Import Required Libraries

Import necessary libraries such as pandas, matplotlib, and seaborn.

In [38]:
```
# Import Required Libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Set seaborn style for better aesthetics
sns.set(style="whitegrid")
```

Load the Data

Load the CSV file into a pandas DataFrame.

In [39]:
```
# Load the Data
# Load the CSV file into a pandas DataFrame
file_path = '../data/mindMonitor_2025-02-15--19-18-29.csv'
df = pd.read_csv(file_path)

# Display the first few rows of the DataFrame to verify the data is loaded correctl
df.head()
```

Out[39]:

	TimeStamp	Delta_TP9	Delta_AF7	Delta_AF8	Delta_TP10	Theta_TP9	Theta_AF7	Theta
0	2025-02-15 19:18:29.097	NaN	NaN	NaN	NaN	NaN	NaN	
1	2025-02-15 19:18:29.257	NaN	NaN	NaN	NaN	NaN	NaN	
2	2025-02-15 19:18:29.261	NaN	NaN	NaN	NaN	NaN	NaN	
3	2025-02-15 19:18:29.597	0.0	0.285171	0.135338	0.0	0.0	0.011946	0.1
4	2025-02-15 19:18:30.098	0.0	0.247423	0.417576	0.0	0.0	-0.103462	0.1

5 rows × 39 columns

Clean the Data

Handle missing values and convert data types as necessary.

In [40]:
```
# Clean the Data
# Handle missing values and convert data types as necessary

# Replace empty strings with NaN
df.replace("", float("NaN"), inplace=True)

# Convert columns to appropriate data types
df['TimeStamp'] = pd.to_datetime(df['TimeStamp'], errors='coerce')

# Fill missing values with forward fill method
df.fillna(method='ffill', inplace=True)

# Verify the data types and check for any remaining missing values
df.info()
df.isnull().sum()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1724 entries, 0 to 1723
Data columns (total 39 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   TimeStamp        1724 non-null   datetime64[ns]
 1   Delta_TP9        1721 non-null   float64
 2   Delta_AF7        1721 non-null   float64
 3   Delta_AF8        1721 non-null   float64
 4   Delta_TP10       1721 non-null   float64
 5   Theta_TP9        1721 non-null   float64
 6   Theta_AF7        1721 non-null   float64
 7   Theta_AF8        1721 non-null   float64
 8   Theta_TP10       1721 non-null   float64
 9   Alpha_TP9        1721 non-null   float64
 10  Alpha_AF7        1721 non-null   float64
 11  Alpha_AF8        1721 non-null   float64
 12  Alpha_TP10       1721 non-null   float64
 13  Beta_TP9         1721 non-null   float64
 14  Beta_AF7         1721 non-null   float64
 15  Beta_AF8         1721 non-null   float64
 16  Beta_TP10        1721 non-null   float64
 17  Gamma_TP9        1721 non-null   float64
 18  Gamma_AF7        1721 non-null   float64
 19  Gamma_AF8        1721 non-null   float64
 20  Gamma_TP10       1721 non-null   float64
 21  RAW_TP9          1721 non-null   float64
 22  RAW_AF7          1721 non-null   float64
 23  RAW_AF8          1721 non-null   float64
 24  RAW_TP10         1721 non-null   float64
 25  AUX_RIGHT        1721 non-null   float64
 26  Accelerometer_X  1721 non-null   float64
 27  Accelerometer_Y  1721 non-null   float64
 28  Accelerometer_Z  1721 non-null   float64
 29  Gyro_X           1721 non-null   float64
 30  Gyro_Y           1721 non-null   float64
 31  Gyro_Z           1721 non-null   float64
 32  HeadBandOn       1721 non-null   float64
 33  HSI_TP9          1721 non-null   float64
 34  HSI_AF7          1721 non-null   float64
 35  HSI_AF8          1721 non-null   float64
 36  HSI_TP10         1721 non-null   float64
 37  Battery          1721 non-null   float64
 38  Elements         1724 non-null   object
dtypes: datetime64[ns](1), float64(37), object(1)
memory usage: 525.4+ KB
```

```
/tmp/ipykernel_3259316/4278126818.py:11: FutureWarning: DataFrame.fillna with 'metho
d' is deprecated and will raise in a future version. Use obj.ffill() or obj.bfill()
instead.
  df.fillna(method='ffill', inplace=True)
```

