In [88]:

```
import numpy as np
import scipy as sp
import scipy.signal as sg
import librosa as lib
import matplotlib.pyplot as plt
```

In [89]:

```
# Part 4
```

In [90]:

```
# Part 4.1
```

In [91]:

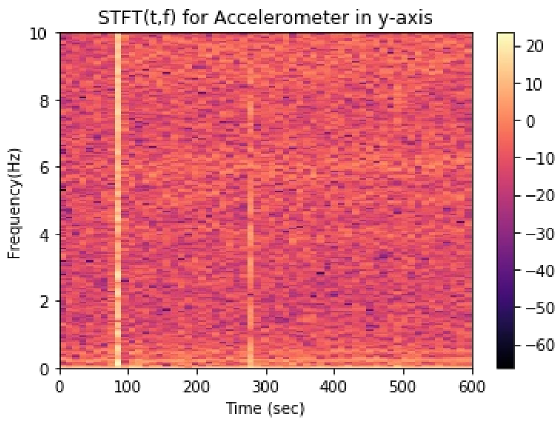
```
# loading of signal
sleep_01 = np.load('sleep_01.npz')

# accelerometer in y-axis
accY = sleep_01['acc'][:,1]
# gyroscope in y-axis
gyrY = sleep_01['gyr'][:,1]

# signals
signals = [accY, gyrY]
# frequency
fs = 20

# overlap*freq
epik_freq = 10*fs
# length*freq
length_freq = 20*fs

signals_stft = [lib.stft(accY,hop_length=epik_freq,win_length=length_freq), lib.stft(gyrY,hop_length=epik_freq,wi
n_length=length_freq)]
[r,c] = np.shape(signals_stft[0])
```

In [92]:

```
# t = linspace (0,deigmata/fdeigmatol,stfttdeigmata)
t = np.linspace(0, 600, c)
# f = linspace (0, fdeigmatol/2, stftfdeigmata)
f = np.linspace(0, fs/2, r)

# Spectrum Depiction

plt.pcolormesh(t,f,20*np.log10(abs(signals_stft[0])),cmap='magma')
plt.colorbar()
plt.title('STFT(t,f) for Accelerometer in y-axis')
plt.xlabel('Time (sec)')
plt.ylabel('Frequency(Hz)')
```

Out[92]:

```
Text(0, 0.5, 'Frequency(Hz)')
```

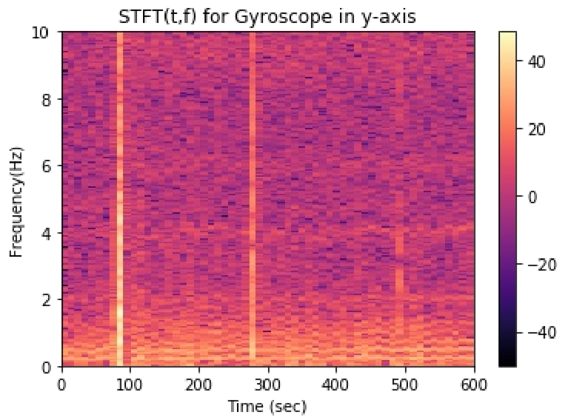


In [93]:

```
plt.pcolormesh(t,f,20*np.log10(abs(signals_stft[1])),cmap='magma')
plt.colorbar()
plt.title('STFT(t,f) for Gyroscope in y-axis')
plt.xlabel('Time (sec)')
plt.ylabel('Frequency(Hz)')
```

Out[93]:

```
Text(0, 0.5, 'Frequency(Hz)')
```

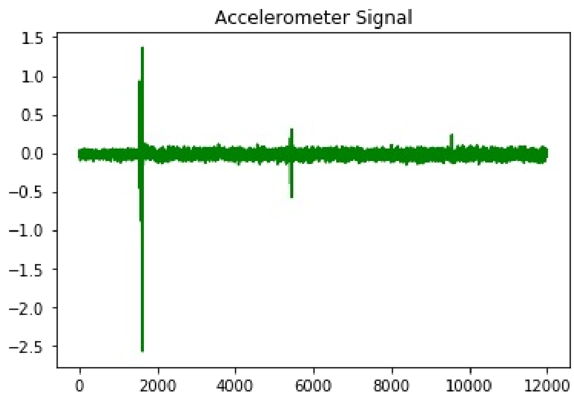


In [94]:

```
# Signal Depiction

plt.plot(signals[0], color='g')
plt.title('Accelerometer Signal')
```

Out[94]:

```
Text(0.5, 1.0, 'Accelerometer Signal')
```

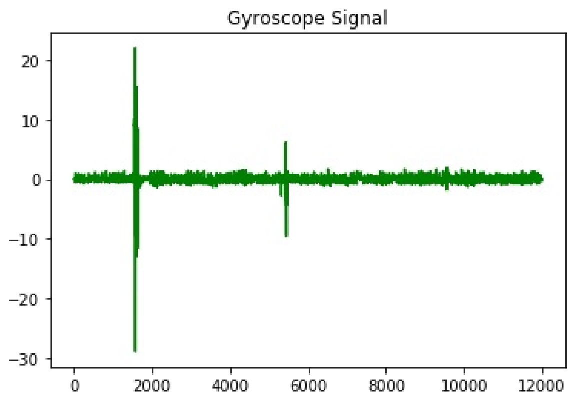


In [95]:

```
plt.plot(signals[1],color='g')
plt.title('Gyroscope Signal')
```

Out[95]:

```
Text(0.5, 1.0, 'Gyroscope Signal')
```



In [96]:

```
# Observation:

# Due to the fact that the accelerometers and the gyroscopes are built with
# microelectronic circuits, they might present noise (mechanic or electric).
# It is obvious that when the user is motionless, a large part of the spectrum
# appears with some 'special' values (these are the intense purple spots which
# correspond to noise).
```

In [97]:

```
# Part 4.2
```

In [98]:

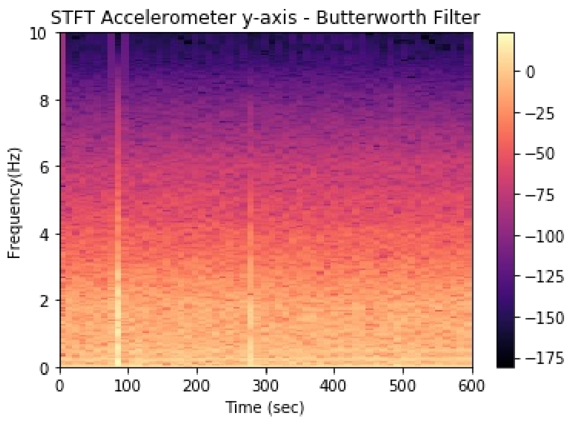
```
# butterworth function

def butterworth(signal, fs):
    # filter's order
    filter_order = 5
    # least cutoff frequency
    cutoff_freq = 2
    nyq = 0.5*fs
    # normalization
    lowcut_normal = cutoff_freq/nyq

    a, b = sg.butter(filter_order, lowcut_normal, btype = 'lowpass')
    signal_butter = sg.lfilter(a, b, signal)

    return signal_butter

signals_butter = [butterworth(signals[0],fs), butterworth(signals[1],fs)]

signals_butter_stft = [lib.stft(signals_butter[0],hop_length=epik_freq,win_length=length_freq),lib.stft(signals_b
utter[1],hop_length=epik_freq,win_length=length_freq)]
```

