|  |  |  |  |
| --- | --- | --- | --- |
| AIUB | **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 2 :Quantized Signal**

**Figure 1 :Analog 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 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 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 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];  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 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 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];  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 6 : Differential Manchester (Zoomed in)**

**Figure 5 : Differential Manchester**