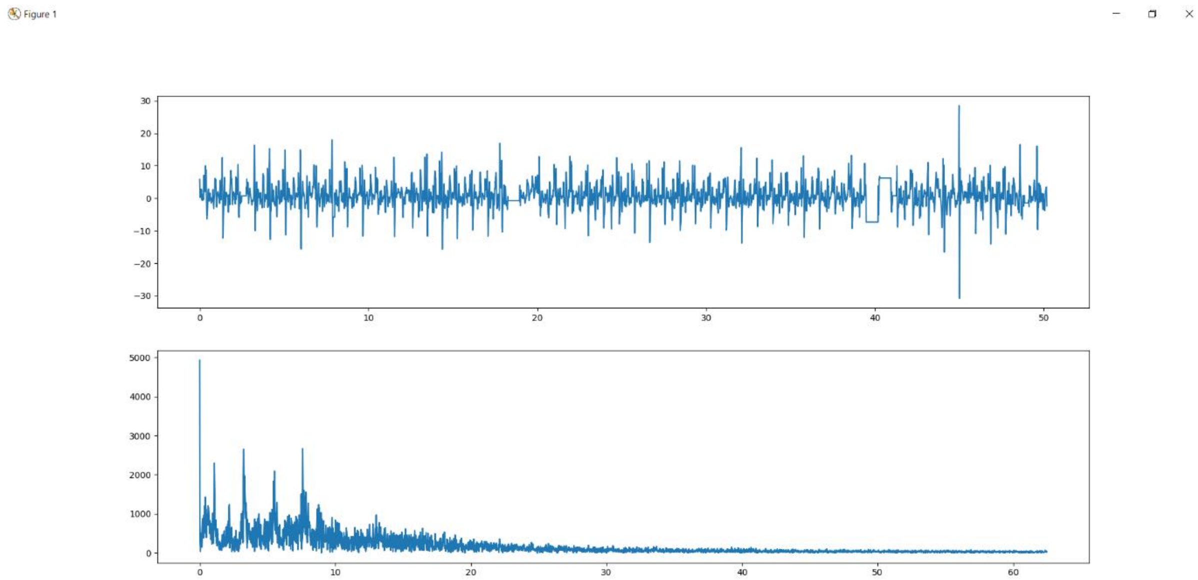

Figure 1: x-direction magnitude by time plot and corresponding Fourier transform

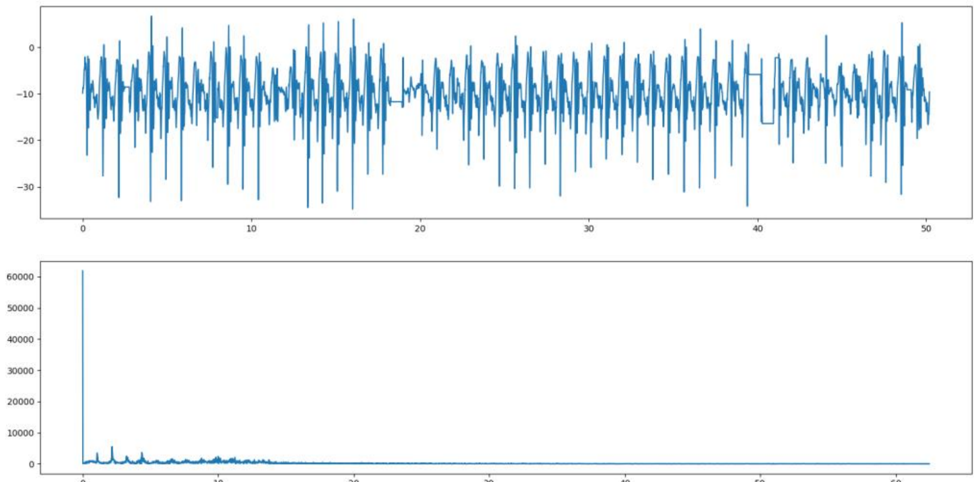

Figure 2: y-direction magnitude by time plot and corresponding Fourier transform

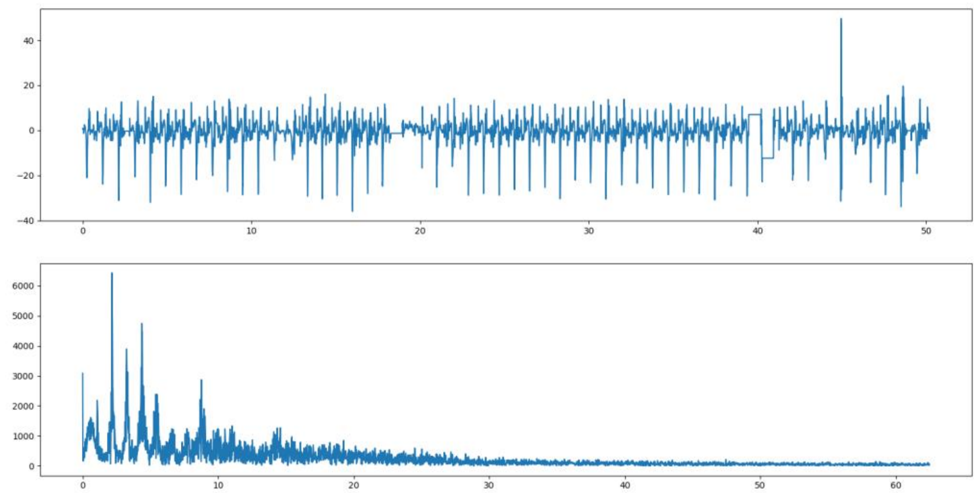

Figure 3: z-direction magnitude by time plot and corresponding Fourier transform

```python
import numpy as np
import scipy as sy
import scipy.fftpack as syfp
import matplotlib.pyplot as plt

array = np.loadtxt("D:\\Desktop\\accelxyz.csv", delimiter=',')
column_num = 2  #0 means x-axis, 1 means y-axis and 2 means z-axis

length = len(array[:,column_num]) #Number of data points
x = sy.linspace(0.005, length*0.008, num=length) #Return evenly spaced numbers as x-axis values

yf = syfp.fft(array[:,column_num]) #Discrete Fourier transform of array
f = syfp.fftfreq(length, np.mean(np.diff(x))) #Return the Discrete Fourier Transform sample frequencies

plt.subplot(211) #Create signal magnitude by time plot
plt.plot(x, array[:,column_num]) #Plot values
plt.subplot(212) #Create magnitude by frequency plot
plt.plot(abs(f), abs(yf)) #Plot values
plt.show()
```

