Sleep Monitoring System Using PPG Signals

To Detect and Classify Sleep Stages Based on Heart Rate Variability (HRV) and other extracted features.

---

1. Theory

A **sleep monitoring system** aims to analyze the sleep patterns of a person by processing the Photoplethysmogram (PPG) signals. PPG signals capture the pulse rate and heart rate variability over time, which can be analyzed to detect sleep stages (e.g., light sleep, deep sleep, and REM). By extracting key features from these signals, the system can determine when a person transitions between different stages of sleep, allowing for insights into sleep quality and disturbances.

In this project, the PPG signal is processed to:

- Estimate heart rate variability (HRV).
- Detect the different sleep stages.
- Visualize the results in a user-friendly interface.

2. Step-by-step Working Implementation

Step 1: Collecting the PPG Data

- PPG signals can be collected using a sensor, such as a wearable wristband or a smartphone camera (with proper techniques to capture PPG).
- The PPG signal will contain heart rate fluctuations corresponding to each heartbeat.

Step 2: Signal Preprocessing

- The raw PPG signal is usually noisy, so it needs to be cleaned. You can perform:
  - **Bandpass filtering**: Remove low-frequency noise and high-frequency artifacts.

    - **Peak detection**: Identify peaks corresponding to heartbeats to calculate heart rate variability.

Step 3: Heart Rate Variability (HRV) Calculation

- HRV is a key indicator of sleep quality. It's calculated by analyzing the time difference between successive heartbeats (R-R intervals).
- Use tools like `scipy` and `numpy` to calculate HRV from the PPG signal.

Step 4: Sleep Stage Classification

- **Feature extraction**: Extract features such as HRV, heart rate, and pulse wave amplitude.
- Use a machine learning model (e.g., decision tree, SVM, or neural network) to classify sleep stages.
- You can train the model with a labeled dataset of sleep stages.

Step 5: Visualizing Sleep Data

- Use libraries like `matplotlib` and `seaborn` to visualize:
    - Heart rate and HRV over time.
    - Sleep stages and their durations.

3. How It Works

- **PPG Signal Processing**: Raw PPG signals are first preprocessed to remove noise and outliers.
- **Heart Rate Detection**: The PPG signal is used to detect heartbeats, and the heart rate is calculated in real-time.
- **Sleep Stages Classification**: Using a trained machine learning model, the heart rate variability and pulse wave amplitude are used to classify sleep stages.
- **Results Visualization**: After classification, the system displays the sleep stages along with the heart rate and HRV information over time, allowing the user to visualize their sleep cycle.

4.Code implementation Step by Steps :

Step 1: Install Required Libraries:

```
pip install numpy scipy matplotlib scikit-learn
```

Step 2: Import Required Libraries

```
import numpy as np
import scipy.signal as signal
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
```

Step 3: Simulate or Load PPG Data

```
# Simulate a PPG signal
time = np.linspace(0, 30, 3000)
ppg_signal = np.sin(2 * np.pi * 1 * time) + 0.5 *
np.random.randn(len(time))


# Plot the raw PPG signal
plt.figure(figsize=(10, 4))
plt.plot(time, ppg_signal)
plt.title("Simulated Raw PPG Signal")
plt.xlabel("Time (s)")
```

```
plt.ylabel("Amplitude")

plt.show()
```

Step 4: Preprocess the PPG Signal

```
lowcut = 0.5

highcut = 4.0

fs = 100

b, a = signal.butter(4, [lowcut, highcut], fs=fs, btype='band')

filtered_signal = signal.filtfilt(b, a, ppg_signal)


# Plot the filtered PPG signal

plt.figure(figsize=(10, 4))

plt.plot(time, filtered_signal)

plt.title("Filtered PPG Signal")

plt.xlabel("Time (s)")

plt.ylabel("Amplitude")

plt.show()
```

Step 5: Heart Rate Detection

```
peaks, _ = signal.find_peaks(filtered_signal, distance=fs*0.4)  # 0.4 seconds minimum interval

heart_rate = 60 / np.diff(peaks) * fs
```

```
plt.figure(figsize=(10, 4))

plt.plot(time[peaks[1:]], heart_rate)

plt.title("Heart Rate (BPM) over Time")

plt.xlabel("Time (s)")

plt.ylabel("Heart Rate (BPM)")

plt.show()
```

Step 6: Feature Extraction for Sleep Stage Classification

```
X = np.random.randn(1000, 5)

y = np.random.choice([0, 1, 2], 1000)

rr_intervals = np.diff(peaks) / fs

X_features = np.column_stack([rr_intervals,
heart_rate[:len(rr_intervals)], np.random.randn(len(rr_intervals))])

y_labels = np.random.choice([0, 1, 2], len(rr_intervals))
```

