Experiment No: 02

Experiment Name: Detecting Heartbeats from PPG Signal Peaks

Objective:

1. Identify peaks in the PPG signal corresponding to individual heartbeats to calculate the heart rate.

2. Analyze the number, amplitude, and intervals of these peaks to assess the quality of the PPG signal.

Theory:

Photoplethysmography (PPG) is an optical technique used to measure blood volume changes in peripheral circulation. To identify heartbeats or peak values from a PPG signal, the following steps are typically followed:

1. Obtain the Raw PPG Signal

2. Introduce Noise

3. Apply Filtering

4. Normalize the Signal

5. Detect Peaks

Steps Explained:

1. Adding Noise to the PPG Signal

In real-world scenarios, PPG signals are often affected by noise such as white noise, motion artifacts, and baseline wander. To simulate this, noise can be added mathematically as:

Noisy Signal = Raw Signal + Noise

Noise can be generated using functions like numpy.random.normal to simulate white noise.

2. Filtering the Signal Using a Butterworth Filter

To recover the original signal, noise is removed using a filter. A Butterworth filter is designed to remove unwanted components while preserving the signal of interest. This is implemented using scipy.signal.butter to design the filter and scipy.signal.filtfilt for zero-phase distortion filtering. The filter is applied with specified low and high cutoff frequencies.

3. Normalizing the Signal

Normalization scales the signal's amplitude to a uniform range for easier and more consistent analysis. The normalization formula is:

$$\textbf{Normalized Signal} = \frac{\textbf{Signal} - \textbf{min}(\textbf{Signal})}{\textbf{max}(\textbf{Signal}) - \textbf{min}(\textbf{Signal})}$$

4. Detecting Peaks in the PPG Signal

Peaks in the PPG signal correspond to heartbeats and typically occur at regular intervals. The strength of these peaks reflects the blood flow's intensity. Peaks can be identified using the scipy.signal.find_peaks function, which locates the positions of these heartbeats.

This process ensures accurate heart rate calculation and quality assessment of the PPG signal.

Source Code:

1. **Raw ppg signal**:

```
import numpy as np
import matplotlib.pyplot as plt
fs=100
t=np.linespace(0,10,fs*10)
ppg_signal = 0.6*np.sin(2*np.pi*1,2*t) + np.random.normal(0,0.05,len(t))

plt.plot(t,ppg_signal)
plt.title("raw ppg signal")
plt.xlabel("time (seconds)")
plt.ylabel("amplitude")
plt.show()
```

2. **Original signal**:

```
import numpy as np
import  matplotlib.pyplot as plt
fs = 100
t=np.linespace(0,10,fs*10)
original_signal = 0.6*np.sin(2*np.pi*1,2*t)

plt.plot(t,original _signal)
plt.title("Original")
plt.xlabel("time (seconds)")
plt.ylabel("amplitude")
plt.show()
```

3. **Noise add**:

```
import numpy as np
import  matplotlib.pyplot as plt
fs = 100
t=np.linespace(0,10,fs*10)
ppg_signal = np.random.normal(0,0.05,len(t))

plt.plot(t,noise_signal)
plt.title("add noise")
plt.xlabel("time (seconds)")
plt.ylabel("amplitude")
plt.show()
```

4. **Filter signal:**

```
from scipy.signal import butter, filtfilt
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_singal,0.5,5,fs)
plt.plot(t,filtered_ppg)
plt.title("filtered ppg signal")
plt.xlabel("time (seconds)")
plt.ylabel("amplitude")
plt.show()
```

5. **Normalized signal**:

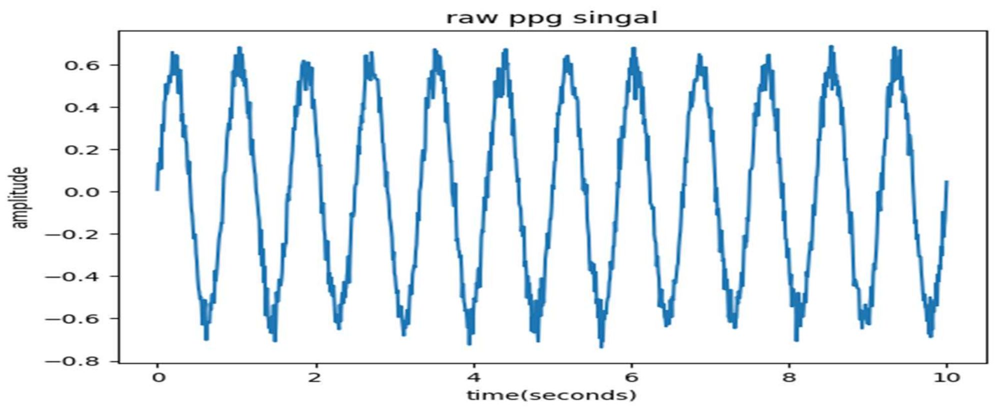
```
normalized_ppg = (filtered_ppg –
np.min(filtered_ppg))/(np.max(filtered_ppg) – np.min(filtered_ppg))

plt.plot(t,normalized_ppg)
plt.title(“normalized ppg signal”)
plt.xlabel(“time(seconds)”)
plt.ylabel(“normalized amplitude”)
plt.show()
```

