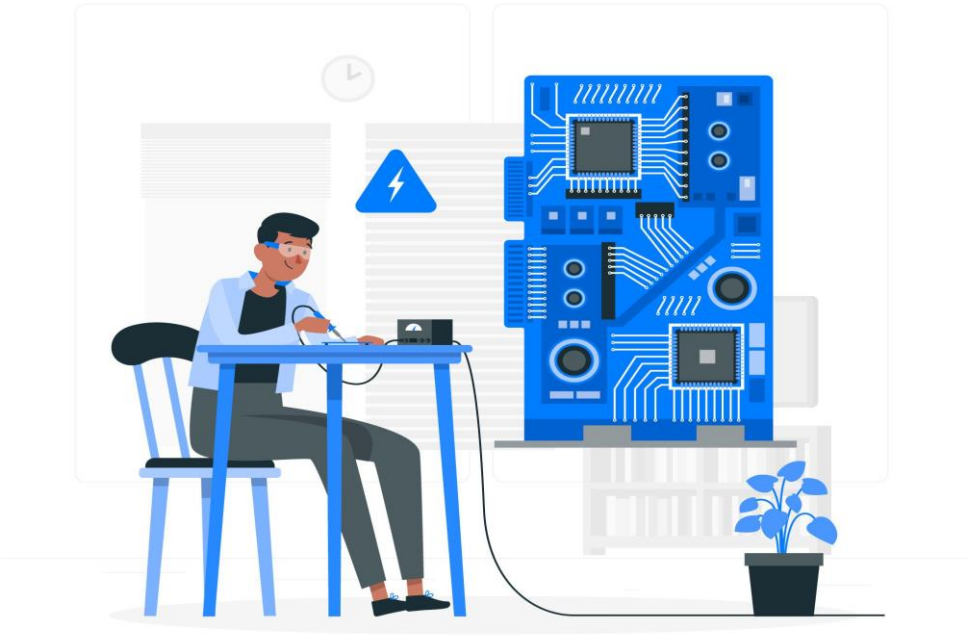


# Lab 9

Akshar panchani- ID 202101522

CT303 Digital Communication

11/19/23

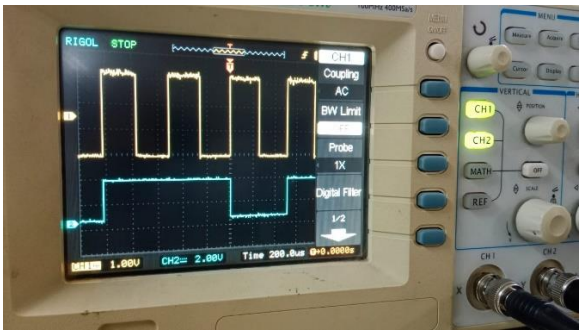
# Experiment 1:

Data Bit: 32-bit

Data Rate: 2KHz

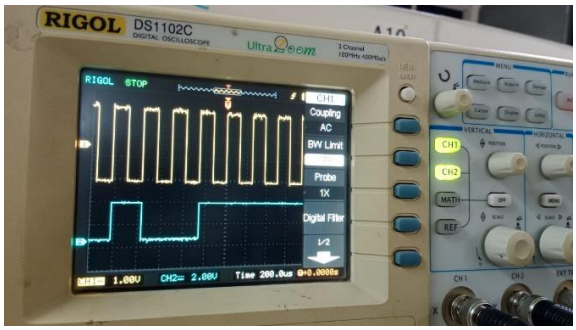
CH1: TP1

CH2: TP2

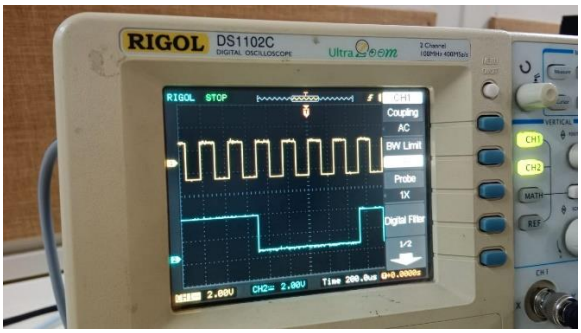



CH1: TP1

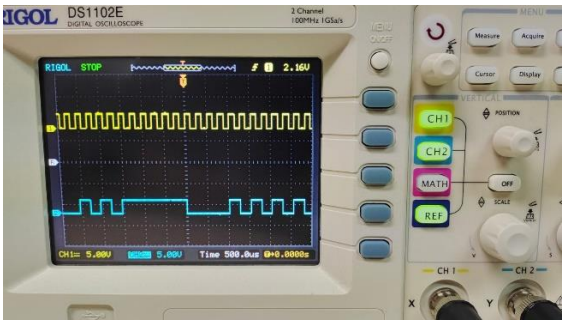
CH2:TP2

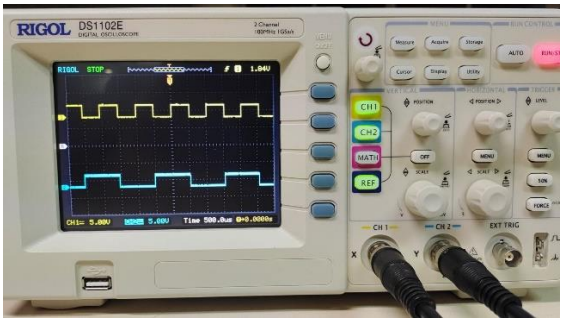



CH1: TP1 CH2: TP2

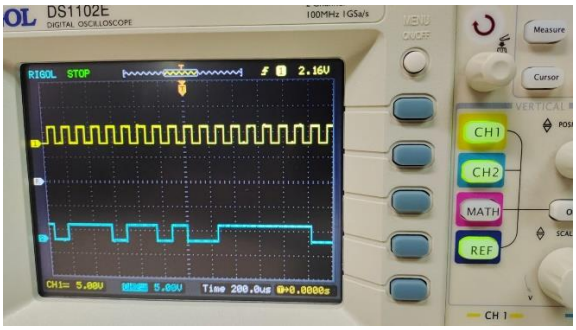


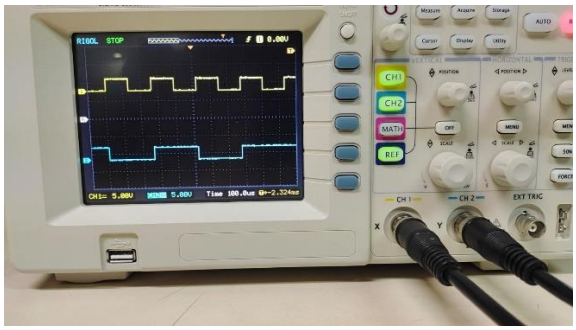

CH1: TP1 CH2: TP2 CH3:TP3 CH4:TP4



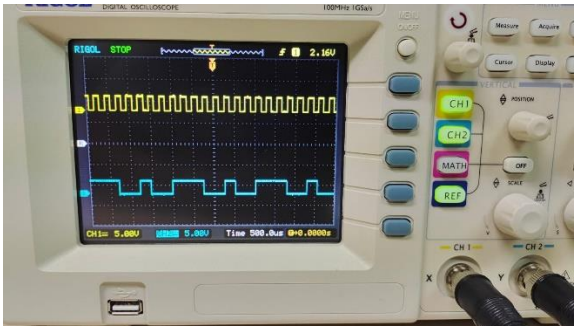

CH1: TP1 CH2: TP2 CH3:TP3 CH4:TP4

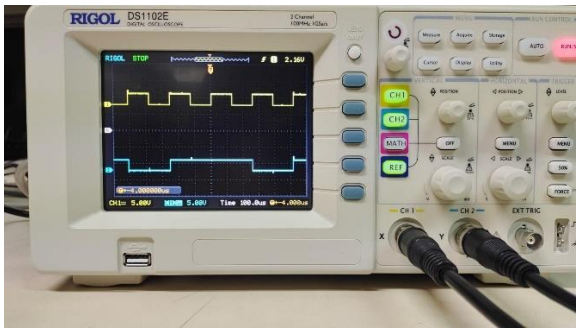


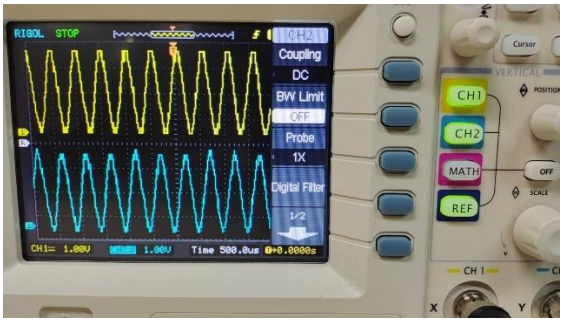
Digital Communication

CH1: TP1 CH2: TP2 CH3:TP3 CH4:TP4



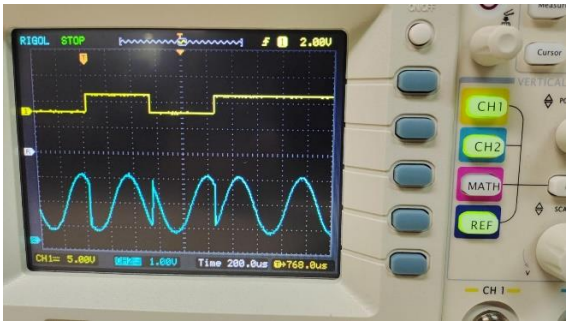



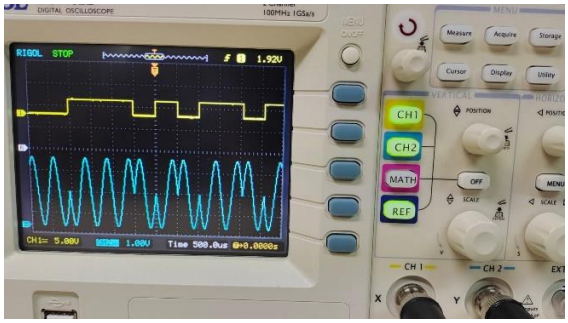

CH1: TP5 CH2: TP6



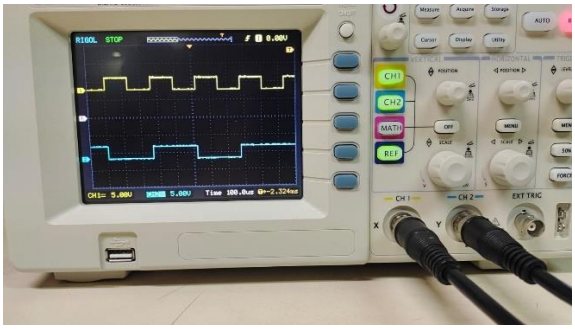

CH1: TP3 CH1: TP4

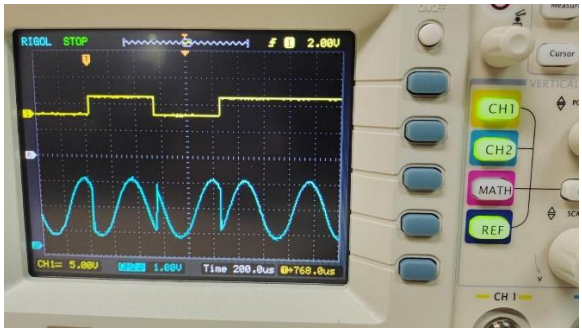
CH2: TP7 CH2:TP8



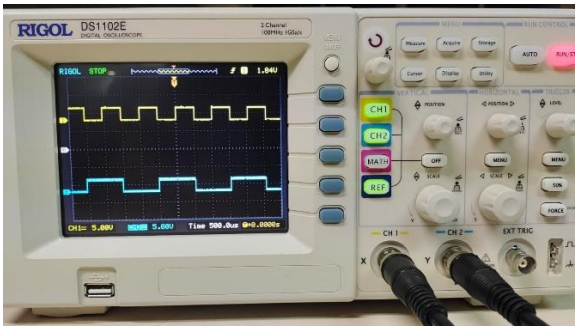



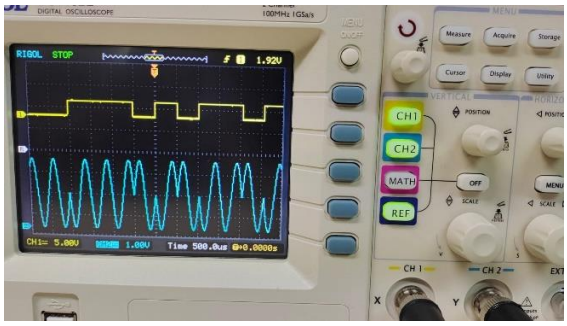

CH1: TP3 CH2: TP4 CH3:TP9



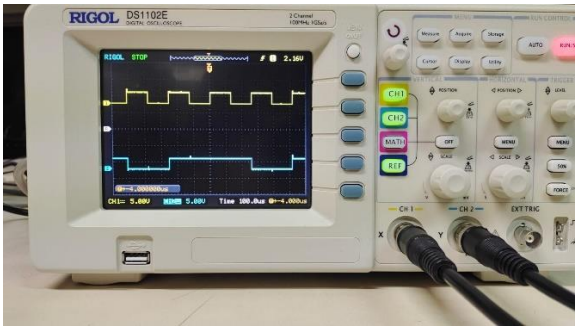



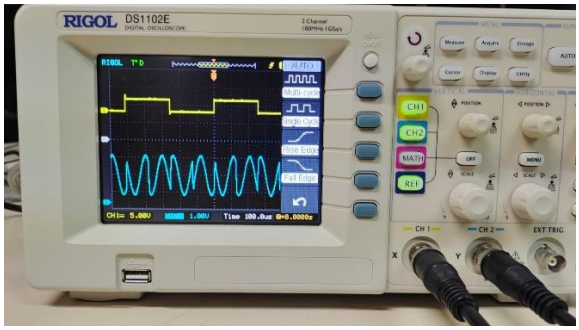

