Name : Sohel Ahamed **Roll No** : 220634

A **PPG (Photoplethysmography) signal** is an optical technique used to measure blood volume changes in the microvascular tissue. It is commonly used for heart rate monitoring and oxygen saturation (SpO_2) measurement.

Working Principle of PPG Signal:

1. **Light Emission & Absorption:** An LED emits light onto the skin, which is either absorbed or reflected by the blood vessels.
2. **Detection:** A photodetector measures the amount of reflected or transmitted light, which varies with blood flow.
3. **Signal Processing:** The detected variations are converted into a PPG waveform, which represents the pulsatile changes in blood volume with each heartbeat.

Good ppg signal:

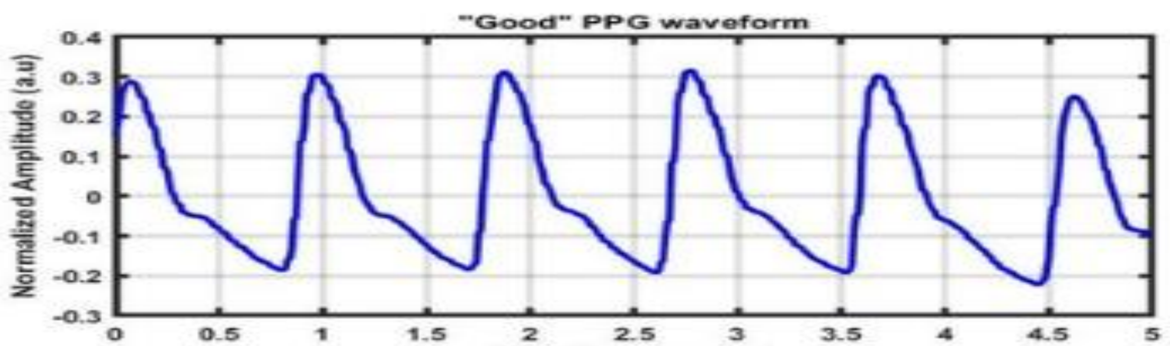


Fig-01:*Good Ppg signal*

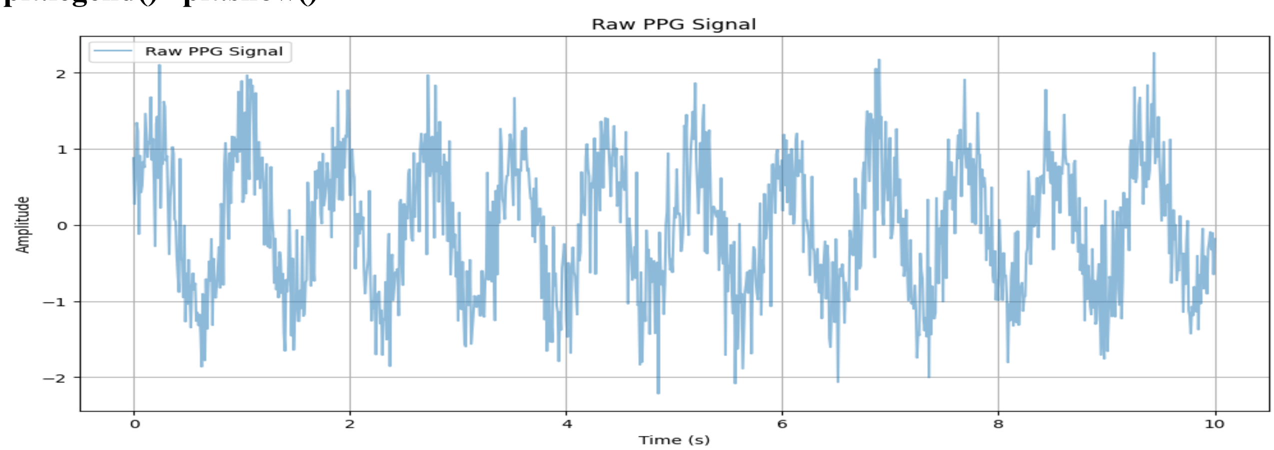
Features of PPG (Photoplethysmography) Signal

1.Raw ppg signal: A raw PPG (Photoplethysmogram) signal represents the variation in light absorption due to changes in blood volume in the microvascular bed of tissue. It's typically measured using a photodetector, like in smartwatches or pulse oximeters, and provides information about heart rate, blood oxygen levels, and other cardiovascular parameters.The raw PPG signal consists of **AC Component** and **DC Component**.

