Experiment No. : 01

Experiment Name: To understand the modulation procedure of Amplitude Shift Keying (ASK), Frequency Shift Keying (FSK) and Phase Shift Keying (PSK).

Objectives:

In this experiment, we will –

- Learn about the basics of Amplitude Shift Keying (ASK), Frequency Shift Keying (FSK) and Phase Shift Keying (PSK).
- Simulate ASK, FSK and PSK using Google Collab.
- Generate carrier, input and output signal and verify the signals.
- Investigate and compare the performance of these modulation techniques in transmitting digital data over a communication channel.

Thoery:

Amplitude Shift Keying (ASK): A type of digital modulation technique where the amplitude of the carrier signal is varied in accordance with the digital data (0s and 1s). A high-frequency sinusoidal wave is used as the carrier. The implementation process of ASK is simple and it has low bandwidth requirement. But it is highly susceptible to noise and interference and less reliable in long-distance or wireless communication compared to other modulation types.

Working Process:

- A high-frequency carrier signal is modulated by a digital binary signal (0s and 1s).
- The amplitude of the carrier wave changes according to the binary input-
  Binary '1' ensures carrier is transmitted with full amplitude.
  Binary '0' ensures carrier is transmitted with zero or reduced amplitude.
- The frequency and phase remain constant; only amplitude varies.
- At the receiver end, the signal is demodulated by detecting changes in amplitude to recover the original binary data.

Frequency Shift Keying (FSK): A digital modulation technique where the frequency of the carrier signal is varied according to the binary data. Binary '1' is represented by one frequency, and binary '0' by another frequency. The amplitude and phase of the carrier remain constant. FSK is more resistant to noise than ASK, making it more reliable in wireless communication. It is used in applications like modems, RFID, and radio transmission systems.

Working Principle:

- The digital binary signal (0s and 1s) is used to modulate the frequency of a carrier wave.
- The carrier signal switches between two distinct frequencies:
  Binary '1' ensures transmitted using a high (or specific) frequency.
  Binary '0' ensures transmitted using a low (or different) frequency.
- The amplitude and phase remain unchanged during modulation.
- At the receiver, the signal is demodulated by detecting the frequency variations to retrieve the original binary data.

Phase Shift Keying (PSK): It is a digital modulation technique where the phase of the carrier signal is varied according to the binary data. The amplitude and frequency of the carrier remain constant. PSK is more noise-resistant than ASK and FSK but requires more complex receivers. Common types: BPSK, QPSK (Quadrature PSK), and 8-PSK. Used in Wi-Fi, satellite communication, and RFID systems.

Working Principle:

- The phase of a constant frequency carrier wave is changed to represent digital binary data.
- The phase shift corresponds to the input bits-
  In Binary PSK (BPSK):
  Binary '1' represents carrier with 0° phase
  Binary '0' represents carrier with 180° phase shift
- More advanced versions like QPSK use 4 phase shifts to represent 2 bits per symbol (e.g., 0°, 90°, 180°, 270°).
- The amplitude and frequency stay constant, only the phase changes.
- At the receiver, a phase detector compares the received signal with a reference to recover the original binary data.

Implementation Step:

The simulation of ASK was run using Python and the code is given below-

```
import numpy as np
import matplotlib.pyplot as plt

bitStream = input('Enter the bit stream (e.g., 0101110): ')
b = np.array(list(map(int, bitStream)))
n = len(b)

samples_per_bit = 100
total_samples = n * samples_per_bit

t = np.linspace(0, n, total_samples)
bitWaveform = np.repeat(b, samples_per_bit)

sinWave = np.sin(2 * np.pi * t )
modulatedSignal = bitWaveform * sinWave

plt.subplot(3,1,1)
plt.plot(t, bitWaveform)
plt.grid(True)
plt.axis([0, n, -2, 2])
plt.xlabel('Time')
plt.ylabel('Amplitude')
plt.title('Bit Stream')

plt.subplot(3,1,2)
plt.plot(t, sinWave)
plt.grid(True)
plt.axis([0, n, -2, 2])
plt.xlabel('Time')
plt.ylabel('Amplitude')
plt.title('Sinusoidal Wave')

plt.subplot(3,1,3)
plt.plot(t, modulatedSignal)
plt.grid(True)
plt.axis([0, n, -2, 2])
plt.xlabel('Time')
plt.ylabel('Amplitude')
plt.title('Amplitude Shift Keying (ASK)')

plt.tight_layout()
plt.show()
```