CH1: TP3 CH2: TP4 CH3:TP9



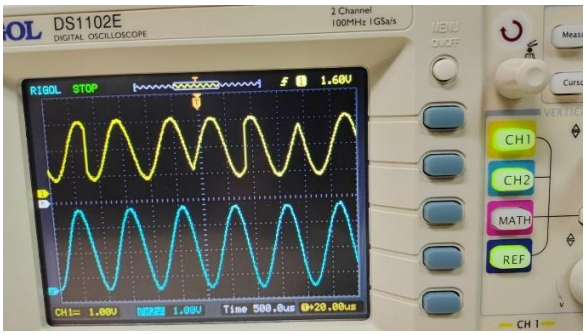

CH1: TP3 CH2: TP4 CH3:TP9

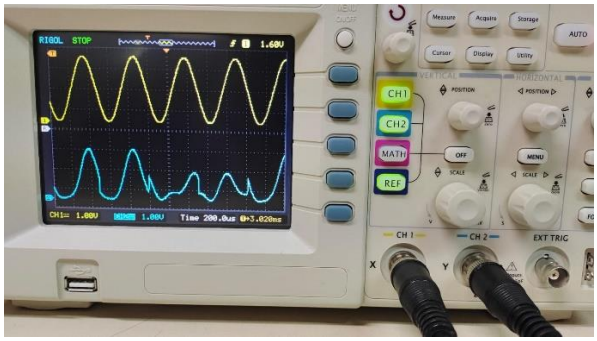



CH1: TP9
CH2: TP10

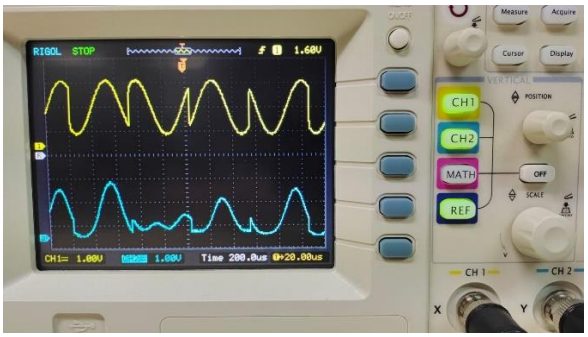
CH1: TP9
CH2:TP11

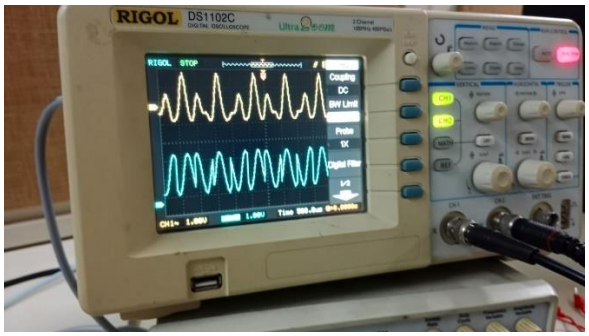



CH1: TP9
CH2: TP12

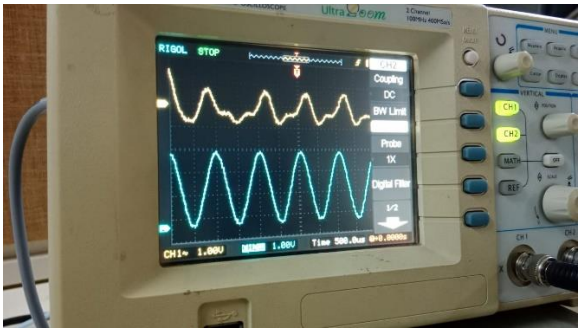
CH1: TP9
CH2:TP12




CH1: TP10
CH2: TP12

CH1: TP10
CH2:TP12

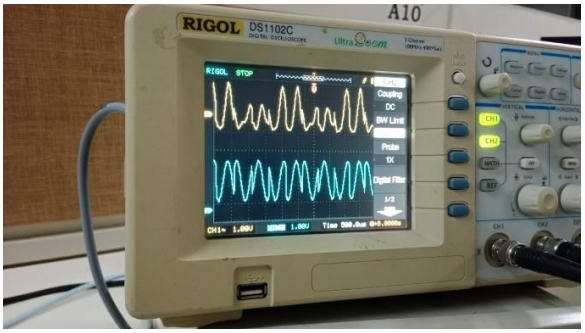


CH1: TP9

CH2: TP13

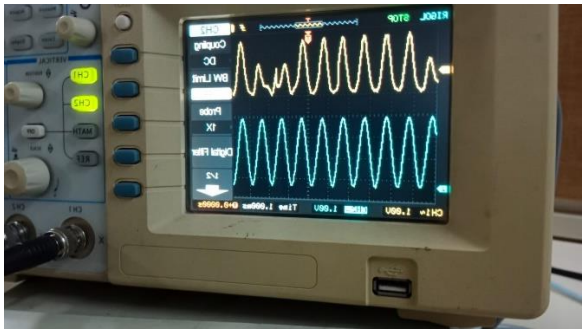
CH1: TP9

CH2:TP18



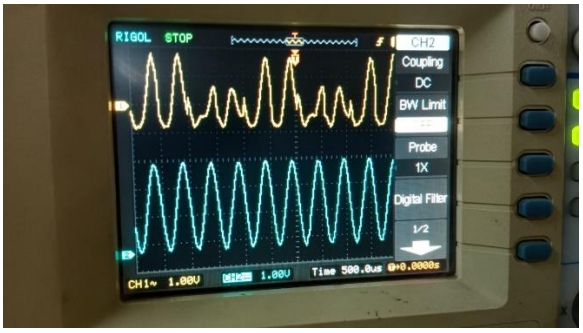
CH1: TP11

CH2: TP13



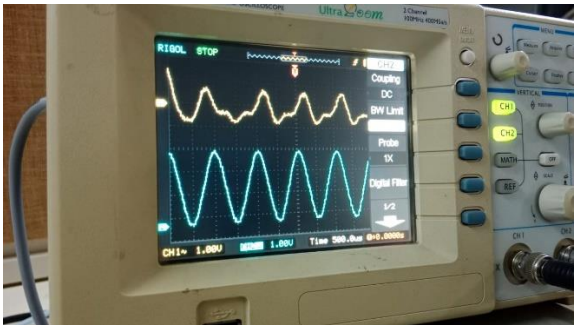
CH1: TP11

CH2:TP13



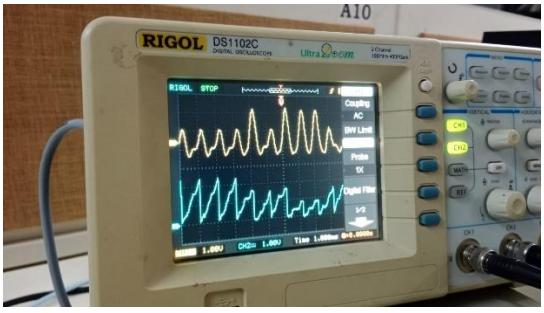
CH1: TP12

CH2: TP14



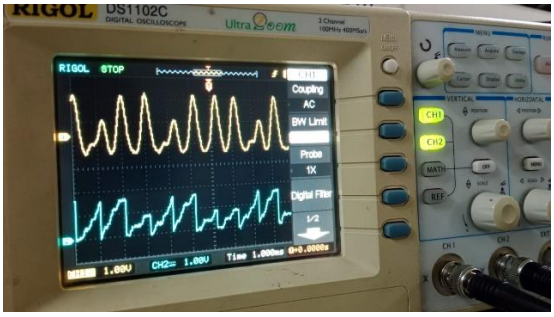
CH1: TP12

CH2:TP14



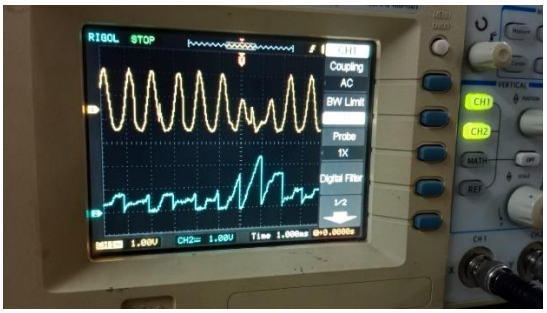
CH1: TP13

CH2: TP15



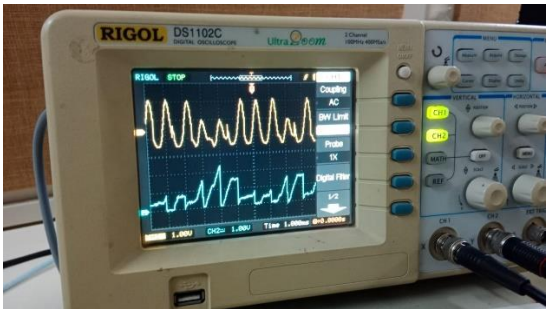
CH1: TP13

CH2:TP15



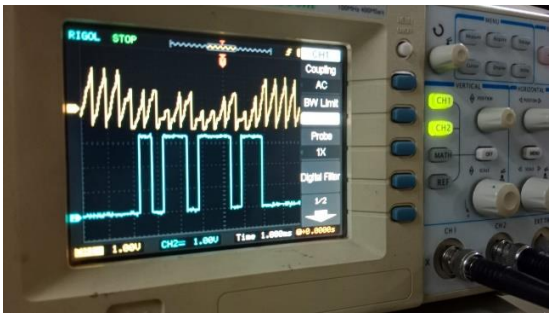

CH1: TP14

CH2: TP16




CH1: TP14

CH2:TP16




CH1: TP14

