Name : Md. Noman Bhuiyan
Roll no : 220608

PPG (Photoplethysmogram) signal is a non-invasive optical technique used to measure blood volume changes in the microvascular tissue. It is commonly used for heart rate (HR) and oxygen saturation (SpO_2) monitoring.

Pure ppg signal:

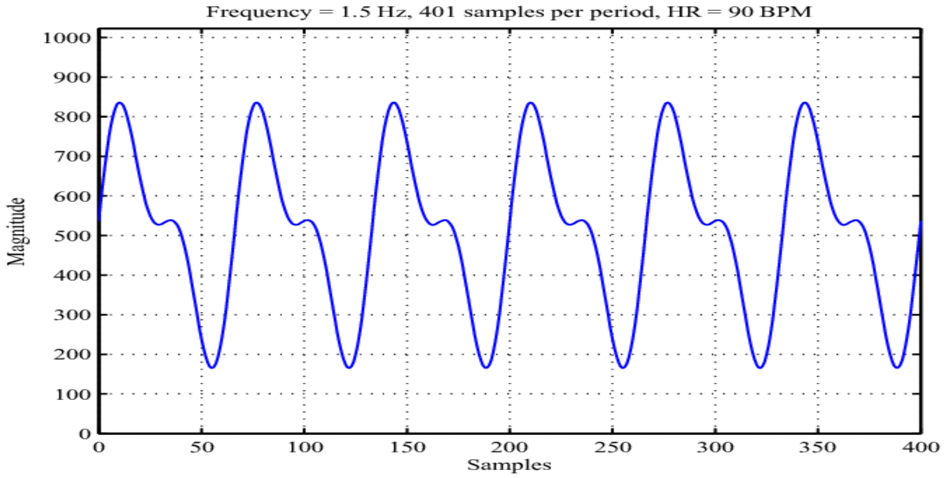


Figure 01: pure ppg signal

Components of a PPG Signal:

A typical PPG waveform consists of:

1. Systolic Peak – The highest point, corresponding to blood ejection from the heart.
2. Dicrotic Notch – A small dip, representing the closing of the aortic valve.
3. Diastolic Phase – The downward slope, indicating blood returning to baseline.

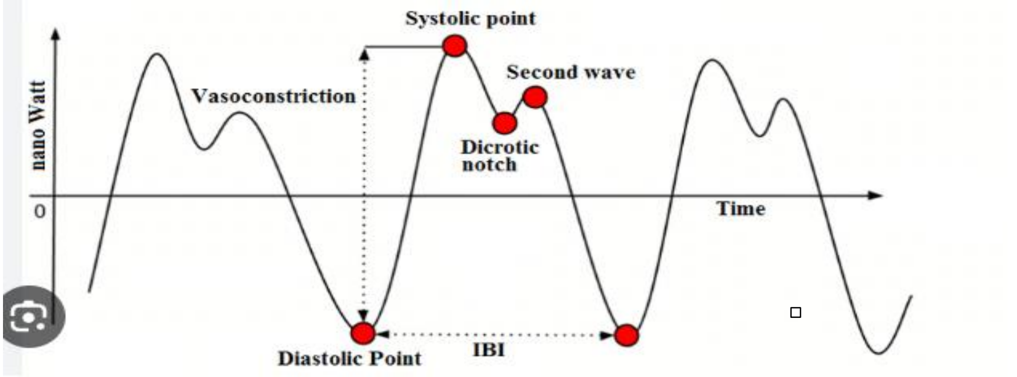


Figure 02: component of a ppg signal

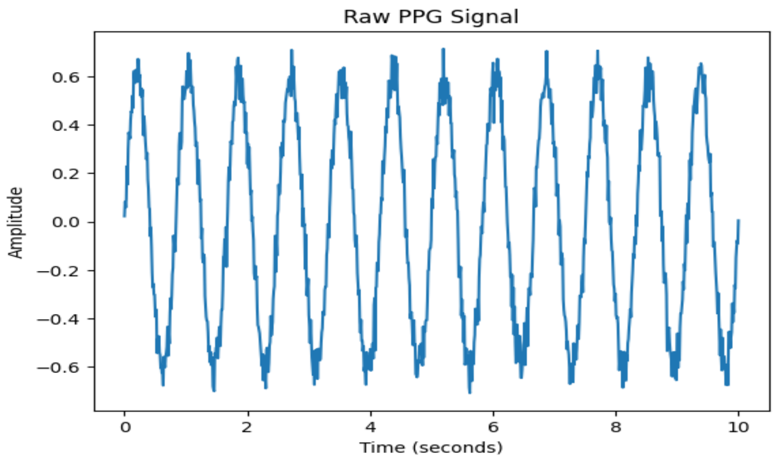
Pocedure for find the feature of ppg signal:

- Generate a PPG Signal: A sinusoidal wave with 1.2 Hz frequency is created with added Gaussian noise to simulate real-world PPG signals.
- Plot the Raw PPG Signal: The generated signal is visualized to observe its waveform.
- Generate and Plot Noise Only: Pure random noise is generated separately to compare with the PPG signal.
- Generate and Plot the Original Signal (Without Noise): A clean sine wave (without noise) is plotted for reference.
- Apply a Bandpass Filter: A Butterworth bandpass filter (0.5–5 Hz) removes noise and unwanted frequency components.
- Normalize the Signal: The filtered signal is scaled between 0 and 1 to make peak detection easier.
- Detect Heartbeats (Peak Detection): Peaks are detected using find_peaks() function, ensuring peaks are spaced properly (avoiding false positives).
- Calculate Heart Rate (BPM): The inter-beat interval (IBI) is calculated from peak differences, and heart rate is computed using 60 / IBI.
- Plot the PPG Signal with Detected Peaks: The detected heartbeats are marked on the normalized PPG signal.
- Display the Heart Rate: The average heart rate (BPM) is printed as the final output.

Code implimentation of ppg feature:

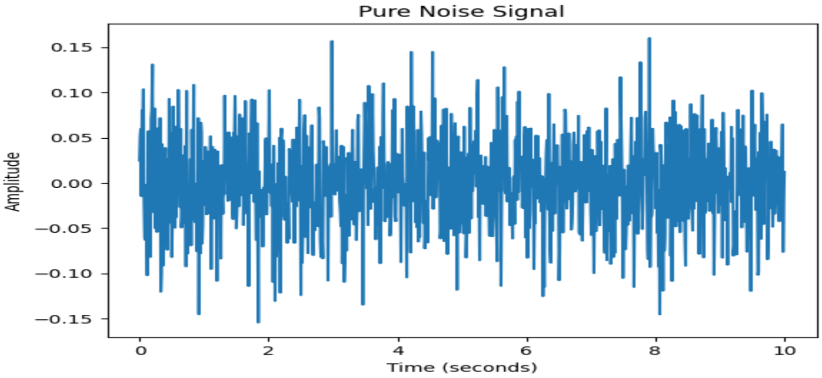
1.Plot a Raw ppg signal:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.signal import butter, filtfilt, find_peaks

# Sampling rate
fs = 100  # 100 Hz sampling frequency

t = np.linspace(0, 10, fs * 10)

# Generate PPG signal: Sine wave (1.2 Hz) + Gaussian noise
ppg_signal = 0.6 * np.sin(2 * np.pi * 1.2 * t) + np.random.normal(0, 0.05, len(t))

# Plot the Raw PPG Signal
plt.figure()
plt.plot(t, ppg_signal)
plt.title("Raw PPG Signal")
plt.xlabel("Time (seconds)")
plt.ylabel("Amplitude")
```



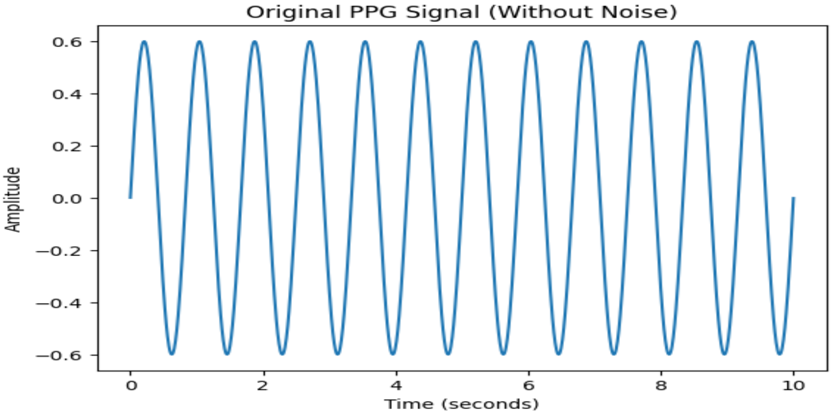
2. Generate and plot noise only:

```
noise_signal = np.random.normal(0, 0.05, len(t))
plt.figure()
plt.plot(t, noise_signal)
plt.title("Pure Noise Signal")
plt.xlabel("Time (seconds)")
plt.ylabel("Amplitude")
plt.show()
```



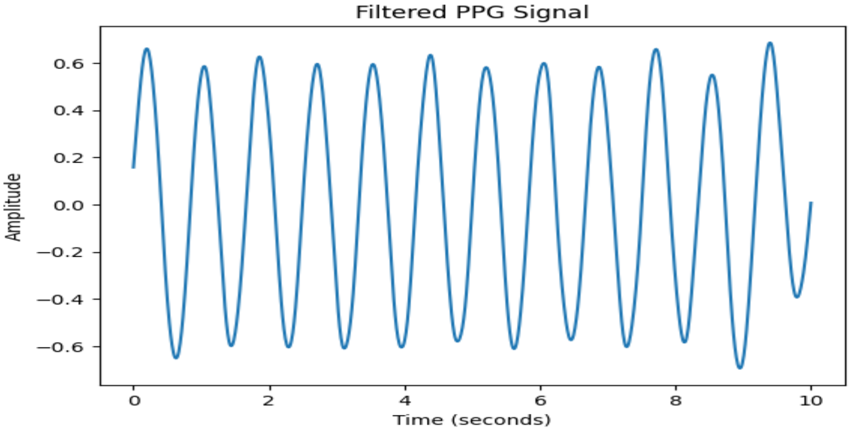
3. Generate and Plot Original Signal (Without Noise)