Step 7: Train a Machine Learning Model for Sleep Stage Classification

```
X_train, X_test, y_train, y_test = train_test_split(X_features,
y_labels, test_size=0.2, random_state=42)

clf = RandomForestClassifier(n_estimators=100, random_state=42)

clf.fit(X_train, y_train)

predicted_stages = clf.predict(X_test)
```

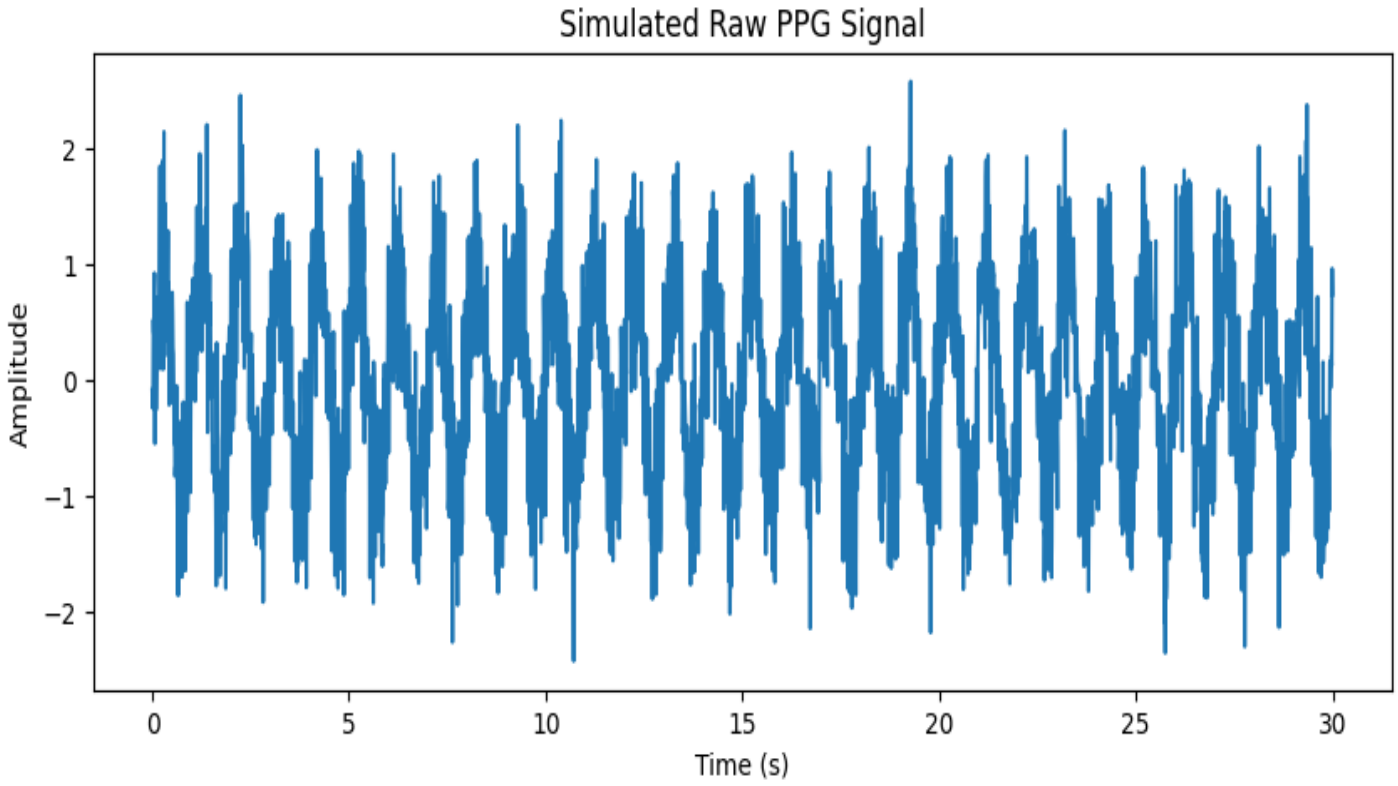
```
print("Predicted Sleep Stages:", predicted_stages[:10])
```