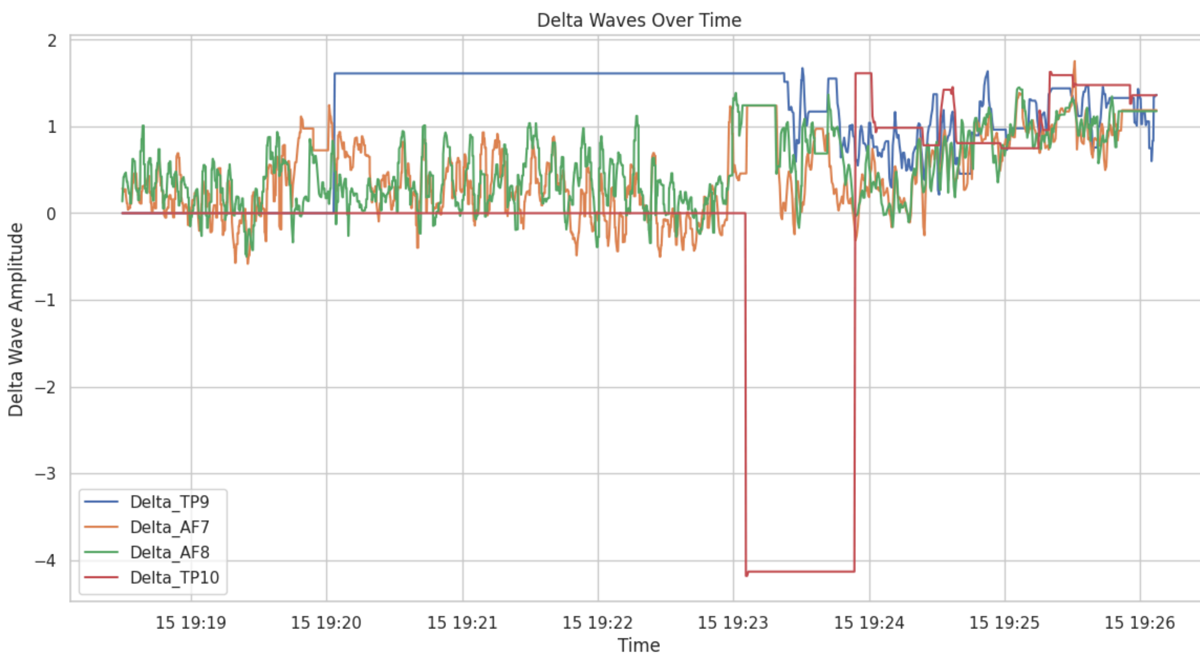
Out[40]:
```
TimeStamp          0
Delta_TP9          3
Delta_AF7          3
Delta_AF8          3
Delta_TP10         3
Theta_TP9          3
Theta_AF7          3
Theta_AF8          3
Theta_TP10         3
Alpha_TP9          3
Alpha_AF7          3
Alpha_AF8          3
Alpha_TP10         3
Beta_TP9           3
Beta_AF7           3
Beta_AF8           3
Beta_TP10          3
Gamma_TP9          3
Gamma_AF7          3
Gamma_AF8          3
Gamma_TP10         3
RAW_TP9            3
RAW_AF7            3
RAW_AF8            3
RAW_TP10           3
AUX_RIGHT          3
Accelerometer_X    3
Accelerometer_Y    3
Accelerometer_Z    3
Gyro_X             3
Gyro_Y             3
Gyro_Z             3
HeadBandOn         3
HSI_TP9            3
HSI_AF7            3
HSI_AF8            3
HSI_TP10           3
Battery            3
Elements           0
dtype: int64
```

Plot Delta Waves

Plot the Delta wave data from the different channels over time.