In [99]:

```
# Spectrum Depiction

plt.pcolormesh(t,f,20*np.log10(abs(signals_butter_stft[0])),cmap='magma')
plt.colorbar()
plt.title("STFT Accelerometer y-axis - Butterworth Filter")
plt.xlabel('Time (sec)')
plt.ylabel('Frequency(Hz)')
```

Out[99]:

```
Text(0, 0.5, 'Frequency(Hz)')
```

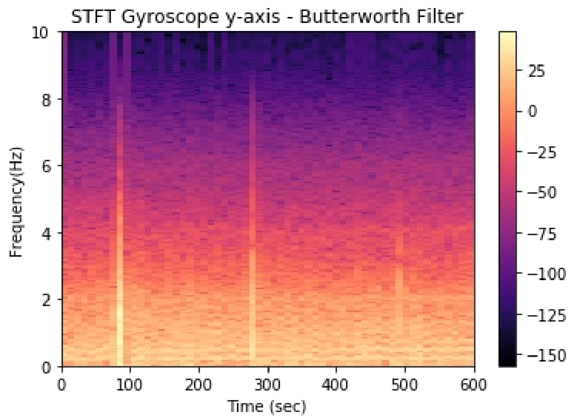


In [100]:

```
plt.pcolormesh(t,f,20*np.log10(abs(signals_butter_stft[1])),cmap='magma')
plt.colorbar();
plt.title("STFT Gyroscope y-axis - Butterworth Filter")
plt.xlabel('Time (sec)')
plt.ylabel('Frequency(Hz)')
```

Out[100]:

```
Text(0, 0.5, 'Frequency(Hz)')
```

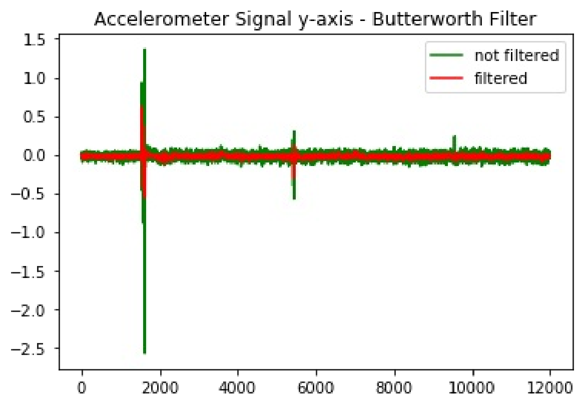


In [101]:

```
# Signal Depiction

plt.plot(signals[0], color='g')
plt.plot(signals_butter[0], color='r')
plt.title("Accelerometer Signal y-axis - Butterworth Filter")
plt.legend(['not filtered','filtered'])
```

Out[101]:

```
<matplotlib.legend.Legend at 0x18bc32474c8>
```

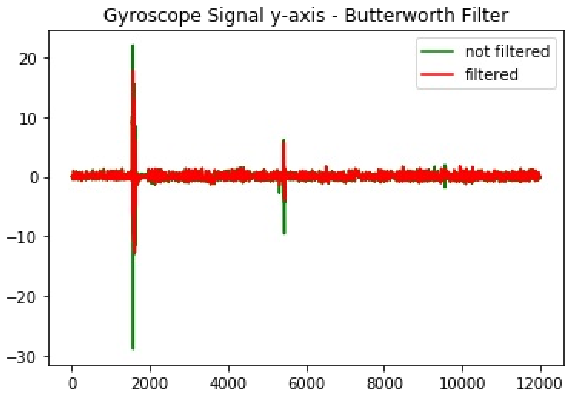


In [102]:

```
plt.plot(signals[1], color='g')
plt.plot(signals_butter[1], color='r')
plt.title("Gyroscope Signal y-axis - Butterworth Filter")
plt.legend(['not filtered','filtered'])
```

Out[102]:

```
<matplotlib.legend.Legend at 0x18bd9954948>
```



In [103]:

```
# Observation:

# It is obvious from the reduction of the amplitude in noise intervals of the
# signal and also from the reduction of the purple spots in the Spectrum that
# with the use of the Butterworth Filter, the noise is reduced.
# However, the Butterworth  Filter is not optimal because, as far as the
# Gyroscope's signal is concerned, the reduction of the noise's amplitude is
# not that obvious.
```

In [104]:

```
# Part 4.3
```

In [105]:

```
# function for the calculation of Power Spectrum

def psd(signal):
    Power = np.sqrt(abs(np.fft.fft(signal))) / len(signal)
    return Power
```

In [106]:

```
# Wiener filter function

def wiener_filter(signal, Pu):
    # signal with zero padding
    signal_padded = np.pad(signal, (0, 8), 'constant', constant_values=(0))
    # window length
    L = 400
    # initialization with zeros
    signal_wiener = np.zeros(len(signal_padded))

    for i in range(int(len(signal_padded)/L)):
        first = i*L
        last = first+L

        # Power Spectrum of a signal with noise
        Pw = psd(signal_padded[first:last])
        # Power Spectrum of a signal without noise
        Pwo = Pw - Pu

        # zero of negative values
        for i in range(len(Pwo)):
            if Pwo[i] < 0:
                Pwo[i] = 0

        H = Pwo/(Pwo+Pu)
        # the imaginary part may be discarded
        S = np.fft.fft(signal_padded[first:last])
        signal_wiener[first:last] = np.fft.ifft(H*S)

    return signal_wiener
```

In [107]:

```
# Power Spectrum of noise
Pu = [psd(signals[0][4000:4400]), psd(signals[1][4000:4400])]
signals_wiener = [wiener_filter(signals[0],Pu[0]), wiener_filter(signals[1],Pu[1])]

# STFT with Wiener
signals_wiener_stft = [lib.stft(signals_wiener[0],hop_length=epik_freq,win_length=length_freq),lib.stft(signals_w
iener[1],hop_length=epik_freq,win_length=length_freq)]
[r,c] = np.shape(signals_wiener_stft[0])

# time
t = np.linspace(0, 600, c)
# frequency
f = np.linspace(0, fs/2, r)
```

```
C:\Users\Chris Tsoufis\anaconda3\lib\site-packages\ipykernel_launcher.py:28: ComplexWarning: Casting
complex values to real discards the imaginary part
```

