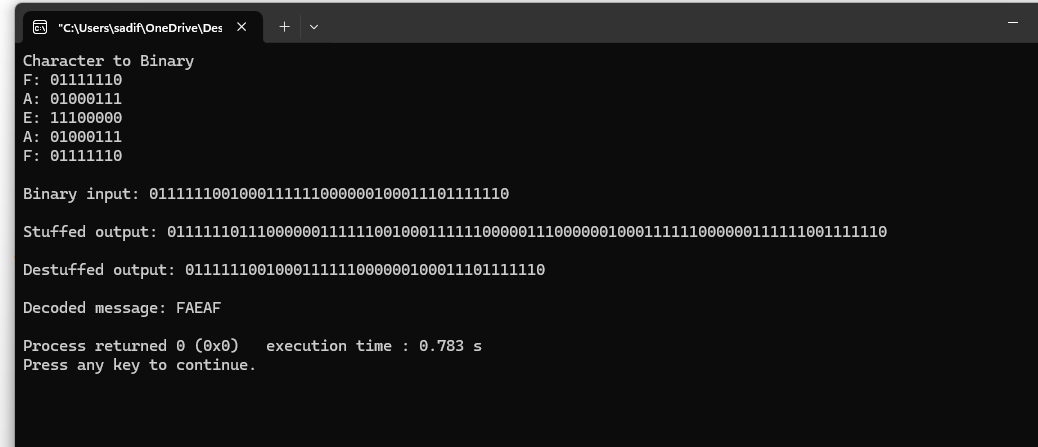
**MD Ashiqul Islam   
Id: 212002056**

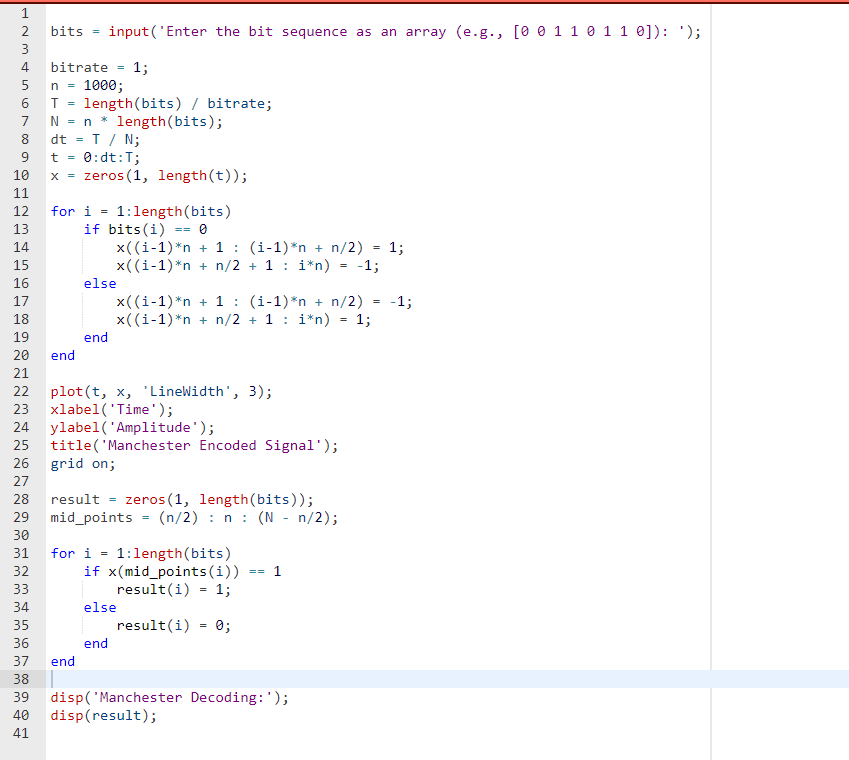
|  |
| --- |
| **// Ashiqul Islam sadif**  **#include<bits/stdc++.h>**  **using namespace std;**  **#define ll long long**  **#define N INT\_MAX**  **#define ss string**  **#define mod 1000000007**  **#define fr(i, s, n) for (ll i = s; i < n; i++)**  **const double PI = 3.141592653589793238460;**  **const ss A = "01000111";**  **const ss FLAG = "01111110";**  **const ss ESC = "11100000";**  **ss ctb(char c)**  **{**  **switch (c)**  **{**  **case 'A':**  **return A;**  **case 'F':**  **return FLAG;**  **case 'E':**  **return ESC;**  **default:**  **return "";**  **}**  **}**  **ss sTbnry(const ss& s)**  **{**  **ss bout;**  **for (char c : s)**  **{**  **bout += ctb(c) + " ";**  **}**  **return bout;**  **}**  **ss crcStf(const ss& s, const ss& flagWord, const ss& escapeWord)**  **{**  **ss sout;**  **sout += flagWord;**  **for (size\_t i = 0; i < s.size(); i += 8)**  **{**  **ss byte = s.substr(i, 8);**  **if (byte == flagWord || byte == escapeWord)**  **{**  **sout += escapeWord;**  **}**  **sout += byte;**  **}**  **sout += flagWord;**  **return sout;**  **}**  **ss crcDStf(const