Roll: 220625

Name: MD Sazzad Hossain Shakkhor

PPG Signal Processing and Abnormality Detection

1. Introduction

Photoplethysmography (PPG) is a non-invasive optical technique used to measure blood volume changes in tissues. It is widely used for heart rate monitoring and detecting cardiovascular abnormalities. This assignment involves generating a synthetic PPG signal and processing it using Python with the help of libraries such as NumPy, Matplotlib, SciPy, and NeuroKit2.

2. Project Objectives

The objective of this assignment is to understand the process of acquiring and analyzing a photoplethysmography (PPG) signal using signal processing techniques. This includes generating a synthetic PPG signal, filtering noise, detecting peaks, and identifying abnormalities based on heart rate.

3. Required Libraries

The project uses the following Python libraries:

- import numpy as np
- import matplotlib.pyplot as plt
- import scipy.signal as signal
- import neurokit2 as nk

4. Methodology

The methodology for this study consists of multiple steps, including signal generation, preprocessing, peak detection, feature extraction, and abnormality analysis. The following outlines the approach:

1. Generating Synthetic PPG Signal:

A synthetic PPG signal is generated using the `ppg_simulate` function from the NeuroKit2 library. This function simulates a realistic PPG waveform with predefined physiological characteristics.

2. Signal Preprocessing:

Preprocessing is performed to enhance signal quality and remove noise. This step includes:

- **Bandpass Filtering:** A Butterworth bandpass filter is applied to retain frequencies between

0.5 Hz and 5 Hz, which are relevant for heart rate detection.

- **Normalization:** The filtered signal is normalized to scale values between 0 and 1, making it easier to process and analyze.

3. Peak Detection:

The `find_peaks` function from the SciPy library is used to detect pulse peaks. Parameters such as minimum peak height and distance are set based on physiological considerations to ensure accurate detection of heartbeats.

4. Feature Extraction:

Key features extracted from the PPG signal include:

- **Heart Rate (BPM):** Calculated based on the number of detected peaks over the duration of the signal.
- **Pulse Amplitude:** The difference between the maximum and minimum values at detected peak locations.
- **RR Intervals:** The time differences between successive peaks, which help assess heart rate variability.

5. Abnormality Detection:

A simple threshold-based abnormality detection mechanism is implemented based on heart rate. The heart rate is classified as abnormal if it falls below 60 BPM or exceeds 100 BPM, which is outside the normal resting heart rate range for most individuals.

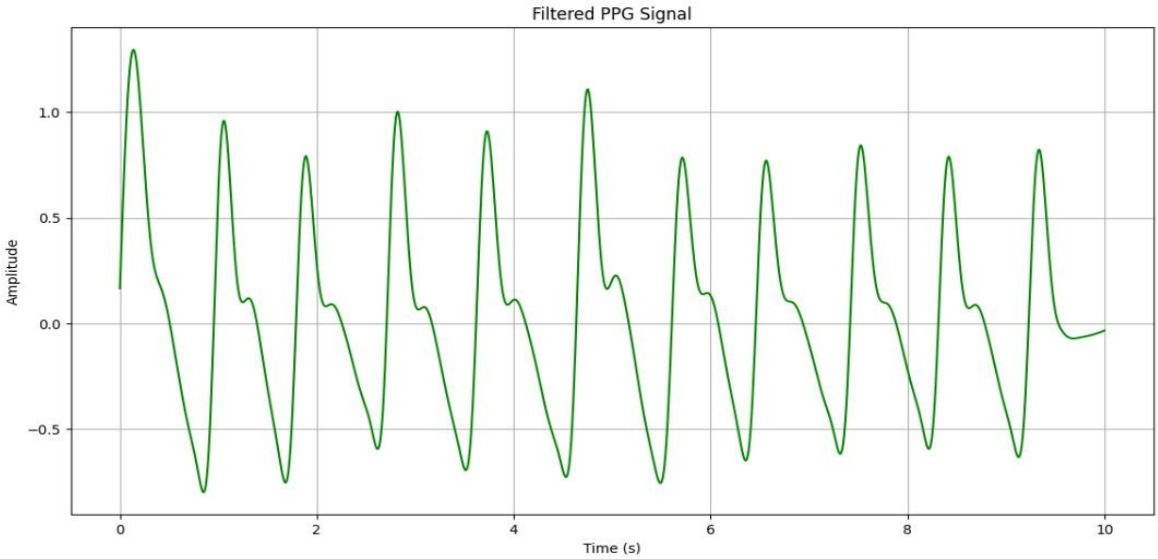
5.Source Code:

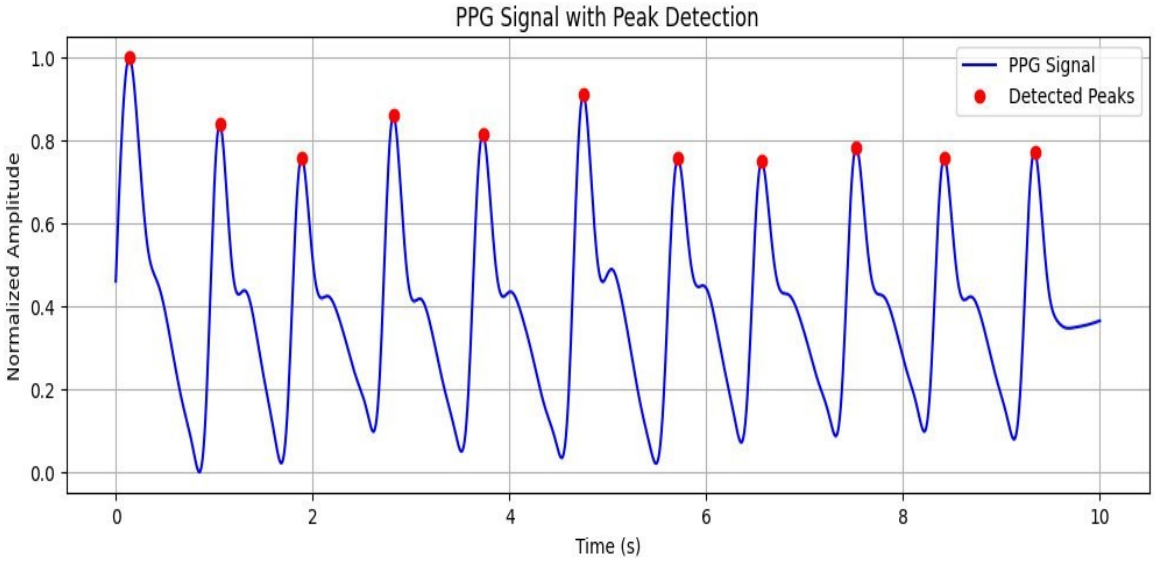
```
import numpy as np

import matplotlib.pyplot as plt

import scipy.signal as signal

import neurokit2 as nk



# Step 1: Generate Synthetic PPG Signal

fs = 100  # Sampling frequency (Hz)

duration = 10  # seconds

time = np.linspace(0, duration, fs * duration)
```

