In [ ]:

```
%pip install pandas numpy matplotlib scipy
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.fft import fft, rfft, fftfreq, rfftfreq
import os, re
# os.chdir("/home")
print(os.getcwd())
```

```
Requirement already satisfied: pandas in ./.venv/lib/python3.11/site-packages (2.2.2)
Requirement already satisfied: numpy in ./.venv/lib/python3.11/site-packages (1.26.4)
Requirement already satisfied: matplotlib in ./.venv/lib/python3.11/site-packages (3.9.0)
Requirement already satisfied: scipy in ./.venv/lib/python3.11/site-packages (1.13.1)
Requirement already satisfied: python-dateutil>=2.8.2 in ./.venv/lib/python3.11/site-packages (from pandas) (2.9
.0.post0)
Requirement already satisfied: pytz>=2020.1 in ./.venv/lib/python3.11/site-packages (from pandas) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in ./.venv/lib/python3.11/site-packages (from pandas) (2024.1)
Requirement already satisfied: contourpy>=1.0.1 in ./.venv/lib/python3.11/site-packages (from matplotlib) (1.2.1
)
Requirement already satisfied: cycler>=0.10 in ./.venv/lib/python3.11/site-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in ./.venv/lib/python3.11/site-packages (from matplotlib) (4.52
.1)
Requirement already satisfied: kiwisolver>=1.3.1 in ./.venv/lib/python3.11/site-packages (from matplotlib) (1.4.
5)
Requirement already satisfied: packaging>=20.0 in ./.venv/lib/python3.11/site-packages (from matplotlib) (24.0)
Requirement already satisfied: pillow>=8 in ./.venv/lib/python3.11/site-packages (from matplotlib) (10.3.0)
Requirement already satisfied: pyparsing>=2.3.1 in ./.venv/lib/python3.11/site-packages (from matplotlib) (3.1.2
)
Requirement already satisfied: six>=1.5 in ./.venv/lib/python3.11/site-packages (from python-dateutil>=2.8.2->pa
ndas) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
/home/bb/drones_prediction
```

Reading data from three different flights (with the first file containing concatenated data from two flights)

In [ ]:

```
data_15 = pd.read_csv('data/gyro_acc_lot_03_15.csv')
data_35 = pd.read_csv('data/gryo_acc_lot_10_35.csv')
```

In [ ]:

```
data_15.head()
```

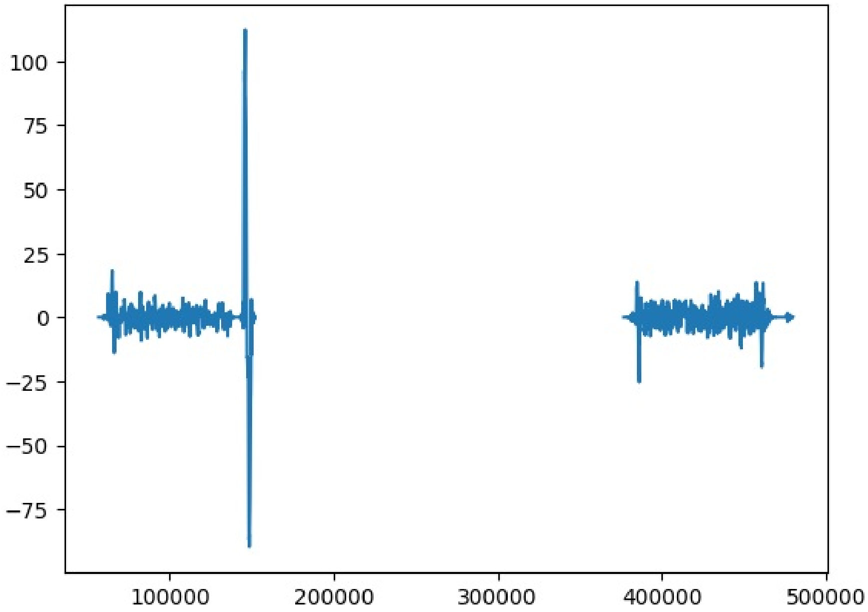
Out[ ]:

	timestamp(ms)	degrees(IMU[0].GyrX)	degrees(IMU[0].GyrY)	degrees(IMU[0].GyrZ)	IMU[0].AccX	IMU[0].AccY
0	56995.851	-0.032568	-0.106268	0.069404	-0.215246	0.459727
1	57095.851	-0.050129	-0.080863	0.047958	-0.218031	0.458423
2	57195.851	-0.039158	-0.043306	0.006790	-0.217225	0.457481
3	57295.851	-0.034236	-0.098692	0.028611	-0.215400	0.461080
4	57395.851	-0.060214	-0.060917	0.029124	-0.221406	0.463030

Verifying the obtained data; this file contains data from a flight with healthy rotor blades and one with a 1.5 cm damage.

