American International University- Bangladesh (AIUB)

Faculty of Engineering

Course Name:	Data Communication	**Course Code:**	COE 3201
Semester:	Fall 2023-2024	**Term:**	Mid
Total Marks:	…	**Submission Date:**	09-11-2023
Faculty Name:	Mr. Abrar Fahim Liaf	**Assignment:**	Mid-Lab-Exam

Student Information:

Student Name:	**MD. SHAHRIAR PARVEZ SHAMIM**	**Student ID:**	**21-44998-2**
Section:	**I**	**Department:**	**CSE**

Answer to the question number 1

Converting Analog Signal to Digital Data

Code :

```
>> %ID : 21-44998-2
A=2;
B=1;
C=4;
D=4;
E=9;
F=9;
G=8;
H=2;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Analog to Digital Conversion
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
time_duration = 0.2;
%% Analog-like signal's representation
% Analog signal generation is not possible in MATLAB
a1 = B + 1;
a2 = C + 3;
a3 = D + 2;
f1 = E + 5;
f2 = F + 7;
f3 = G + 1;
a=[a1 a2 a3];
f=[f1 f2 f3];
analog_t = 0:0.0001:time_duration;
>> analog_sig = a1*sin(2*pi*f1*analog_t) +
a2*cos(2*pi*f2*analog_t) + a3*cos(2*pi*f3*analog_t);
>> figure
hold on
subplot(1,2,1)
plot(analog_t, analog_sig,'linewidth',1.5)
grid on
xlabel('time in seconds')
ylabel('amplitude in volts')
title('analog signal')
%% Sampling Frequency
fs = 250;
ts = 1/fs;
%% Sampling
samp_t = 0:1/fs:time_duration;
samp_sig = a1*sin(2*pi*f1*samp_t)
+a2*cos(2*pi*f2*samp_t) + a3*cos(2*pi*f3*samp_t);
hold on
subplot(1,2,2)
plot(samp_t, samp_sig,'linewidth',1.5)
grid on
xlabel('time in seconds')
ylabel('amplitude in volts')
title(['sampled signal for ',num2str(fs),' Hz sampling
frequency'])
```

```
%% Levels for Quantization
L = 8;

%% Quantizing
delta = (max(samp_sig) - min(samp_sig))/(L-1); %step size
quant_sig = min(samp_sig) + round((samp_sig-
min(samp_sig))/delta)*delta; % quantized signal
figure
subplot(1,2,1)
plot(samp_t, samp_sig,'linewidth',1.5)
grid on
xlabel('time in seconds')
ylabel('amplitude in volts')
title('sampled signal')
subplot(1,2,2)
plot(samp_t, quant_sig,'linewidth',1.5);
xlabel('time')
ylabel('amplitude')
title('quantized samples')

%% Number of Bits/Sample
nb = log2(L);

%% Encoding
i = round((samp_sig-min(samp_sig))/delta); % index for
encoding
dig_data_matrix = dec2bin(i,nb); % encoded binary bits are as
a matrix here
dig_data = reshape(dig_data_matrix',1,[]); % encoded binary
bits are as an array here
disp(['The index values for encoding from quantization of the
sampled signal are:',num2str(i)])
disp(['The converted bits from the input analog signal are:
',num2str(dig_data)])
```