Explanation of the code:

1. First, it takes a binary input from the user as a string in *bitstream.*
2. Converts the string to a list of integers and stores the number of bits in *n.*
3. Takes 100 samples for each bit for smooth plotting and generate e time vector from 0 to n seconds with *total_samples* points.
4. *np.repeat()* Expands each bit value to occupy 100 samples so it appears as a flat level.
5. *np.sin()* generates a continuous sine wave at 1Hz over the time vector and ASK is performed using *modulatedSignal = bitWaveform * sinWave.*
6. Finally, all the signals are plotted.

The simulation of FSK was run using Python and the code is given below-

```
import numpy as np
import matplotlib.pyplot as pltD

bitStream = input('Enter the bit stream (e.g., 0101110): ')
b = np.array(list(map(int, bitStream)))
n = len(b)

t= np.arange(0, n, 0.01)
bitWaveform = np.zeros(len(t))
bp = np.zeros(n)

for i in range(n):
  if b[i] == 1:
    bp[i] = 1
  else:
    bp[i] = -1
  bitWaveform[int(i*100):int((i+1)*100)] = bp[i]

  f0 = 2 * (2*np.pi*t)
  f = 1 * (2*np.pi*t)
  sinHighfreq = np.sin(f0+f)
  sinLowfreq = np.sin(f0-f)
  sin = np.sin(f0+(bitWaveform*f))

  fig, axes = plt.subplots(4, 1, figsize=(10, 8))

  axes[0].plot(t, bitWaveform)
  axes[0].grid(True)
  axes[0].set_title('Bit Waveform')
  axes[0].axis([0, n, -2, 2])
```

```
axes[1].plot(t, sinHighfreq)
axes[1].grid(True)
axes[1].set_title('Sinu wave with High Frequency')
axes[1].axis([0, n, -2, 2])

axes[2].plot(t, sinLowfreq)
axes[2].grid(True)
axes[2].set_title('Sinu wave with Low Frequency')
axes[2].axis([0, n, -2, 2])

axes[3].plot(t, sin)
axes[3].grid(True)
axes[3].set_title('FSK modulated signal')
axes[3].axis([0, n, -2, 2])

plt.tight_layout()
plt.show()
```

Explanation of the code:

1. *Input()* takes a binary string from the user and converts it to a NumPy array of integers and *n* is the number of bits.
2. *np.arange()* generates a time vector from 0 to n in steps of 0.01 and *bitWaveform* is initialized to store each bit over time and *bp* stores ±1 based on bit value (used for later modulation).
3. Then the bits are converted from 1 to +1 and 0 to -1 and filled the *bitWaveform* with 100 samples of ±1 per bit.
4. Sin waves of two different frequencies are generated and the FSK modulated signal is also generated using *sin = np.sin(f0 + (bitWaveform * f))* .
5. Finally, the signals are plotted in the graph.