Raw ppg signal CODE

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

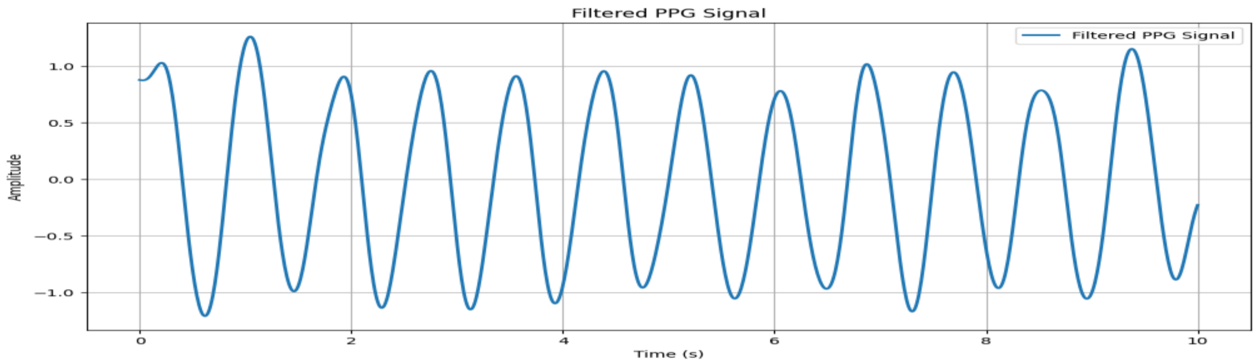
```
# Generate sample PPG signal
np.random.seed(0)
time = np.linspace(0, 10, 1000)
ppg_signal = np.sin(2 * np.pi * 1.2 * time) + 0.5 * np.random.normal(size=len(time))

# Define the lowpass filter function
def butter_lowpass_filter(data, cutoff, fs, order=5):
    nyquist = 0.5 * fs
    normal_cutoff = cutoff / nyquist
    b, a = butter(order, normal_cutoff, btype='low', analog=False)
    y = filtfilt(b, a, data)
    return y
# Plot 1: Raw PPG Signal
plt.figure(figsize=(12, 6))
plt.plot(time, ppg_signal, label="Raw PPG Signal", alpha=0.5)
plt.title("Raw PPG Signal")
plt.xlabel("Time (s)")
plt.ylabel("Amplitude")
plt.grid()
plt.legend()   plt.show()
```



2.Filtered PPG signal:

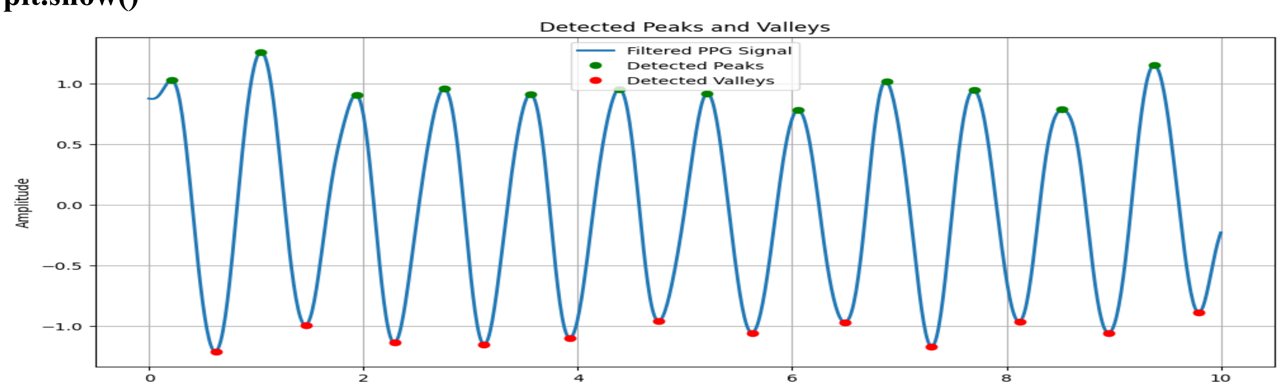
```
# Filter settings
fs = 100  # Sampling frequency in Hz
cutoff = 3  # Cutoff frequency in Hz
filtered_ppg = butter_lowpass_filter(ppg_signal, cutoff, fs)
# Plot 2: Filtered PPG Signal
plt.figure(figsize=(12, 6))
plt.plot(time, filtered_ppg, label="Filtered PPG Signal", linewidth=2)
```

```
plt.title("Filtered PPG Signal")
plt.xlabel("Time (s)")
plt.ylabel("Amplitude")
plt.grid()
plt.legend()
plt.show()
```



