Experiment no. 1
Python code:

```
import numpy as np
c1=[1,1,1,1]
c2=[1,-1,1,-1]
c3=[1,1,-1,-1]
c4=[1,-1,-1,1]
rc=[]

print("Enter the data bits :")

d1=int(input("Enter D1 :"))
d2=int(input("Enter D2 :"))
d3=int(input("Enter D3 :"))
d4=int(input("Enter D4 :"))
r1=np.multiply(c1,d1)
r2=np.multiply(c2,d2)
r3=np.multiply(c3,d3)
r4=np.multiply(c4,d4)
resultant_channel=r1+r2+r3+r4;
print("Resultant Channel",resultant_channel)
Channel=int(input("Enter the station to listen for C1=1 ,C2=2, C3=3
C4=4 : "))

if Channel==1: rc=c1
elif Channel==2: rc=c2
elif Channel==3: rc=c3
elif Channel==4: rc=c4
inner_product = np.multiply(resultant_channel,rc)

print("Inner Product",inner_product)
res1=sum(inner_product)

data=res1/len(inner_product)
print("Data bit that was sent",data)
```

Output:

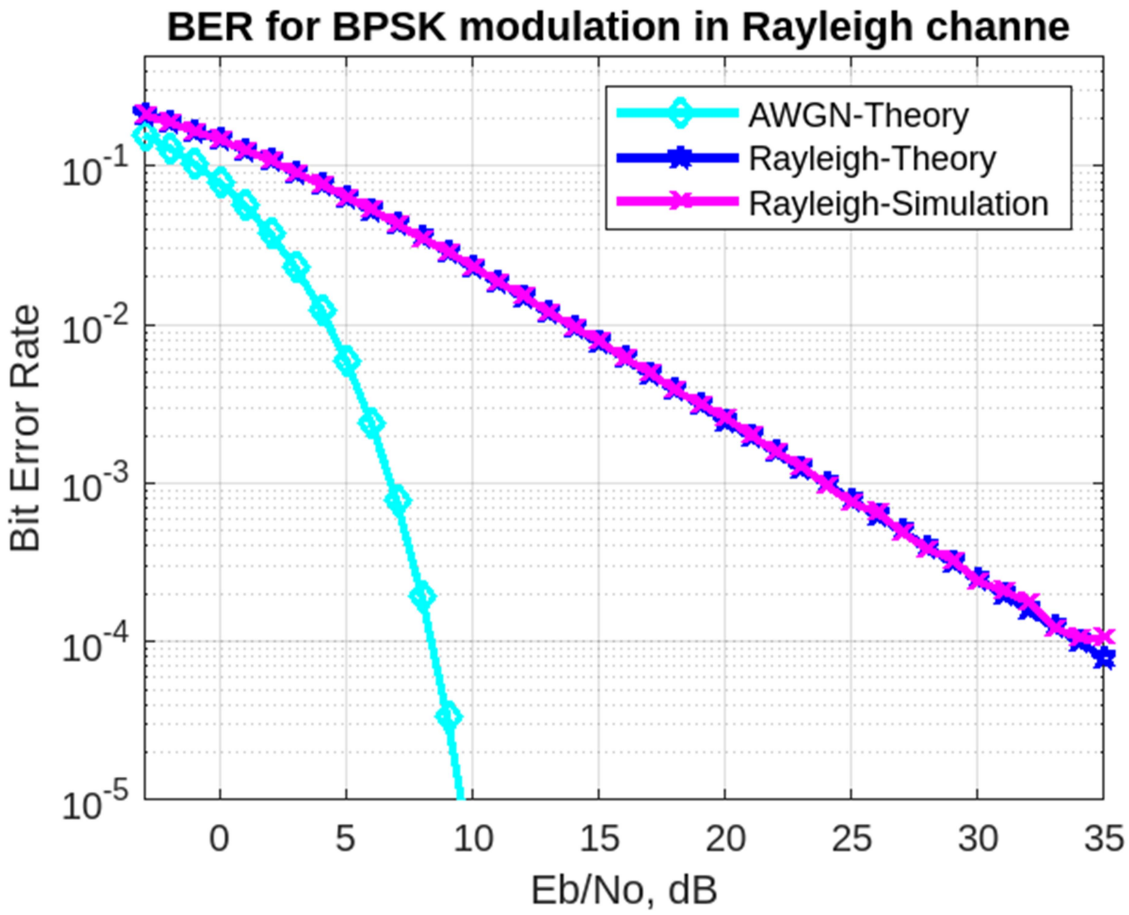
```
Enter the data bits :
Enter D1 :1
Enter D2 :0
Enter D3 :1
Enter D4 :0
Resultant Channel [2 2 0 0]
Enter the station to listen for C1=1 ,C2=2, C3=3 C4=4 : 3
Inner Product [2 2 0 0]
Data bit that was sent 1.0
```

Experiment no. 4
MATLAB Code:

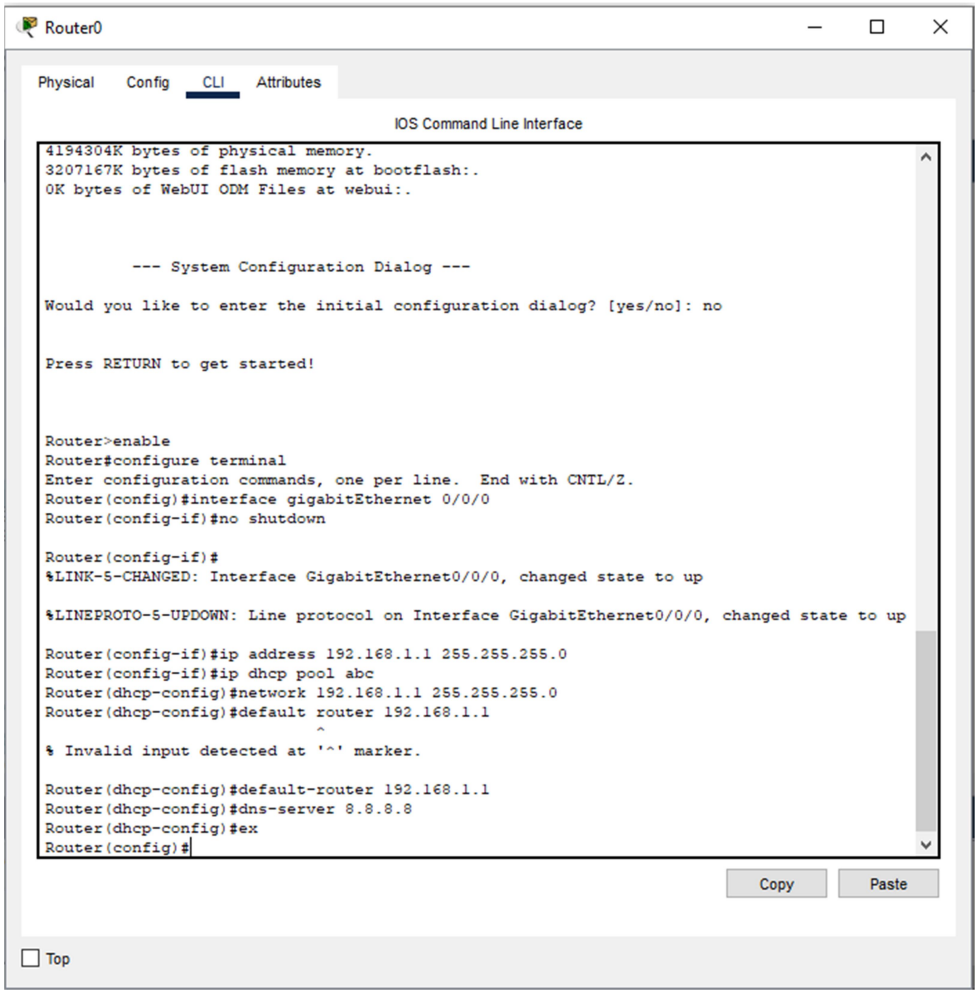
```
clear

N = 10^6 % number of bits or symbols

% Transmitter

ip = rand(1,N)>0.5; % generating 0,1 with equal probability

s = 2*ip-1; % BPSK modulation 0 -> -1; 1 -> 0


Eb_N0_dB = [-3:35]; % multiple Eb/N0 values

for ii = 1:length(Eb_N0_dB)


  n = 1/sqrt(2)*[randn(1,N) + j*randn(1,N)]; % white gaussian noise, 0dB variance

  h = 1/sqrt(2)*[randn(1,N) + j*randn(1,N)]; % Rayleigh channel


  % Channel and noise Noise addition

  y = h.*s + 10^(-Eb_N0_dB(ii)/20)*n;

  % equalization

  yHat = y./h;

  % receiver - hard decision decoding

  ipHat = real(yHat)>0;

  % counting the errors

  nErr(ii) = size(find([ip- ipHat]),2);

end

simBer = nErr/N; % simulated ber

theoryBerAWGN = 0.5*erfc(sqrt(10.^(Eb_N0_dB/10))); % theoretical ber
```

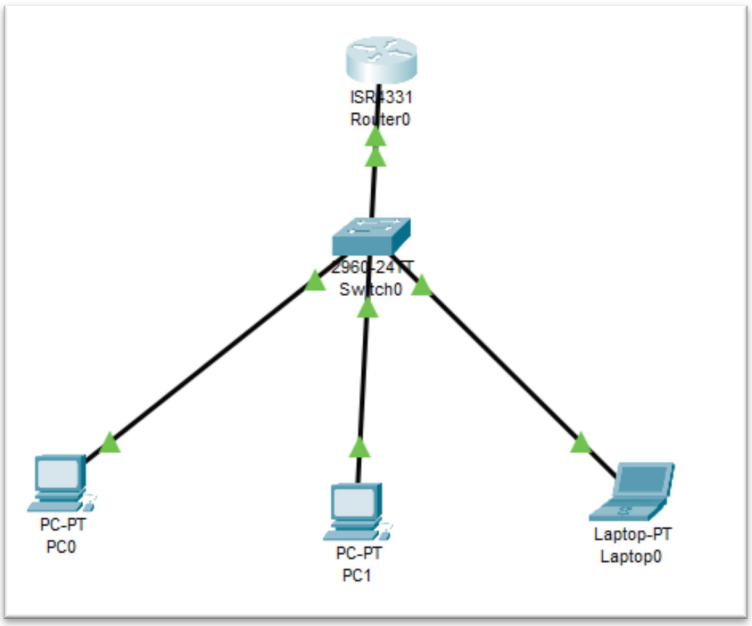
```
EbN0Lin = 10.^(Eb_N0_dB/10);
theoryBer = 0.5.*(1-sqrt(EbN0Lin./(EbN0Lin+1)));
% plot
close all
figure
semilogy(Eb_N0_dB,theoryBerAWGN,'cd-','LineWidth',2);
hold on
semilogy(Eb_N0_dB,theoryBer,'bp-','LineWidth',2);
semilogy(Eb_N0_dB,simBer,'mx-','LineWidth',2);
axis([-3 35 10^-5 0.5])
grid on
legend('AWGN-Theory','Rayleigh-Theory', 'Rayleigh-Simulation');
xlabel('Eb/No, dB');
ylabel('Bit Error Rate');
title('BER for BPSK modulation in Rayleigh channel');
```

