Om Shete
TE C32
2103163

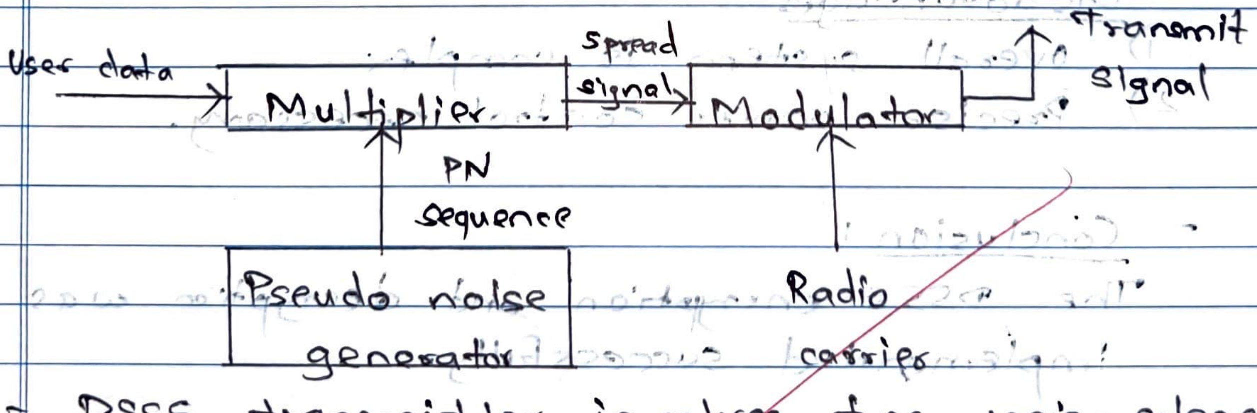
Experiment No : 9

Aim : Write a program to explain the concept of DSSS

Theory :

- In telecommunication, direct-sequence spread spectrum (DSSS) is a spread spectrum modulation technique primarily used to reduce overall signal interference.
- The direct-sequence modulation makes the transmitted signal wider in background than the information bandwidth.
- After the dispreading or removal of the direct sequencing modulation in the receiver, the info bandwidth is restored, while the unitensional and intensional interference is substantially reduced.
- DSSS Transmitter

⇒

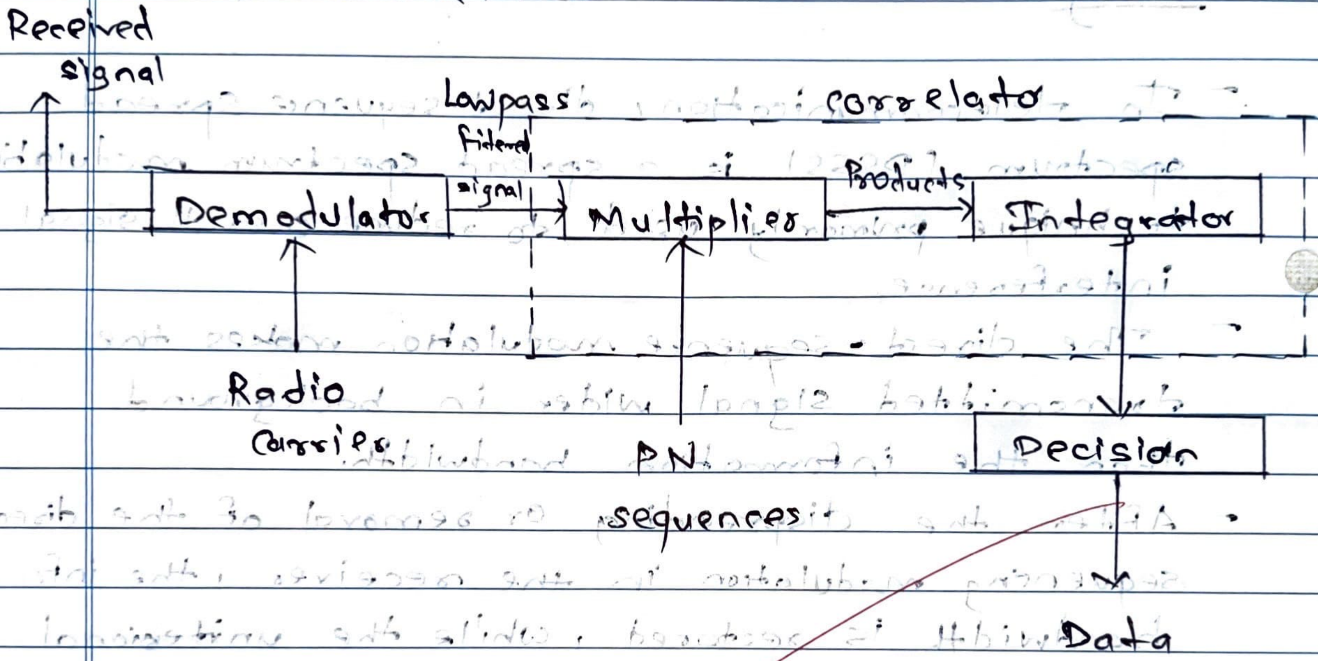


- DSSS transmitter involves two major steps :-
1) Spreading the signal
2) Radio modulation

- DSSS Receiver :

⇒ It also involves three major steps :-

1) Demodulation
2) Correlator
3) Decision making



- Advantages :

1) Resistance to Interception
2) Resistance to Fading

- Disadvantages :

1) Overall system is complex
2) Precise power control necessary.