In [41]:
```
# Plot Delta Waves
plt.figure(figsize=(14, 7))

# Plot Delta_TP9
plt.plot(df['TimeStamp'], df['Delta_TP9'], label='Delta_TP9')

# Plot Delta_AF7
plt.plot(df['TimeStamp'], df['Delta_AF7'], label='Delta_AF7')
```

```python
# Plot Delta_AF8
plt.plot(df['TimeStamp'], df['Delta_AF8'], label='Delta_AF8')

# Plot Delta_TP10
plt.plot(df['TimeStamp'], df['Delta_TP10'], label='Delta_TP10')

# Add title and labels
plt.title('Delta Waves Over Time')
plt.xlabel('Time')
plt.ylabel('Delta Wave Amplitude')

# Add legend
plt.legend()

# Display the plot
plt.show()
```

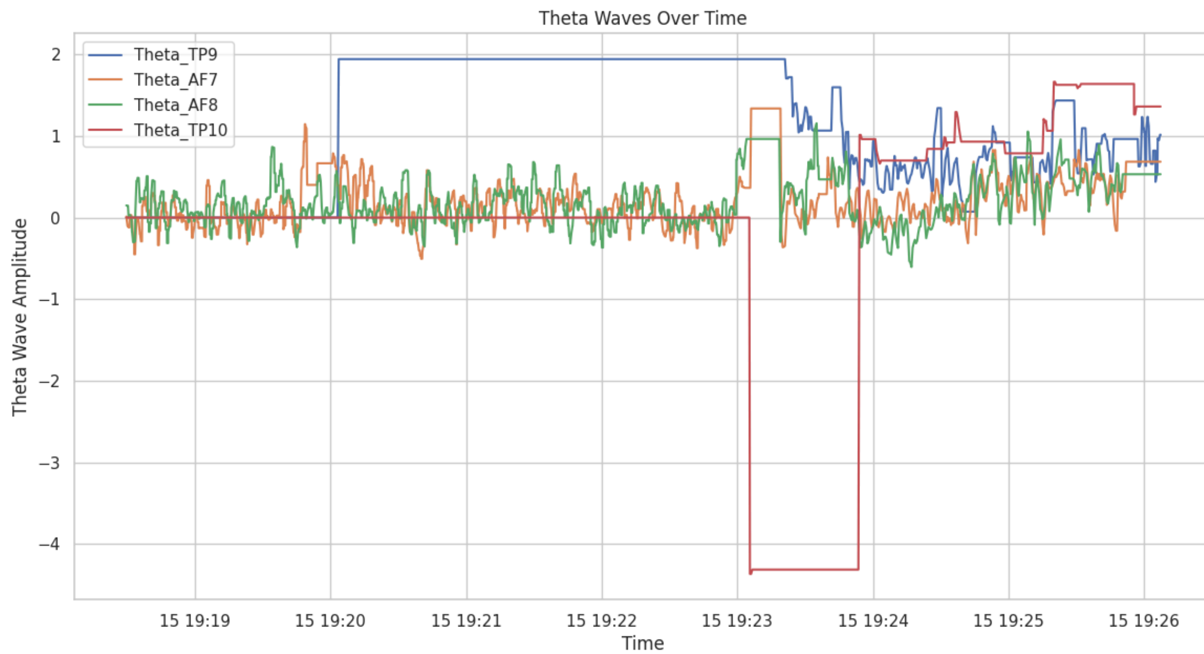


Plot Theta Waves

Plot the Theta wave data from the different channels over time.

In [42]:
```python
# Plot Theta Waves
plt.figure(figsize=(14, 7))

# Plot Theta_TP9
plt.plot(df['TimeStamp'], df['Theta_TP9'], label='Theta_TP9')

# Plot Theta_AF7
plt.plot(df['TimeStamp'], df['Theta_AF7'], label='Theta_AF7')

# Plot Theta_AF8
plt.plot(df['TimeStamp'], df['Theta_AF8'], label='Theta_AF8')
```

```
# Plot Theta_TP10
plt.plot(df['TimeStamp'], df['Theta_TP10'], label='Theta_TP10')

# Add title and labels
plt.title('Theta Waves Over Time')
plt.xlabel('Time')
plt.ylabel('Theta Wave Amplitude')

# Add legend
plt.legend()

# Display the plot
plt.show()
```



Plot Alpha Waves

Plot the Alpha wave data from the different channels over time.