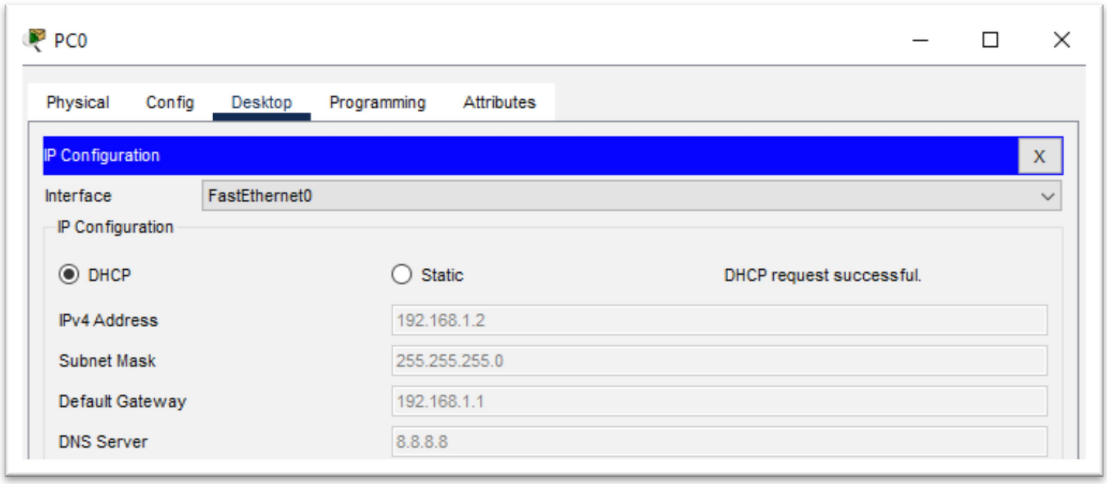
My output:

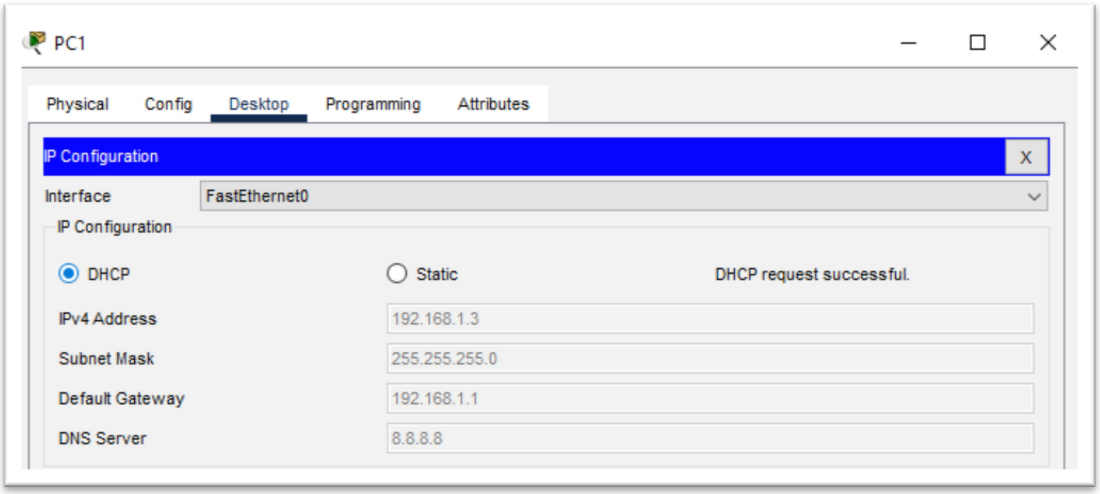

BER for BPSK modulation in Rayleigh channe
AWGN-Theory
Rayleigh-Theory
Rayleigh-Simulation
Bit Error Rate
10-1
10-2
10-3
10-4
10-5
0
5
10
15
20
25
30
35
Eb/No, dB

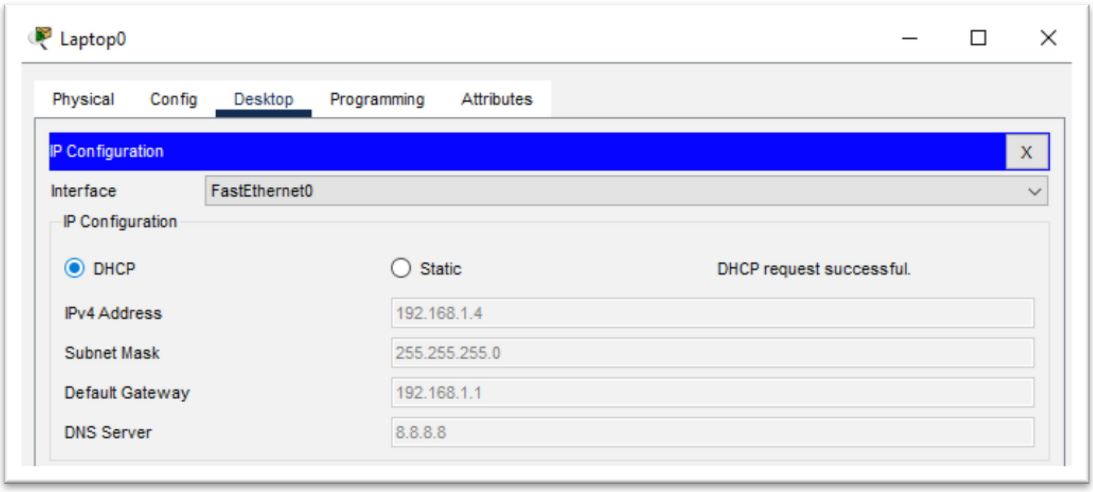
My Output:


Router0
Physical
Config
CLI
Attributes
IOS Command Line Interface
4194304K bytes of physical memory.
3207167K bytes of flash memory at bootflash:.
0K bytes of WebUI ODM Files at webui:.
--- System Configuration Dialog ---
Would you like to enter the initial configuration dialog? [yes/no]: no
Press RETURN to get started!
Router>enable
Router#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#interface gigabitEthernet 0/0/0
Router(config-if)#no shutdown
Router(config-if)#
%LINK-5-CHANGED: Interface GigabitEthernet0/0/0, changed state to up
%LINEPROTO-5-UPDOWN: Line protocol on Interface GigabitEthernet0/0/0, changed state to up
Router(config-if)#ip address 192.168.1.1 255.255.255.0
Router(config-if)#ip dhcp pool abc
Router(dhcp-config)#network 192.168.1.1 255.255.255.0
Router(dhcp-config)#default router 192.168.1.1
^
% Invalid input detected at '^' marker.
Router(dhcp-config)#default-router 192.168.1.1
Router(dhcp-config)#dns-server 8.8.8.8
Router(dhcp-config)#ex
Router(config)#
Copy
Paste
Top