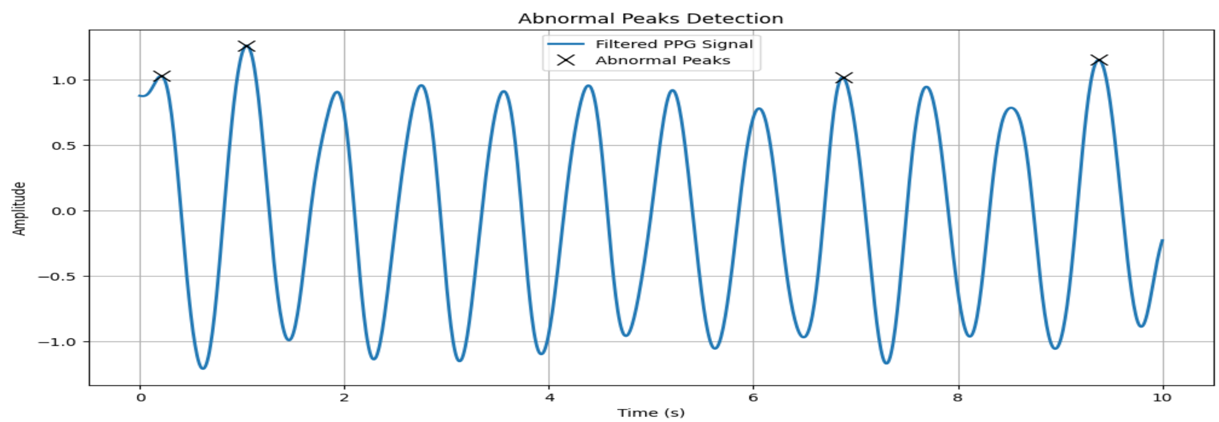
3.Peaks And Value:

```
# Peak and valley detection
peaks, _ = find_peaks(filtered_ppg, height=0.5, distance=fs//2)
valleys, _ = find_peaks(-filtered_ppg, height=0.5, distance=fs//2)
# Plot 3: Peaks and Valleys
plt.figure(figsize=(12, 6))
plt.plot(time, filtered_ppg, label="Filtered PPG Signal", linewidth=2)
plt.plot(time[peaks], filtered_ppg[peaks], "go", label="Detected Peaks")
plt.plot(time[valleys], filtered_ppg[valleys], "ro", label="Detected Valleys")
plt.title("Detected Peaks and Valleys")
plt.xlabel("Time (s)")
plt.ylabel("Amplitude")
plt.grid()
plt.legend()
plt.show()
```