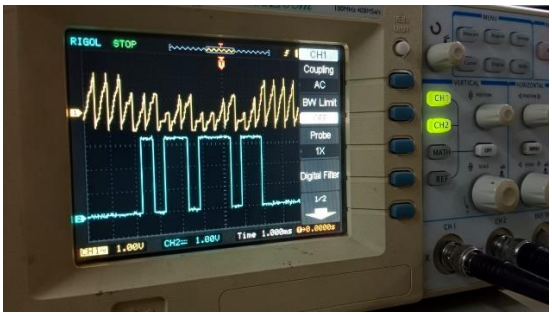
CH2: TP16


CH1: TP15

CH2:TP17

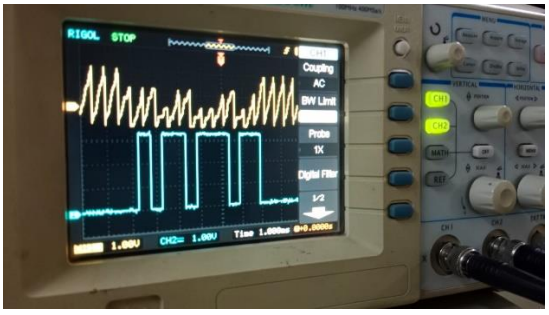

CH1: TP15

CH2: TP17



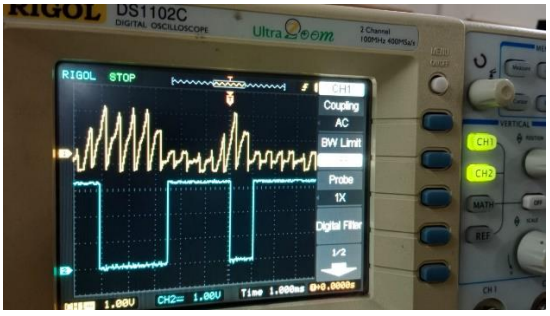

CH1: TP15

CH2:TP17



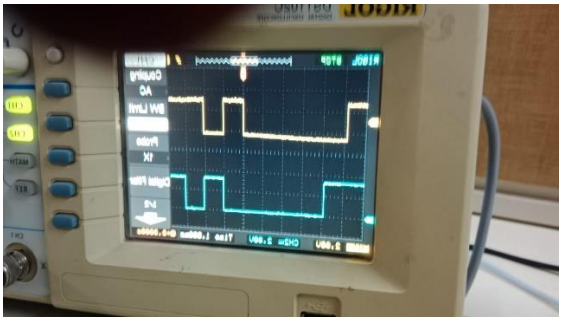

CH1: TP3

CH2: TP16

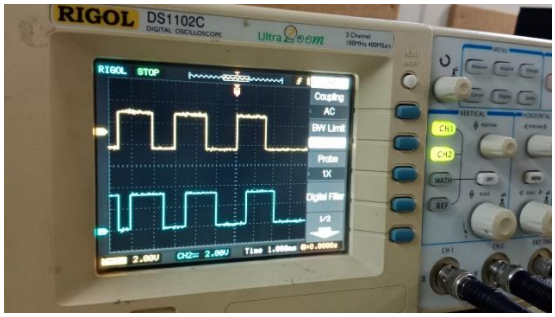



CH1: TP3

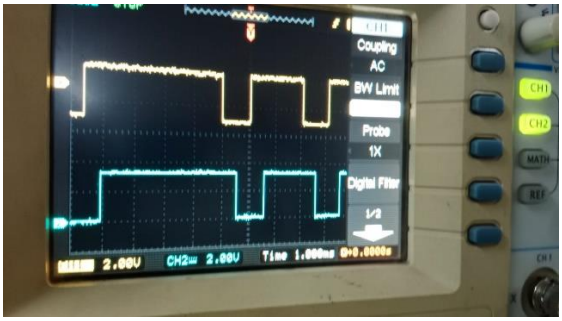
CH2:TP16



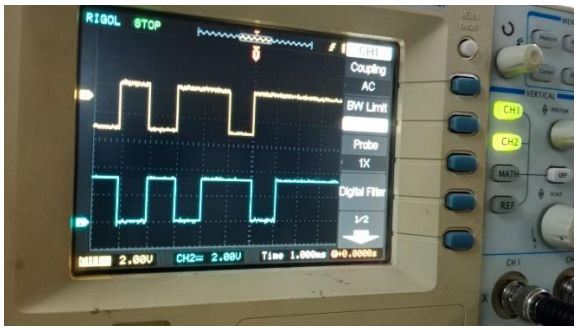
CH1: TP4
CH2: TP17



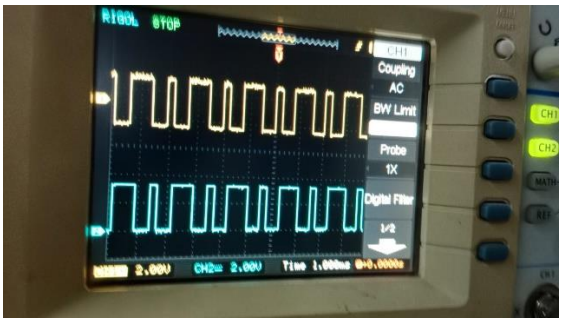

CH1: TP4
CH2:TP17

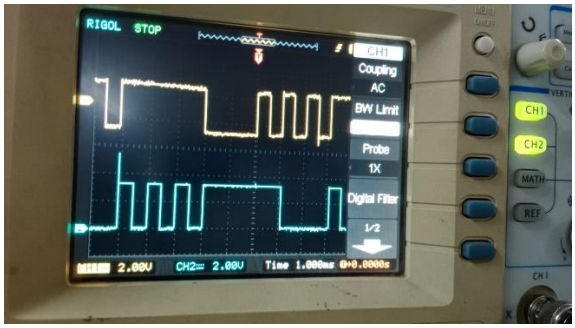



CH1: TP2
CH2: TP18

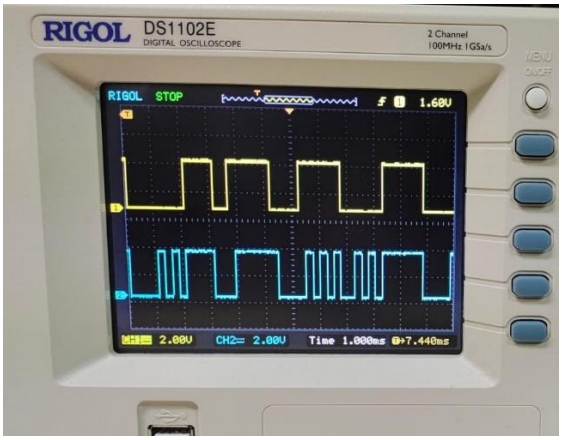


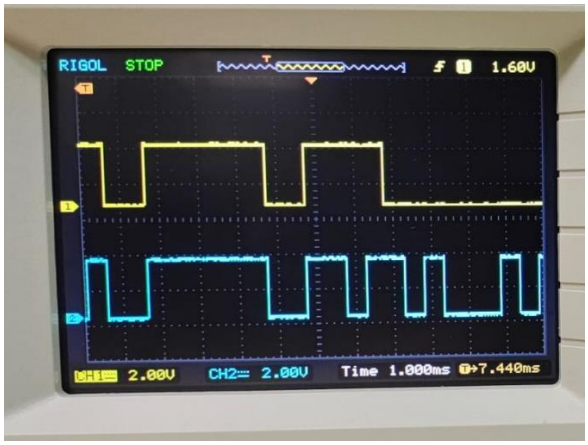

CH1: TP2
CH2:TP18



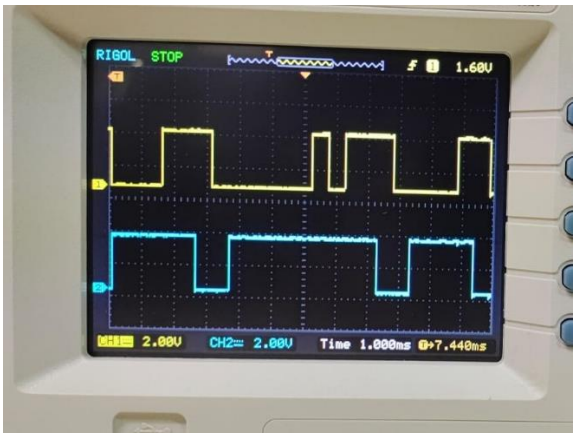



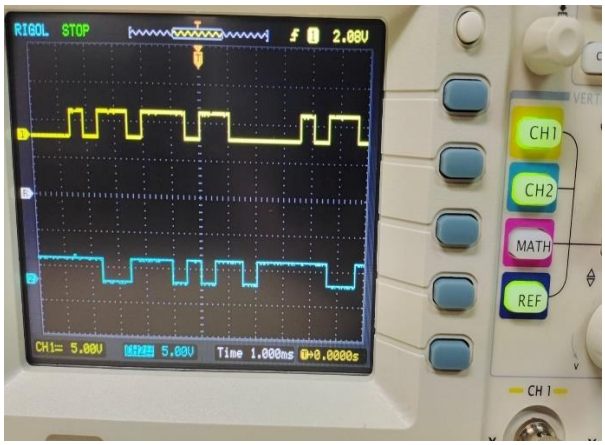

CH1: TP16 CH2: TP17 CH3:TP18



CH1: TP16 CH2: TP17 CH3:TP18

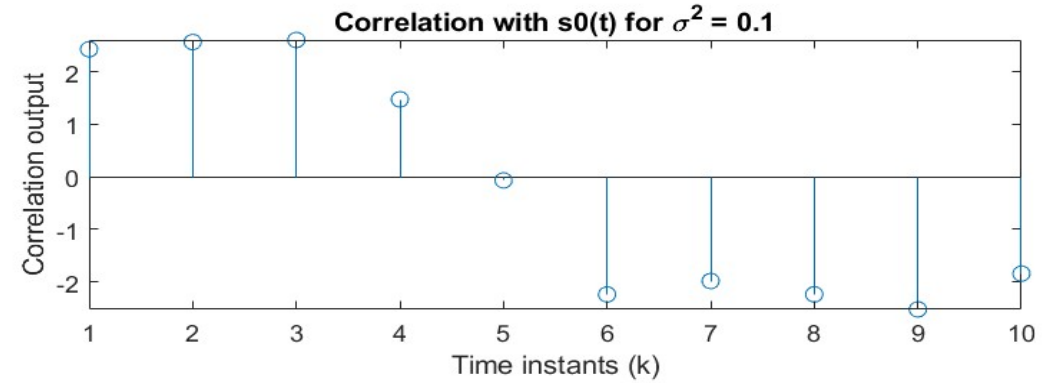


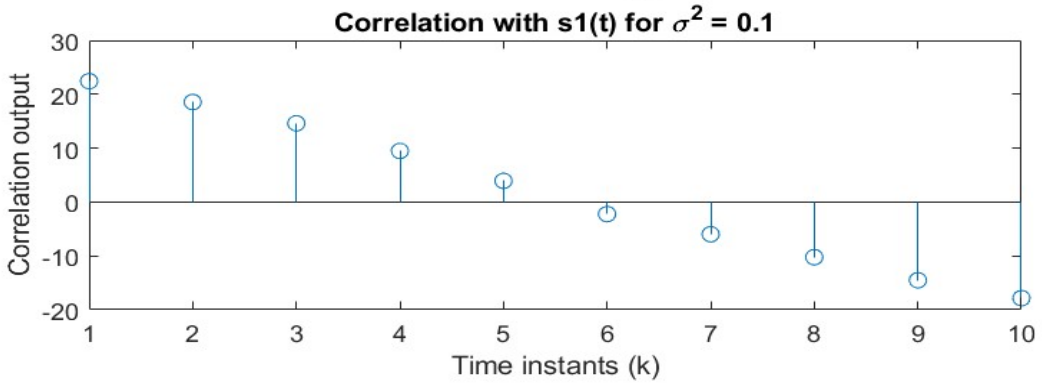



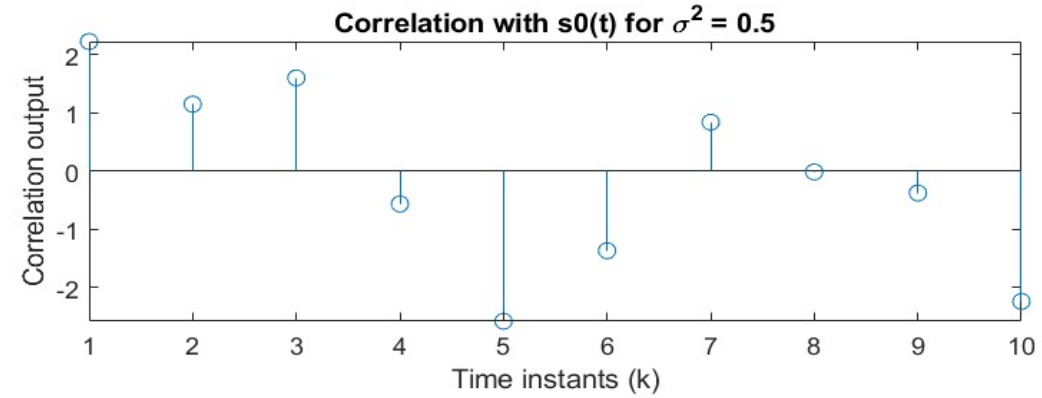

# Experiment 2:

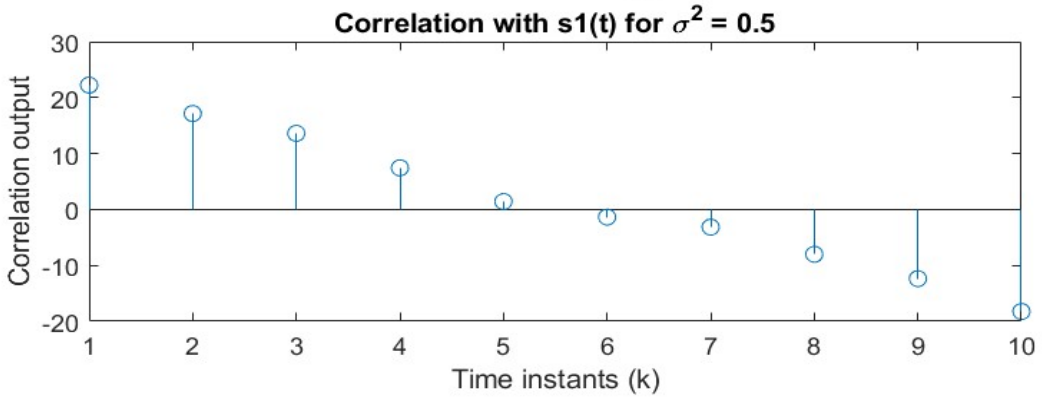
## Problem 5.1:

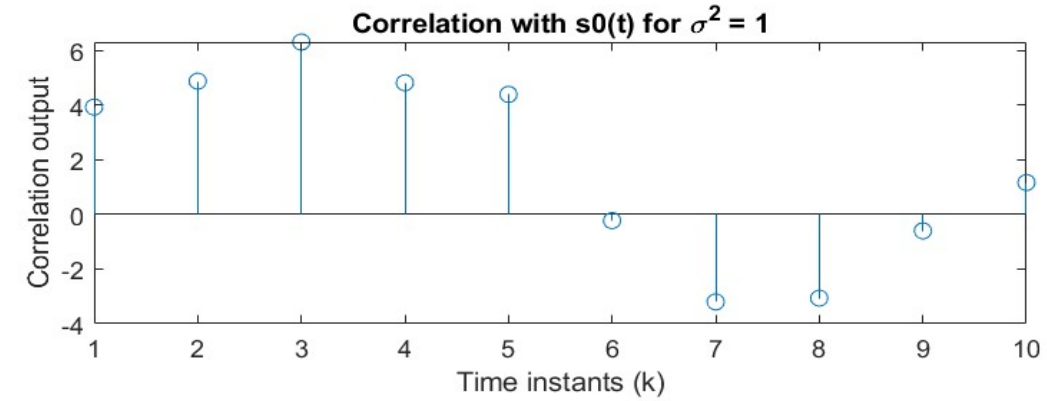
```
A = 1; % Amplitude
Tb = 1; % Bit interval duration
numSamples = 10; % Number of samples per bit interval
sigma_squared_values = [0, 0.1, 0.5, 1.0]; % Values of noise variance
for sigma_squared = sigma_squared_values
    % Generate the noise sequence
    nk = sqrt(sigma_squared) * randn(1, numSamples * 2); % Generate Gaussian noise
    s0_samples = [A * ones(1, numSamples), -A * ones(1, numSamples)];
    received_s0 = A + nk;
    s1_samples = [A * ones(1, numSamples), -A * ones(1, numSamples)];
    received_s1 = [A + nk(1:numSamples), -A + nk(numSamples+1:end)];
    correlation_s0 = zeros(1, numSamples);
    for k = 1:numSamples
        correlation_s0(k) = sum(received_s0 .* circshift(s0_samples, [0, k-1]));
    end
    correlation_s1 = zeros(1, numSamples);
    for k = 1:numSamples
        correlation_s1(k) = sum(received_s1 .* circshift(s1_samples, [0, k-1]));
    end

    figure;
    subplot(2, 1, 1);
    stem(1:numSamples, correlation_s0);
    title(['Correlation with s0(t) for \sigma^2 = ' num2str(sigma_squared)]);
    xlabel('Time instants (k)');
    ylabel('Correlation output');

    subplot(2, 1, 2);
    stem(1:numSamples, correlation_s1);
    title(['Correlation with s1(t) for \sigma^2 = ' num2str(sigma_squared)]);
    xlabel('Time instants (k)');
    ylabel('Correlation output');
end
```

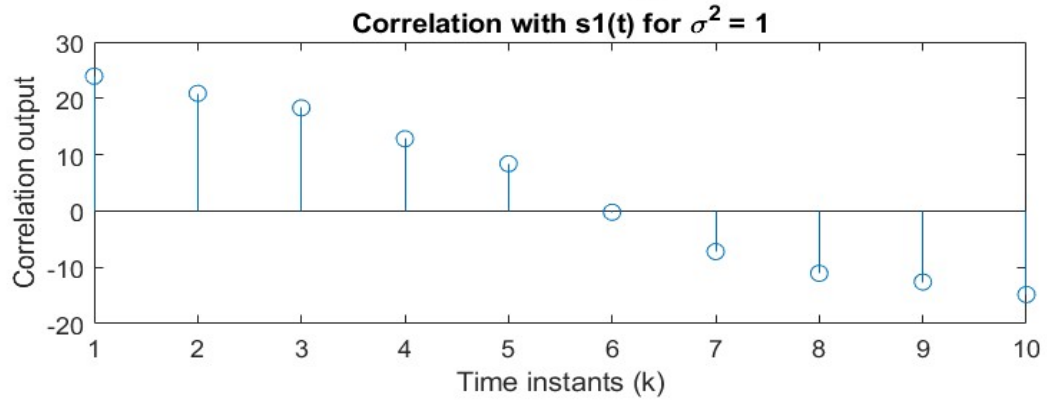

Correlation with s0(t) for $\sigma^2$ = 0.1
Correlation output
Time instants (k)

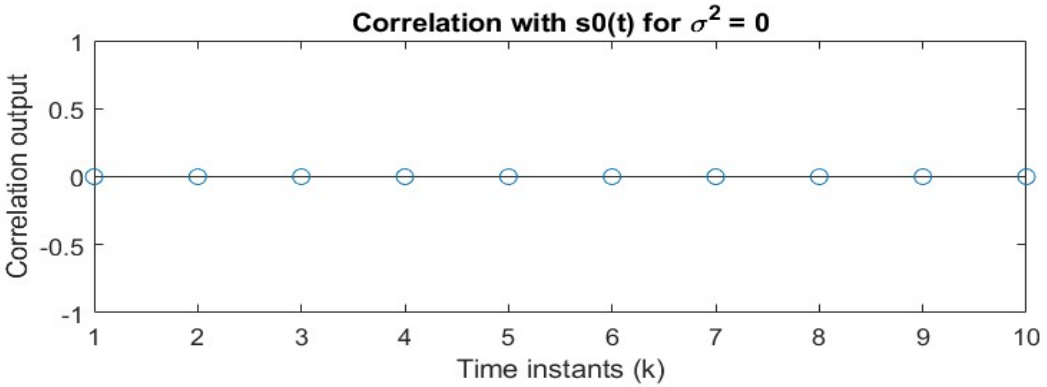

Correlation with s1(t) for $\sigma^2$ = 0.1
Correlation output
Time instants (k)

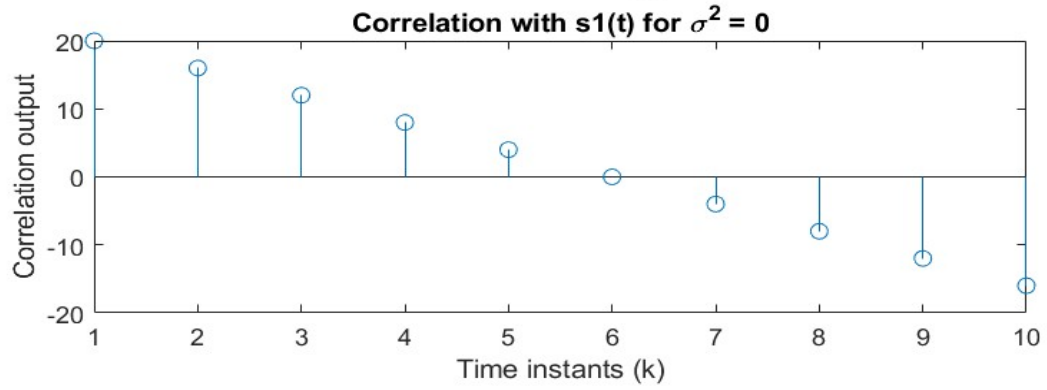

Correlation with s0(t) for $\sigma^2$ = 0.5
Correlation output
Time instants (k)


Correlation with s1(t) for $\sigma^2$ = 0.5
Correlation output
Time instants (k)


Correlation with s0(t) for $\sigma^2$ = 1
Correlation output
Time instants (k)


Correlation with s1(t) for $\sigma^2$ = 1
Correlation output
Time instants (k)


Correlation with s0(t) for $\sigma^2$ = 0
Correlation output
Time instants (k)