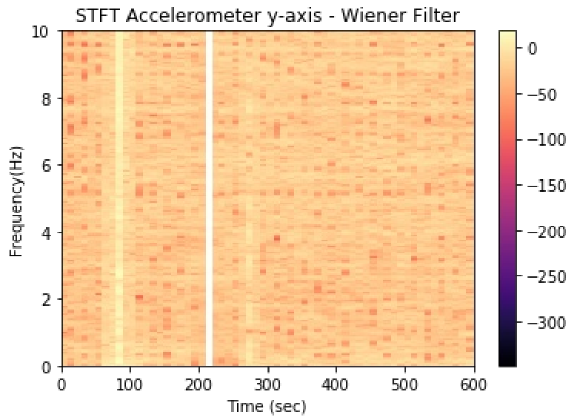
In [108]:

```
# Spectrum Depiction

plt.pcolormesh(t,f,20*np.log10(abs(signals_wiener_stft[0])),cmap='magma')
plt.colorbar()
plt.title("STFT Accelerometer y-axis - Wiener Filter")
plt.xlabel('Time (sec)')
plt.ylabel('Frequency(Hz)')
```

```
C:\Users\Chris Tsoufis\anaconda3\lib\site-packages\ipykernel_launcher.py:3: RuntimeWarning: divide b
y zero encountered in log10
  This is separate from the ipykernel package so we can avoid doing imports until
```

Out[108]:

```
Text(0, 0.5, 'Frequency(Hz)')
```



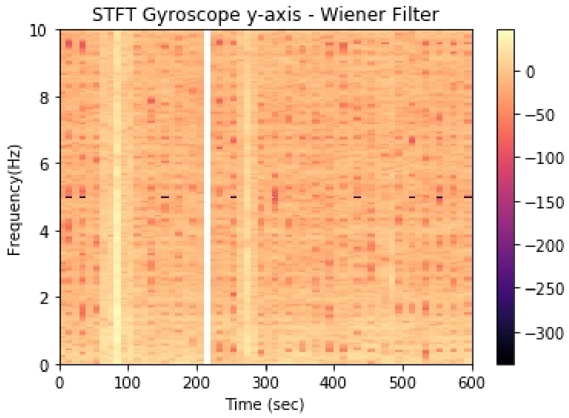
In [109]:

```
plt.pcolormesh(t,f,20*np.log10(abs(signals_wiener_stft[1])),cmap='magma')
plt.colorbar()
plt.title("STFT Gyroscope y-axis - Wiener Filter")
plt.xlabel('Time (sec)')
plt.ylabel('Frequency(Hz)')
```

```
C:\Users\Chris Tsoufis\anaconda3\lib\site-packages\ipykernel_launcher.py:1: RuntimeWarning: divide b
y zero encountered in log10
  """Entry point for launching an IPython kernel.
```

Out[109]:

```
Text(0, 0.5, 'Frequency(Hz)')
```

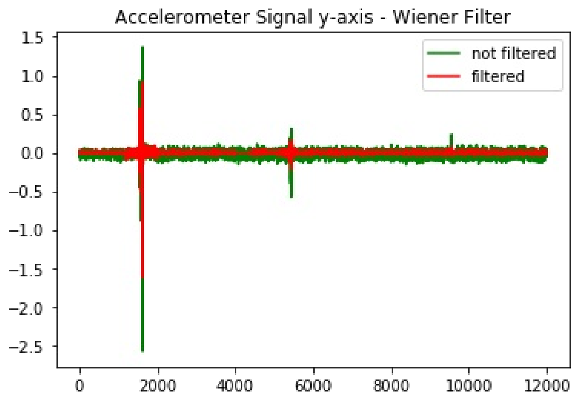


In [110]:

```
# Signal Depiction

plt.plot(signals[0], color='g')
plt.plot(signals_wiener[0], color='r')
plt.title("Accelerometer Signal y-axis - Wiener Filter")
plt.legend(['not filtered','filtered'])
```

Out[110]:

```
<matplotlib.legend.Legend at 0x18bc3144e48>
```

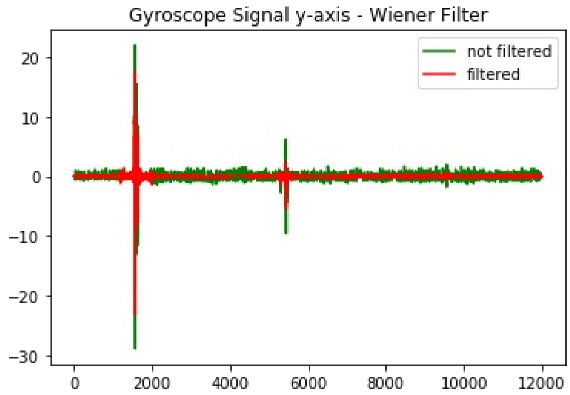


In [111]:

```
plt.plot(signals[1], color='g')
plt.plot(signals_wiener[1], color='r')
plt.title("Gyroscope Signal y-axis - Wiener Filter")
plt.legend(['not filtered','filtered'])
```

Out[111]:

```
<matplotlib.legend.Legend at 0x18bda0dc188>
```



In [112]:

```
# Observation:

# Comparing to the Butterworth Filter, it is observed that with the use of
# the Wiener Filter, better decontamination is achieved. In this case, the
# 'special' values (the purple spots) have been significantly reduced.
```

In [113]:

```
# Part 4.4
```

In [114]:

```
# the following signals have been chosen randomly

signals_names = ['sleep_05.npz','sleep_12.npz','step_07.npz','step_13.npz']
signals = [[],[],[],[]]

for i in range(4):
    signal = np.load(signals_names[i])
    signals[i].append(signal['acc'][:,2])
    signals[i].append(signal['gyr'][:,2])
```

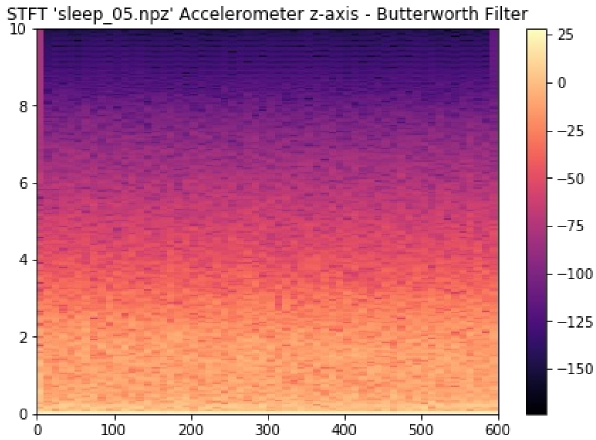
In [115]:

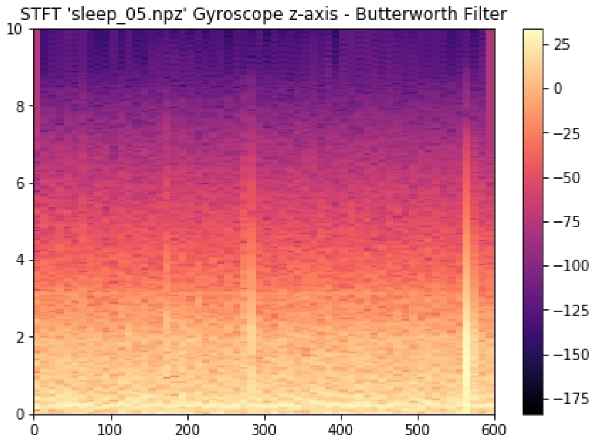
```
Pu = [[psd(signals[0][0][4000:4400]), psd(signals[0][1][4000:4400])], \
      [psd(signals[1][0][5600:6000]), psd(signals[1][1][5600:6000])], \
      [psd(signals[2][0][5700:6100]), psd(signals[2][1][5700:6100])], \
      [psd(signals[3][0][2600:3000]), psd(signals[3][1][2600:3000])]]

signals_butter = [[],[],[],[]]
signals_wiener = [[],[],[],[]]

for i in range(4):
    signals_butter[i].append(butterworth(signals[i][0],fs))
    signals_butter[i].append(butterworth(signals[i][1],fs))

    signals_wiener[i].append(wiener_filter(signals[i][0],Pu[i][0]))
    signals_wiener[i].append(wiener_filter(signals[i][1],Pu[i][1]))
```

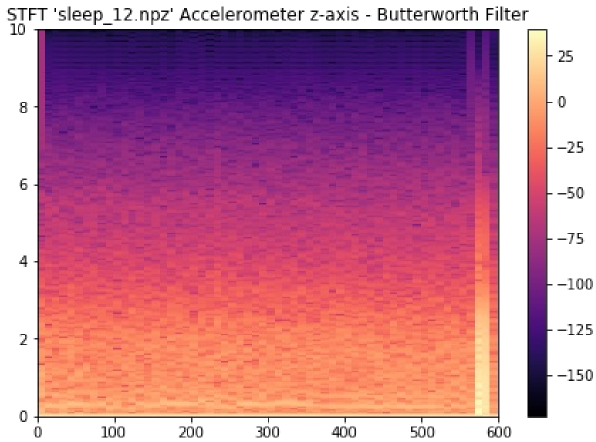
```
C:\Users\Chris Tsoufis\anaconda3\lib\site-packages\ipykernel_launcher.py:28: ComplexWarning: Casting
complex values to real discards the imaginary part
```

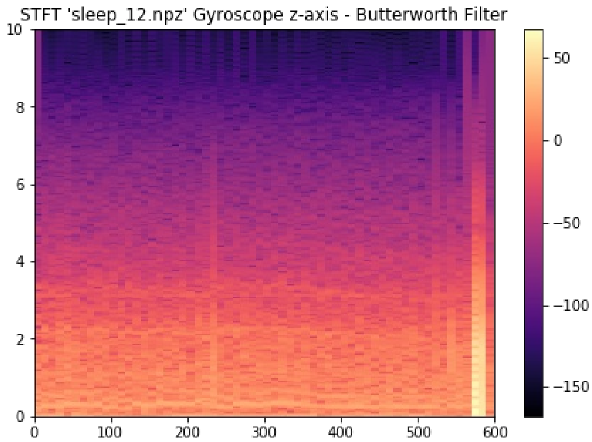
In [116]:

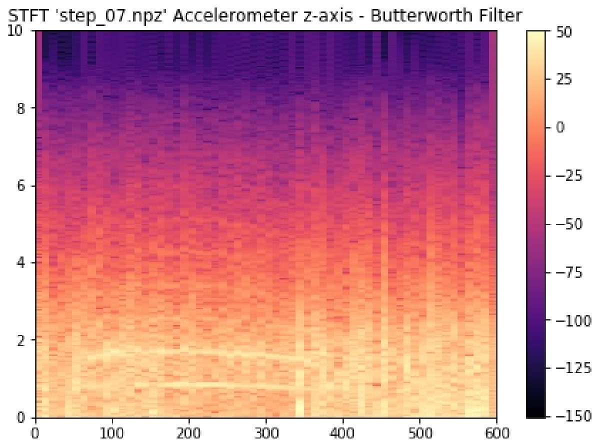
```python
# Butterworth

for i in range(4):

    # Spectrum Depiction
    signals_butter_stft = [lib.stft(signals_butter[i][0],hop_length=epik_freq,win_length=length_freq), \
                           lib.stft(signals_butter[i][1],hop_length=epik_freq,win_length=length_freq)]

    plt.figure(figsize=(16,5))
    plt.subplot(1,2,1)
    plt.pcolormesh(t,f,20*np.log10(abs(signals_butter_stft[0])),cmap='magma')
    plt.colorbar()
    plt.title("STFT '" + signals_names[i] + "' Accelerometer z-axis - Butterworth Filter")

    plt.subplot(1,2,2)
    plt.pcolormesh(t,f,20*np.log10(abs(signals_butter_stft[1])),cmap='magma')
    plt.colorbar()
    plt.title("STFT '" + signals_names[i] + "' Gyroscope z-axis - Butterworth Filter")
```