Simulation of BPSK was run using Python and the code is given below-

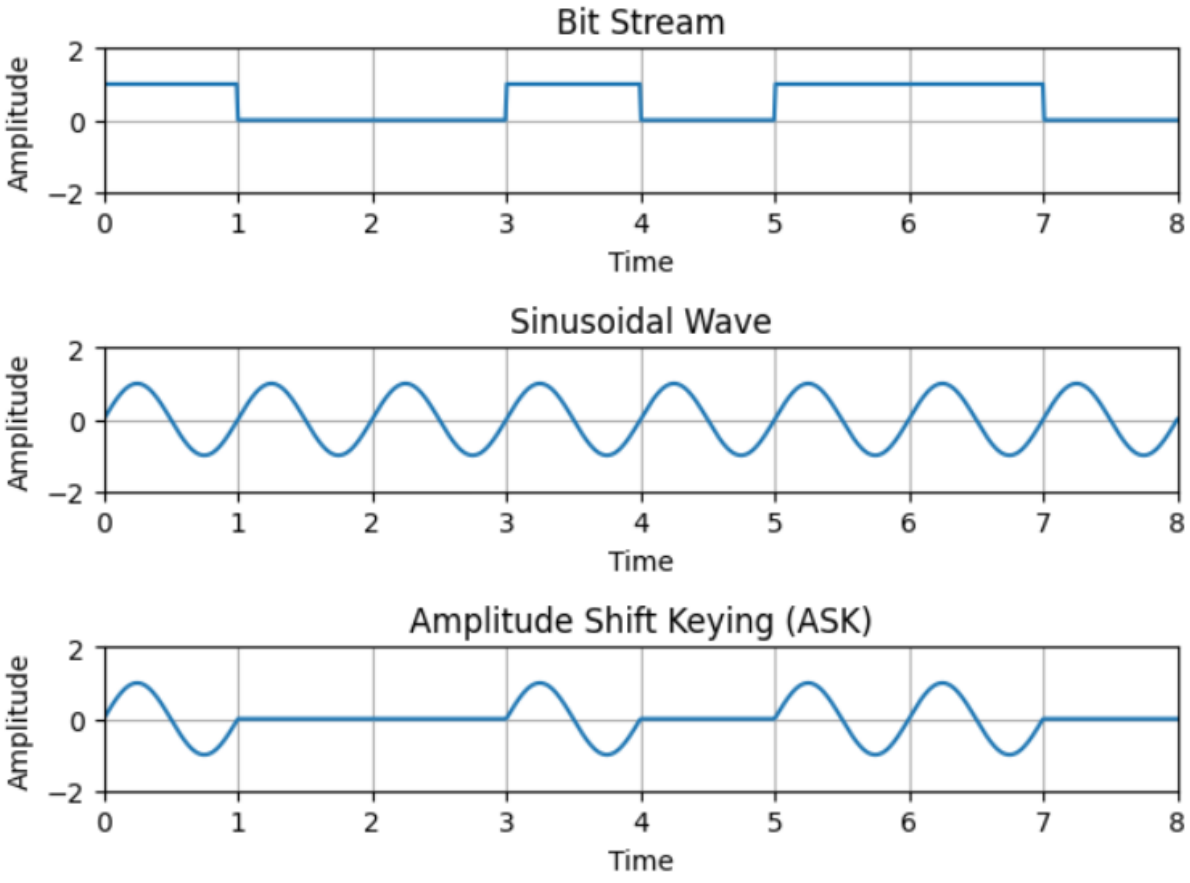
```
import numpy as np
import matplotlib . pyplot as plt

bitStream = input('Enter the bit stream (e.g., 0101110): ')
b = np.array(list(map(int, bitStream)))
n = len(b)

t = np.arange(0, n, 0.01)
bitWaveform = np.zeros(len(t))

for i in range(n):
  if b[i] == 0:
    bp = -1
  else:
    bp = 1
  bitWaveform[i*100:(i+1)*100] = bp

sin = np.sin(2*np.pi*t)
modulatedSignal = bitWaveform * sin

plt.subplot(3, 1, 1)
plt.plot(t, bitWaveform)
plt.grid(True)
plt.axis([0, n, -2, 2])
plt.title('Bit Stream')
plt.xlabel('Time')
plt.subplot(3, 1, 2)
plt.plot(t, sin)
plt.grid(True)
plt.axis([0, n, -2, 2])
plt.title('Sine Wave')
plt.xlabel('Time')

plt.subplot(3, 1, 3)
plt.plot(t, modulatedSignal)
plt.grid(True)
plt.axis([0, n, -2, 2])
plt.title('Phase Shift Keying (PSK)')
plt.xlabel('Time')

plt.tight_layout()
plt.show()
```