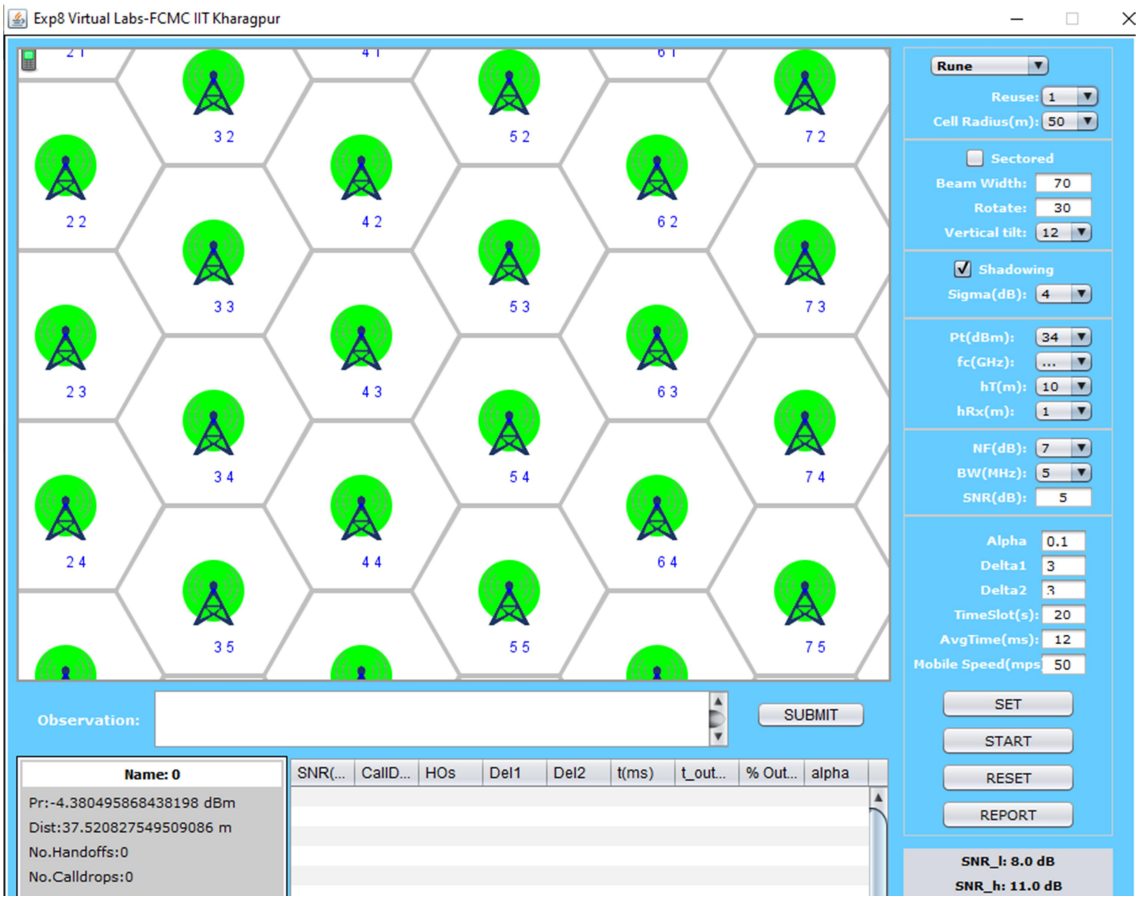

ISR4331
Router0
2960-24TT
Switch0
PC-PT
PC0
PC-PT
PC1
Laptop-PT
Laptop0


PC0
Physical
Config
Desktop
Programming
Attributes
IP Configuration
X
Interface
FastEthernet0
IP Configuration
DHCP
Static
DHCP request successful.
IPv4 Address
192.168.1.2
Subnet Mask
255.255.255.0
Default Gateway
192.168.1.1
DNS Server
8.8.8.8


PC1
Physical
Config
Desktop
Programming
Attributes
IP Configuration
X
Interface
FastEthernet0
IP Configuration
DHCP
Static
DHCP request successful.
IPv4 Address
192.168.1.3
Subnet Mask
255.255.255.0
Default Gateway
192.168.1.1
DNS Server
8.8.8.8


Laptop0
Physical
Config
Desktop
Programming
Attributes
IP Configuration
X
Interface
FastEthernet0
IP Configuration
DHCP
Static
DHCP request successful.
IPv4 Address
192.168.1.4
Subnet Mask
255.255.255.0
Default Gateway
192.168.1.1
DNS Server
8.8.8.8