In [43]:
```
# Plot Alpha Waves
plt.figure(figsize=(14, 7))

# Plot Alpha_TP9
plt.plot(df['TimeStamp'], df['Alpha_TP9'], label='Alpha_TP9')

# Plot Alpha_AF7
plt.plot(df['TimeStamp'], df['Alpha_AF7'], label='Alpha_AF7')

# Plot Alpha_AF8
plt.plot(df['TimeStamp'], df['Alpha_AF8'], label='Alpha_AF8')

# Plot Alpha_TP10
plt.plot(df['TimeStamp'], df['Alpha_TP10'], label='Alpha_TP10')
```

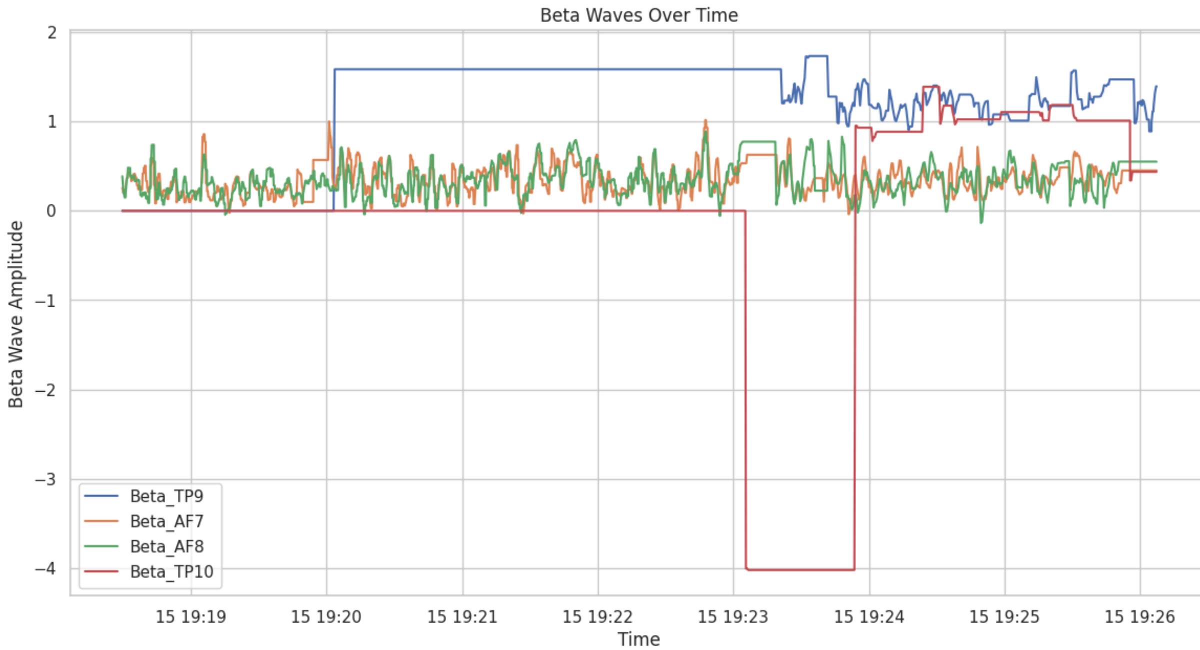
```python
# Add title and labels
plt.title('Alpha Waves Over Time')
plt.xlabel('Time')
plt.ylabel('Alpha Wave Amplitude')

# Add legend
plt.legend()

# Display the plot
plt.show()
```

Plot Beta Waves

Plot the Beta wave data from the different channels over time.