ss& s, const ss& flagWord, const ss& escapeWord)**  **{**  **ss dout;**  **size\_t pos = flagWord.size();**  **size\_t end = s.size() - flagWord.size();**  **while (pos < end)**  **{**  **ss byte = s.substr(pos, 8);**  **if (byte == escapeWord)**  **{**  **pos += 8;**  **byte = s.substr(pos, 8);**  **}**  **dout += byte;**  **pos += 8;**  **}**  **return dout;**  **}**  **int main()**  **{**  **ss s = "FAEAF";**  **cout << "Character to Binary " << endl;**  **for (char c : s)**  **{**  **cout << c << ": " << ctb(c) << endl;**  **}**  **ss binaryInput;**  **for (char c : s)**  **{**  **binaryInput += ctb(c);**  **}**  **cout << "\nBinary input: " << binaryInput << endl;**  **ss stf = crcStf(binaryInput, FLAG, ESC);**  **cout << "\nStuffed output: " << stf << endl;**  **ss dtf = crcDStf(stf, FLAG, ESC);**  **cout << "\nDestuffed output: " << dtf << endl;**  **ss DcMsg;**  **for (size\_t i = 0; i < dtf.size(); i += 8)**  **{**  **ss byte = dtf.substr(i, 8);**  **if (byte == A)**  **{**  **DcMsg += 'A';**  **}**  **else if (byte == FLAG)**  **{**  **DcMsg += 'F';**  **}**  **else if (byte == ESC)**  **{**  **DcMsg += 'E';**  **}**  **}**  **cout << "\nDecoded message: " << DcMsg << endl;**  **return 0;**  **}** |