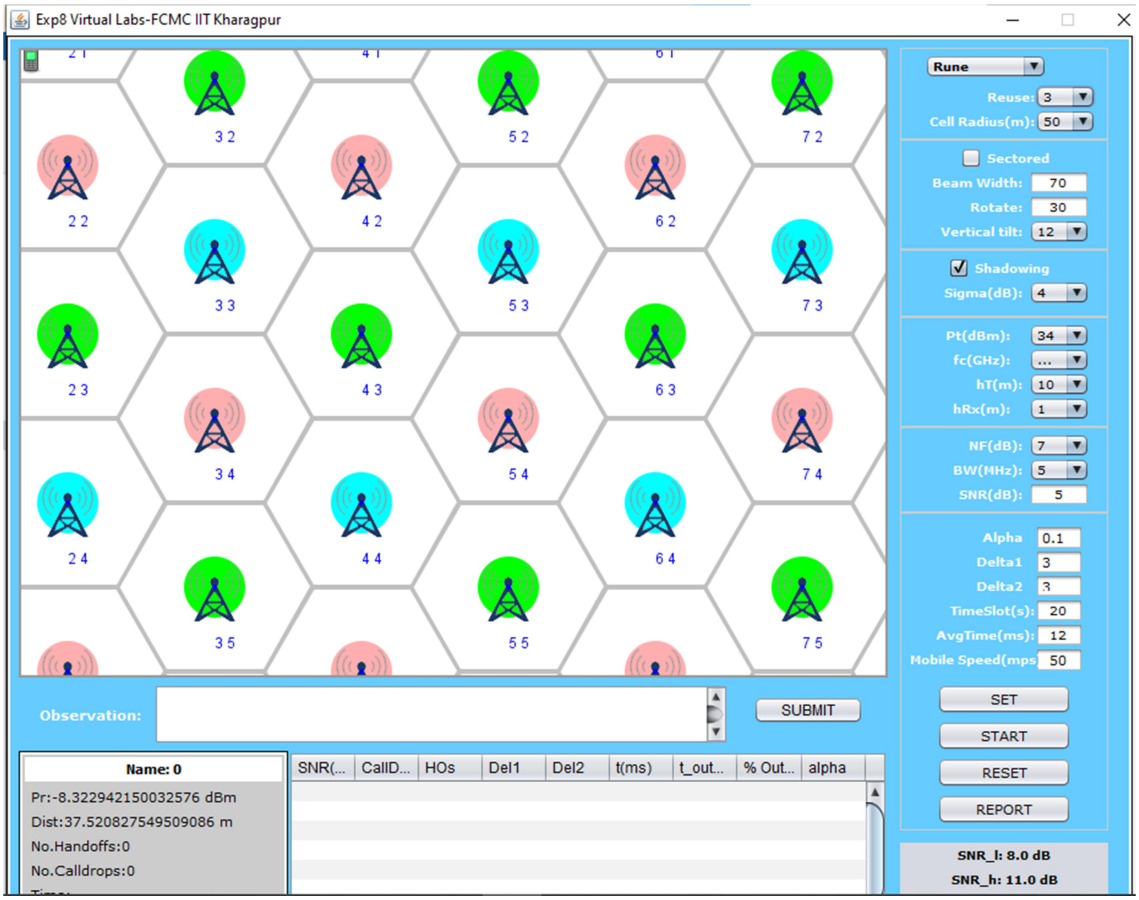
Freq Reuse = 1 & Mobile speed = 50 mps



Input Parameters	
Reuse: 1 ,Model: Rune	Pt(dBm): 34
fc(GHz): 0.8	Beam Width(deg): 70
Rotate(deg): 30	Cell Radius(m): 50
hT(m): 10	hM(m): 1
Sigma(dB): 4	Vertical Tilt(deg): 12
SNR(dB): 5	Band Width(MHz): 5
Noise Figure(dB): 7	Noise Power(dBm): -100.01
Pr0(dBm): -95.01	Time Slot(s): 20

Exp. Results								
SNR	No.Calldrops	No.Handoffs	Delta1	Delta2	Reading Time(ms)	Outage Time(ms)	% Outage	Alpha
5.0	6.0	6.0	3.0	3.0	20016.0	11376.0	56.83	0.1

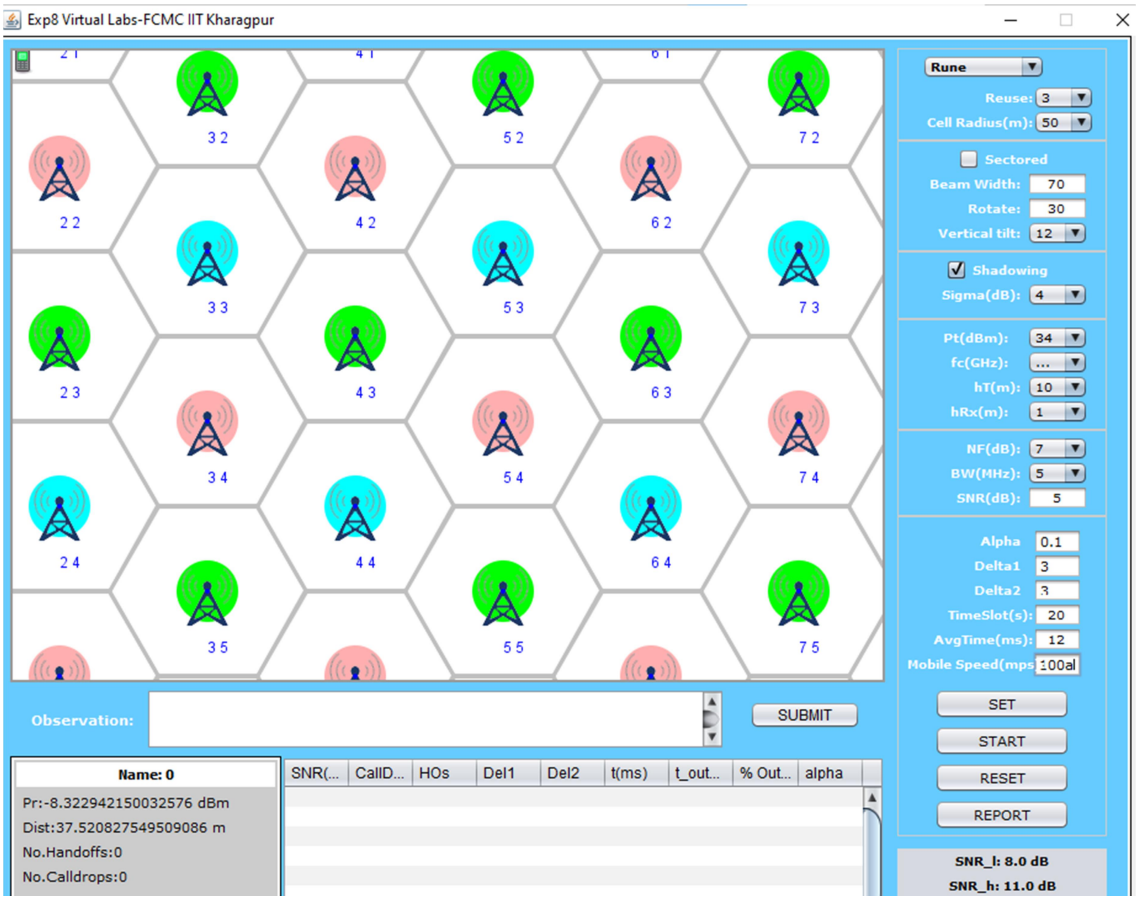
Freq. Reuse = 3 & Mobile Speed = 50 mps



Input Parameters	
Reuse: 3 ,Model: Rune	Pt(dBm): 34
fc(GHz): 0.8	Beam Width(deg): 70
Rotate(deg): 30	Cell Radius(m): 50
hT(m): 10	hM(m): 1
Sigma(dB): 4	Vertical Tilt(deg): 12
SNR(dB): 5	Band Width(MHz): 5
Noise Figure(dB): 7	Noise Power(dBm): -100.01
Pr0(dBm): -95.01	Time Slot(s): 20