In [ ]:

```
plt.plot(data_15['timestamp(ms)'],data_15['degrees(IMU[0].GyrZ)'])
plt.show()
```



In [ ]:
```
def get_first_last_none_index(df, column_name):
    column = df[column_name]
    first_none_index = column.index[column.isnull()].min()
    last_none_index = column.index[column.isnull()].max()
    return first_none_index, last_none_index

def split_array(array, segment_size, overlap):
    splitted_segments = []
    i = 0
    while i < len(array):
        if len(array[i:i+segment_size]) == segment_size:
            splitted_segments.append(array[i:i+segment_size])
        i += segment_size - overlap
    return splitted_segments

def split_aggregation(array,first_segment,agg):
    splitted_segments = []
    i = 0
    while first_segment+agg*i < len(array):
        splitted_segments.append(array[:first_segment+agg*i])
        i += 1
    return splitted_segments

def get_acc(arr1,arr2,arr3):
    sum_of_squares = np.square(arr1) + np.square(arr2) + np.square(arr3)
    result = np.sqrt(sum_of_squares)
    return result

def normalization(data):
    min_data = min(data)
    diff_data = max(data) - min_data
    return [(x-min_data)/diff_data for x in data]

def FFT_data(data_array,df=True):
    data_dict = {}
    signal_size = 500
    sample_rate = 1000
    skip_unit_jump = 10 #on the first values is noise as unit jump, so we skip them
    fft_freq = rfftfreq(signal_size, d=1/sample_rate)

    data_dict['Frequencies'] = fft_freq[skip_unit_jump:]
    for index,data in enumerate(data_array):
        fft_val = abs(rfft(data,n=signal_size))
        data_dict[f'Sample {index+1}'] = normalization(fft_val[skip_unit_jump:])
    if df:
        return pd.DataFrame.from_dict(data_dict)
    else:
        return data_dict
```