Explanation of the code:

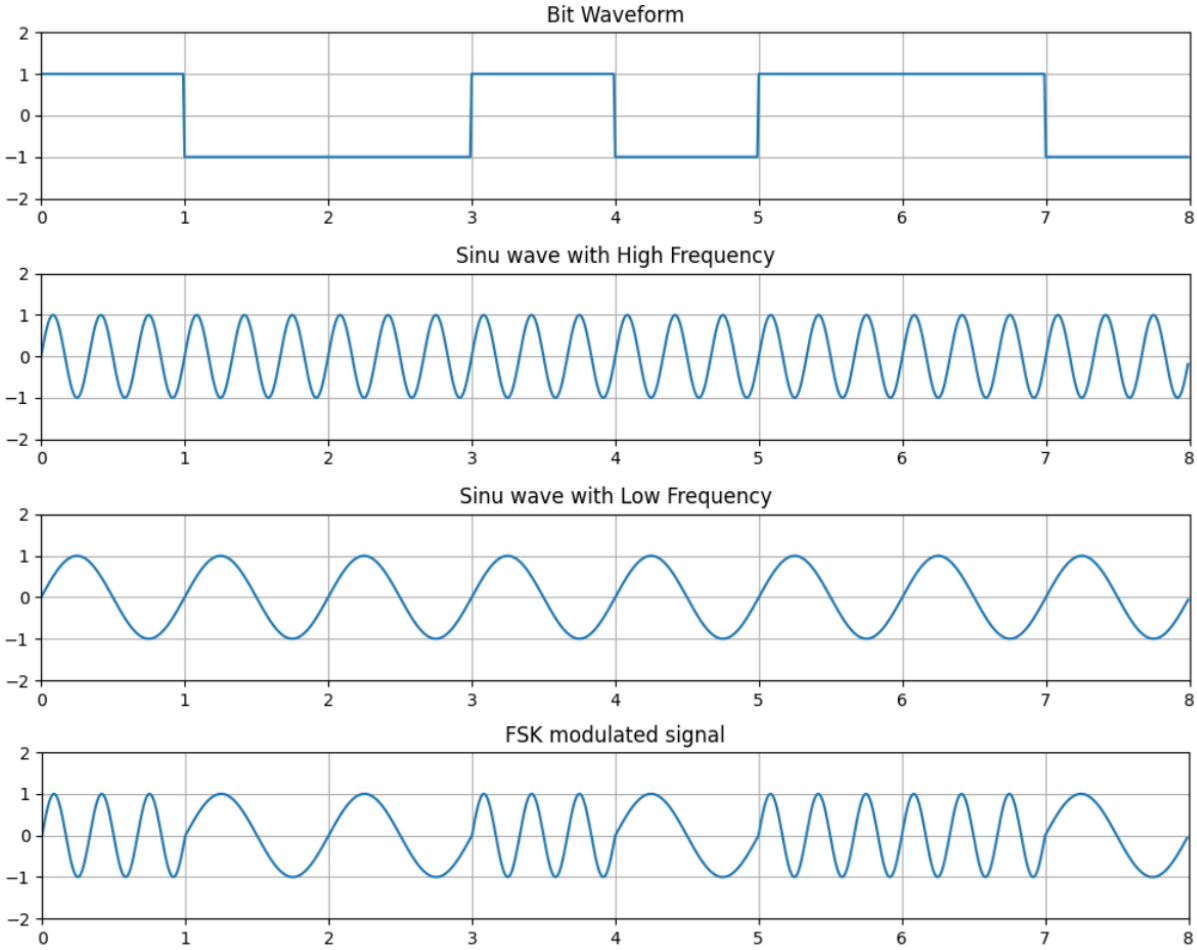
1. *Input()* takes a binary string from the user and converts it to a NumPy array of integers and *n* is the number of bits.
2. *np.arange()* generates a time vector from 0 to n in steps of 0.01 and *bitWaveform* is initialized to store each bit over time.
3. Created bit waveform for modulation, +1 if the bit is 1 and -1 if the bit is 0 and 100 samples per bit are assigned to maintain time resolution.
4. *np.sin()* is used to generate carrier wave and used in modulation and BPSK modulated signal is generated using *modulatedSignal = bitWaveform * sin*
5. Finally, all the figures are plotted for all signals and bit stream also.

