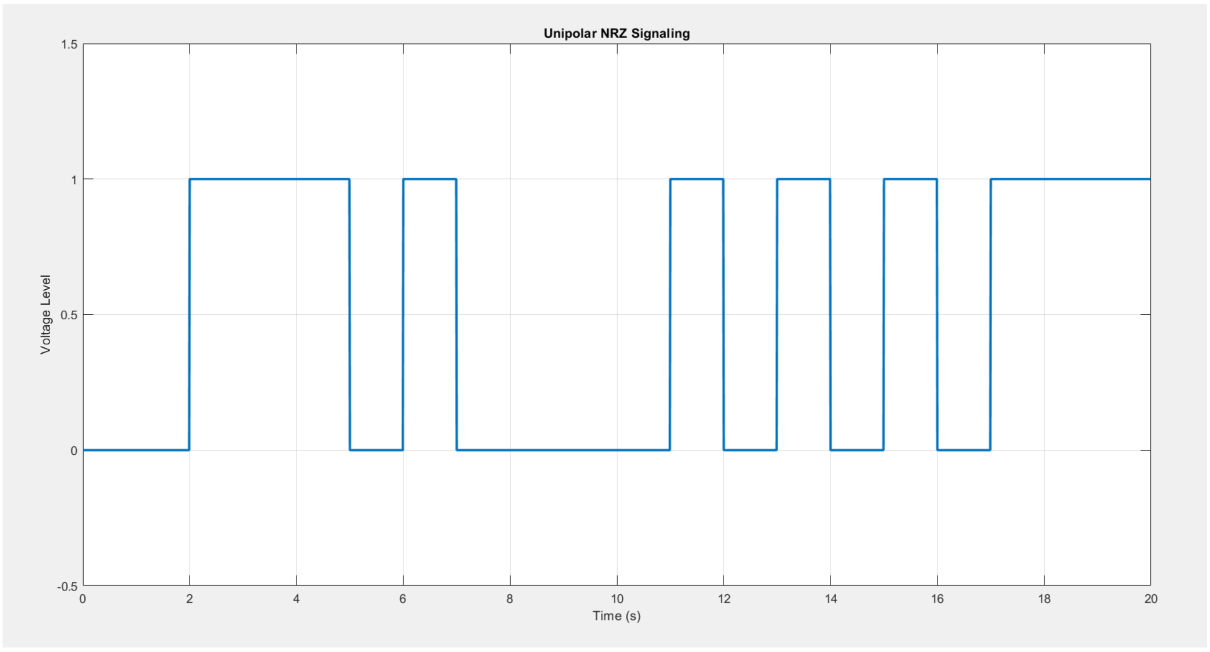
Code:

```
clc;
clear;
close all;
% Parameters
N = 20;  % Number of bits
bit_duration = 1;  % Duration of each bit (in seconds)
sampling_rate = 100; % Sampling rate (samples per second)
% Generate random binary data
binary_data = randi([0, 1], 1, N);
% Create time vector
t = 0:1/sampling_rate:N*bit_duration - 1/sampling_rate;
% Generate Unipolar NRZ signal
unipolar_signal = repmat(binary_data, sampling_rate*bit_duration,
1);
unipolar_signal = unipolar_signal(:); % Convert to column vector
% Plotting
plot(t, unipolar_signal, 'linewidth', 2);
axis([0, N*bit_duration, -0.5, 1.5]);  % Set axis limits
grid on;
title("Unipolar NRZ Signaling");
xlabel('Time (s)');
ylabel('Voltage Level');
```

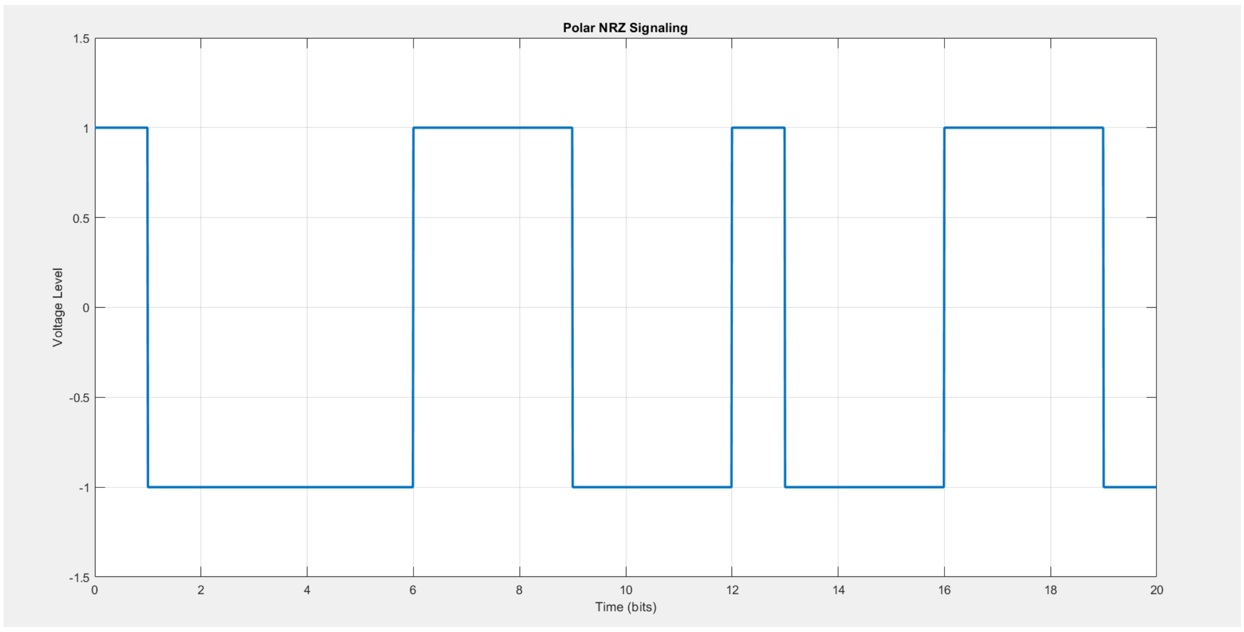
Output:



Code:

```
clc;
clear;
close all;
% Number of bits
N = 20;
% Generate random binary data
binary_data = randi([0, 1], 1, N);
% Mapping Function
polar_data = 2 * binary_data - 1;
% Time vector setup
t = 0:0.01:N-0.01;
% Extend binary data for plotting
extended_data = repelem(polar_data, 100);
% Plotting
plot(t, extended_data, 'linewidth', 2);
axis([0, N, -1.5, 1.5]); % Set axis limits
grid on;
title("Polar NRZ Signaling");
xlabel('Time (bits)');
ylabel('Voltage Level');
```

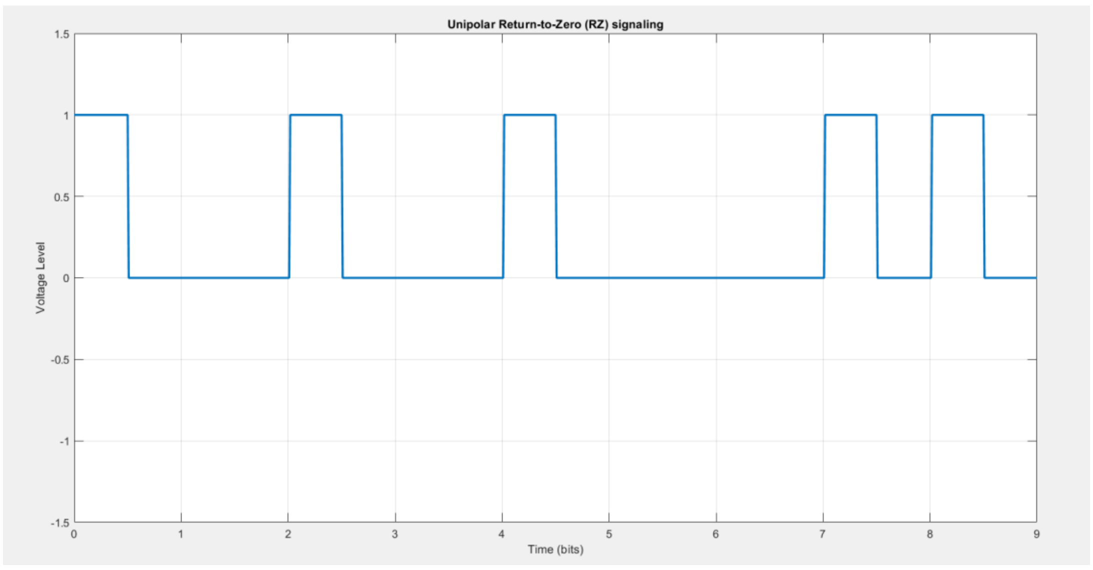
Output:



CODE:

```
clc;
clear ;
close all;
N = 10; % Number of bits
binary_data = randi([0, 1], 1, N); %Generaterandombinary
% RZ Pulse Shaping
i = 1;
a = 0; % Initial value for the first half cycle
b = 0.5; % Initial value for the second half cycle
t = 0:0.01:(N-1); % Time vector
y = zeros(size(t)); % Initialize the output signal
for j = 1:length(t)
    if t(j) >= a && t(j) <= b
        y(j) = binary_data(i); % Assign first 50 values
    elseif t(j) > b && t(j) <= i
        y(j) = 0;
    else
        i = i + 1; % Binary input data index increment
        a = a + 1;
        b = b + 1;
    end
end
% Plotting
plot(t, y, 'lineWidth', 2); % Linewidth 2 for clear visualization
axis([0, N-1, -1.5, 1.5]); % Axis set-up
grid on;
title('Unipolar Return-to-Zero (RZ) signaling');
xlabel('Time (bits)');
ylabel('Voltage Level');
```

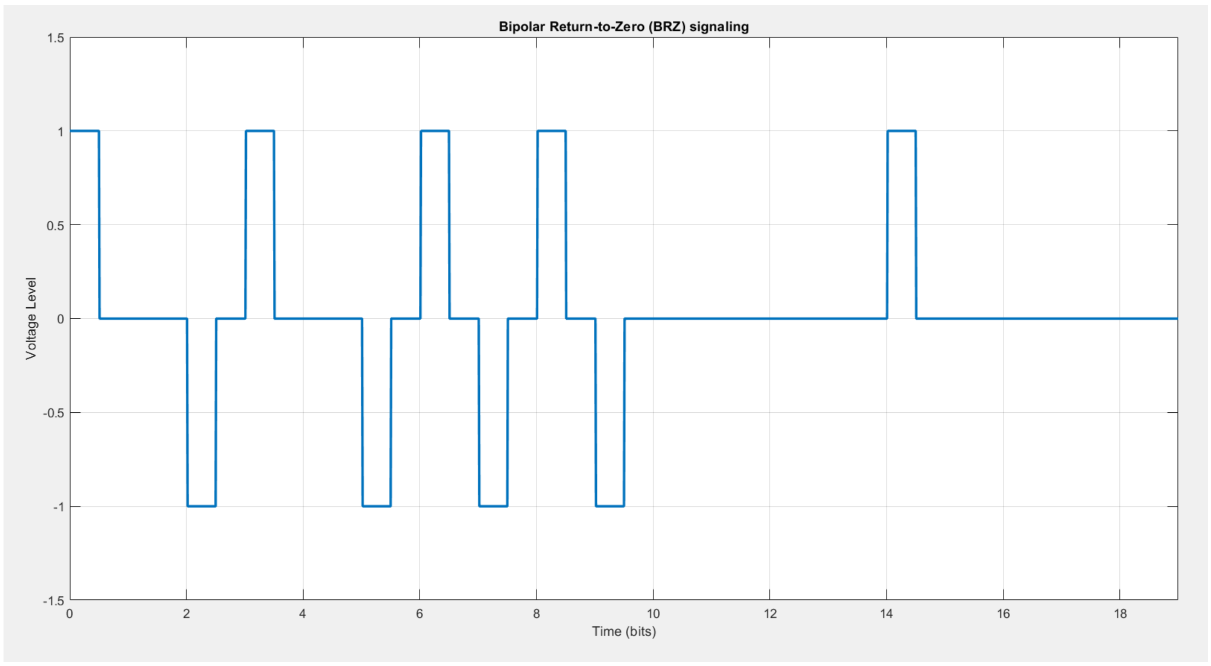
OUTPUT:



CODE:

```
clc;
clear all;
close all;
% Number of bits
N = 20;
% Generate random binary data
binary_data = randi([0, 1], 1, N);
% Binary to Bipolar Conversion
f = 1;
for m = 1:N
    if binary_data(m) == 1
        if f == 1
            bipolar_data(m) = 1;
            f = -1;
        else
            bipolar_data(m) = -1;
            f = 1;
        end
    else
        bipolar_data(m) = 0;
    end
end
% Bipolar RZ Pulse Shaping
i = 1;
a = 0; % Initial value for the first half cycle
b = 0.5; % Initial value for the second half cycle
t = 0:0.01:(N-1); % Time vector
y = zeros(size(t)); % Initialize the output signal
for j = 1:length(t)
    if t(j) >= a && t(j) <= b % Condition for the first half cycle
        y(j) = bipolar_data(i); % Assign first 50 values
    elseif t(j) > b && t(j) <= i % ConditionfortheSecond half cycle
        y(j) = 0; % Set all values to 0 for the second half cycle
    else
        i = i + 1; % Binary input data index increment
        a = a + 1; % Initial value for the first halfcycle increment
        b = b + 1; % Initial value for second halfcycle increment
    end
end
% Plotting
plot(t, y, 'lineWidth', 2); % Linewidth 2 for clear visualization
axis([0, N-1, -1.5, 1.5]); % Axis set-up
grid on;
title('Bipolar Return-to-Zero (BRZ) signaling');
xlabel('Time (bits)');
ylabel('Voltage Level');
```

OUTPUT:

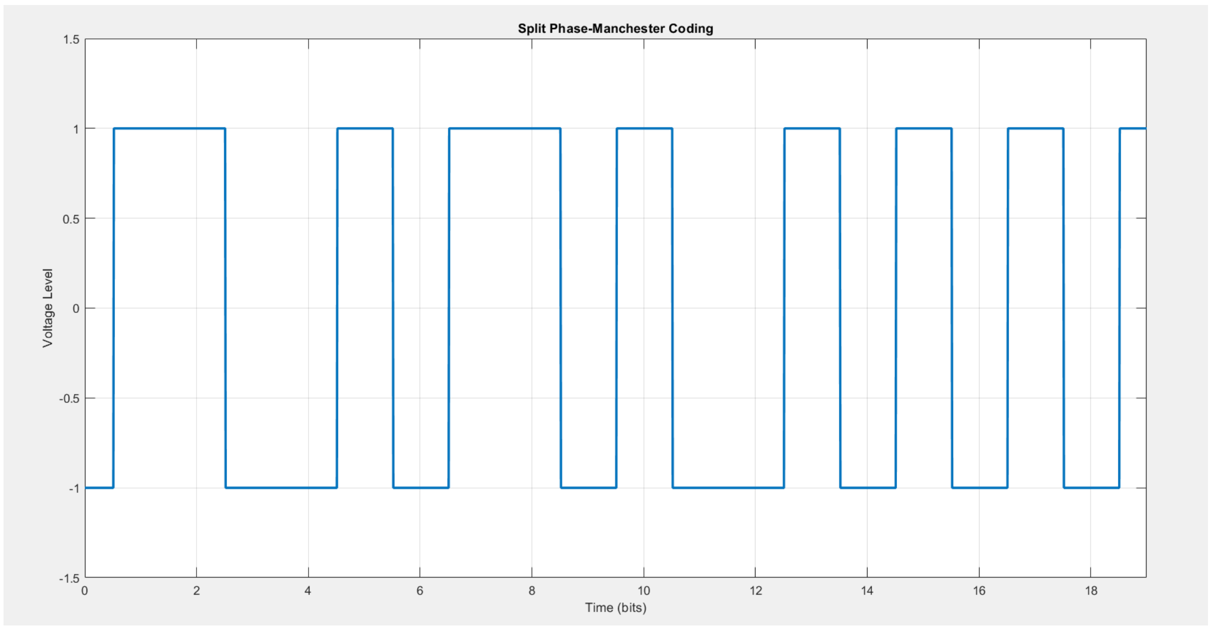


Bipolar Return-to-Zero (BRZ) signaling
Voltage Level
Time (bits)
1.5
1
0.5
0
-0.5
-1
-1.5
0
2
4
6
8
10
12
14
16
18