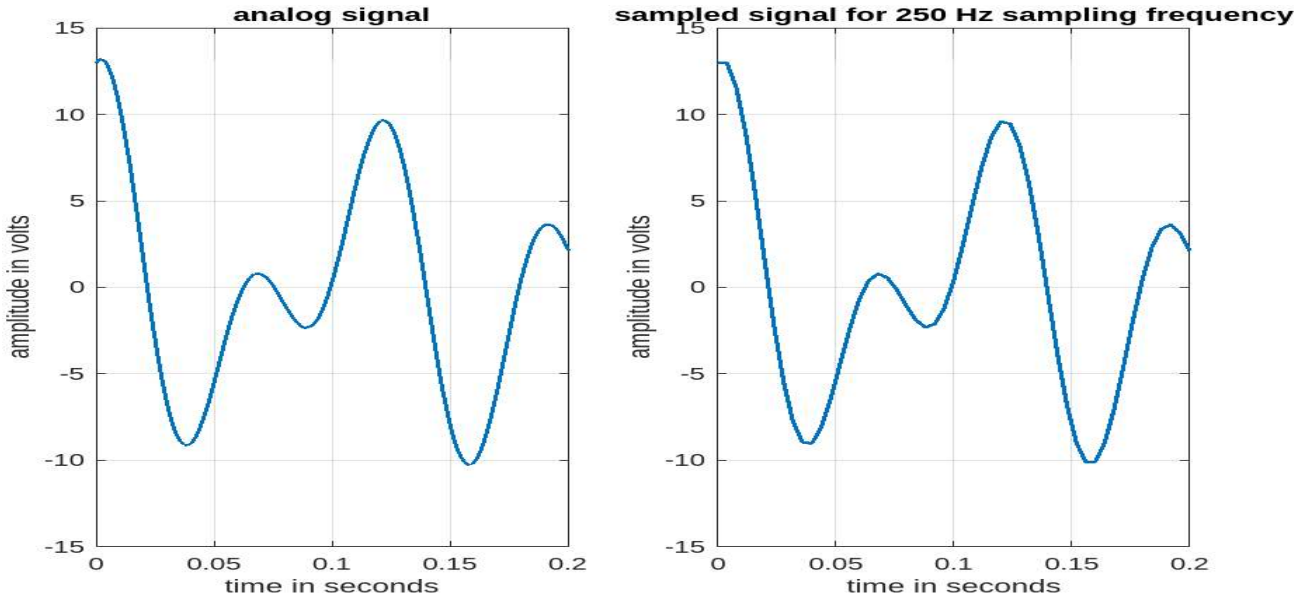

Figure 1 :Analog Signal

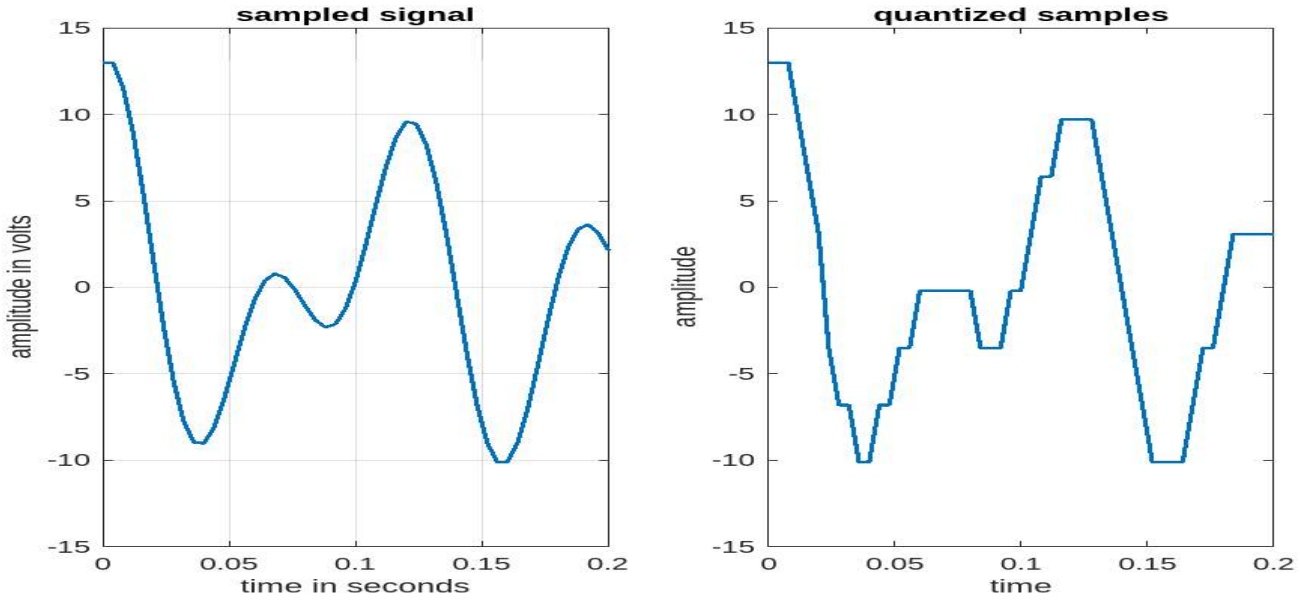

Figure 2 :Quantized Signal

The index values for encoding from quantization of the sampled signal are:
7 7 7 6 5 4 2 1 1 0 0 1 1 2 2 3 3 3 3 3 3 2 2 2 3
3 4 5 5 6 6 6 6 5 4 3 2 1 0 0 0 0 1 2 2 3 4 4 4 4 4