```
original_signal = 0.6 * np.sin(2 * np.pi * 1.2 * t)
plt.figure()
plt.plot(t, original_signal)
plt.title("Original PPG Signal (Without Noise)")
plt.xlabel("Time (seconds)")
plt.ylabel("Amplitude")
plt.show()
```



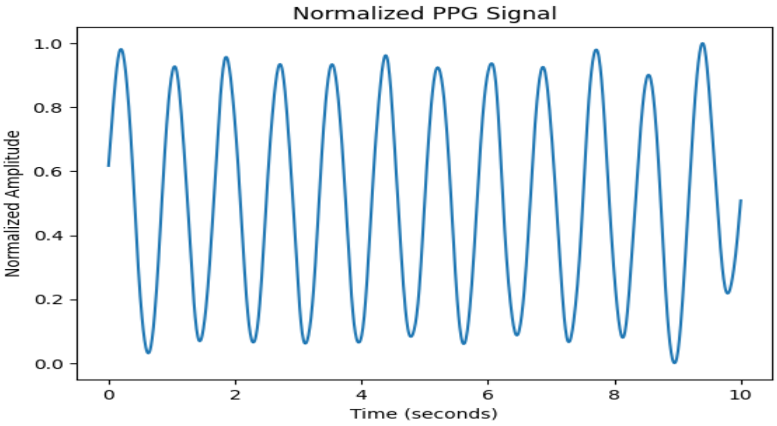
4. Apply Bandpass Filter

```
def bandpass_filter(signal, lowcut, highcut, fs, order=4):
  nyquist = 0.5 * fs
  low = lowcut / nyquist
  high = highcut / nyquist
  b, a = butter(order, [low, high], btype='band')
  return filtfilt(b, a, signal)

filtered_ppg = bandpass_filter(ppg_signal, 0.5, 5, fs)
plt.figure()
plt.plot(t, filtered_ppg)
plt.title("Filtered PPG Signal")
plt.xlabel("Time (seconds)")
plt.ylabel("Amplitude")
plt.show()
```



5. Normalize the Signal

```
normalized_ppg = (filtered_ppg - np.min(filtered_ppg)) / (np.max(filtered_ppg) -
np.min(filtered_ppg))
plt.figure()
plt.plot(t, normalized_ppg)
plt.title("Normalized PPG Signal")
plt.xlabel("Time (seconds)")
plt.ylabel("Normalized Amplitude")
plt.show()
```

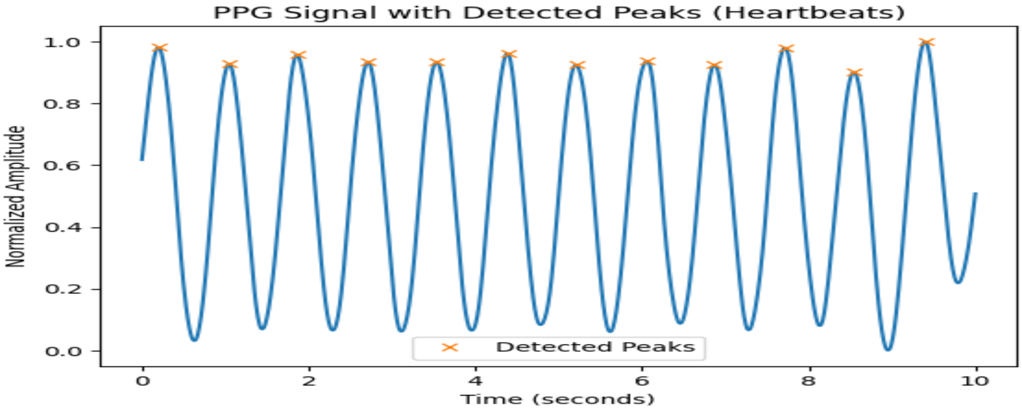


6. Detect Heartbeats (Peak Detection):

```
peaks, _ = find_peaks(normalized_ppg, distance=fs*0.6)
```

7. Calculate Heart Rate (BPM) and Plot the PPG Signal with Detected Peaks:

```
ibi = np.diff(peaks) / fs  # Inter-beat intervals in seconds
heart_rate = 60 / ibi  # Convert to BPM

plt.figure()
plt.plot(t, normalized_ppg)
plt.plot(t[peaks], normalized_ppg[peaks], "x", label="Detected Peaks")
plt.title("PPG Signal with Detected Peaks (Heartbeats)")
plt.xlabel("Time (seconds)")
plt.ylabel("Normalized Amplitude")
plt.legend()
plt.show()
```



8. Display the Heart Rate:

```
print("Estimated Heart Rate:", np.mean(heart_rate), "BPM")
```

output: Estimated Heart Rate: **71.84599594262268** BPM