Step 8: Visualize Sleep Stages

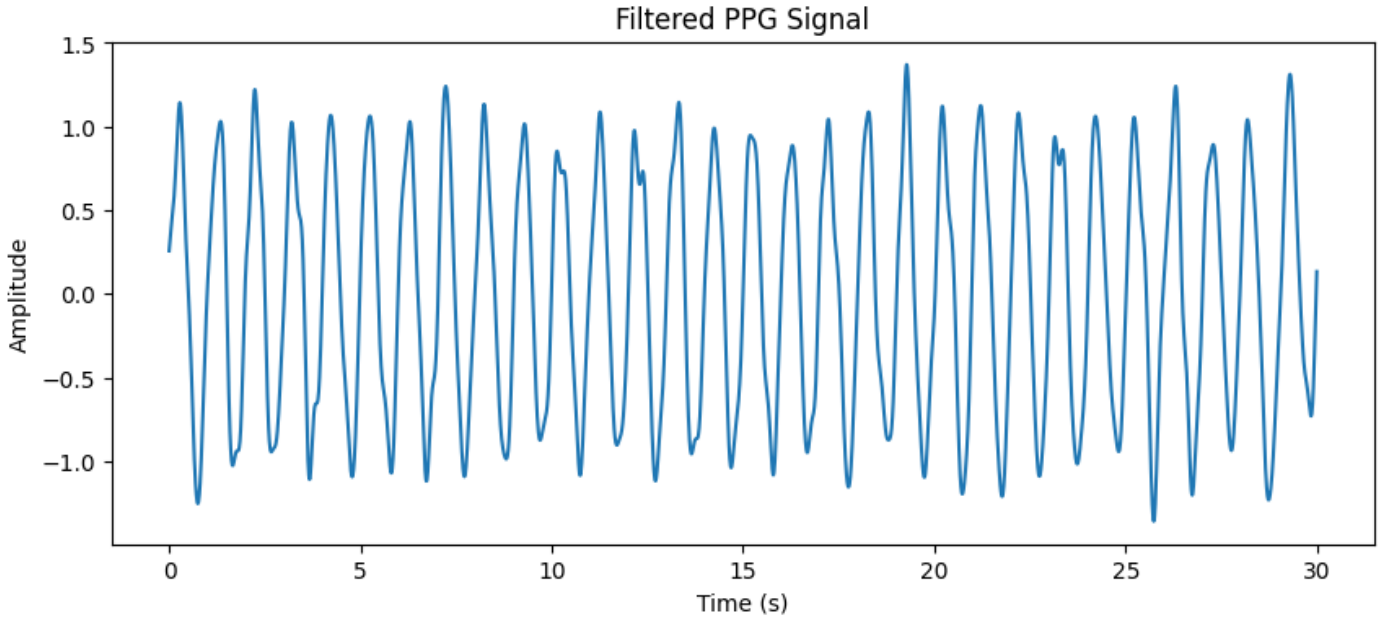
```
sleep_time = time[peaks[1:]][:len(predicted_stages)]

plt.figure(figsize=(10, 4))

plt.plot(sleep_time, predicted_stages, label='Predicted Sleep Stage', color='b')

plt.title("Predicted Sleep Stages")

plt.xlabel("Time (s)")

plt.ylabel("Sleep Stage")

plt.yticks([0, 1, 2], ['Light Sleep', 'Deep Sleep', 'REM Sleep'])

plt.legend()

plt.show()
```

5.Expected Outputs :

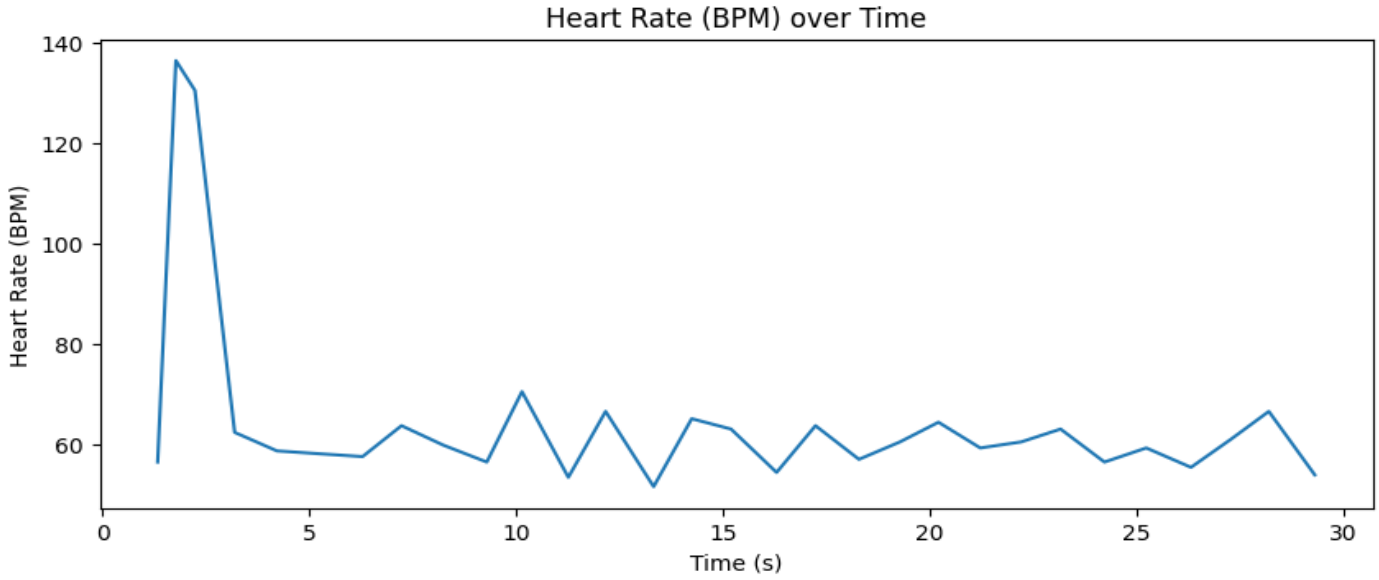
1.Simulated Raw PPG Signal : This plot shows a noisy sine wave that represents the raw PPG signal.