The converted bits from the input analog signal are: 1 1 1 1 1 1 1 1 1 1 1 0 1 0
1 1 0 0 0 1 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 1 0 0 1 0 1 0 0 1 0 0 1 1 0 1 1 0 1 1
0 1 1 0 1 1 0 1 1 0 1 0 0 1 0 0 1 0 0 1 1 0 1 1 1 0 0 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0
1 1 0 1 0 1 1 0 0 0 1 1 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 1 0 0 1 1
1 0 0 1 0 0 1 0 0 1 0 0 1 0 0

Answer to the question number 2

Converting Digital Data to Digital Signal

Code :

Unipolar NRZ	Differential Manchester
bit_stream = [1 1 1 1 1 1 1 1 1 1 1 0 1 0 1 1 0 0 0 1 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 1 0 0 1 0 1 0 0 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 0 0 1 0 0 1 0 0 1 1 0 1 1 1 0 0 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 0 1 0 1 1 0 0 0 1 1 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 1 0 0 1 0 0 1 1 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0]; no_bits = length(bit_stream); bit_rate = 1000; % 1 kbps pulse_per_bit = 1; % for unipolar nrz pulse_duration = 1/((pulse_per_bit)*(bit_rate)); no_pulses = no_bits*pulse_per_bit; samples_per_pulse = 500; fs = (samples_per_pulse)/(pulse_duration); %sampling frequency % including pulse duration in sampling frequency % ensures having enough samples in each pulse t = 0:1/fs:(no_pulses)*(pulse_duration); % sampling interval % total duration = (no_pulse)*(pulse_duration) no_samples = length(t); % total number of samples dig_sig = zeros(1,no_samples); max_voltage = 5; min_voltage = 0; for i = 1:no_bits if bit_stream(i) == 1 dig_sig(((i-1)*(samples_per_pulse)+1):i*(samples_per_pulse)) = max_voltage*ones(1,samples_per_pulse); else dig_sig(((i-1)*(samples_per_pulse)+1):i*(samples_per_pulse)) = min_voltage*ones(1,samples_per_pulse); end end plot(t,dig_sig,'linewidth',1.5) grid on xlabel('time in seconds') ylabel('Voltage') ylim([(min_voltage - (max_voltage)*0.2) (max_voltage+max_voltage*0.2)]) title(['Unipolar NRZ for ',num2str(bit_stream),''])	bit_stream = [1 1 1 1 1 1 1 1 1 1 1 0 1 0 1 1 0 0 0 1 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 1 0 0 1 0 1 0 0 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 0 0 1 0 0 1 0 0 1 0 0 1 1 0 1 1 1 0 0 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 0 1 0 1 1 0 0 0 1 1 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 1 0 0 1 1 1 0 0 1 0 0 1 0 0 1 0 0]; no_bits = length(bit_stream); bit_rate = 1000; % 1 kbps pulse_per_bit = 2; % for differential manchester pulse_duration = 1/((pulse_per_bit)*(bit_rate)); no_pulses = no_bits*pulse_per_bit; samples_per_pulse = 500; fs = (samples_per_pulse)/(pulse_duration); %sampling fr. t = 0:1/fs:(no_pulses)*(pulse_duration); % sampling interval no_samples = length(t); % total number of samples dig_sig = zeros(1,no_samples); max_voltage = +2; min_voltage = -2; inv_bit = 1; % inverting bit last_state = max_voltage; inv_last_state = min_voltage; % inverse of last state for i = 1:no_bits j = (i-1)*2; if bit_stream(i) == inv_bit dig_sig((j*(samples_per_pulse)+1):(j+1)*(samples_per_pulse)) = inv_last_state*ones(1,samples_per_pulse); dig_sig(((j+1)*(samples_per_pulse)+1):(j+2)*(samples_per_pulse)) = last_state*ones(1,samples_per_pulse); else dig_sig((j*(samples_per_pulse)+1):(j+1)*(samples_per_pulse)) = last_state*ones(1,samples_per_pulse); dig_sig(((j+1)*(samples_per_pulse)+1):(j+2)*(samples_per_pulse)) = inv_last_state*ones(1,samples_per_pulse); temp_cons = last_state; % temporary constant last_state = inv_last_state; inv_last_state = temp_cons; end end figure plot(t,dig_sig,'linewidth',1.5) grid on xlabel('time in seconds') ylabel('Voltage') ylim([(min_voltage - (max_voltage)*0.2) (max_voltage+max_voltage*0.2)]) title(['Differential Manchester for ',num2str(bit_stream),', last state = ',num2str(last_state),', inverting bit is ',num2str(inv_bit),''])