Exp. Results								
SNR	No.Calldrops	No.Handoffs	Delta1	Delta2	Reading Time(ms)	Outage Time(ms)	% Outage	Alpha
5.0	10.0	57.0	3.0	3.0	20016.0	756.0	3.78	0.1

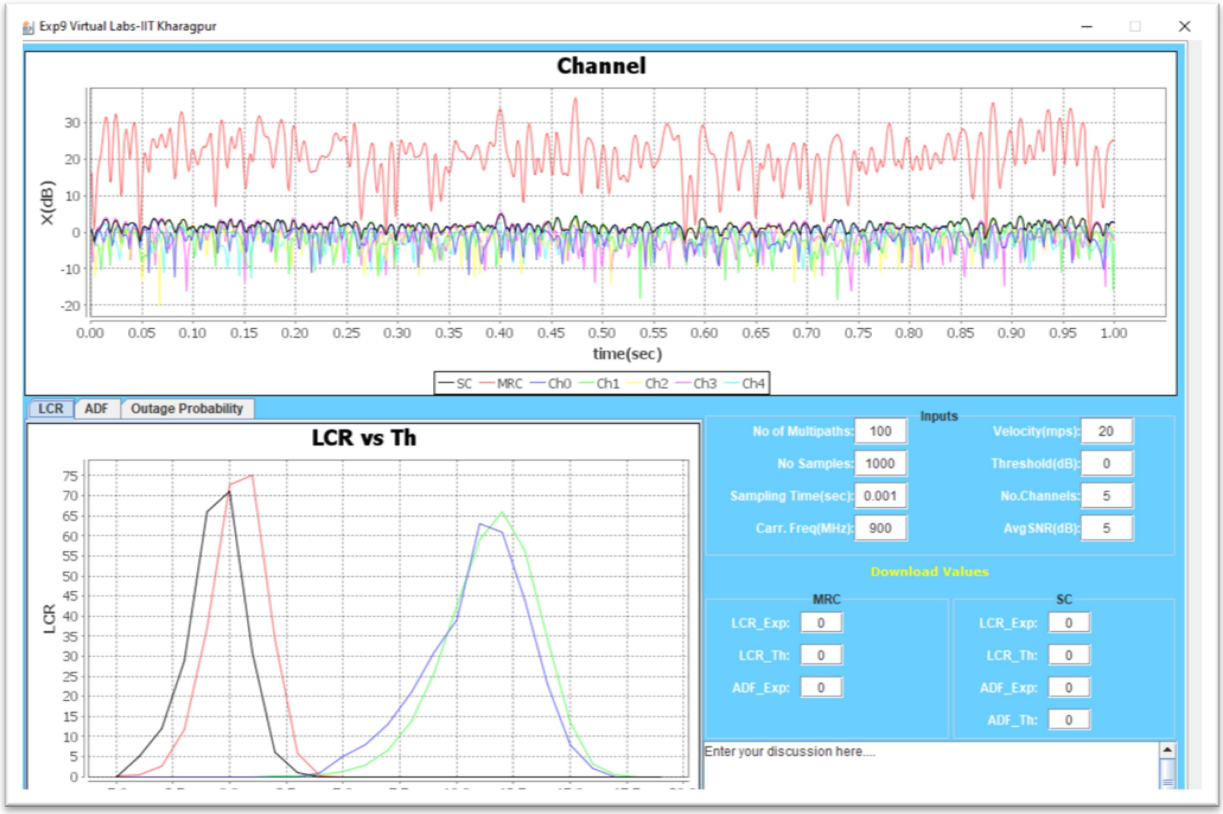
Freq. Reuse = 3 & Mobile Speed = 100 mps