CODE:

```
clc;
clear ;
close all;
% Number of bits
N = 10;

% Generate random binary data
binary_data = randi([0, 1], 1, N);

% Binary to Manchester Conversion
manchester_data = [];
for m = 1:N
    if binary_data(m) == 1
        manchester_data = [manchester_data 1 -1];
    else
        manchester_data = [manchester_data -1 1];
    end
end

% Manchester Coding Pulse Shaping
i = 1;
l = 0.5;
t = 0:0.01:(2*N-1);

y = zeros(size(t)); % Initialize the output signal

for j = 1:length(t)
    if t(j) <= l % Condition for the first half cycle
        y(j) = manchester_data(i); % Assign values for the first
half cycle
    else
        y(j) = manchester_data(i);
        i = i + 1; % Increment index for binary input data
        l = l + 1; % Increment l for the second half cycle
    end
end
% Plotting
plot(t, y, 'lineWidth', 2); % Linewidth 2 for clear visualization
axis([0, 2*N-1, -1.5, 1.5]); % Axis set-up
grid on;
title('Split Phase-Manchester Coding');
xlabel('Time (bits)');
ylabel('Voltage Level');
```

OUTPUT:

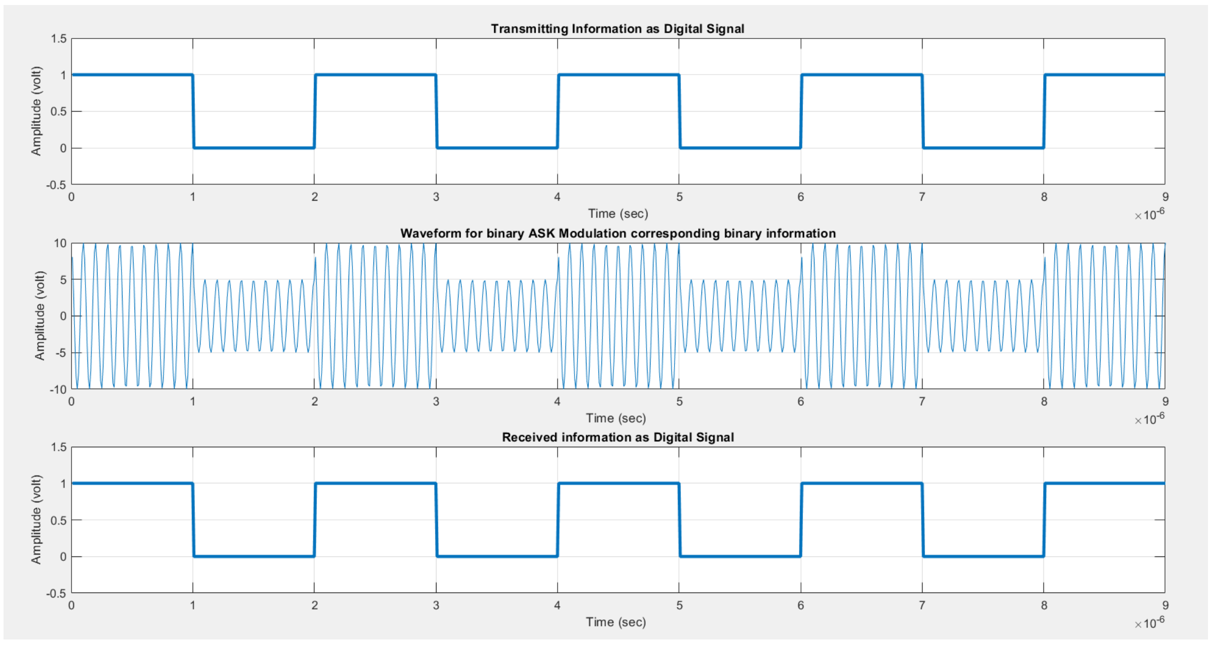

Split Phase-Manchester Coding
Voltage Level
Time (bits)
1.5
1
0.5
0
-0.5
-1
-1.5
0
2
4
6
8
10
12
14
16
18

CODE:

```
clc;
clear;
close all;
x = [1 0 1 0 1 0 1 0 1];
bp = 0.000001;
% Session: 2018-2019
disp('Binary Information at transmitter');
disp(x);
% Representation of Transmitting binary information as digital
signal
bit = [];
for n = 1:length(x)
    if x(n) == 1
        se = ones(1, 100);
    else
        se = zeros(1, 100);
    end
    bit = [bit se];
end
t1 = bp/100:bp/100:100*length(x)*(bp/100);
subplot(3, 1, 1)
plot(t1, bit, 'linewidth', 2.5);
grid on;
axis([0 bp*length(x) -0.5 1.5]);
ylabel('Amplitude (volt)');
xlabel('Time (sec)');
title('Transmitting Information as Digital Signal');
% Binary ASK Modulation
A1 = 10; A2 = 5;
br = 1/bp; f = br*10;
t2 = bp/99:bp/99:bp;
ss = length(t2);
m = [];
for i = 1:length(x)
    if x(i) == 1
        y = A1*cos(2*pi*f*t2);
    else
        y = A2*cos(2*pi*f*t2);
    end
    m = [m y];
end
t3 = bp/99:bp/99:bp*length(x);
subplot(3, 1, 2); plot(t3, m); grid on;
xlabel('Time (sec) '); ylabel('Amplitude (volt)');
title('Waveform for binary ASK Modulation corresponding binary
information');
% Binary ASK Demodulation
mn = [];
```

```
for n = ss:ss:length(m)
    t = bp/99:bp/99:bp;
    y = cos(2*pi*f*t);
    mm = y.*m((n-(ss-1)):n);
    t4 = bp/99:bp/99:bp;
    z = trapz(t4, mm);
    zz = round((2*z/bp));
    if (zz > 7.5)
        a = 1;
    else
        a = 0;
    end
    mn = [mn a];
end
disp('Binary Information at Receiver'); disp(mn);
bit = []; % Represntation of Binary data into Digital signal
for n = 1:length(mn)
    if mn(n) == 1
        se = ones(1, 100);
    else
        se = zeros(1, 100);
    end
    bit = [bit se];
end
t4 = bp/100:bp/100:100*length(mn)*(bp/100);
subplot(3, 1, 3)
plot(t4, bit, 'linewidth', 2.5); grid on;
axis([0 bp*length(mn) -0.5 1.5]);
xlabel('Time (sec) '); ylabel('Amplitude (volt)');
title('Received information as Digital Signal');
```