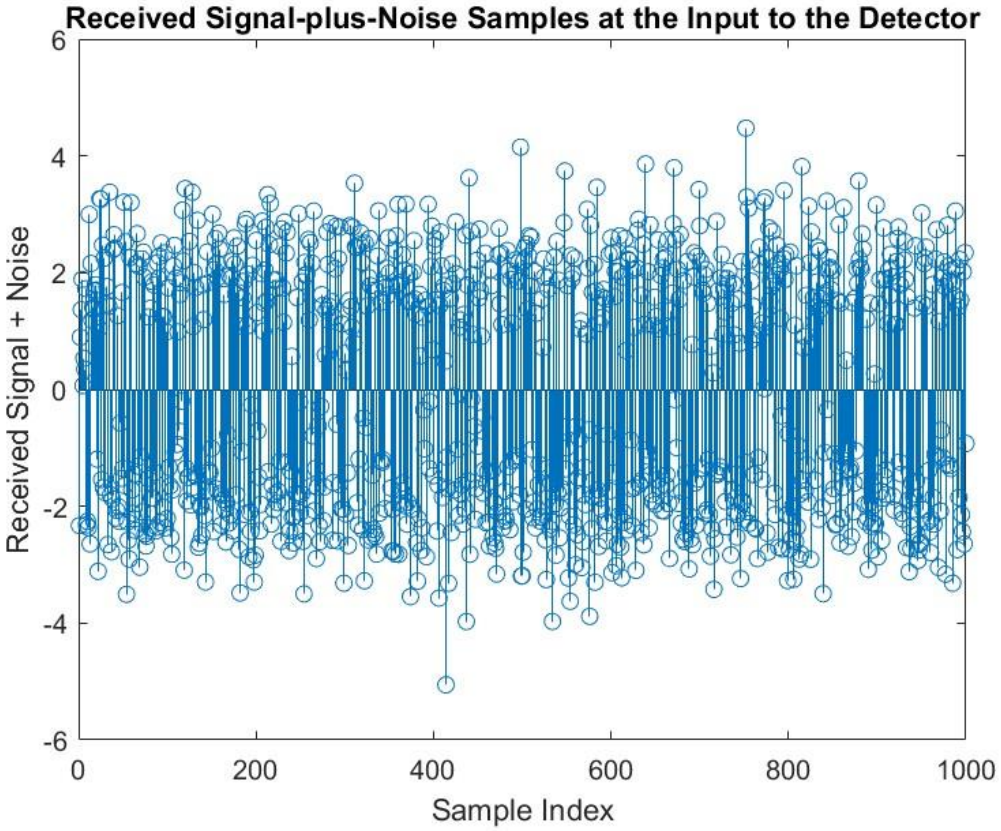

Correlation with s1(t) for $\sigma^2$ = 0
Correlation output
Time instants (k)

## Problem 5.11:

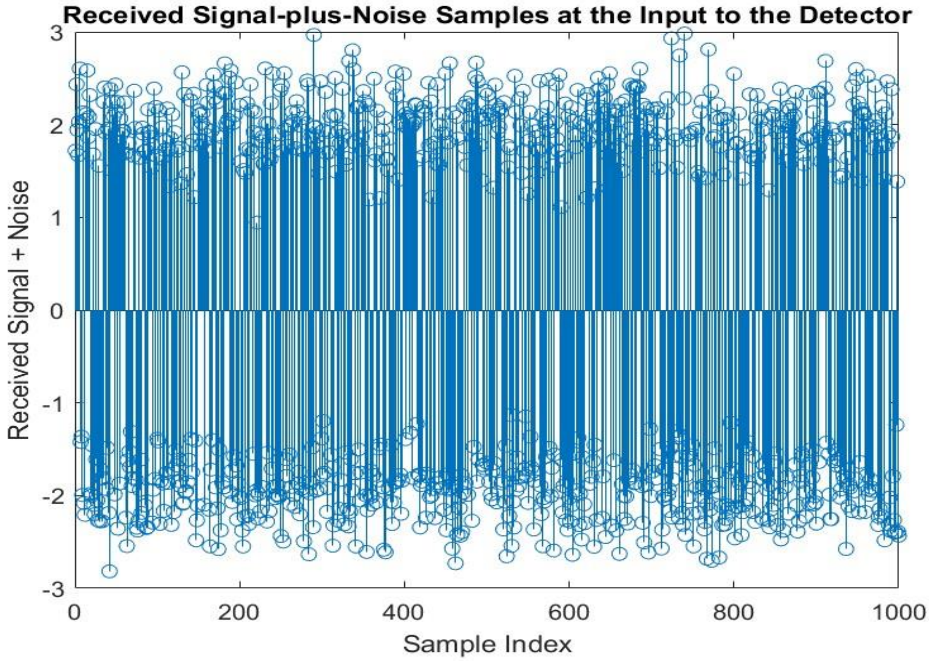
```
num_symbols = 10000;
symbols = 2 * (randi([0, 1], 1, num_symbols) * 2 - 1);
noise_variance = 0; % Setting noise variance to 0
noise = sqrt(noise_variance) * randn(1, num_symbols);
received_symbols = symbols + noise;
decoded_symbols = zeros(size(received_symbols));
decoded_symbols(received_symbols < -2) = -3;
decoded_symbols(received_symbols >= -2 & received_symbols < 0) = -1;
decoded_symbols(received_symbols >= 0 & received_symbols < 2) = 1;
decoded_symbols(received_symbols >= 2) = 3;

symbol_errors = sum(decoded_symbols ~= symbols);
simulated_symbol_error_rate = symbol_errors / num_symbols;

theoretical_symbol_error_rate = 0;

fprintf('Simulated Symbol Error Rate: %.6f\n', simulated_symbol_error_rate);
fprintf('Theoretical Symbol Error Rate: %.6f\n', theoretical_symbol_error_rate);

samples_to_plot = 1000;
if samples_to_plot > num_symbols
    samples_to_plot = num_symbols;
end

figure;
stem(1:samples_to_plot, received_symbols(1:samples_to_plot));
xlabel('Sample Index');
ylabel('Received Signal + Noise');
title('Received Signal-plus-Noise Samples at the Input to the Detector');
```

## Output:

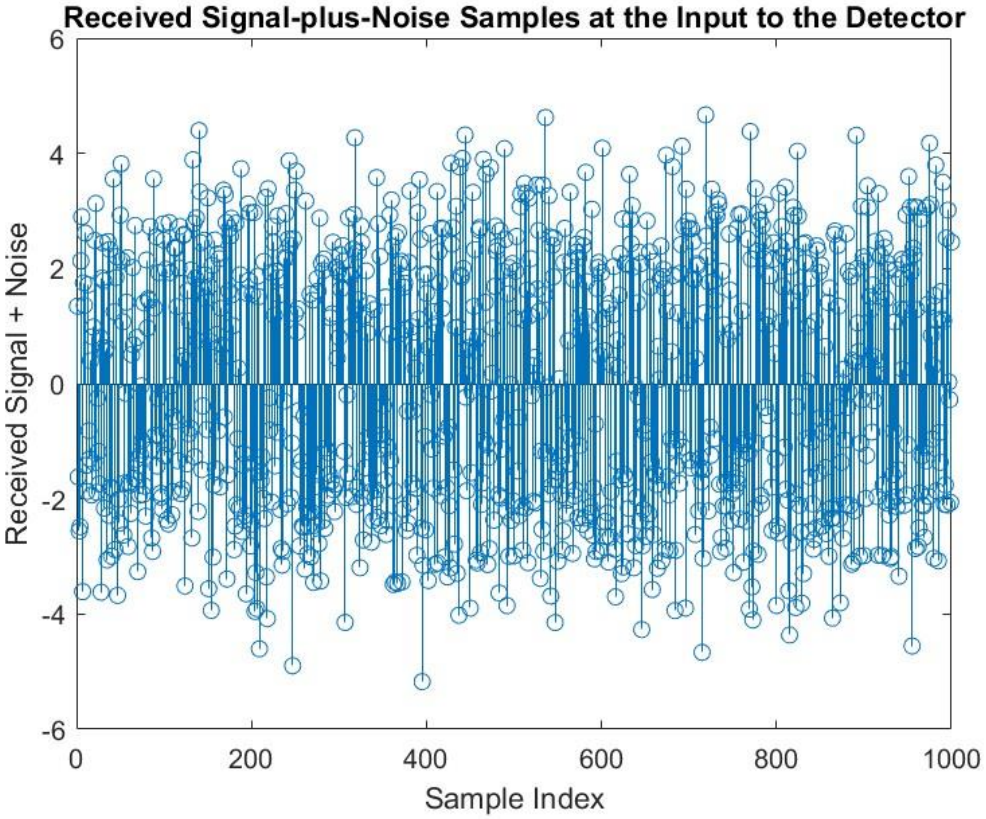
Sigma^2=0, 0.1, 0.5, 1
Simulated Symbol Error Rate: 1.000000
Theoretical Symbol Error Rate: 0.000000



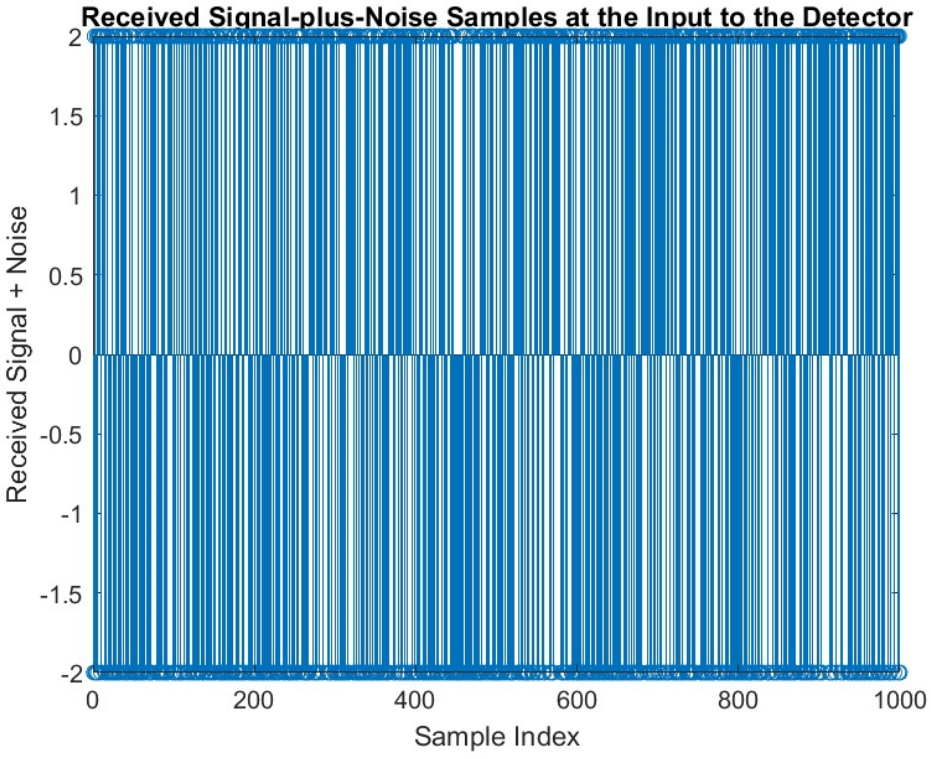

Sigma=1.




