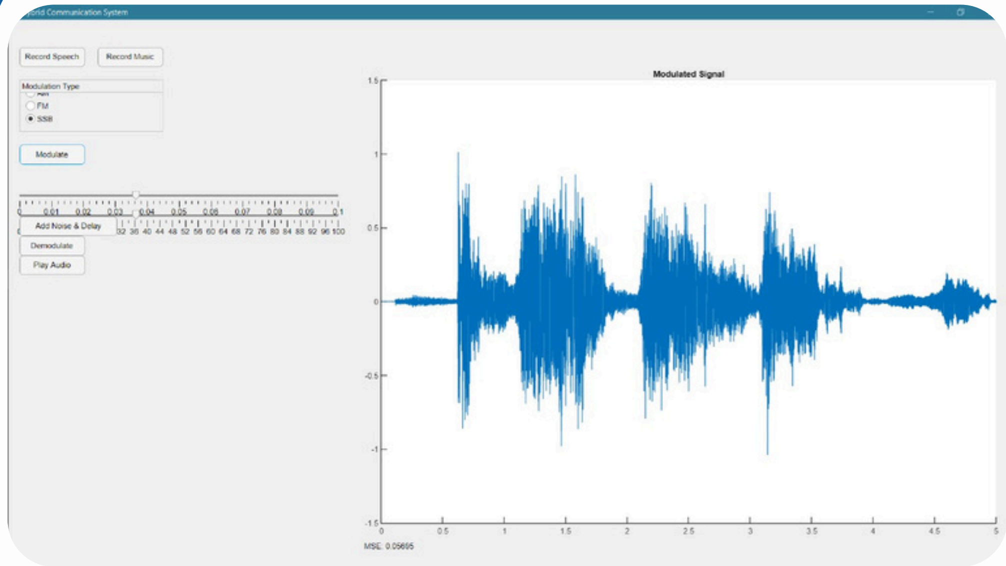
Plays the Received signal
As Audio

```
function playAudio(app)
    if isempty(app.received_signal)
        uialert(app.UIFigure, 'No received signal available!', 'Error');
        return;
    end
    soundsc(app.received_signal, app.fs);
    disp('Playing received audio.');
end
```

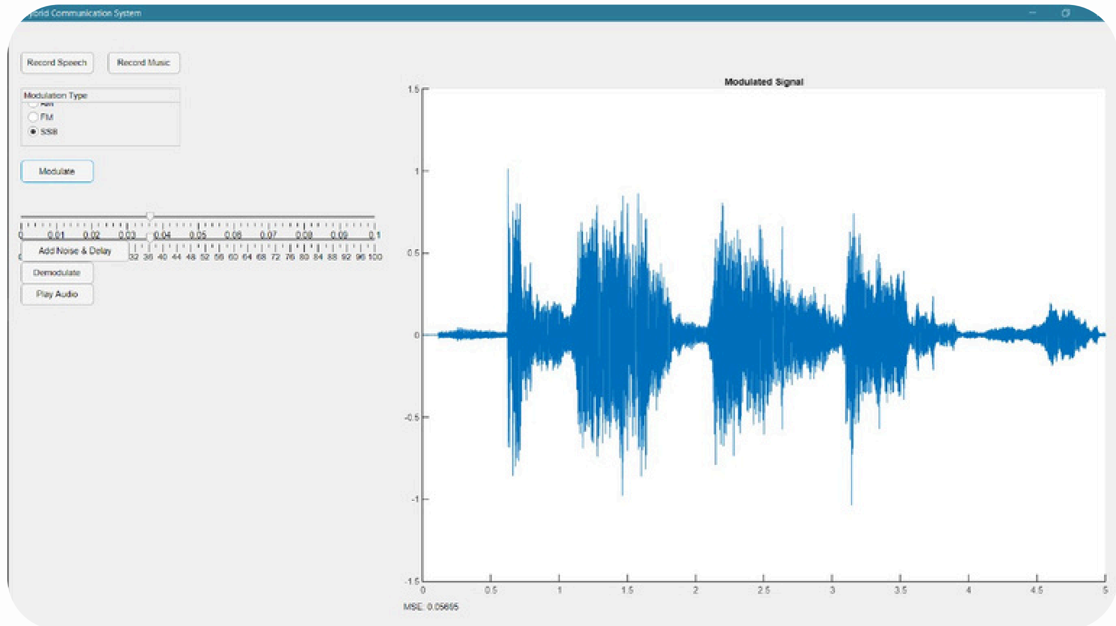
Visualization



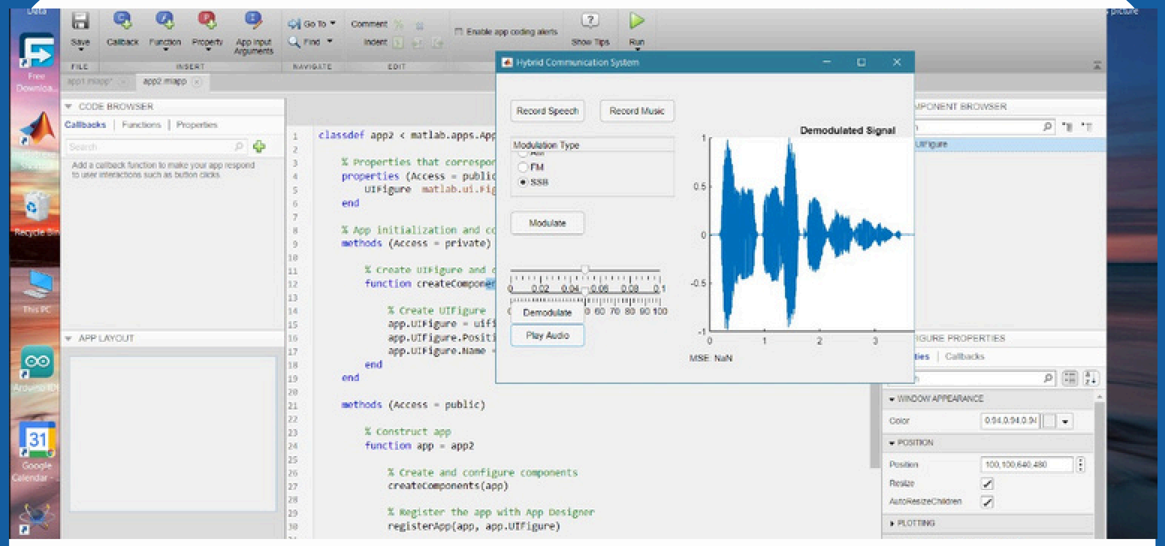
AM Modulation



FM Modulation



SSB Modulation



Demodulation

Challenges Faced

- **Signal Distortion from Noise and Delay**: Managing the impact of noise and delay on the transmitted signals and maintaining signal integrity during modulation and demodulation.
- **Real-time Processin**g: Implementing real-time audio recording, modulation, and demodulation, while ensuring efficient processing for smooth user interaction.
- **Accurate Demodulation:** Ensuring correct demodulation of different signal types (AM, FM, SSB) despite noise and distortions, and maintaining quality in the recovered signal.

SOLUTIONS IMPLEMENTED

1. Implement advanced noise filtering techniques like low-pass filters
2. Use advanced demodulation techniques.

3. Using fast algorithms like FFT (Fast Fourier Transform) for modulation and demodulation.

THANK YOU!