OUTPUT:



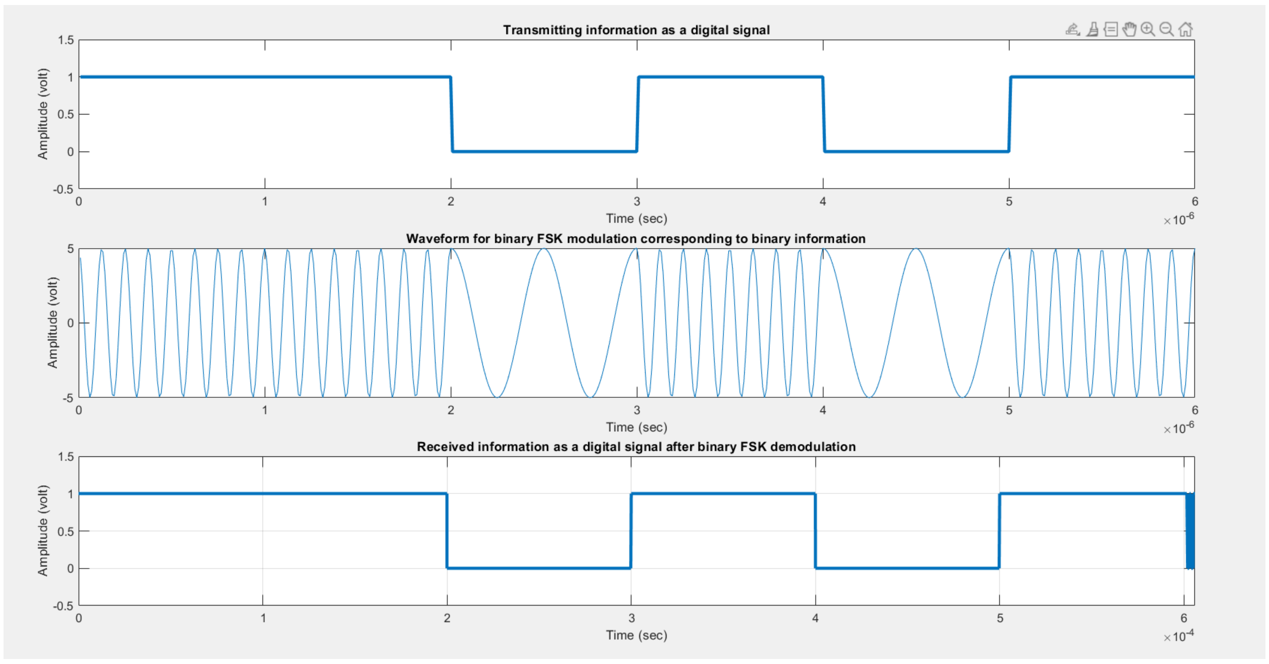
CODE:

```
clc;
clear ;
close all;
% Binary information to be transmitted
x = [1 1 0 1 0 1];
bp = 0.000001; % Bit period
disp('Binary information at transmitter');
disp(x);
% Representation of transmitting binary information as a
digital signal
bit = [];
for n = 1:length(x)
    if x(n) == 1
        se = ones(1, 100);
    else
        se = zeros(1, 100);
    end
    bit = [bit se];
end
tl = bp/100 : bp/100 : 100 * length(x) * (bp/100);
subplot(3, 1, 1);
grid on;
plot(tl, bit, 'linewidth', 2.5);
A = 5;
axis([0 bp * length(x) -0.5 1.5]);
ylabel('Amplitude (volt)');
xlabel('Time (sec)');
title('Transmitting information as a digital signal');
% Binary FSK Modulation
br = 1 / bp;
fl = br * 8;
f2 = br * 2;
t2 = bp / 99 : bp / 99 : bp;
ss = length(t2);
m = [];
for i = 1:length(x)
    if x(i) == 1
        y = A * cos(2 * pi * fl * t2);
    else
        y = A * cos(2 * pi * f2 * t2);
    end
    m = [m y];
end
```

```
t3 = bp / 99 : bp / 99 : bp * length(x);

subplot(3, 1, 2);
plot(t3, m);
xlabel('Time (sec)');
ylabel('Amplitude (volt)');
title('Waveform for binary FSK modulation corresponding to
binary information');

% Binary FSK Demodulation
mn = [];
for n = ss : ss : length(m)
    t = bp / 99 : bp / 99 : bp;
    y1 = cos(2 * pi * f1 * t);
    y2 = cos(2 * pi * f2 * t);
    mn = [mn y1 .* m((n - (ss - 1)) : n)];
    mn = [mn y2 .* m((n - (ss - 1)) : n)];
    t4 = bp / 99 : bp / 99 : bp;
    z1 = trapz(t4, mn(1:ss));
    z2 = trapz(t4, mn(ss + 1 : end));
    zz1 = round(2 * z1 / bp);
    zz2 = round(2 * z2 / bp);
    if zz1 > A / 2
        bit = [bit 1];
    elseif zz2 > A / 2
        bit = [bit 0];
    end
    mn = [];
end
disp('Binary information at Receiver');
disp(bit);
% Representation of binary information as a digital signal
bit_received = [];
for n = 1:length(bit)
    if bit(n) == 1
        se = ones(1, 100);
    else
        se = zeros(1, 100);
    end
    bit_received = [bit_received se];
end
t4 = bp / 100 : bp / 100 : 100 * length(bit) * (bp / 100);
subplot(3, 1, 3);
plot(t4, bit_received, 'Linewidth', 2.5);
grid on;
```

```
axis([0 bp * length(bit) -0.5 1.5]);
ylabel('Amplitude (volt)');
xlabel('Time (sec)');
title('Received information as a digital signal after binary
FSK demodulation');
```