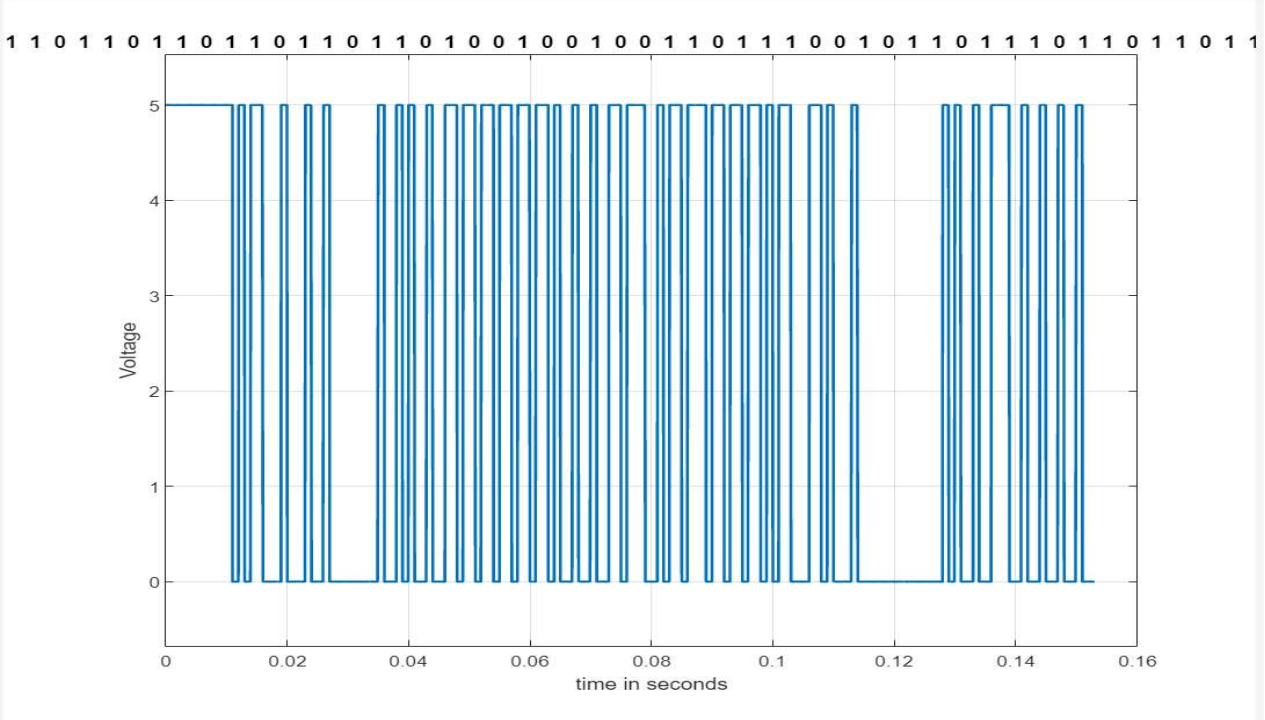

Figure 3 : Unipolar NRZ

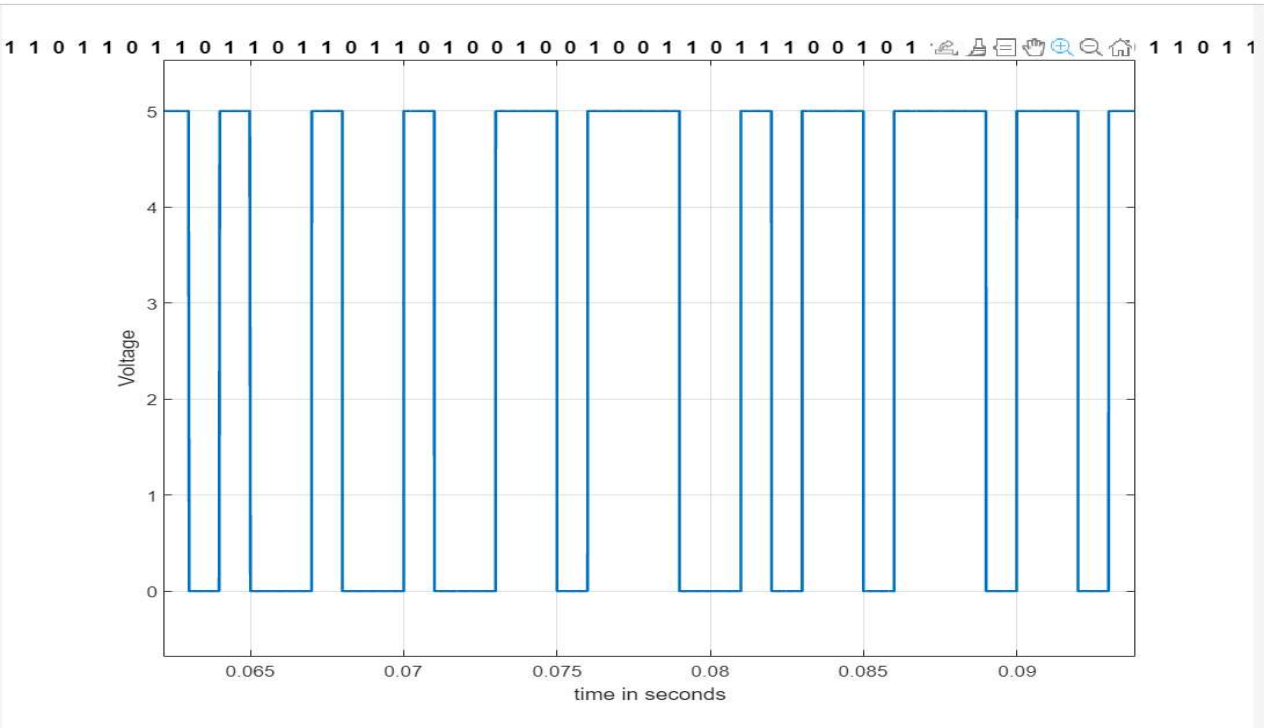

Figure 4 : Unipolar NRZ (Zoomed in)