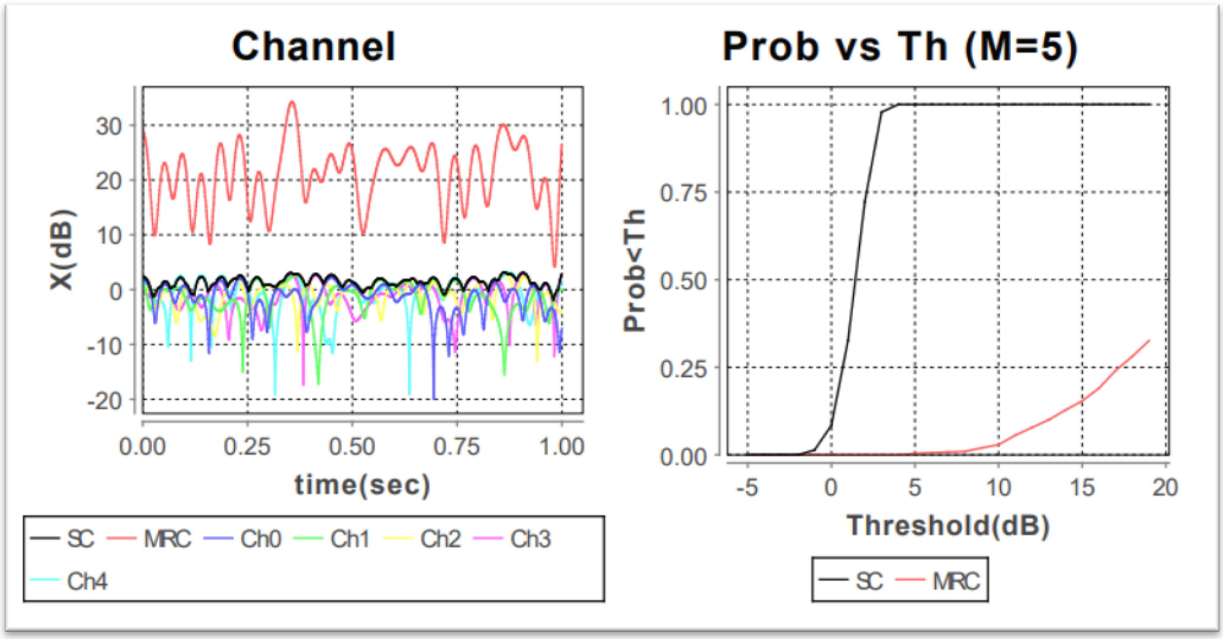


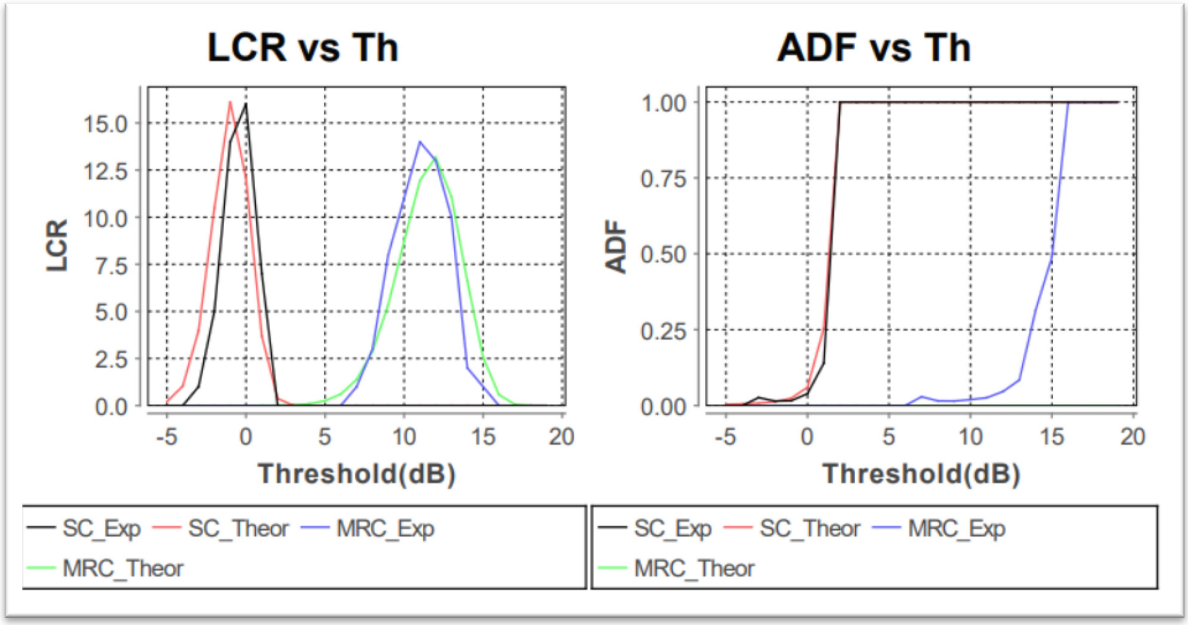
Input Parameters	
Reuse: 3 ,Model: Rune	Pt(dBm): 34
fc(GHz): 0.8	Beam Width(deg): 70
Rotate(deg): 30	Cell Radius(m): 50
hT(m): 10	hM(m): 1
Sigma(dB): 4	Vertical Tilt(deg): 12
SNR(dB): 5	Band Width(MHz): 5
Noise Figure(dB): 7	Noise Power(dBm): -100.01
Pr0(dBm): -95.01	Time Slot(s): 20

Exp. Results								
SNR	No.Calldrops	No.Handoffs	Delta1	Delta2	Reading Time(ms)	Outage Time(ms)	% Outage	Alpha
5.0	0.0	10.0	3.0	3.0	20016.0	720.0	3.6	0.1

My Output:


Exp9 Virtual Labs-IIT Kharagpur
Channel
X(dB)
time(sec)
SC MRC Ch0 Ch1 Ch2 Ch3 Ch4
LCR
ADF
Outage Probability
LCR vs Th
LCR
Inputs
No of Multipaths: 100
Velocity(mps): 20
No Samples: 1000
Threshold(dB): 0
Sampling Time(sec): 0.001
No.Channels: 5
Carr. Freq(MHz): 900
Avg SNR(dB): 5
Download Values
MRC
LCR_Exp: 0
LCR_Th: 0
ADF_Exp: 0
SC
LCR_Exp: 0
LCR_Th: 0
ADF_Exp: 0
ADF_Th: 0
Enter your discussion here....


Channel
X(dB)
time(sec)
SC MRC Ch0 Ch1 Ch2 Ch3
Ch4
Prob vs Th (M=5)
Prob<Th
Threshold(dB)
SC MRC



Input Parameters	
No.of Multipaths	100
No.Samples	1000
Sampling Time(sec)	0.0010
fc(Hz)	9.0E8
Velocity(mps)	4.0
Threshold(dB)	0.0
No.of Branches	5
Avg SNR(dB)	5.0

Results

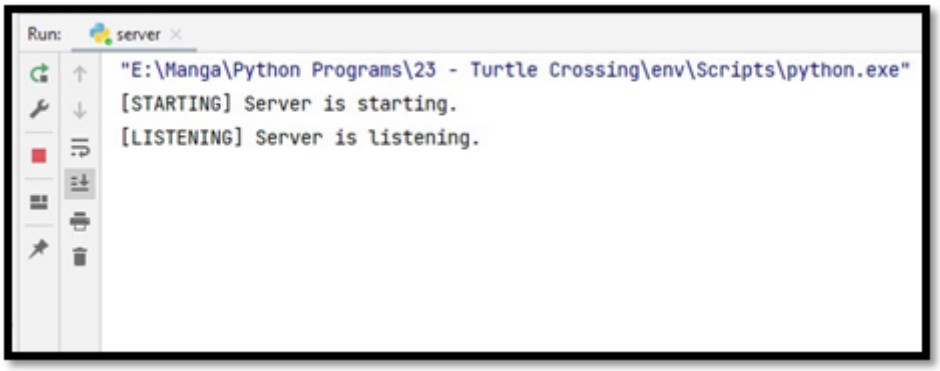
Type	LCR(Exp)	LCR(Th)	ADF(Exp)	ADF(Th)
SC(Actual)	16.0	12.0687	0.039	0.0601
SC(Entered)	10.0	0.0	1.0	0.0
MRC(Actual)	0.0	0.0016	0.0	-
MRC(Entered)	0.0	0.0	0.0	-