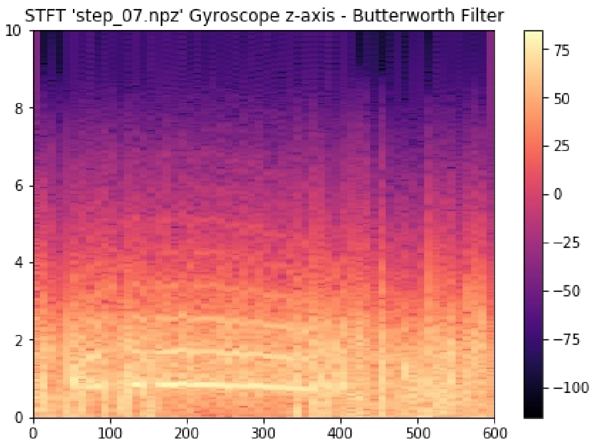

STFT 'sleep_05.npz' Accelerometer z-axis - Butterworth Filter

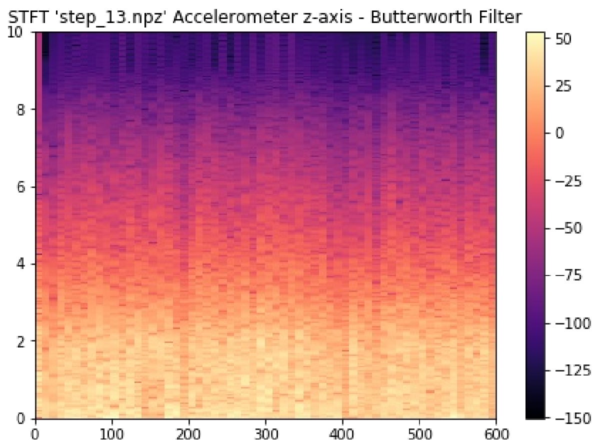

STFT 'sleep_05.npz' Gyroscope z-axis - Butterworth Filter

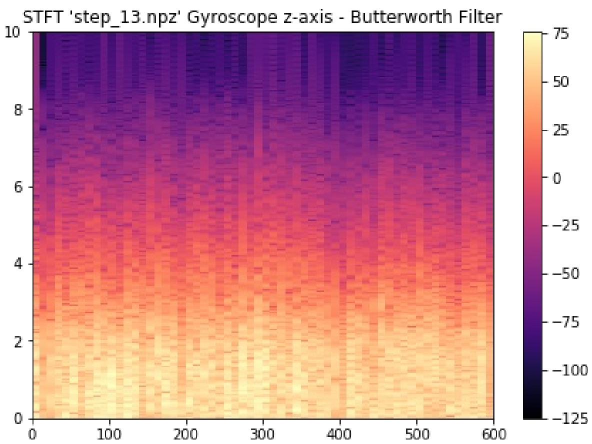

STFT 'sleep_12.npz' Accelerometer z-axis - Butterworth Filter


STFT 'sleep_12.npz' Gyroscope z-axis - Butterworth Filter


STFT 'step_07.npz' Accelerometer z-axis - Butterworth Filter

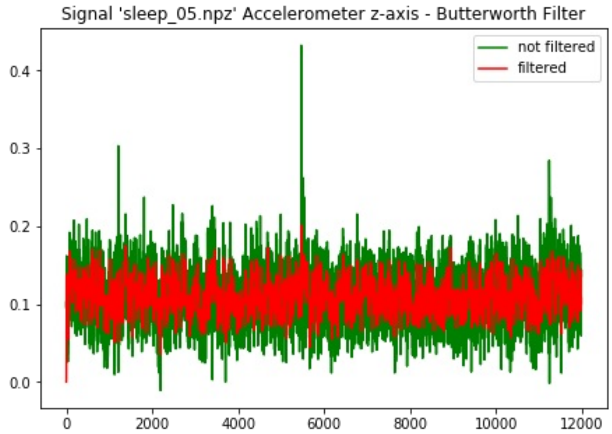

STFT 'step_07.npz' Gyroscope z-axis - Butterworth Filter

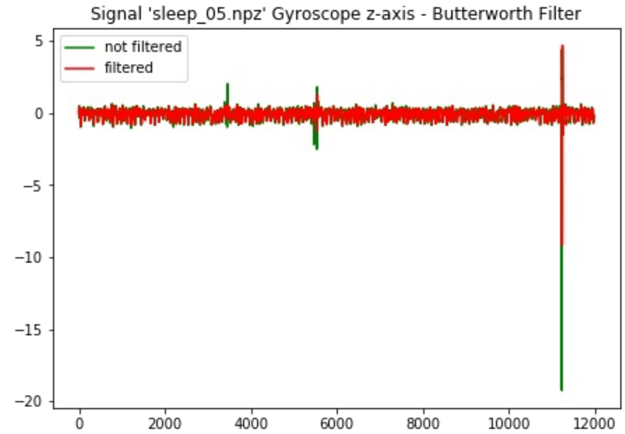

STFT 'step_13.npz' Accelerometer z-axis - Butterworth Filter

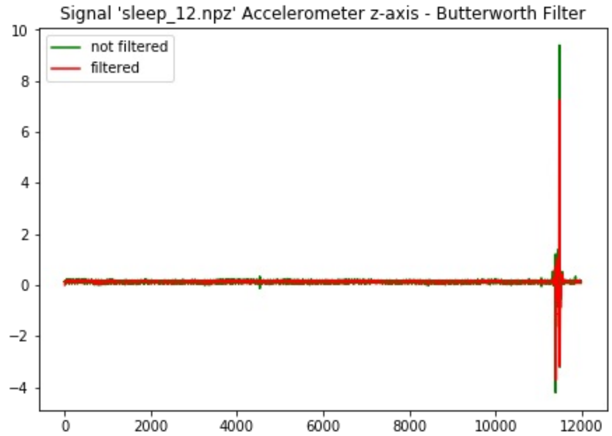

STFT 'step_13.npz' Gyroscope z-axis - Butterworth Filter

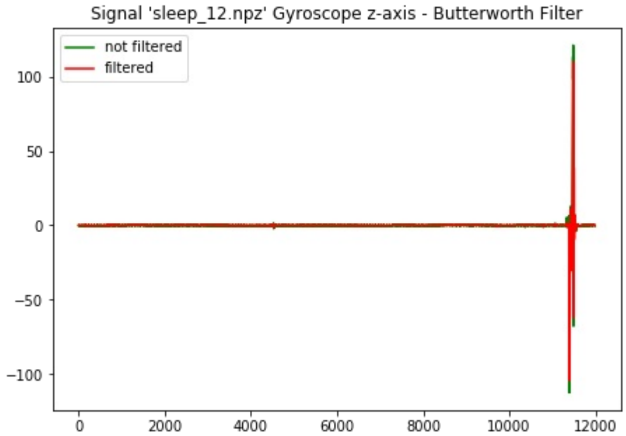
In [117]:

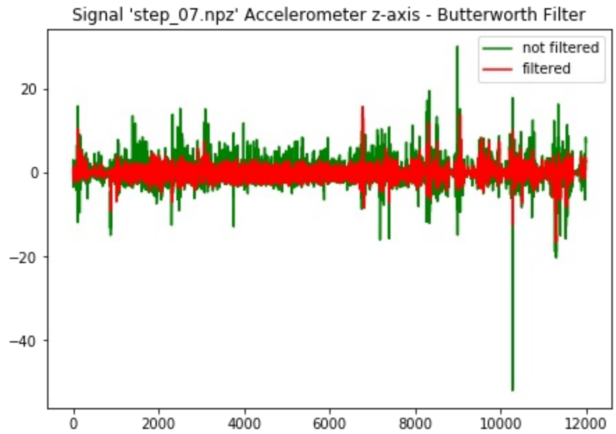
```
# Butterworth

for i in range(4):

    # Signal Depiction
    plt.figure(figsize=(16,5))
    plt.subplot(1,2,1)
    plt.plot(signals[i][0], color='g')
    plt.plot(signals_butter[i][0], color='r')
    plt.title("Signal '" + signals_names[i] + "' Accelerometer z-axis - Butterworth Filter")
    plt.legend(['not filtered','filtered'])

    plt.subplot(1,2,2)
    plt.plot(signals[i][1], color='g')
    plt.plot(signals_butter[i][1], color='r')
    plt.title("Signal '" + signals_names[i] + "' Gyroscope z-axis - Butterworth Filter")
    plt.legend(['not filtered','filtered'])
```