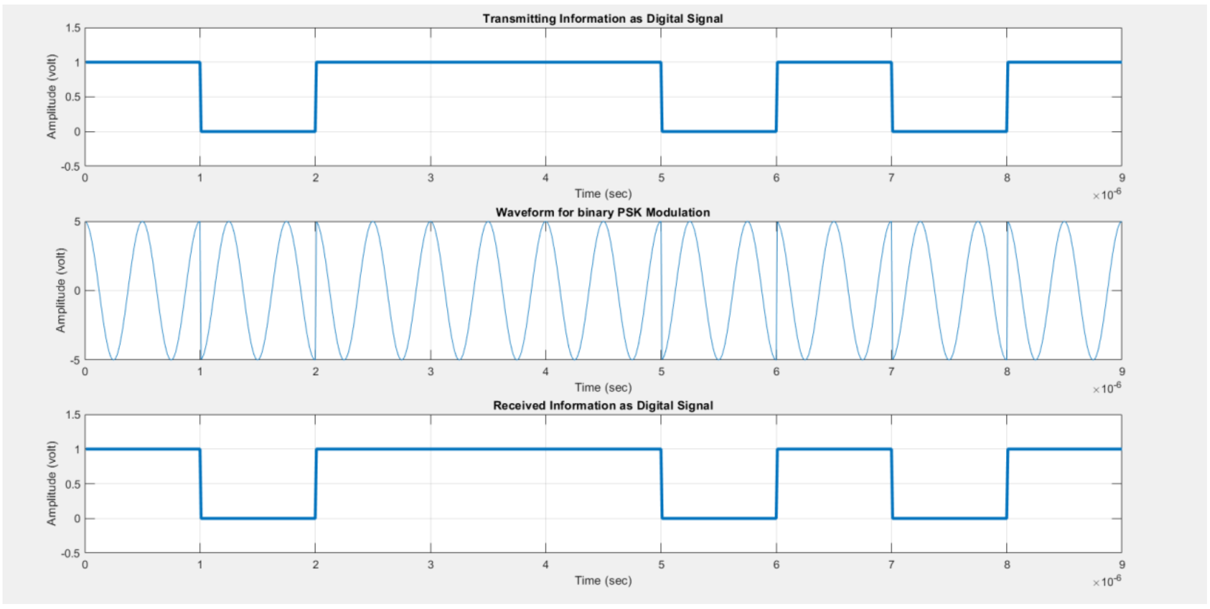
OUTPUT:



CODE:

```
clc;
clear;
close all;
x=[1 0 1 1 1 0 1 0 1];
bp=0.000001;
disp('Binary Information at transmitter');
disp(x);
% Representation of Transmitting binary information as digital
signal
bit=[];
for n=1:length(x)
    if x(n)==1
        se=ones(1,100);
    else
        se=zeros(1,100);
    end
    bit=[bit se];
end
tl=bp/100:bp/100:100*length(x)*(bp/100);
subplot(3,1,1)
plot(tl, bit, 'linewidth', 2.5);
grid on;
axis([0 bp*length(x) -0.5 1.5]);
ylabel('Amplitude (volt)');
xlabel('Time (sec)');
title('Transmitting Information as Digital Signal');
% Binary PSK Modulation
A=5;
br=1/bp;
f=br*2;
t2=bp/99:bp/99:bp;
ss=length(t2);
m=[];
for i=1:length(x)
    if x(i)==1
        y=A*cos(2*pi*f*t2);
    else
        y=A*cos(2*pi*f*t2+pi);
    end
    m=[m y];
end
t3=bp/99:bp/99:bp*length(x);
subplot(3,1,2);plot(t3,m);grid on;
xlabel('Time (sec)'); ylabel('Amplitude (volt)');
title('Waveform for binary PSK Modulation');
mn=[];% Binary PSK Demodulation
for n=ss:ss:length(m)
    t=bp/99:bp/99:bp;
    y=cos(2*pi*f*t); % Carrier signal
    mm=y.*m((n-(ss-1)):n);
    t4=bp/99:bp/99:bp;
    z=trapz(t4,mm); % Integration
    zz=round(2*z/bp);
    if(zz>0)
        a=1;
    else
        a=0;
    end
    mn=[mn a];
end
disp('Binary Information at Receiver After PSK Demodulation');
disp(mn);% Representation of Binary data into Digital signal
bit=[];
for n=1:length(mn)
    if mn(n)==1
        se=ones(1,100);
    else
        se=zeros(1,100);
    end
    bit=[bit se];
end
t4=bp/100:bp/100:100*length(mn)*(bp/100);
subplot(3,1,3);plot(t4,bit,'linewidth',2.5);
grid on;axis([0 bp*length(mn) -0.5 1.5]);
xlabel('Time (sec)') ;ylabel('Amplitude (volt)');
title('Received Information as Digital Signal');
```