Experiment no. 8
Client Python Code:

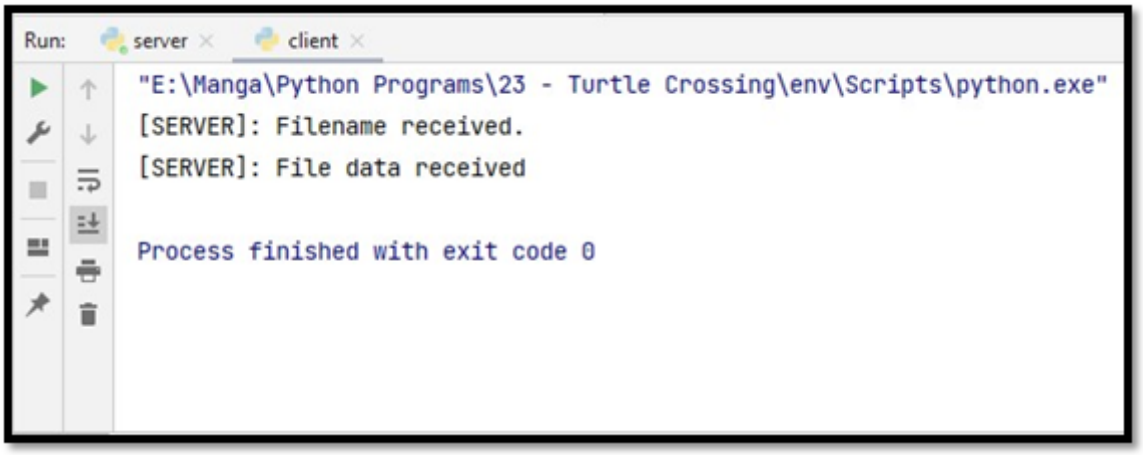
```
import socket

IP = socket.gethostbyname(socket.gethostname())
PORT = 4455
ADDR = (IP, PORT)
FORMAT = "utf-8"
SIZE = 1024

def main():
    """ Staring a TCP socket. """
    client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

    """ Connecting to the server. """
    client.connect(ADDR)

    """ Opening and reading the file data. """
    file = open("files/message.txt", "r")
    data = file.read()

    """ Sending the filename to the server. """
    client.send("message.txt".encode(FORMAT))
    msg = client.recv(SIZE).decode(FORMAT)
    print(f"[SERVER]: {msg}")

    """ Sending the file data to the server. """
    client.send(data.encode(FORMAT))
    msg = client.recv(SIZE).decode(FORMAT)
    print(f"[SERVER]: {msg}")

    """ Closing the file. """
    file.close()

    """ Closing the connection from the server. """
    client.close()

if __name__ == "__main__":
    main()
```

Server Python Code:

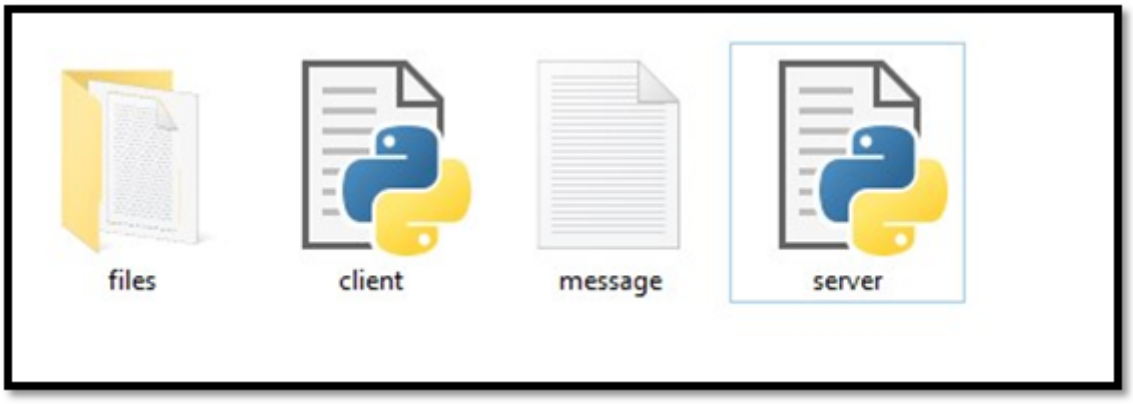
```
import socket

IP = socket.gethostbyname(socket.gethostname())
PORT = 4455
ADDR = (IP, PORT)
SIZE = 1024
FORMAT = "utf-8"

def main():
    print("[STARTING] Server is starting.")
    """ Staring a TCP socket. """
    server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

    """ Bind the IP and PORT to the server. """
    server.bind(ADDR)

    """ Server is listening, i.e., server is now waiting for the client
to connected. """
    server.listen()
    print("[LISTENING] Server is listening.")

    while True:
        """ Server has accepted the connection from the client. """
        conn, addr = server.accept()
        print(f"[NEW CONNECTION] {addr} connected.")

        """ Receiving the filename from the client. """
        filename = conn.recv(SIZE).decode(FORMAT)
        print(f"[RECV] Receiving the filename.")
        file = open(filename, "w")
        conn.send("Filename received.".encode(FORMAT))

        """ Receiving the file data from the client. """
        data = conn.recv(SIZE).decode(FORMAT)
        print(f"[RECV] Receiving the file data.")
        file.write(data)
        conn.send("File data received".encode(FORMAT))

        """ Closing the file. """
        file.close()

        """ Closing the connection from the client. """
        conn.close()
        print(f"[DISCONNECTED] {addr} disconnected.")

if __name__ == "__main__":
    main()
```



Server Listening



Client Code Running



Message received