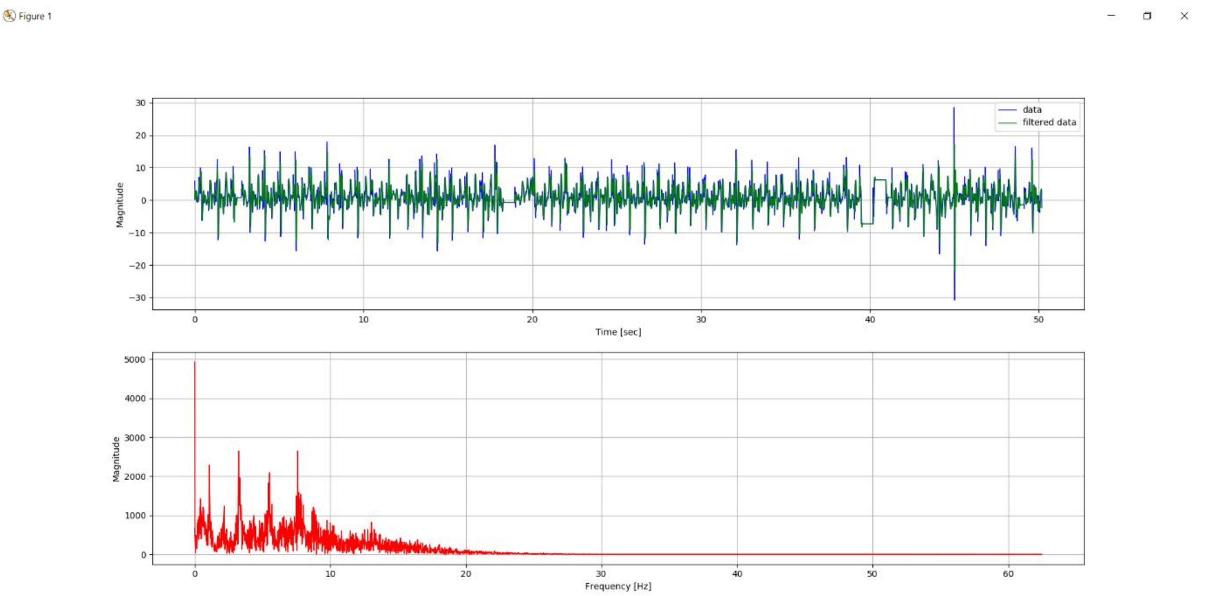

Figure 4: Original and filtered sensor data in the x-direction along with Fourier transform of filtered sensor data

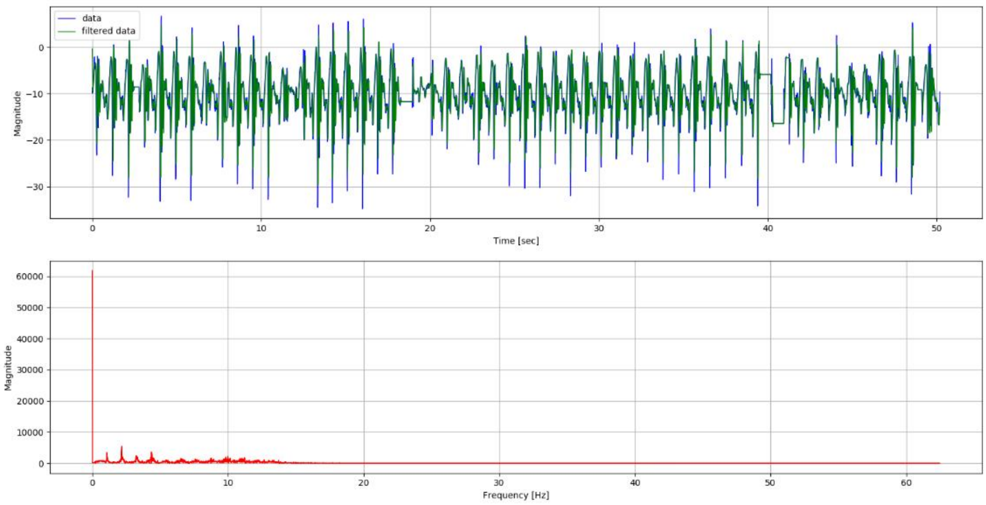

Figure 5: Original and filtered sensor data in the y-direction along with Fourier transform of filtered sensor data