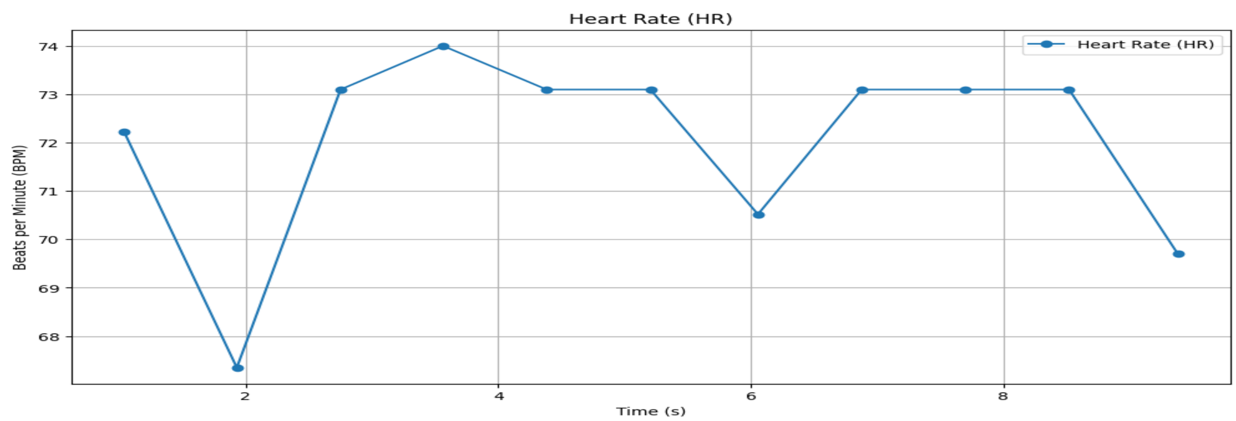
4. Abnormal Peaks:

```
# Abnormal peaks detection
peak_heights = filtered_ppg[peaks]
abnormal_peaks = peaks[peak_heights > 1]  # Threshold for high spikes
# Plot 4: Abnormal Peaks
plt.figure(figsize=(12, 6))
plt.plot(time, filtered_ppg, label="Filtered PPG Signal", linewidth=2)
plt.plot(time[abnormal_peaks], filtered_ppg[abnormal_peaks], "kx", label="Abnormal
Peaks", markersize=10)
plt.title("Abnormal Peaks Detection")
plt.xlabel("Time (s)")
plt.ylabel("Amplitude")
plt.grid()
plt.legend()
plt.show()
```



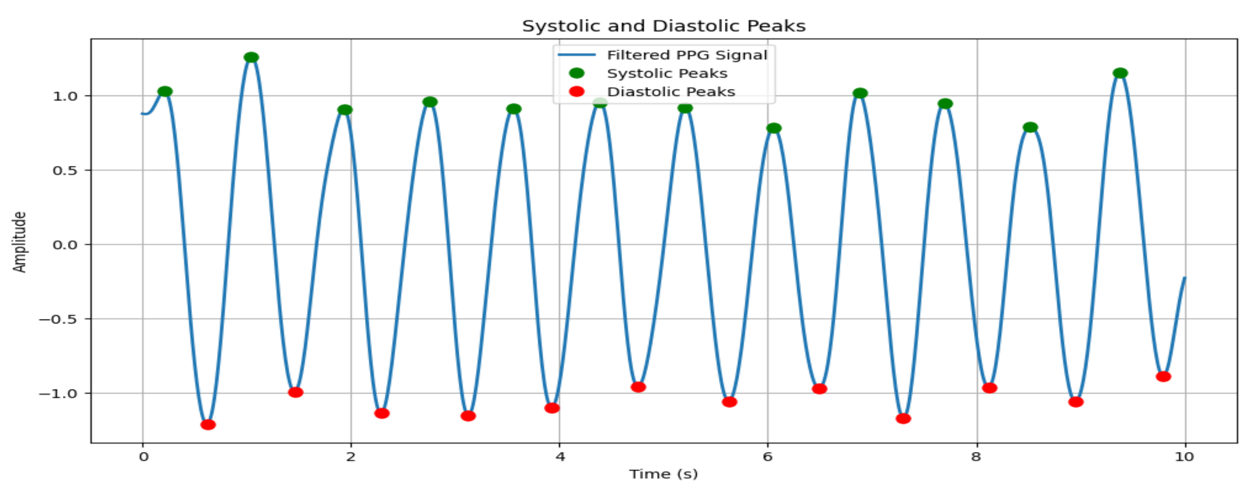
5. Heart Rate (HR)

```
# Heart Rate (HR) calculation
peak_times = time[peaks]
hr = 60 / np.diff(peak_times)  # Beats per minute
# Plot 5: Heart Rate (HR)
plt.figure(figsize=(12, 6))
plt.plot(peak_times[1:], hr, label="Heart Rate (HR)", marker='o')
plt.title("Heart Rate (HR)")
plt.xlabel("Time (s)")
plt.ylabel("Beats per Minute (BPM)")
plt.grid()
plt.legend()
plot.show()
```