OUTPUT:

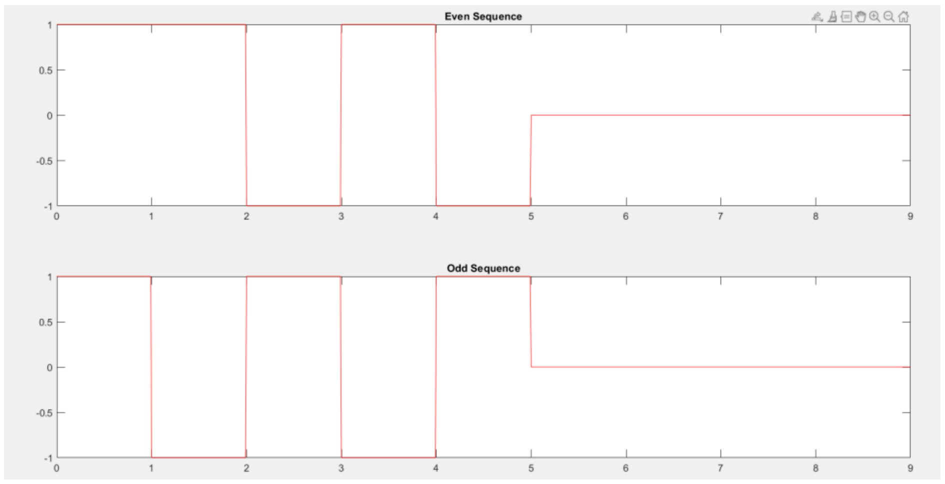


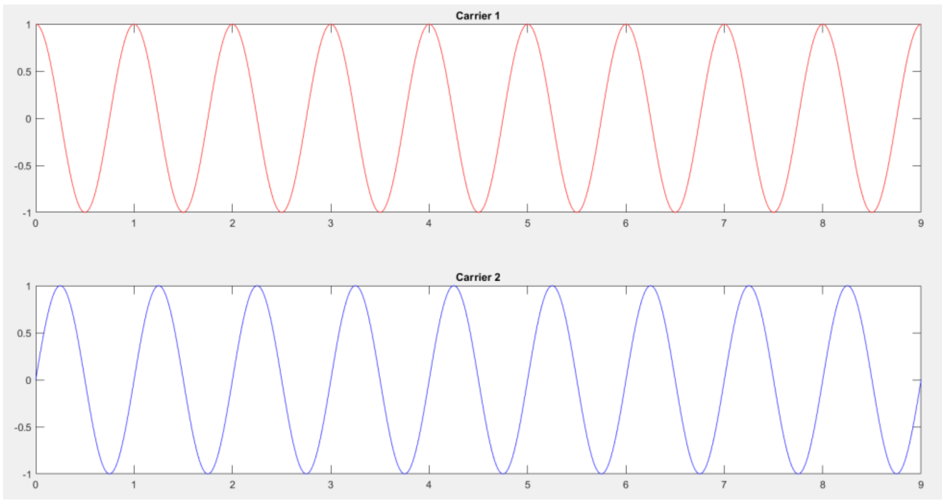
CODE:

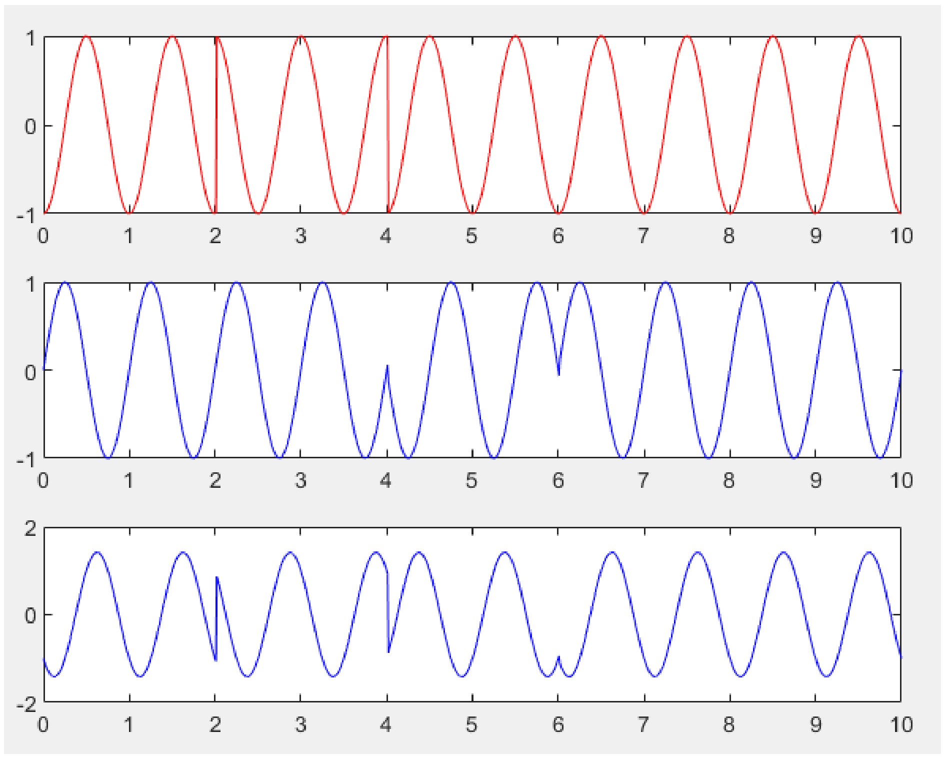
```
%QPSK waveform generation
clc;
clear ;
close all;
x=[0 1 1 0 1 0 0 1];%input bits
%x=randi([0,1],1,10)

%Bits to polar
for i=1:length(x)
    if x(i)==0
        p(i)=-1;
    else
        p(i)=1;
    end
end

%separation of even and odd sequence
even_seq = p(1:2:length(x));
odd_seq = p(2:2:length(x));


t = 0:0.01:length(x);
%Unipolar-NRZ polar line coder signal generation
i = 1;
m= 2:2:length(x);
for j=1:length(t)
    if t(j)<=m(i)
        even_ps(j)=even_seq(i);
    else
        even_ps(j)=even_seq(i);
        i=i+1;
    end
end

i=1;
m=2:2:length(x);
for j=1:length(t)
    if t(j)<=m(i)
        odd_ps(j)=odd_seq(i);
    else
        odd_ps(j)=odd_seq(i);
        i=i+1;
    end
end
```

```
%Showing the first figure
figure(1)
subplot(2,1,1);
plot(t,even_ps,'r');
title('Even Sequences');
subplot(212);
plot(t,odd_ps,'r');
title('Odd Sequences');

%carrier signals generation
c1=cos(2*pi*1*t);
c2=sin(2*pi*1*t);


%Showing the Second figure
figure(2)
subplot(2,1,1);
plot(t,c1,'r');
title('The First Carrier Signal');
subplot(2,1,2)
plot(t,c2,'b');
title('The Second Carrier Signal');


%QPSK waveform generation
r1=even_ps.*c1;
r2=odd_ps.*c2;
qpsk_sig=r1-r2;


%Showing the Second figure
figure(3)
subplot(3,1,1)
plot(t,r1,'r');
title('Modulated Even Sequence');
subplot(3,1,2)
plot(t,r2,'b');
title('Modulated Odd Sequence');
subplot(3,1,3);
plot(t,qpsk_sig,'b');
title('QPSK Waveform');
```