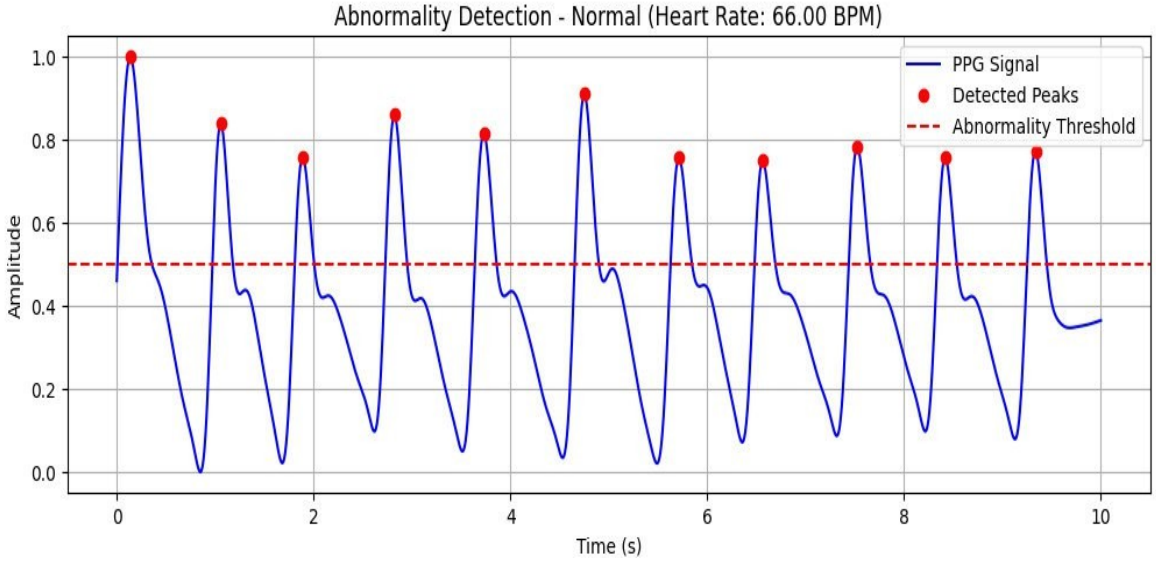
```
# Create synthetic PPG signal

ppg = nk.ppg_simulate(duration=duration, sampling_rate=fs)



# Step 2: Apply Bandpass Filter (Remove Noise)

def bandpass_filter(signal_data, lowcut=0.5, highcut=5, fs=100, order=3):

    nyquist = 0.5 * fs

    low = lowcut / nyquist

    high = highcut / nyquist

    b, a = signal.butter(order, [low, high], btype='band')

    return signal.filtfilt(b, a, signal_data)



filtered_ppg = bandpass_filter(ppg)



# Step 3: Normalize Signal

normalized_ppg = (filtered_ppg - np.min(filtered_ppg)) / (np.max(filtered_ppg) -
np.min(filtered_ppg))



# Step 4: Detect Peaks

peaks, _ = signal.find_peaks(normalized_ppg, distance=fs//2, height=0.5)



# Step 5: Extract Features

heart_rate = len(peaks) / (duration / 60)  # BPM

pulse_amplitude = np.max(normalized_ppg[peaks]) - np.min(normalized_ppg)
```

```
rr_intervals = np.diff(peaks) / fs  # Time between peaks (seconds)


# Step 6: Abnormality Detection (Basic)
abnormal = heart_rate < 60 or heart_rate > 100
status = "Abnormal" if abnormal else "Normal"


# Plot Figure 1: Filtered Signal
plt.figure(figsize=(12, 4))
plt.plot(time, filtered_ppg, color="green")
plt.xlabel("Time (s)")
plt.ylabel("Amplitude")
plt.title("Filtered PPG Signal")
plt.grid()
plt.show()


# Plot Figure 2: Peak Detection
plt.figure(figsize=(12, 4))
plt.plot(time, normalized_ppg, label="PPG Signal", color="blue")
plt.plot(time[peaks], normalized_ppg[peaks], "ro", label="Detected Peaks")
plt.xlabel("Time (s)")
plt.ylabel("Normalized Amplitude")
plt.title("PPG Signal with Peak Detection")
plt.legend()
```

```
plt.grid()

plt.show()



# Plot Figure 3: Abnormality Detection

plt.figure(figsize=(12, 4))

plt.plot(time, normalized_ppg, label="PPG Signal", color="blue")

plt.plot(time[peaks], normalized_ppg[peaks], "ro", label="Detected Peaks")

plt.axhline(y=0.5, color='r', linestyle='--', label="Abnormality Threshold")

plt.xlabel("Time (s)")

plt.ylabel("Amplitude")

plt.title(f"Abnormality Detection - {status} (Heart Rate: {heart_rate:.2f} BPM)")

plt.legend()

plt.grid()

plt.show()
```


Filtered PPG Signal
Amplitude
Time (s)


PPG Signal with Peak Detection
PPG Signal
Detected Peaks
Normalized Amplitude
Time (s)


Abnormality Detection - Normal (Heart Rate: 66.00 BPM)
PPG Signal
Detected Peaks
Abnormality Threshold
Amplitude
Time (s)

7. Saving the Output

To save the output data, including heart rate, pulse amplitude, and RR intervals, use the following:

```
with open("ppg_output.txt", "w") as f:
```

- f.write(f"Heart Rate: {heart_rate:.2f} BPM\n")
- f.write(f"Pulse Amplitude: {pulse_amplitude:.2f}\n")
- f.write(f"RR Intervals (s): {rr_intervals}\n")
- f.write(f"Status: {status}\n")

8. Conclusion

This assignment demonstrated the essential steps in processing a PPG signal, including filtering, normalization, peak detection, and abnormality identification. The implementation successfully identified heart rate and classified the signal as normal or abnormal based on predefined heart rate thresholds. Future improvements could involve advanced machine learning techniques for anomaly detection.