In [44]:
```python
# Plot Beta Waves
plt.figure(figsize=(14, 7))

# Plot Beta_TP9
plt.plot(df['TimeStamp'], df['Beta_TP9'], label='Beta_TP9')

# Plot Beta_AF7
plt.plot(df['TimeStamp'], df['Beta_AF7'], label='Beta_AF7')

# Plot Beta_AF8
plt.plot(df['TimeStamp'], df['Beta_AF8'], label='Beta_AF8')

# Plot Beta_TP10
plt.plot(df['TimeStamp'], df['Beta_TP10'], label='Beta_TP10')

# Add title and labels
plt.title('Beta Waves Over Time')
```

```
plt.xlabel('Time')
plt.ylabel('Beta Wave Amplitude')

# Add legend
plt.legend()

# Display the plot
plt.show()
```



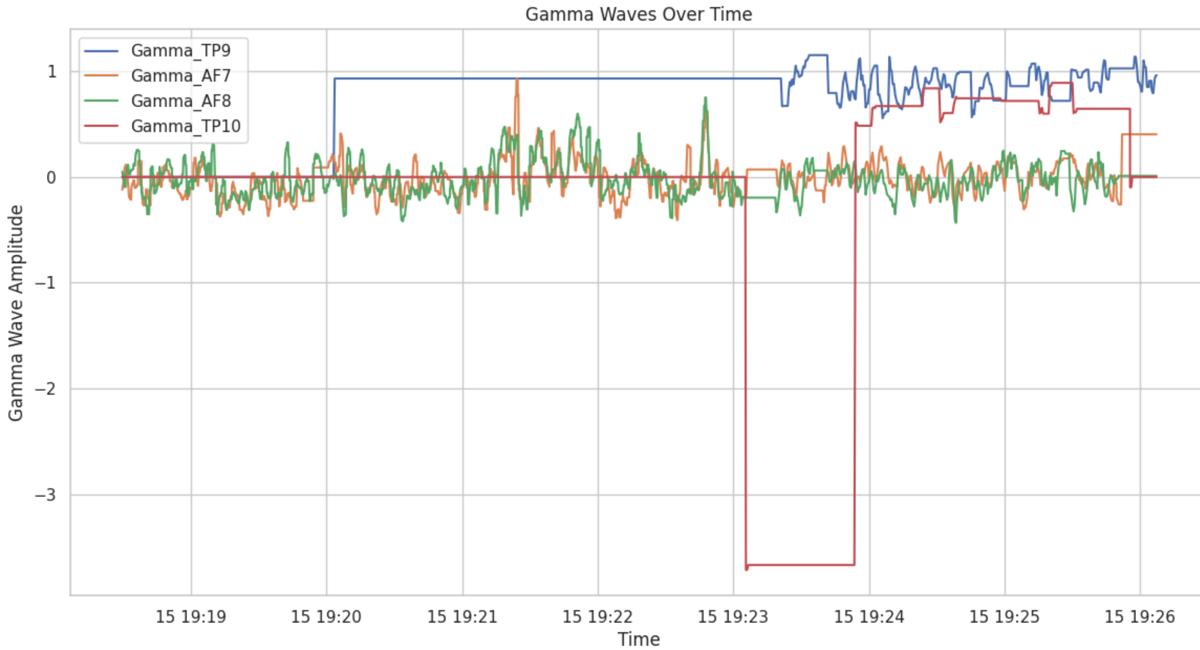
Plot Gamma Waves

Plot the Gamma wave data from the different channels over time.

In [45]:

```
# Plot Gamma Waves
plt.figure(figsize=(14, 7))

# Plot Gamma_TP9
plt.plot(df['TimeStamp'], df['Gamma_TP9'], label='Gamma_TP9')

# Plot Gamma_AF7
plt.plot(df['TimeStamp'], df['Gamma_AF7'], label='Gamma_AF7')

# Plot Gamma_AF8
plt.plot(df['TimeStamp'], df['Gamma_AF8'], label='Gamma_AF8')

# Plot Gamma_TP10
plt.plot(df['TimeStamp'], df['Gamma_TP10'], label='Gamma_TP10')

# Add title and labels
plt.title('Gamma Waves Over Time')
plt.xlabel('Time')
plt.ylabel('Gamma Wave Amplitude')
```