- Conclusion :

The DSSS encryption and decryption was implemented successfully.

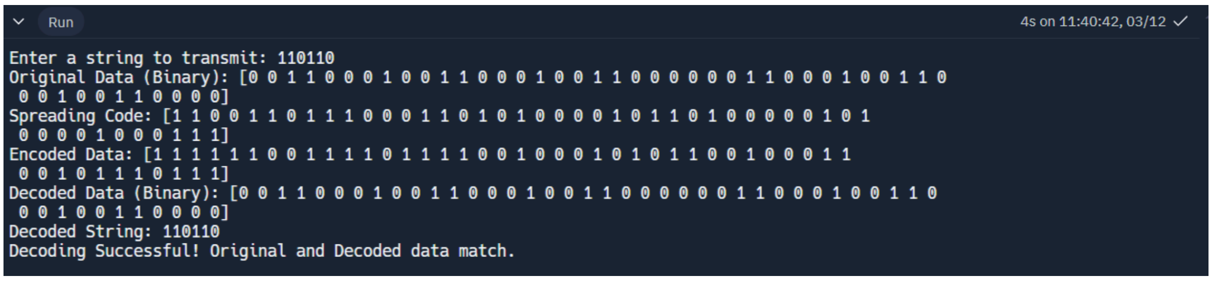
[illegible] A+

Code:

```
import numpy as np


def generate_spreading_code(length):
  spreading_code = np.random.randint(0, 2, size=length)
  return spreading_code


def string_to_binary(input_string):
  binary_data = ''.join(format(ord(char), '08b') for char in
input_string)
  return np.array(list(map(int, binary_data)))


def binary_to_string(binary_data):
  binary_string = ''.join(map(str, binary_data))
  string_data = ''.join(
      chr(int(binary_string[i:i + 8], 2))
      for i in range(0, len(binary_string), 8))
  return string_data


def dsss_encode(data, spreading_code):
  encoded_data = np.bitwise_xor(data, spreading_code)
  return encoded_data


def dsss_decode(encoded_data, spreading_code):
  decoded_data = np.bitwise_xor(encoded_data, spreading_code)
  return decoded_data


def main():
  user_input = input("Enter a string to transmit: ")

  data = string_to_binary(user_input)
  print("Original Data (Binary):", data)

  spreading_code_length = len(data)

  spreading_code = generate_spreading_code(spreading_code_length)
```

```python
  print("Spreading Code:", spreading_code)

  encoded_data = dsss_encode(data, spreading_code)
  print("Encoded Data:", encoded_data)

  decoded_data = dsss_decode(encoded_data, spreading_code)
  print("Decoded Data (Binary):", decoded_data)

  decoded_string = binary_to_string(decoded_data)
  print("Decoded String:", decoded_string)

  if decoded_string == user_input:
    print("Decoding Successful! Original and Decoded data match.")
  else:
    print("Decoding Failed! Original and Decoded data do not match.")


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

Output:



```
Enter a string to transmit: 110110
Original Data (Binary): [0 0 1 1 0 0 0 1 0 0 1 1 0 0 0 1 0 0 1 1 0 0 0 0 0 0 1 1 0 0 0 1 0 0 1 1 0
 0 0 1 0 0 1 1 0 0 0 0]
Spreading Code: [1 1 0 0 1 1 0 1 1 1 0 0 0 1 1 0 1 0 1 0 0 0 0 1 0 1 1 0 1 0 0 0 0 0 1 0 1
 0 0 0 0 1 0 0 0 1 1 1]
Encoded Data: [1 1 1 1 1 1 0 0 1 1 1 1 0 1 1 1 1 0 0 1 0 0 0 1 0 1 0 1 1 0 0 1 0 0 0 1 1
 0 0 1 0 1 1 1 0 1 1 1]
Decoded Data (Binary): [0 0 1 1 0 0 0 1 0 0 1 1 0 0 0 1 0 0 1 1 0 0 0 0 0 0 1 1 0 0 0 1 0 0 1 1 0
 0 0 1 0 0 1 1 0 0 0 0]
Decoded String: 110110
Decoding Successful! Original and Decoded data match.
```