6. **Peak find**:

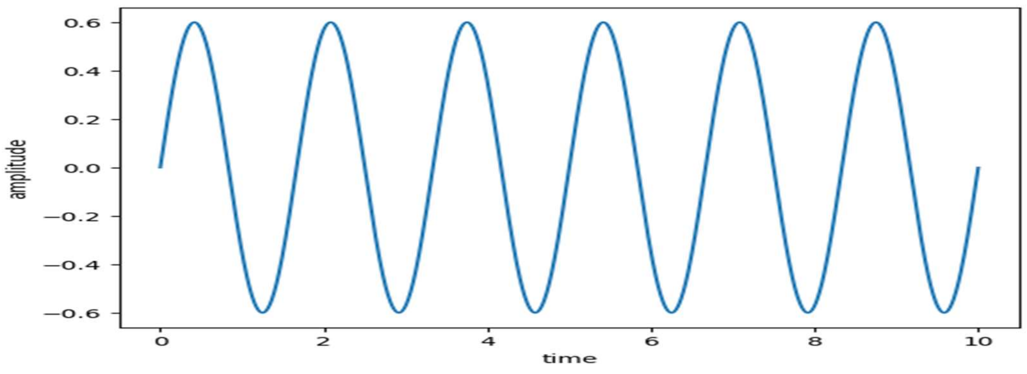
```
from scipy.signal import find_peaks
peaks,_=find_peaks(normalized_ppg,distance = fs*0.6)
ibi = np.diff(peaks)/fs
heart_rate = 60 / ibi

plt.plot(t,normalize_ppg)
plt.plot(t,[peaks],normalized_ppg[peaks],”x”)
plt.title(“ppg signal with detected peaks (heartbeats)”)
plt.xlabel(“time(seconds)”)
plt.ylabel(“normalized amplitude”)
plt.show()
print(“heart rate:”, np.mean(heart_rate), “BPM”)
```

output:

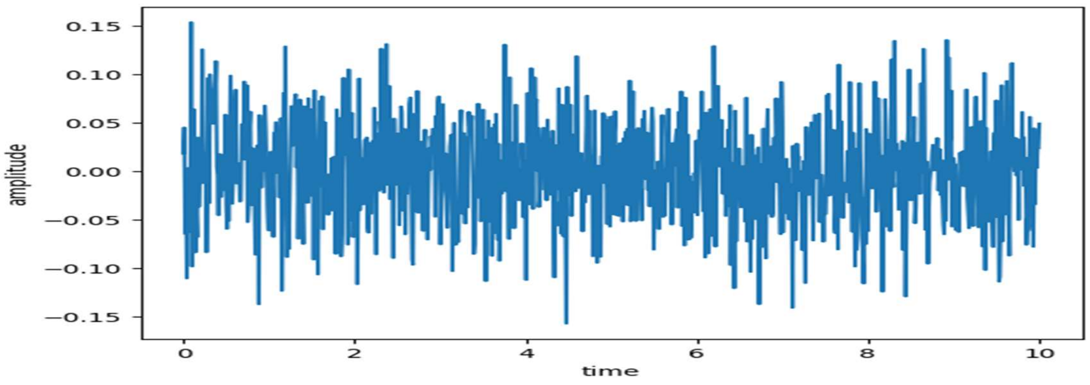
1.Raw signal:



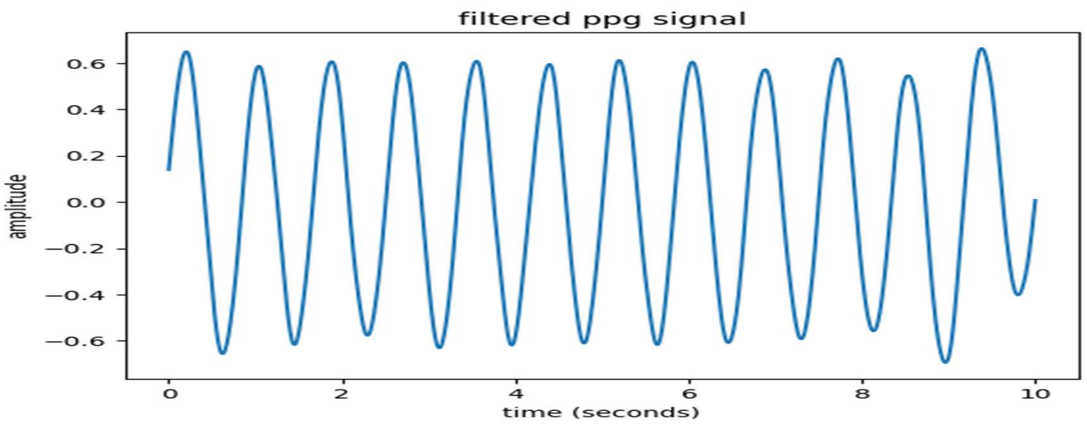
2.Original signal:



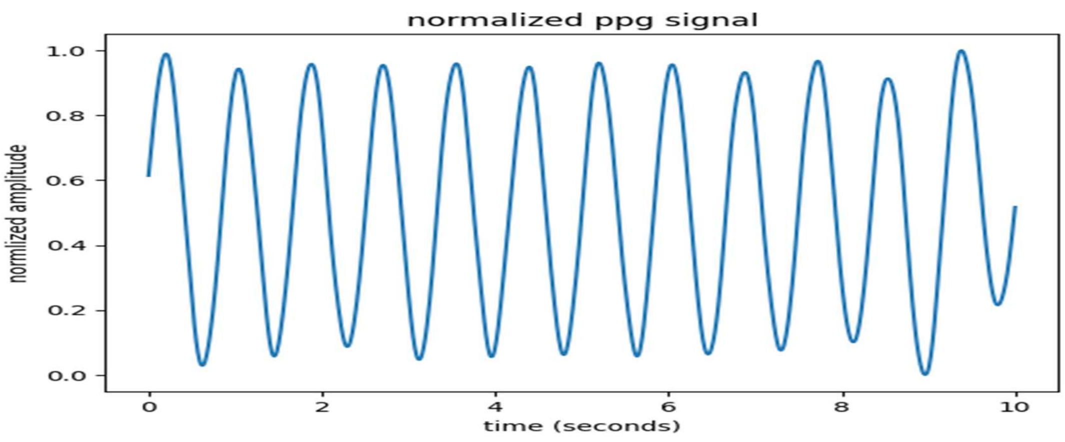
3.<u>**Noise signal**</u>**:**



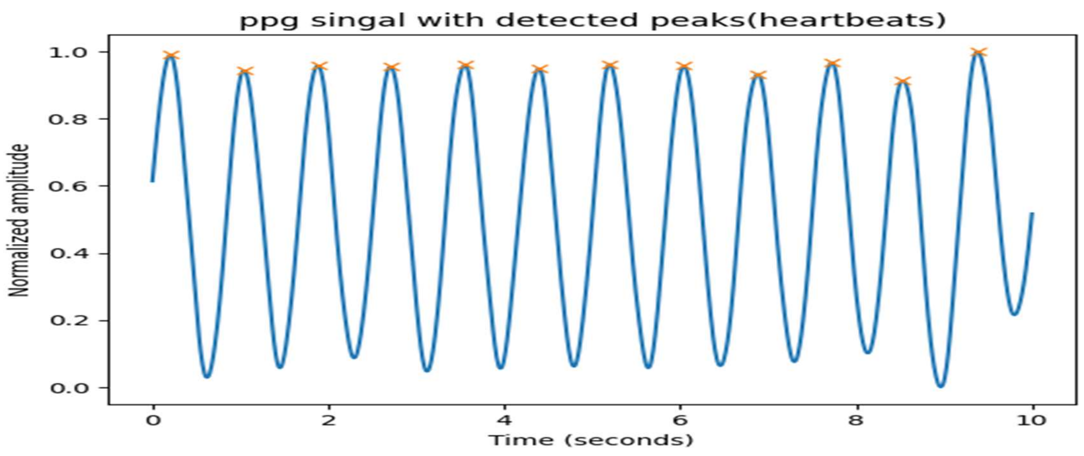
4.<u>**filter signal**</u>:



5.<u>**Normalized PPG signal**</u>:



6.**PPG signal peaks detect(Heartbeats):**


ppg singal with detected peaks(heartbeats)
Normalized amplitude
Time (seconds)