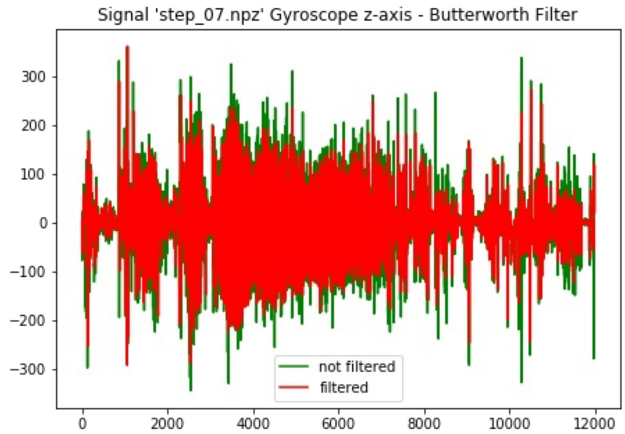

Signal 'sleep_05.npz' Accelerometer z-axis - Butterworth Filter
not filtered
filtered

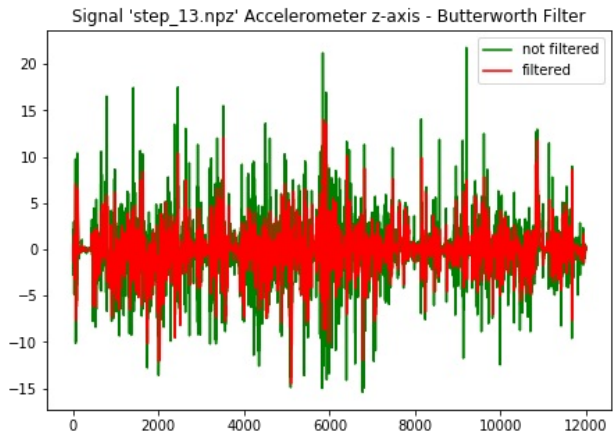

Signal 'sleep_05.npz' Gyroscope z-axis - Butterworth Filter
not filtered
filtered

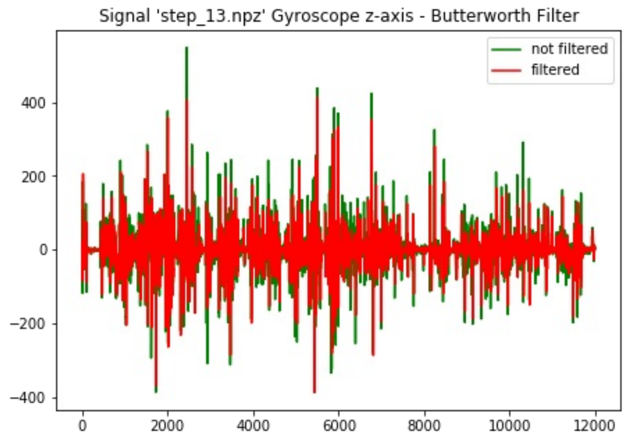

Signal 'sleep_12.npz' Accelerometer z-axis - Butterworth Filter
not filtered
filtered


Signal 'sleep_12.npz' Gyroscope z-axis - Butterworth Filter
not filtered
filtered


Signal 'step_07.npz' Accelerometer z-axis - Butterworth Filter
not filtered
filtered


Signal 'step_07.npz' Gyroscope z-axis - Butterworth Filter
not filtered
filtered

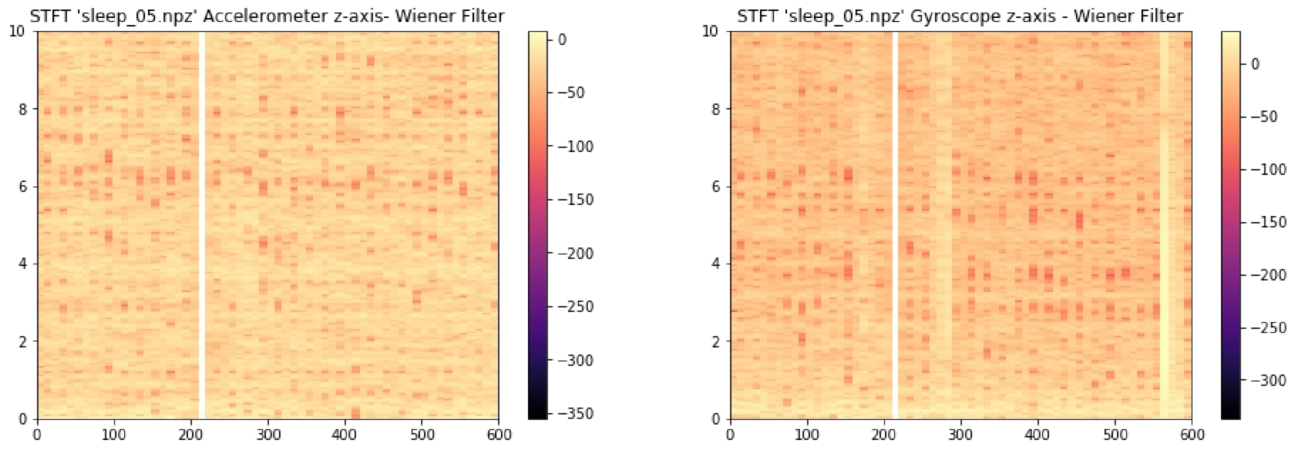

Signal 'step_13.npz' Accelerometer z-axis - Butterworth Filter
not filtered
filtered