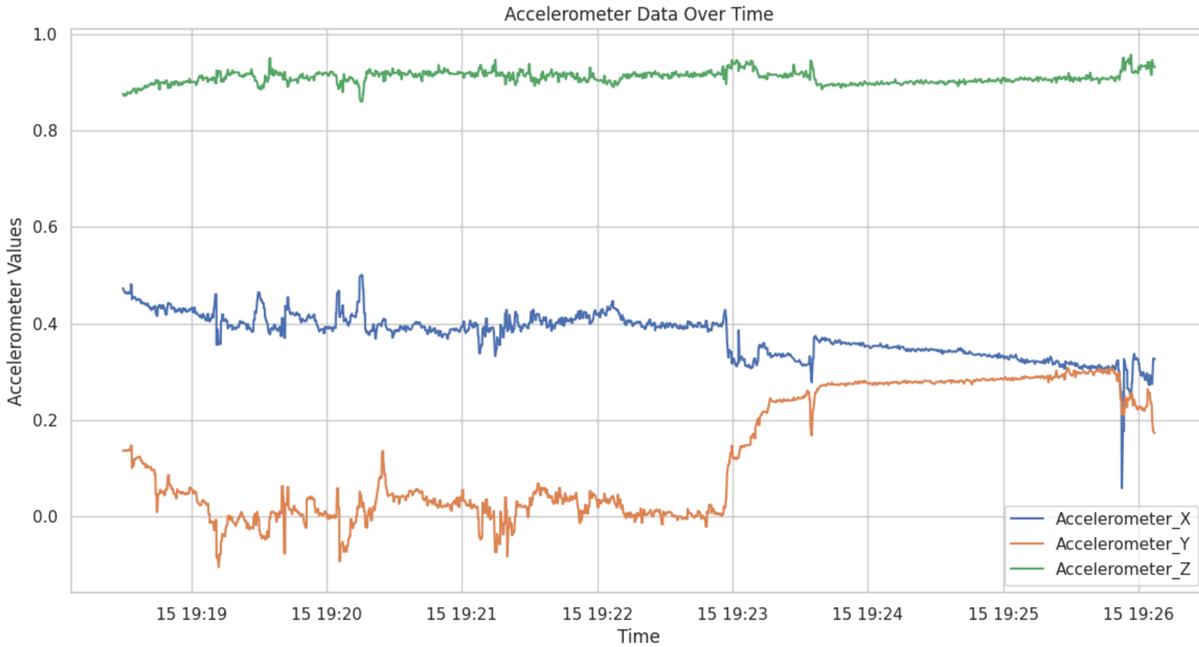
```python
# Add legend
plt.legend()

# Display the plot
plt.show()
```



Plot Accelerometer Data

Plot the accelerometer data (X, Y, Z) over time.