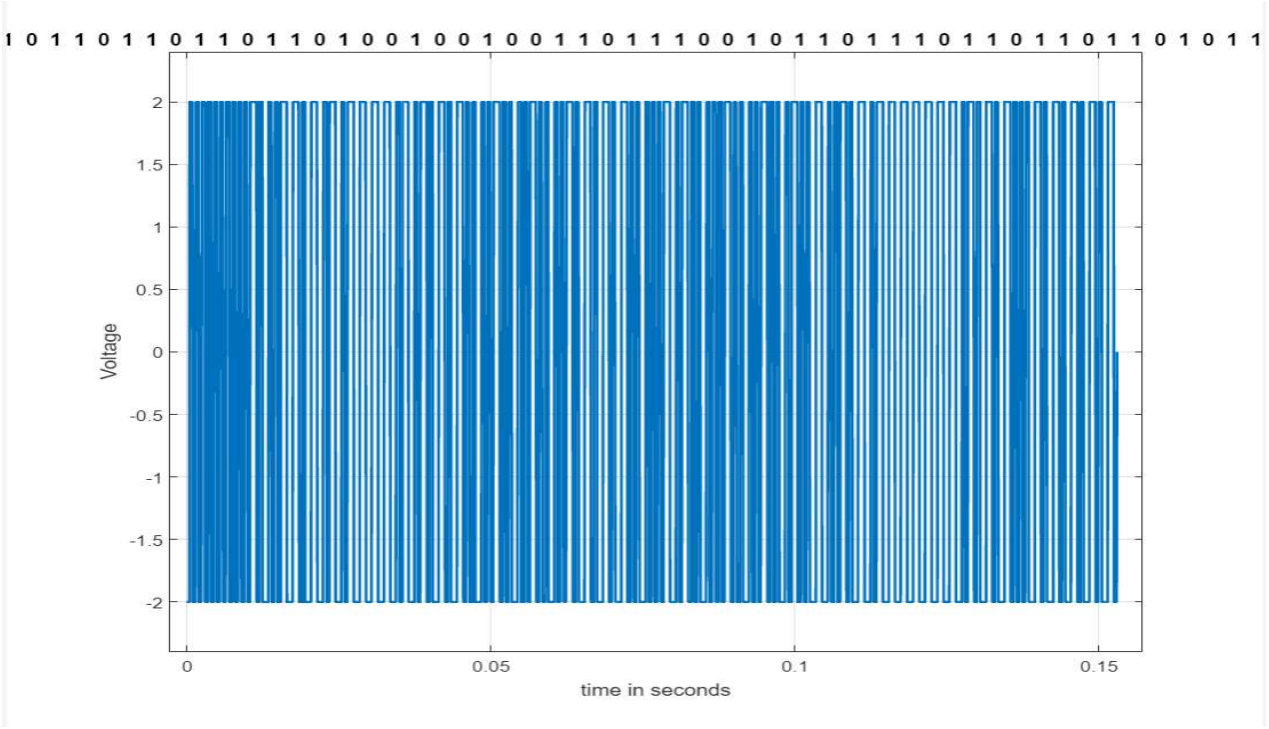

Figure 5 : Differential Manchester