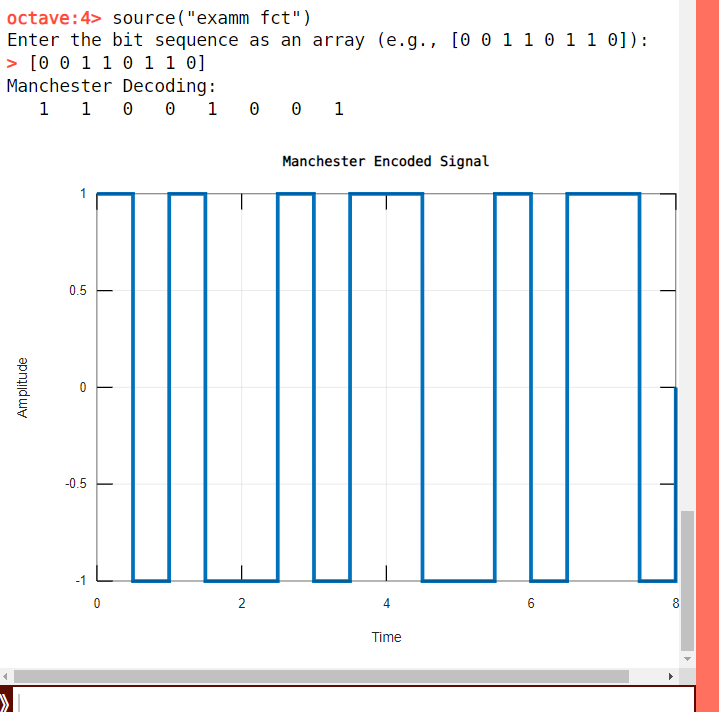
****

**…………**

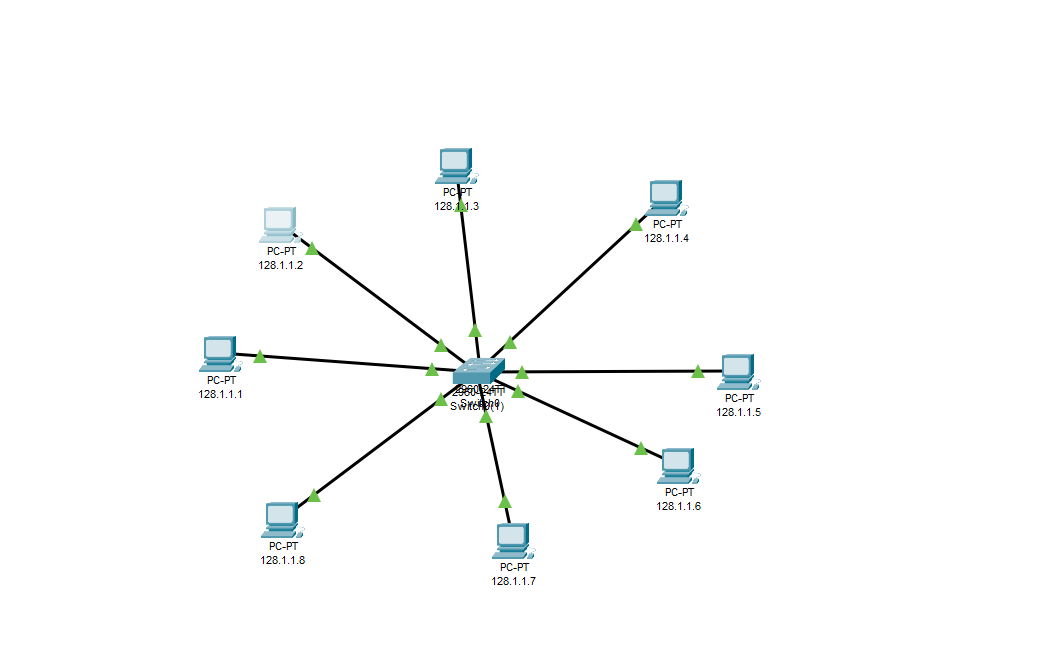
|  |
| --- |
| **bits = input('Enter the bit sequence as an array (e.g., [0 0 1 1 0 1 1 0]): ');**  **bitrate = 1;**  **n = 1000;**  **T = length(bits) / bitrate;**  **N = n \* length(bits);**  **dt = T / N;**  **t = 0:dt:T;**  **x = zeros(1, length(t));**  **for i = 1:length(bits)**  **if bits(i) == 0**  **x((i-1)\*n + 1 : (i-1)\*n + n/2) = 1;**  **x((i-1)\*n + n/2 + 1 : i\*n) = -1;**  **else**  **x((i-1)\*n + 1 : (i-1)\*n + n/2) = -1;**  **x((i-1)\*n + n/2 + 1 : i\*n) = 1;**  **end**  **end**  **plot(t, x, 'LineWidth', 3);**  **xlabel('Time');**  **ylabel('Amplitude');**  **title('Manchester Encoded Signal');**  **grid on;**  **result = zeros(1, length(bits));**  **mid\_points = (n/2) : n : (N - n/2);**  **for i = 1:length(bits)**  **if x(mid\_points(i)) == 1**  **result(i) = 1;**  **else**  **result(i) = 0;**  **end**  **end**  **disp('Manchester Decoding:');**  **disp(result);** |

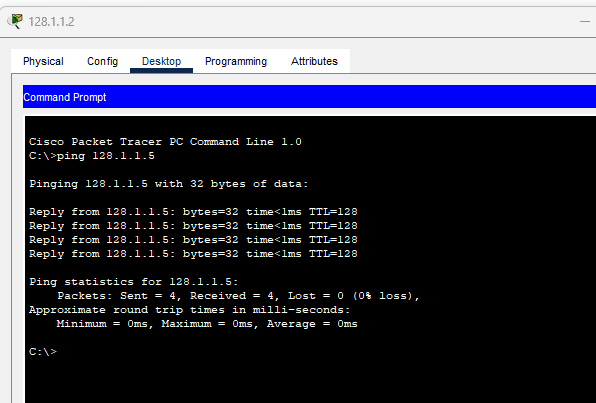
**Ss**

****

****

**..**

****

****