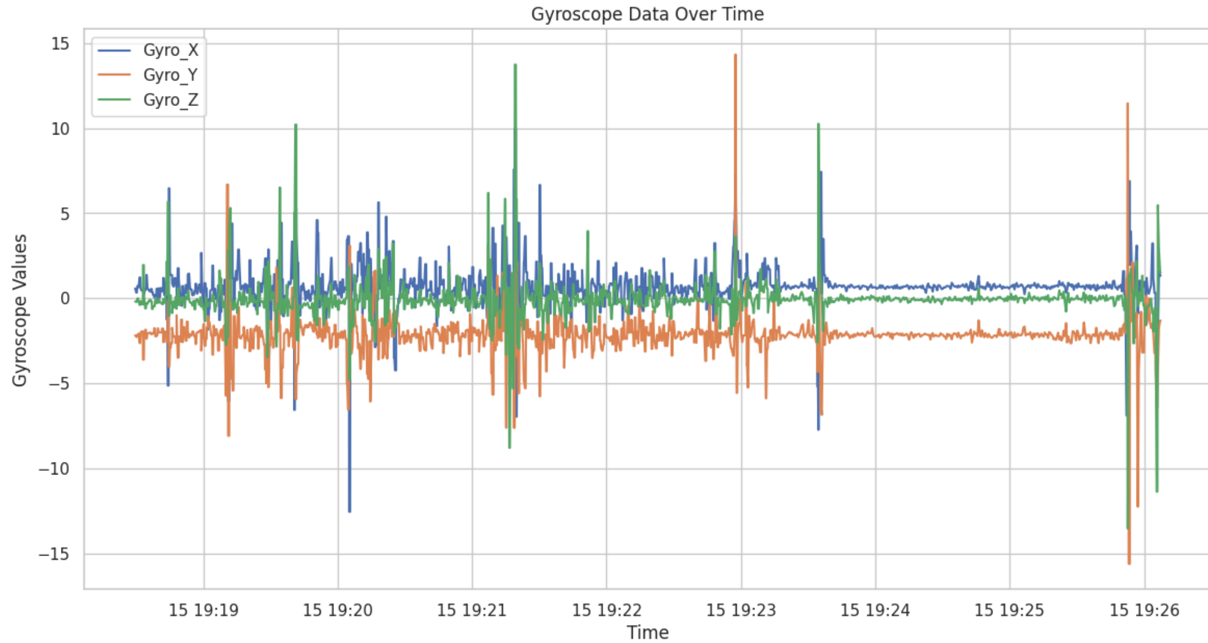
In [46]:
```python
# Plot Accelerometer Data
plt.figure(figsize=(14, 7))

# Plot Accelerometer_X
plt.plot(df['TimeStamp'], df['Accelerometer_X'], label='Accelerometer_X')

# Plot Accelerometer_Y
plt.plot(df['TimeStamp'], df['Accelerometer_Y'], label='Accelerometer_Y')

# Plot Accelerometer_Z
plt.plot(df['TimeStamp'], df['Accelerometer_Z'], label='Accelerometer_Z')

# Add title and labels
plt.title('Accelerometer Data Over Time')
plt.xlabel('Time')
plt.ylabel('Accelerometer Values')

# Add legend
plt.legend()

# Display the plot
plt.show()
```



Plot Gyroscope Data

Plot the gyroscope data (X, Y, Z) over time.

In [47]:
```
# Plot Gyroscope Data
plt.figure(figsize=(14, 7))

# Plot Gyro_X
plt.plot(df['TimeStamp'], df['Gyro_X'], label='Gyro_X')

# Plot Gyro_Y
plt.plot(df['TimeStamp'], df['Gyro_Y'], label='Gyro_Y')

# Plot Gyro_Z
plt.plot(df['TimeStamp'], df['Gyro_Z'], label='Gyro_Z')

# Add title and labels
plt.title('Gyroscope Data Over Time')
plt.xlabel('Time')
plt.ylabel('Gyroscope Values')

# Add legend
plt.legend()

# Display the plot
plt.show()
```