Sigma=0.5.




Sigma=0.1

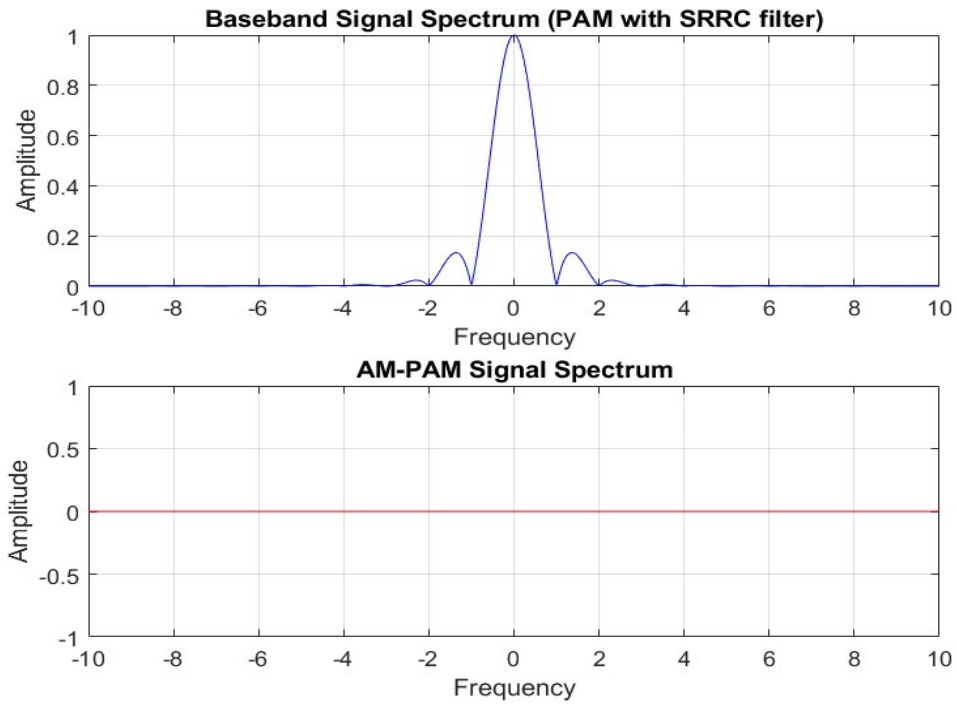



Sigma=0

# Experiment: 3:

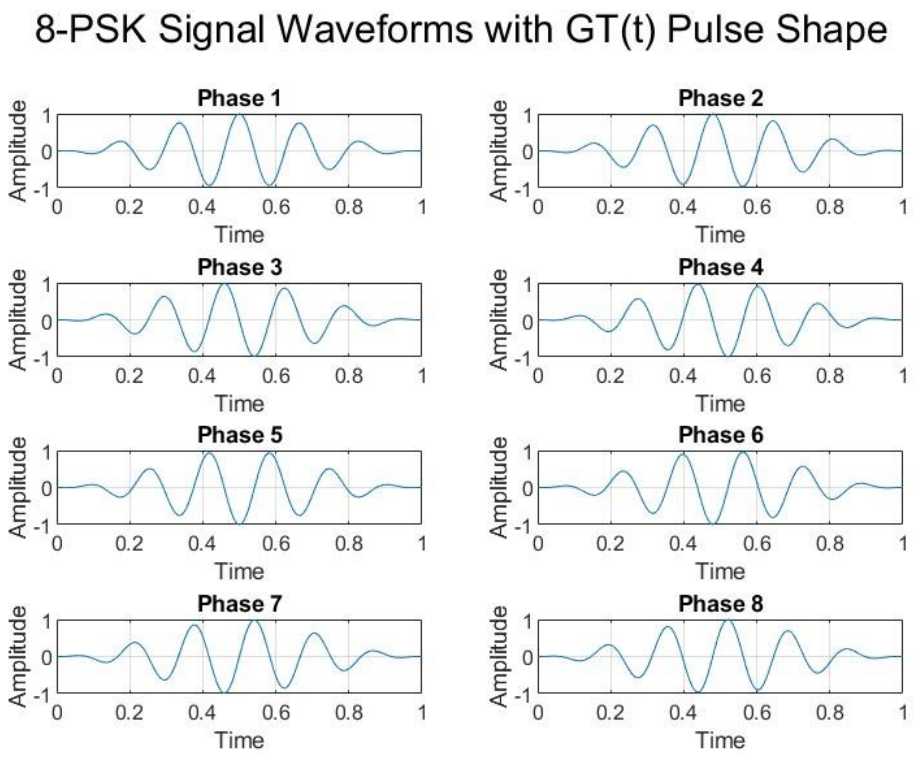
## Problem 7.1:

```
ex = 0.5; % Roll-off factor for square-root raised-cosine filter
T = 1; % Time period
fc = 40 / T; % Carrier frequency
f = linspace(-10/T, 10/T, 1000);
X_baseband = sinc(f * T) .* cos(pi * ex * f * T) ./ (1 - (2 * ex * f * T).^2);
carrier = zeros(size(f));
carrier(f == fc) = 1;
carrier(f == -fc) = 1;
Y_AM_PAM = conv(X_baseband, carrier, 'same');
figure;
subplot(2, 1, 1);
plot(f, abs(X_baseband), 'b');
title('Baseband Signal Spectrum (PAM with SRRC filter)');
xlabel('Frequency');
ylabel('Amplitude');
grid on;
subplot(2, 1, 2);
plot(f, abs(Y_AM_PAM), 'r');
title('AM-PAM Signal Spectrum');
xlabel('Frequency');
ylabel('Amplitude');
grid on;
```



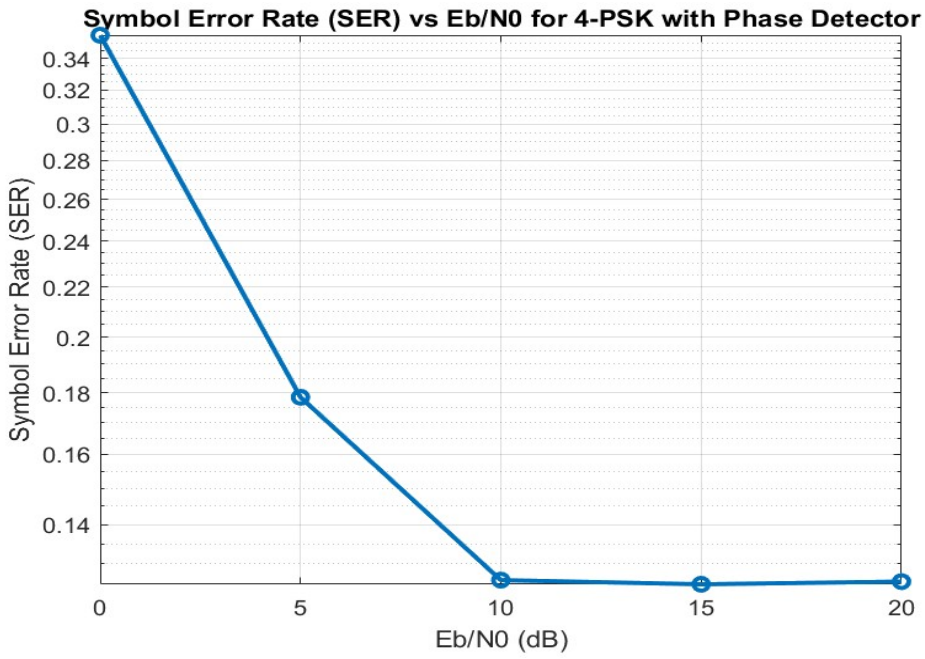
## Problem 7.7:

```
T = 1; % Time period
fc = 6 / T; % Carrier frequency
M = 8; % Number of phases for PSK
t = linspace(0, T, 1000); % Time vector
GT = @(t) 0.5 * (1 - cos(2 * pi * t / T)) .* (0 <= t & t <= T);
theta = linspace(0, 2 * pi, M + 1); % Phase angles
theta = theta(1:M); % Remove the last angle to avoid overlap
PSK_waveforms = zeros(M, length(t));

for i = 1:M
    PSK_waveforms(i, :) = cos(2 * pi * fc * t + theta(i)) .* GT(t);
end
figure;

for i = 1:M
    subplot(M/2, 2, i);
    plot(t, PSK_waveforms(i, :));
    title(['Phase ' num2str(i)]);
    xlabel('Time');
    ylabel('Amplitude');
    grid on;
end

sgtitle('8-PSK Signal Waveforms with GT(t) Pulse Shape');
```