Output:

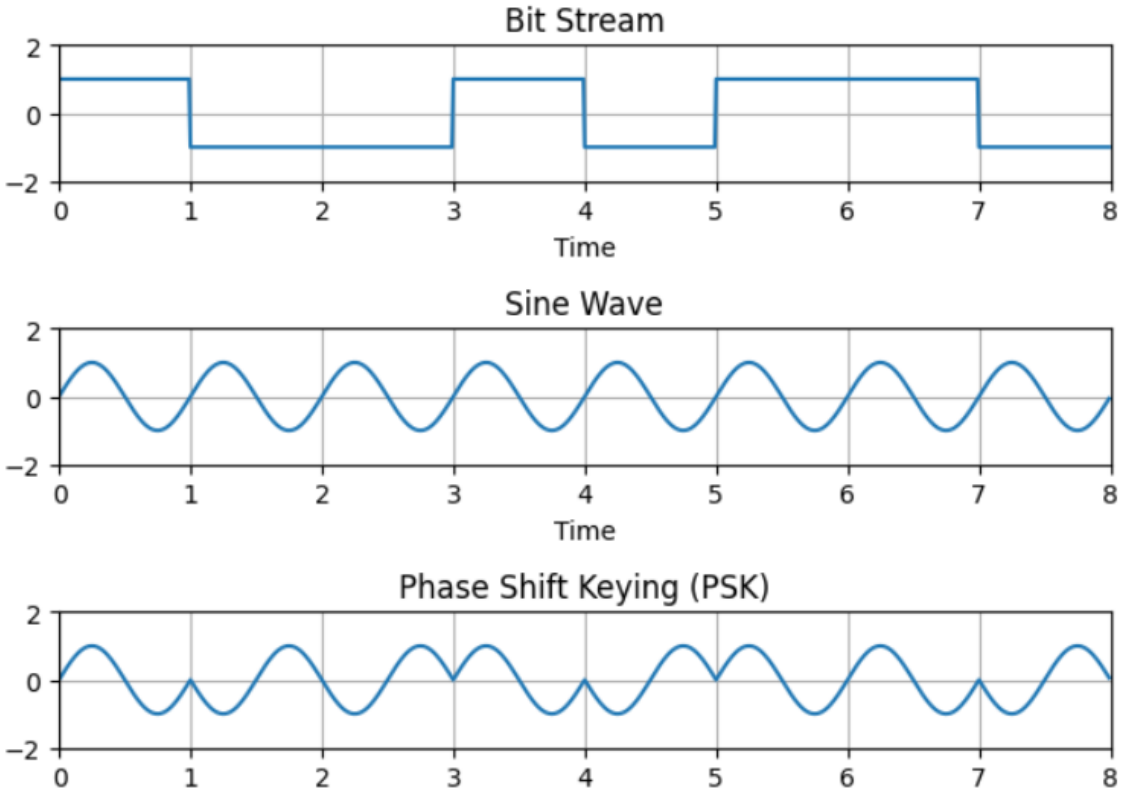
The output waveform of ASK modulation for bit stream: 10010110



The output waveform of FSK modulation for bit stream: 10010110



The output waveform of PSK(BPSK) modulation for bit stream: 10010110



Discussion:

In this experiment, we have-

1. Learned the basics of ASK, FSK and PSK modulation.
2. Implemented those modulations using Python and saw the output of different types of modulation.
3. Seen how to generate bit stream by taking input a string from the user and using that bit stream implement different types of modulation.
4. Known how to generate carrier wave and modulated signal from that carrier wave.