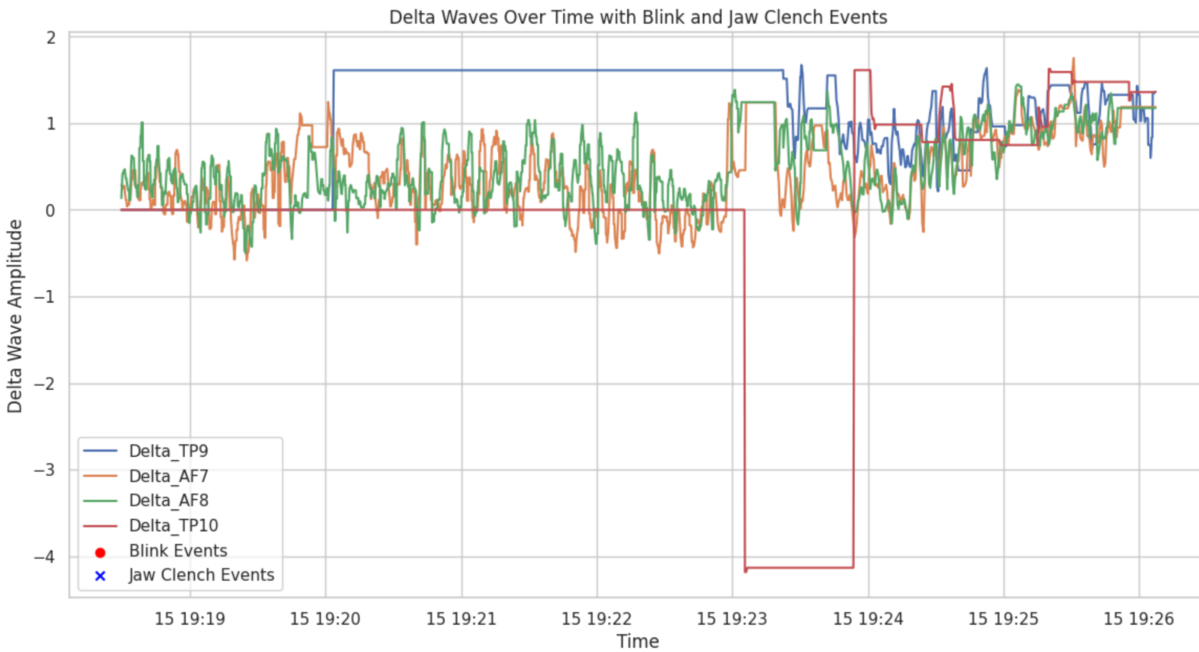


Plot Blink and Jaw Clench Events

Highlight the blink and jaw clench events on the plots.

In [48]:
```
# Plot Blink and Jaw Clench Events

# Extract blink and jaw clench events
blink_events = df[df['Elements'] == '/muse/elements/blink']
jaw_clench_events = df[df['Elements'] == '/muse/elements/jaw_clench']

# Plot Delta Waves with Blink and Jaw Clench Events
plt.figure(figsize=(14, 7))

# Plot Delta_TP9
plt.plot(df['TimeStamp'], df['Delta_TP9'], label='Delta_TP9')

# Plot Delta_AF7
plt.plot(df['TimeStamp'], df['Delta_AF7'], label='Delta_AF7')

# Plot Delta_AF8
plt.plot(df['TimeStamp'], df['Delta_AF8'], label='Delta_AF8')

# Plot Delta_TP10
plt.plot(df['TimeStamp'], df['Delta_TP10'], label='Delta_TP10')

# Highlight blink events
plt.scatter(blink_events['TimeStamp'], [max(df['Delta_TP9'])] * len(blink_events),

# Highlight jaw clench events
plt.scatter(jaw_clench_events['TimeStamp'], [max(df['Delta_TP9'])] * len(jaw_clench

# Add title and labels
plt.title('Delta Waves Over Time with Blink and Jaw Clench Events')
```

```
plt.xlabel('Time')
plt.ylabel('Delta Wave Amplitude')

# Add legend
plt.legend()

# Display the plot
plt.show()
```



In [49]:

```
# Plot all waves (Alpha, Beta, Gamma, Delta, Theta) on the same plot

# Define the wave types and their corresponding columns
wave_types = {
    'Delta': ['Delta_TP9', 'Delta_AF7', 'Delta_AF8', 'Delta_TP10'],
    'Theta': ['Theta_TP9', 'Theta_AF7', 'Theta_AF8', 'Theta_TP10'],
    'Alpha': ['Alpha_TP9', 'Alpha_AF7', 'Alpha_AF8', 'Alpha_TP10'],
    'Beta': ['Beta_TP9', 'Beta_AF7', 'Beta_AF8', 'Beta_TP10'],
    'Gamma': ['Gamma_TP9', 'Gamma_AF7', 'Gamma_AF8', 'Gamma_TP10']
}

plt.figure(figsize=(20, 10))

# Plot each wave type
for wave_type, columns in wave_types.items():
    for column in columns:
        plt.plot(df['TimeStamp'], df[column], label=f'{wave_type}_{column.split("_"

# Add title and labels
plt.title('EEG Waves Over Time')
plt.xlabel('Time')
plt.ylabel('Amplitude')

# Add legend
plt.legend()
```

```python
# Display the plot
plt.show()
```