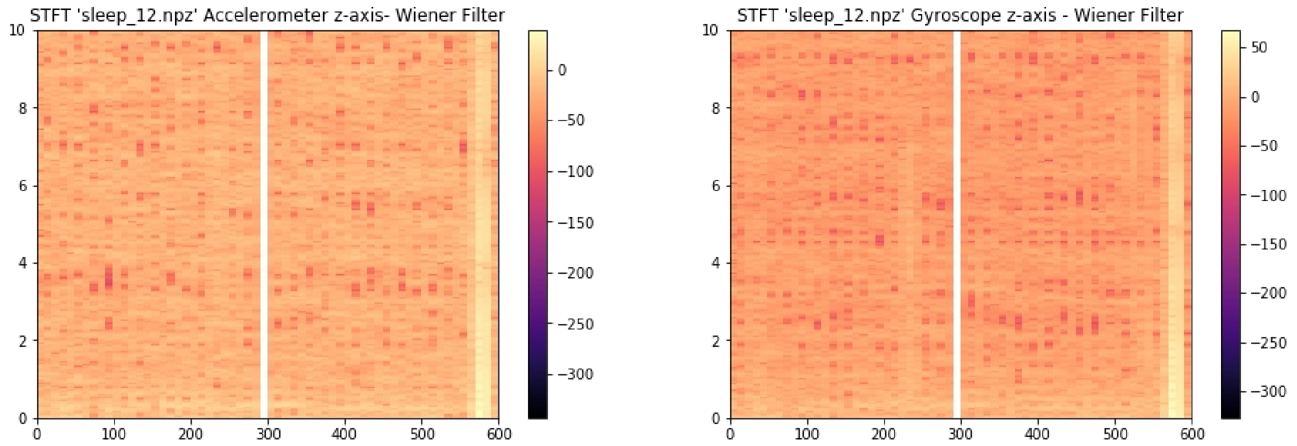

Signal 'step_13.npz' Gyroscope z-axis - Butterworth Filter
not filtered
filtered

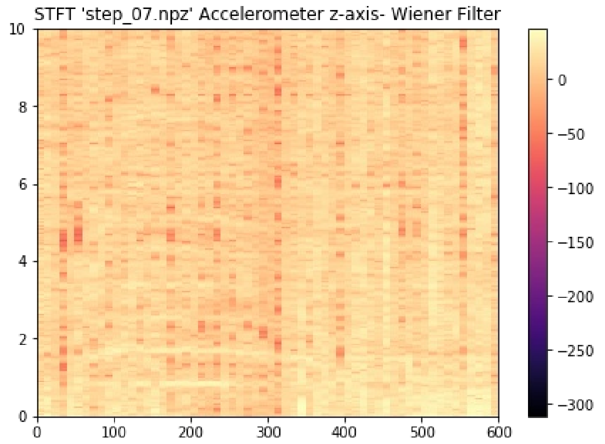
In [118]:

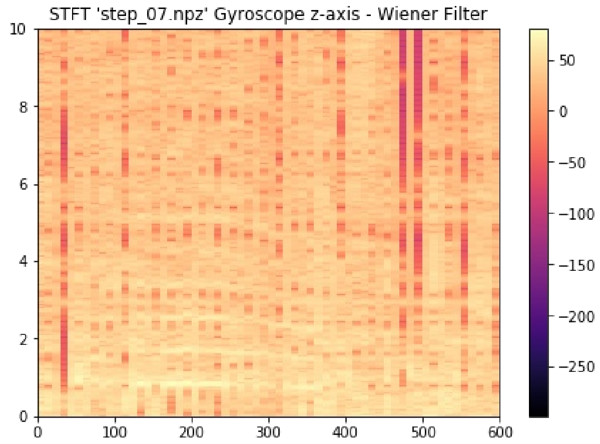
```python
# Wiener

for i in range(4):

    # Spectrum Depiction

    t = np.linspace(0, 600, c)
    f = np.linspace(0, fs/2, r)
    [r,c] = np.shape(signals_wiener_stft[0])

    signals_wiener_stft = [lib.stft(signals_wiener[i][0],hop_length=epik_freq,win_length=length_freq), \
                           lib.stft(signals_wiener[i][1],hop_length=epik_freq,win_length=length_freq)]

    plt.figure(figsize=(16,5))
    plt.subplot(1,2,1)
    plt.pcolormesh(t,f,20*np.log10(abs(signals_wiener_stft[0])),cmap='magma')
    plt.colorbar()
    plt.title("STFT '" + signals_names[i] + "' Accelerometer z-axis- Wiener Filter")

    plt.subplot(1,2,2)
    plt.pcolormesh(t,f,20*np.log10(abs(signals_wiener_stft[1])),cmap='magma')
    plt.colorbar()
    plt.title("STFT '" + signals_names[i] + "' Gyroscope z-axis - Wiener Filter")
```

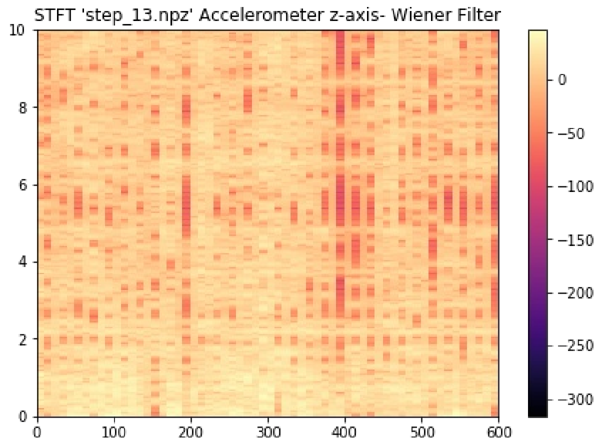
```
C:\Users\Chris Tsoufis\anaconda3\lib\site-packages\ipykernel_launcher.py:16: RuntimeWarning: divide
by zero encountered in log10
  app.launch_new_instance()
C:\Users\Chris Tsoufis\anaconda3\lib\site-packages\ipykernel_launcher.py:21: RuntimeWarning: divide
by zero encountered in log10
```