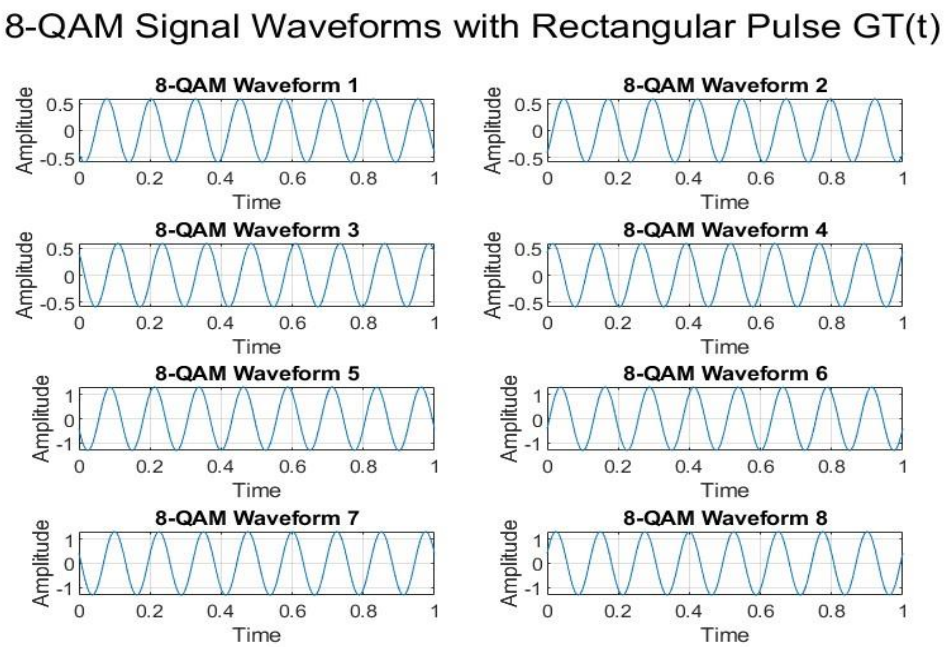
## Problem 7.9:

```
M = 4; % Number of phases for PSK
Eb_N0_dB = 0:5:20; % Eb/N0 values in dB
num_symbols = 10^5; % Number of symbols to simulate
constellation = exp(1i * (0:2*pi/M:2*pi*(1-1/M))); % PSK constellation
for k = 1:length(Eb_N0_dB)
    Eb_N0 = 10^(Eb_N0_dB(k) / 10); % Convert dB to linear scale
    info_symbols = randi([1 M], 1, num_symbols);
    transmitted_symbols = constellation(info_symbols); % Map symbols to constellation
    noise_var = 1 / (2 * Eb_N0); % Calculate noise variance for given Eb/N0
    noise = sqrt(noise_var) * (randn(1, num_symbols) + 1i * randn(1, num_symbols)); %
    received_symbols = transmitted_symbols + noise; % Received symbols with noise

    detected_symbols = zeros(1, num_symbols);
    for i = 1:num_symbols
        received_phase = angle(received_symbols(i));
        [~, index] = min(abs(received_phase - angle(constellation)));
        detected_symbols(i) = constellation(index);
    end
    symbol_errors = nnz(detected_symbols - transmitted_symbols);
    symbol_error_rate(k) = symbol_errors / num_symbols;
end

figure;
semilogy(Eb_N0_dB, symbol_error_rate, 'o-', 'linewidth', 2);
title('Symbol Error Rate (SER) vs Eb/N0 for 4-PSK with Phase Detector');
xlabel('Eb/N0 (dB)');
ylabel('Symbol Error Rate (SER)');
grid on;
```



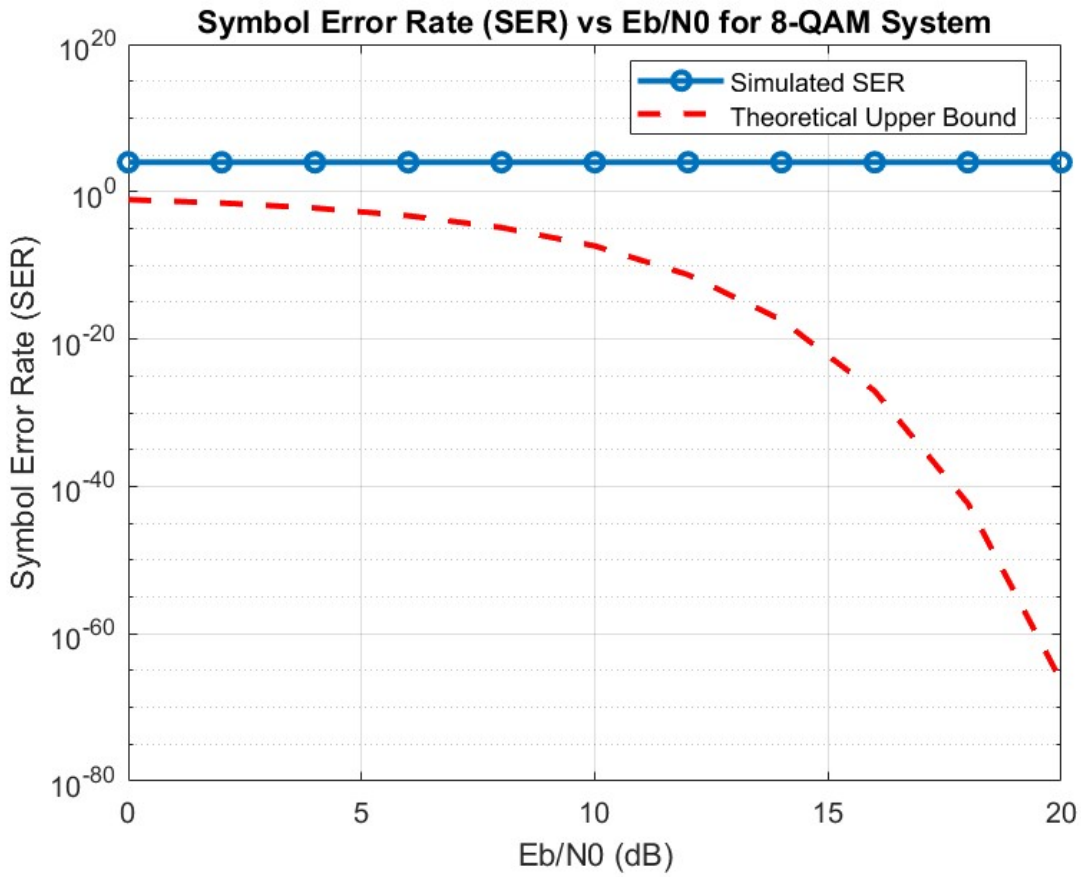
## Problem 7.12:

```
T = 1; % Time period
fc = 8 / T; % Carrier frequency
A = sqrt(2) - 1; % Signal constellation amplitude factor
t = linspace(0, T, 1000); % Time vector
constellation = [
    -A - 1i * A;
    -A + 1i * A;
    A - 1i * A;
    A + 1i * A;
    -A - 3i * A;
    -A + 3i * A;
    A - 3i * A;
    A + 3i * A
];
figure;

for i = 1:size(constellation, 1)
    % Generate waveform for each constellation point using rectangular pulse
    waveform = real(constellation(i)) * cos(2 * pi * fc * t) + imag(constellation(i))
* sin(2 * pi * fc * t);
    subplot(4, 2, i);
    plot(t, waveform);
    title(['8-QAM Waveform ' num2str(i)]);
    xlabel('Time');
    ylabel('Amplitude');
    grid on;
end

sgtitle('8-QAM Signal Waveforms with Rectangular Pulse GT(t)');
```



## Problem 7.17:

```
M = 8; % Number of symbols in 8-QAM
num_symbols = 10000; % Number of symbols to simulate
Eb_N0_dB = 0:2:20; % Eb/N0 values in dB
Eb_N0 = 10.^(Eb_N0_dB / 10); % Convert Eb/N0 values to linear scale

constellation = [
    -3-3i;
    -3-1i;
    -3+3i;
    -3+1i;
    -1-3i;
    -1-1i;
    -1+3i;
    -1+1i
];
SER_simulated = zeros(1, length(Eb_N0));
SER_upper_bound = zeros(1, length(Eb_N0));
for k = 1:length(Eb_N0)

    info_symbols = randi([1 M], 1, num_symbols);
    transmitted_symbols = constellation(info_symbols); % Map symbols to constellation
    normalized_symbols = transmitted_symbols /
sqrt(mean(abs(transmitted_symbols).^2));

    sigma = 1 / sqrt(2 * Eb_N0(k)); % Calculate noise standard deviation
    noise = sigma * (randn(1, num_symbols) + 1i * randn(1, num_symbols)); % Complex
Gaussian noise
    received_symbols = normalized_symbols + noise;
    detected_symbols = zeros(size(received_symbols));
    for i = 1:num_symbols
        [~, index] = min(abs(received_symbols(i) - constellation));
        detected_symbols(i) = constellation(index);
    end
    symbol_errors = nnz(detected_symbols - transmitted_symbols);
    SER_simulated(k) = symbol_errors / num_symbols;
    SER_upper_bound(k) = 2 * qfunc(sqrt(3 * Eb_N0(k))); % Theoretical upper bound
formula
end

figure;
semilogy(Eb_N0_dB, SER_simulated, 'o-', 'linewidth', 2, 'DisplayName', 'Simulated
SER');
hold on;
semilogy(Eb_N0_dB, SER_upper_bound, 'r--', 'linewidth', 2, 'DisplayName',
'Theoretical Upper Bound');
title('Symbol Error Rate (SER) vs Eb/N0 for 8-QAM System');
xlabel('Eb/N0 (dB)');
ylabel('Symbol Error Rate (SER)');
legend('Location', 'best');
grid on;
```


Symbol Error Rate (SER) vs Eb/N0 for 8-QAM System
Simulated SER
Theoretical Upper Bound
Symbol Error Rate (SER)
Eb/N0 (dB)
10^20
10^0
10^-20
10^-40
10^-60
10^-80
0
5
10
15
20