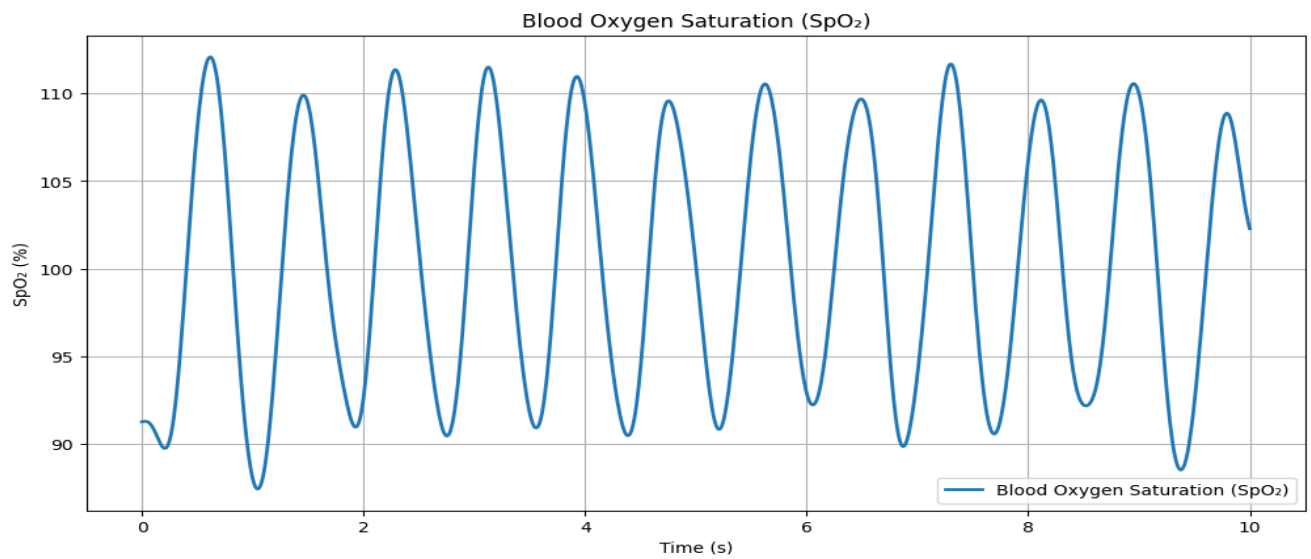
6. Systolic & Diastolic Peaks

```
# Systolic & Diastolic Peaks
systolic_peaks = peaks
diastolic_peaks = valleys
# Plot 6: Systolic & Diastolic Peaks
plt.figure(figsize=(12, 6))
plt.plot(time, filtered_ppg, label="Filtered PPG Signal", linewidth=2)
plt.plot(time[systolic_peaks], filtered_ppg[systolic_peaks], "go", label="Systolic Peaks",
markersize=8)
plt.plot(time[diastolic_peaks], filtered_ppg[diastolic_peaks], "ro", label="Diastolic Peaks",
markersize=8)
plt.title("Systolic and Diastolic Peaks")
plt.xlabel("Time (s)")
plt.ylabel("Amplitude")
plt.grid()
plt.legend()
plt.show()
```



7 Blood Oxygen Saturation (SpO₂).

```
# Blood Oxygen Saturation (SpO₂) simulation
# Assuming SpO₂ is inversely proportional to the amplitude of the PPG signal
spo2 = 100 - (filtered_ppg * 10)  # Simulated SpO₂ values

# Plot 7: Blood Oxygen Saturation (SpO₂)
plt.figure(figsize=(12, 6))
plt.plot(time, spo2, label="Blood Oxygen Saturation (SpO₂)", linewidth=2)
plt.title("Blood Oxygen Saturation (SpO₂)")
plt.xlabel("Time (s)")
plt.ylabel("SpO₂ (%)")
plt.grid()
plt.legend()
plt.show()
```



8. Pulse Transit Time (PTT)

```
# Pulse Transit Time (PTT) simulation
# Assuming PTT is the time difference between systolic and diastolic peaks
ptt = np.zeros(len(systolic_peaks))
for i in range(len(systolic_peaks)):
    if i < len(diastolic_peaks):
        ptt[i] = time[diastolic_peaks[i]] - time[systolic_peaks[i]]

# Plot 8: Pulse Transit Time (PTT)
plt.figure(figsize=(12, 6))
plt.plot(time[systolic_peaks[:len(ptt)]], ptt, label="Pulse Transit Time (PTT)", marker='o')
plt.title("Pulse Transit Time (PTT)")
```

```
plt.xlabel("Time (s)")
plt.ylabel("PTT (s)")
plt.grid()
plt.legend()
plt.show()
```