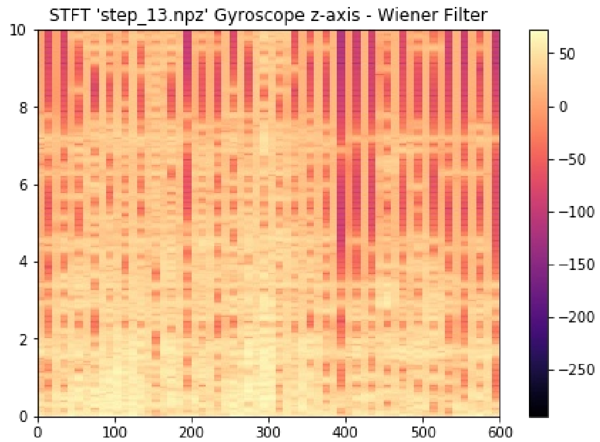






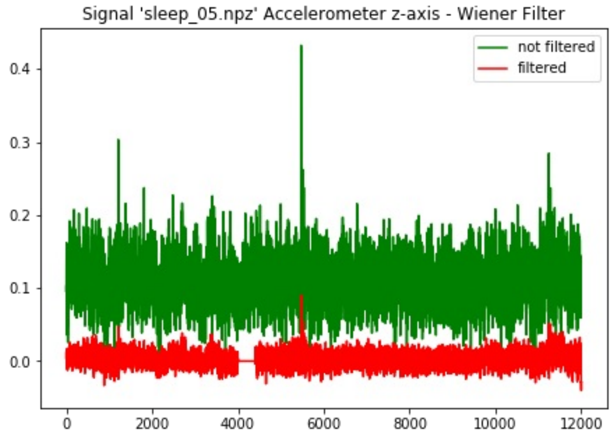


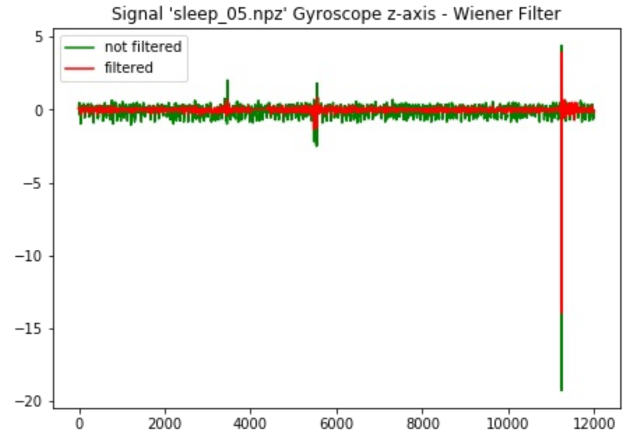


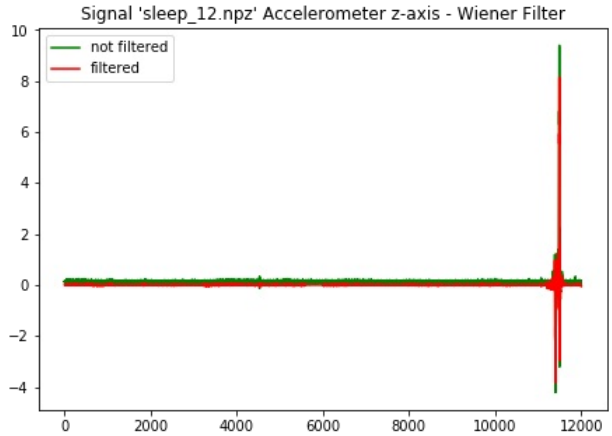


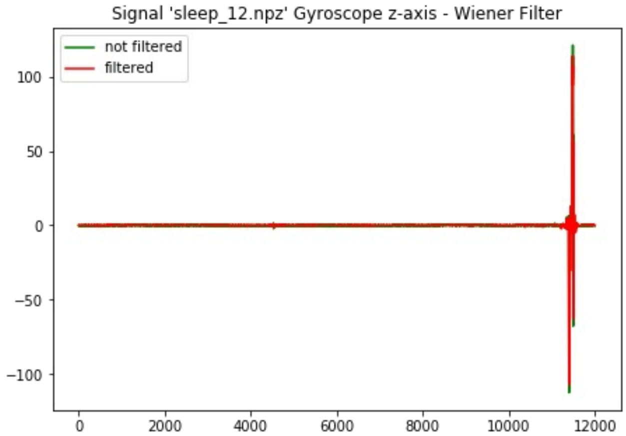
In [119]:

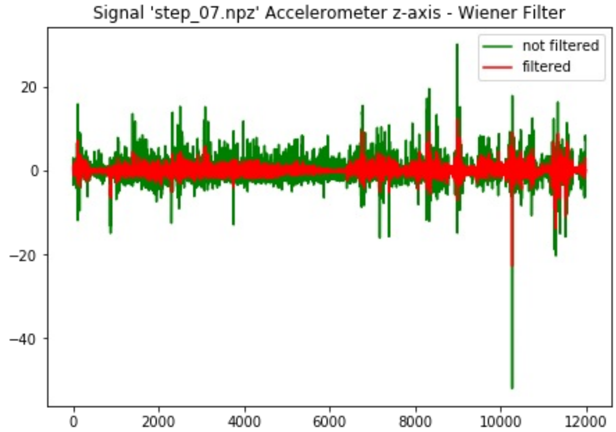
```
# Wiener

for i in range(4):

    # Spectrum Depiction

    plt.figure(figsize=(16,5))
    plt.subplot(1,2,1)
    plt.plot(signals[i][0], color='g')
    plt.plot(signals_wiener[i][0], color='r')
    plt.title("Signal '" + signals_names[i] + "' Accelerometer z-axis - Wiener Filter")
    plt.legend(['not filtered','filtered'])

    plt.subplot(1,2,2)
    plt.plot(signals[i][1], color='g')
    plt.plot(signals_wiener[i][1], color='r')
    plt.title("Signal '" + signals_names[i] + "' Gyroscope z-axis - Wiener Filter")
    plt.legend(['not filtered','filtered'])
```

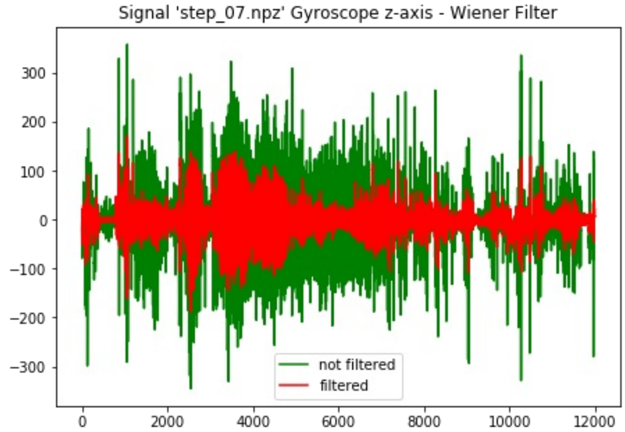


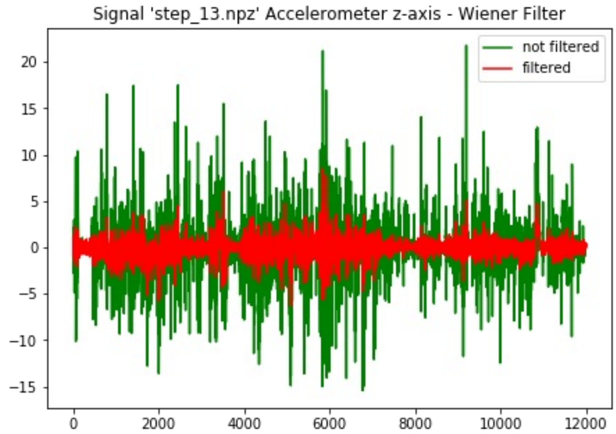


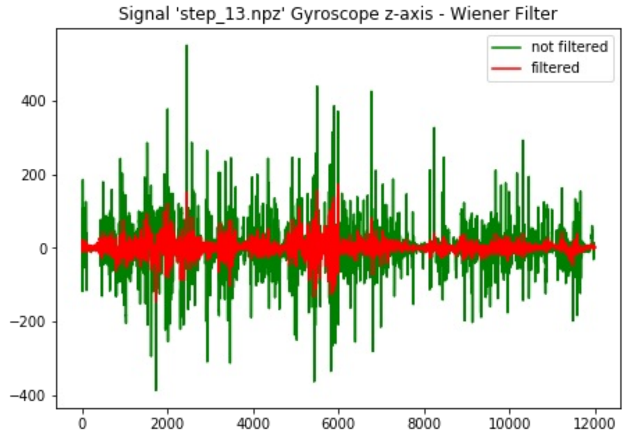












In [120]:

```
# Observation:

# The decontamination for the chosen signals is better with the Wiener Filter.
```