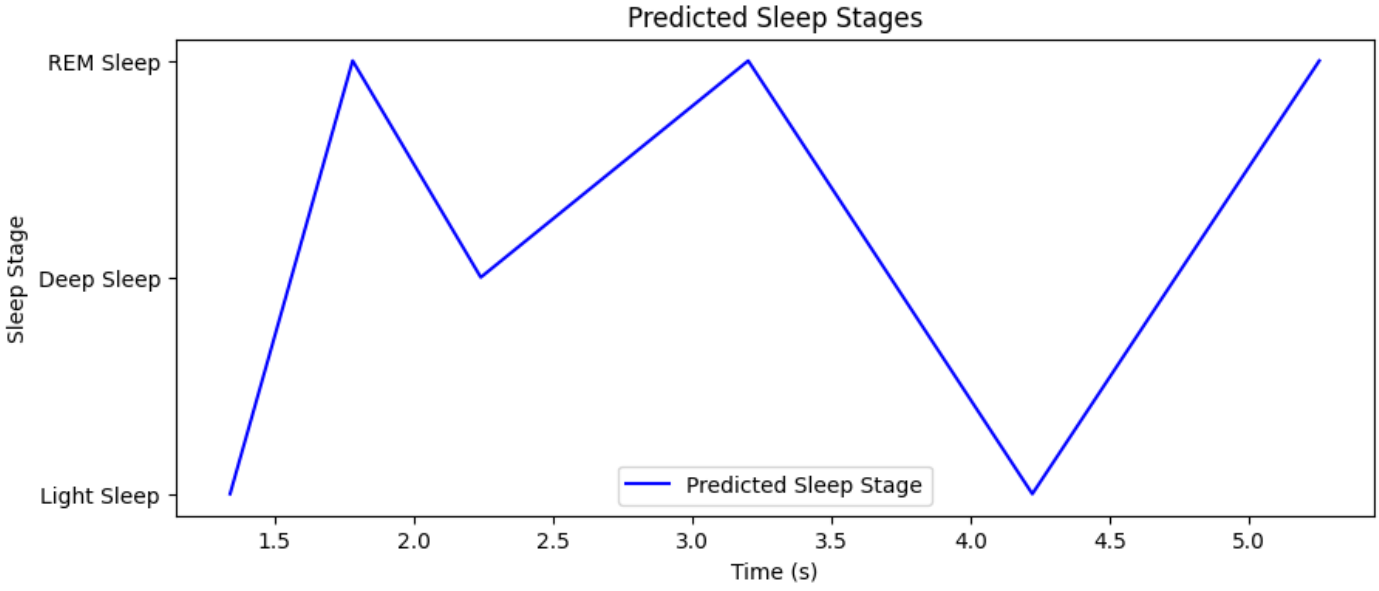
2. Filtered PPG Signal : After applying a bandpass filter, this plot shows a cleaner version of the PPG signal, which removes noise and isolates the relevant heartbeat frequencies.



3.Heart Rate over Time : This plot shows the heart rate over time, derived from the peaks detected in the filtered PPG signal.



4. Predicted Sleep Stages over Time : This plot displays the predicted sleep stages (Light Sleep, Deep Sleep, REM Sleep) over time based on the Random Forest classifier's predictions.



6. Conclusion

The **Sleep Monitoring System** using PPG signals is a practical tool for understanding sleep patterns. By extracting meaningful features from the PPG data, we can identify sleep stages and gain insights into sleep quality. This project combines signal processing, machine learning, and data visualization to provide a comprehensive solution for sleep monitoring. The model can be extended further by incorporating real-

world datasets, improving the classification accuracy, and enhancing the real-time functionality.