OUTPUT:



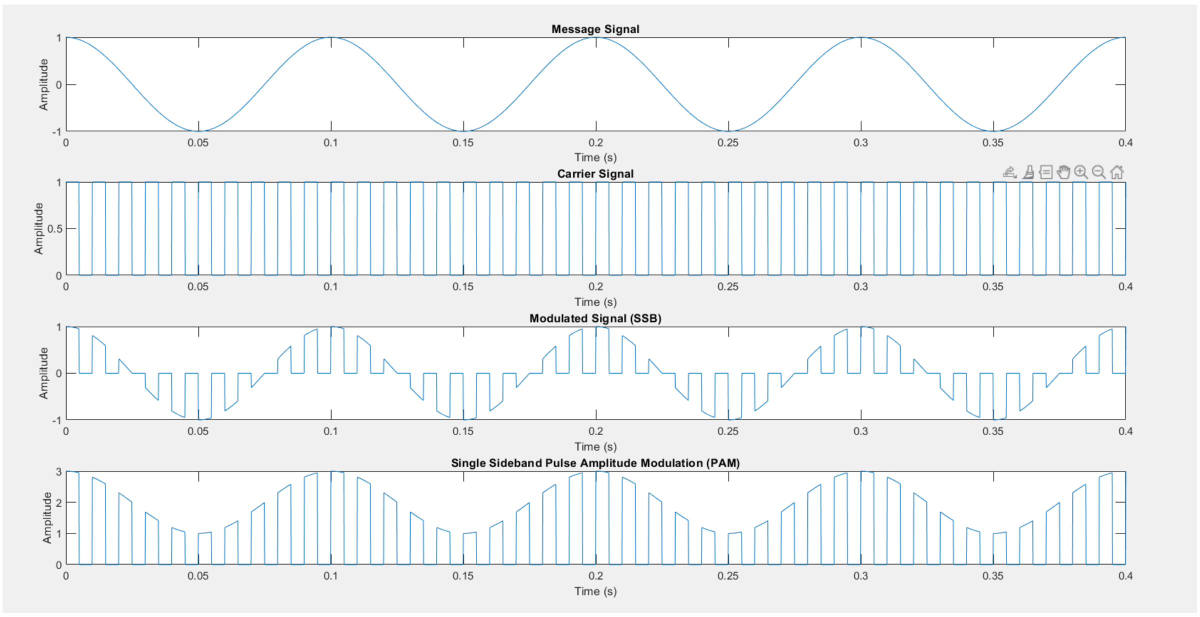




CODE:

```
clc;clear ;close all;
fc = 100;fm = fc / 10;
fs = 100 * fc;
t = 0:1/fs:4/fm;mt = cos(2*pi*fm*t);
ct = 0.5*square(2*pi*fc*t) + 0.5;
st = mt.*ct;tt = [];
% Single-sided PAM
for i = 1:length(st)
    if st(i) == 0
        tt = [tt, st(i)];
    else
        tt = [tt, st(i) + 2];
    end
end
subplot(4,1,1);plot(t, mt);
title('Message Signal');
xlabel('Time (s)');ylabel('Amplitude');
subplot(4,1,2);plot(t, ct);
title('Carrier Signal');
xlabel('Time (s)');ylabel('Amplitude');
subplot(4,1,3);plot(t, st);
title('Modulated Signal (SSB)');
xlabel('Time (s)');ylabel('Amplitude');
subplot(4,1,4);plot(t, tt);
title('Single Sideband Pulse Amplitude Modulation (PAM)');
xlabel('Time (s)');ylabel('Amplitude');
```

OUTPUT:



INDEX