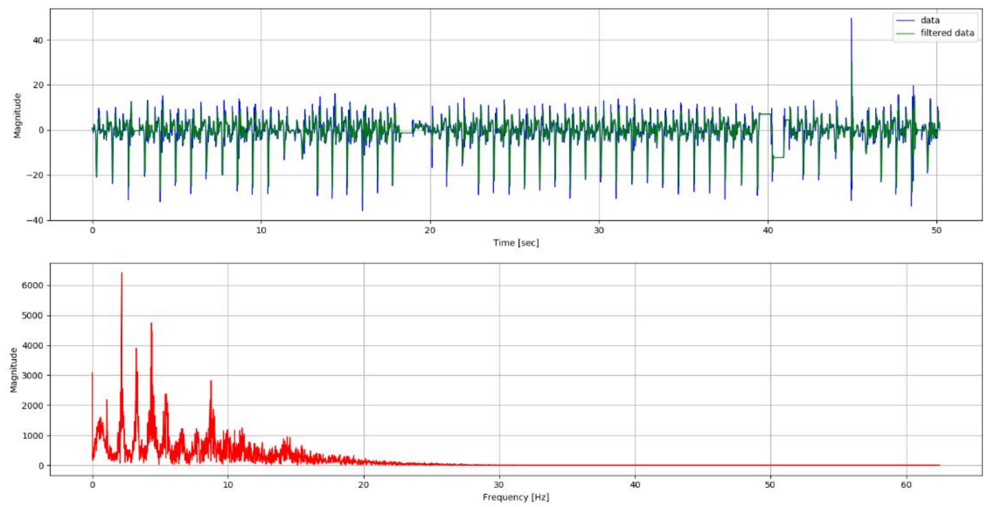

Figure 6: Original and filtered sensor data in the z-direction along with Fourier transform of filtered sensor data

```python
import numpy as np
import scipy as sy
import scipy.fftpack as syfp
import matplotlib.pyplot as plt
from scipy.signal import butter, lfilter, freqz

def butter_lowpass(cutoff, fs, order=3):
    nyq = 0.5 * fs
    normal_cutoff = cutoff / nyq
    b, a = butter(order, normal_cutoff, btype='low', analog=False)
    return b, a

def butter_lowpass_filter(data, cutoff, fs, order=3):
    b, a = butter_lowpass(cutoff, fs, order=order)
    y = lfilter(b, a, data)
    return y

array = np.loadtxt("D:\\Desktop\\accelxyz.csv", delimiter=',')
column_num = 0 #0 means x-axis, 1 means y-axis and 2 means z-axis

order = 3 # filter order
fs = 125.0 # sample rate, Hz
cutoff = 15 # cut-off frequency

length = len(array[:,column_num]) #Number of data points
x = sy.linspace(0.005, length*0.008, num=length) #Return evenly spaced numbers as x-axis values

y = butter_lowpass_filter(array[:,column_num], cutoff, fs, order)

plt.subplot(2, 1, 1)
plt.plot(x, array[:,column_num], 'b-', linewidth=1, label='data')
plt.plot(x, y, 'g-', linewidth=1, label='filtered data')
plt.xlabel('Time [sec]')
plt.ylabel('Magnitude')
plt.grid()
plt.legend()

yf = syfp.fft(y) #Discrete Fourier transform of array
f = syfp.fftfreq(length, np.mean(np.diff(x))) #Return the Discrete Fourier Transform sample frequencies

plt.subplot(212) #Create magnitude by frequency plot
plt.plot(abs(f), abs(yf), 'r-', linewidth=1) #Plot values
plt.xlabel('Frequency [Hz]')
plt.ylabel('Magnitude')
plt.grid()
plt.show()
```

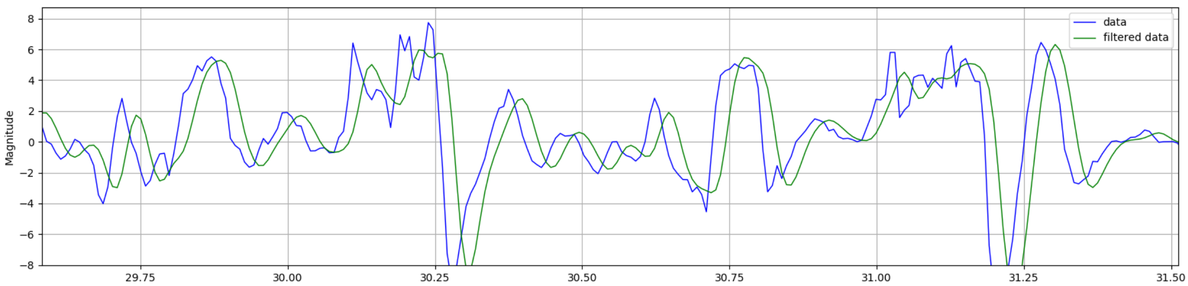

Figure 7: Closer look at raw data and filtered data