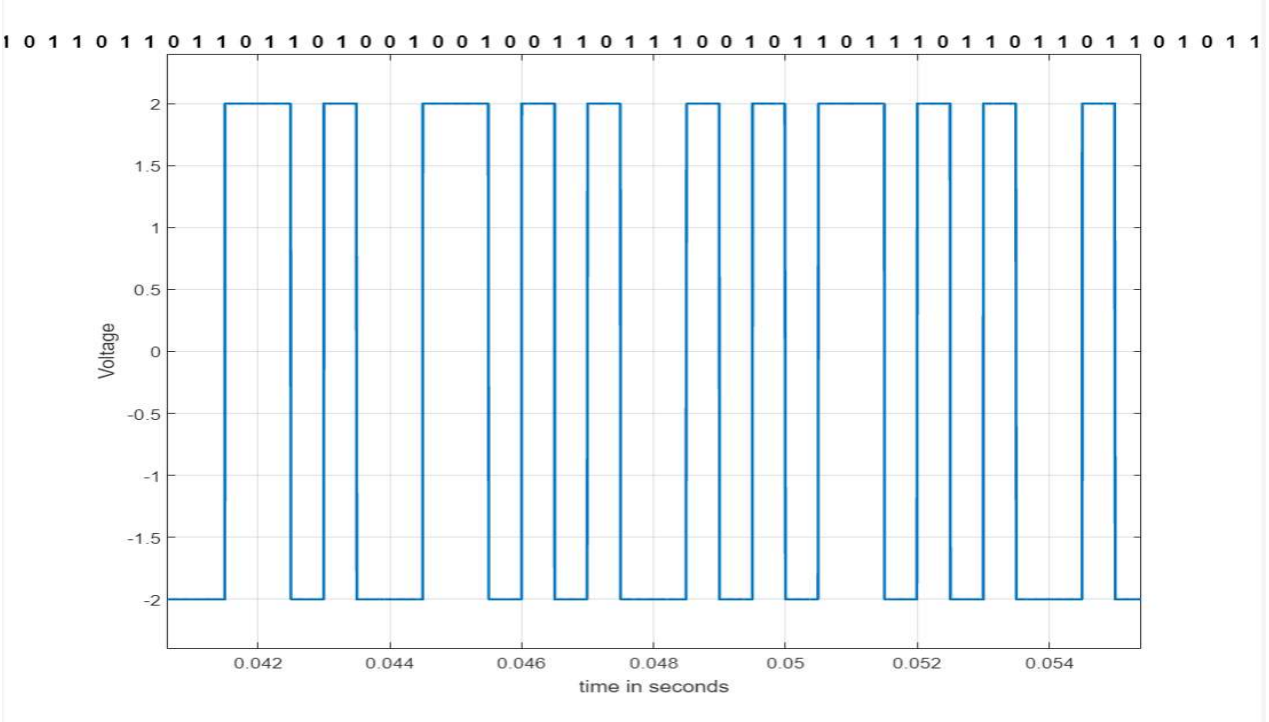

Figure 6 : Differential Manchester (Zoomed in)