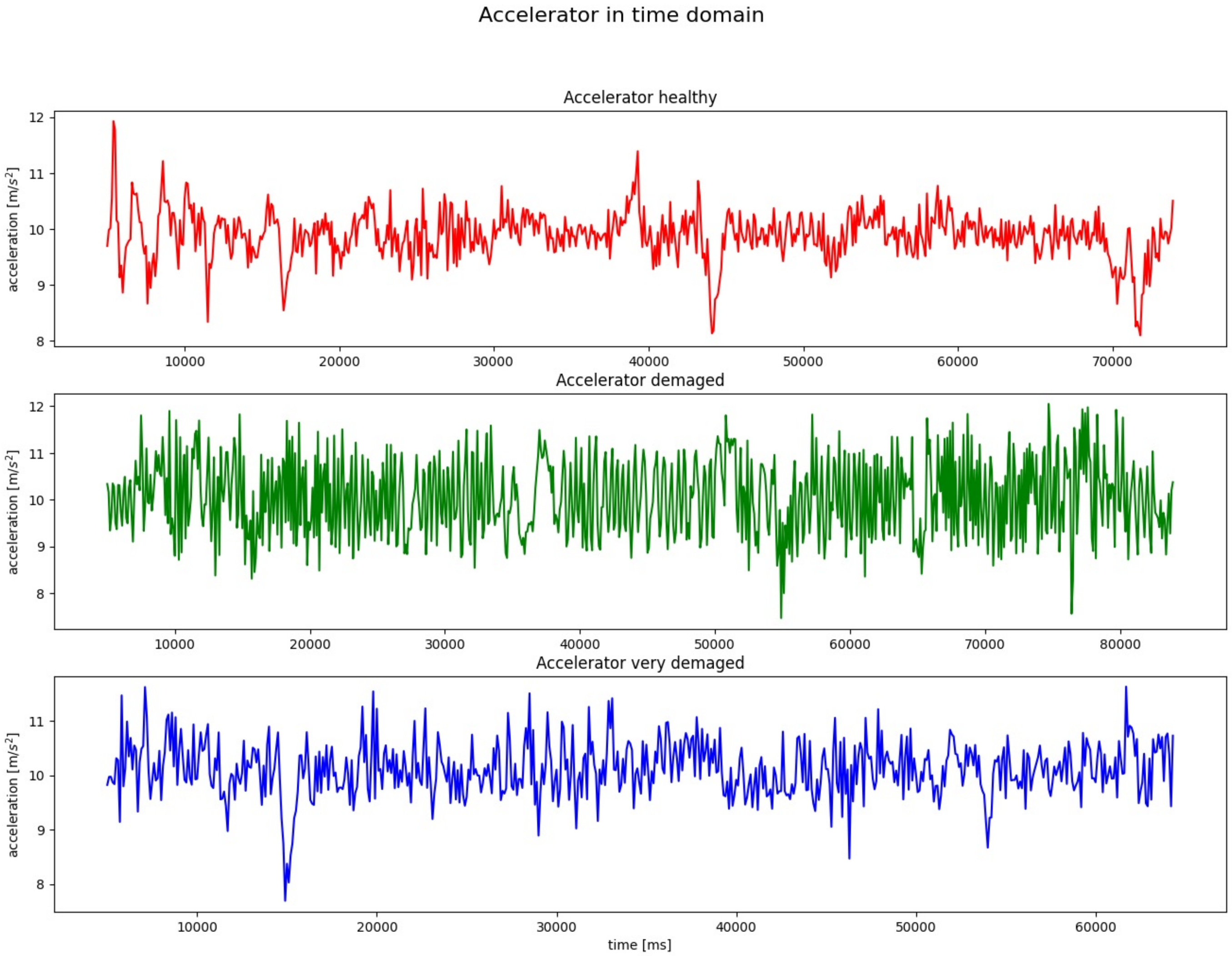
In [ ]:
```
# Get the first and last indices for data_15
first, second = get_first_last_none_index(data_15, 'IMU[0].AccY')

# Calculate time for "healthy" data
time_healthy = data_15['timestamp(ms)'].values[:first] - min(data_15['timestamp(ms)'].values[:first])

# Calculate time for "damaged" data
```

```python
time_damaged = data_15['timestamp(ms)'].values[second + 1:] - min(data_15['timestamp(ms)'].values[second + 1:])

# Calculate time for "very damaged" data
time_very_damaged = data_35['timestamp(ms)'].values - min(data_35['timestamp(ms)'].values)

# Get accelerometer data for "healthy" state
acc_X_healthy = data_15['IMU[0].AccX'].values[:first]
acc_Y_healthy = data_15['IMU[0].AccY'].values[:first]
acc_Z_healthy = data_15['IMU[0].AccZ'].values[:first]
acc_healthy = get_acc(acc_X_healthy, acc_Y_healthy, acc_Z_healthy)  # 50 210

# Get gyroscope data for "healthy" state
gyro_X_healthy = data_15['degrees(IMU[0].GyrX)'].values[:first]
gyro_Y_healthy = data_15['degrees(IMU[0].GyrY)'].values[:first]
gyro_Z_healthy = data_15['degrees(IMU[0].GyrZ)'].values[:first]
gyro_healthy = get_acc(gyro_X_healthy, gyro_Y_healthy, gyro_Z_healthy)  # 50 210

# Get accelerometer data for "damaged" state
acc_X_damaged = data_15['IMU[0].AccX'].values[second + 1:]
acc_Y_damaged = data_15['IMU[0].AccY'].values[second + 1:]
acc_Z_damaged = data_15['IMU[0].AccZ'].values[second + 1:]
acc_damaged = get_acc(acc_X_damaged, acc_Y_damaged, acc_Z_damaged)  # 50 190

# Get gyroscope data for "damaged" state
gyro_X_damaged = data_15['degrees(IMU[0].GyrX)'].values[second + 1:]
gyro_Y_damaged = data_15['degrees(IMU[0].GyrY)'].values[second + 1:]
gyro_Z_damaged = data_15['degrees(IMU[0].GyrZ)'].values[second + 1:]
gyro_damaged = get_acc(gyro_X_damaged, gyro_Y_damaged, gyro_Z_damaged)  # 50 190

# Get accelerometer data for "very damaged" state
acc_X_very_damaged = data_35['IMU[0].AccX'].values
acc_Y_very_damaged = data_35['IMU[0].AccY'].values
acc_Z_very_damaged = data_35['IMU[0].AccZ'].values

# Get gyroscope data for "very damaged" state
gyro_X_very_damaged = data_35['degrees(IMU[0].GyrX)'].values
gyro_Y_very_damaged = data_35['degrees(IMU[0].GyrY)'].values
gyro_Z_very_damaged = data_35['degrees(IMU[0].GyrZ)'].values

# Calculate accelerometer values for "very damaged" state
acc_very_damaged = get_acc(acc_X_very_damaged, acc_Y_very_damaged, acc_Z_very_damaged)  # 5 135

# Calculate gyroscope values for "very damaged" state
gyro_very_damaged = get_acc(gyro_X_very_damaged, gyro_Y_very_damaged, gyro_Z_very_damaged)  # 5 135
```

In [ ]:
```python
start = 50
stop = 150
```

Comparing acceleration signals after drone takeoff with different rotor blade conditions.

In [ ]:
```python
fig,ax = plt.subplots(3,1,figsize=[16,11])


ax[0].set_title('Acceleration')
ax[0].plot(time_healthy[start:len(time_healthy)-210], acc_healthy[start:len(time_healthy)-210], label='Acc_heal
ax[1].plot(time_damaged[start:len(time_damaged)-190], acc_damaged[start:len(time_damaged)-190], label='Acc_damag
ax[2].plot(time_very_damaged[start:len(time_very_damaged)-135], acc_very_damaged[start:len(time_very_damaged)-1

# ax[0].legend()
# ax[1].legend()
# ax[2].legend()
ax[0].set_ylabel(r'acceleration [m/$s^2$]')
ax[1].set_ylabel(r'acceleration [m/$s^2$]')

ax[2].set_ylabel(r'acceleration [m/$s^2$]')
ax[2].set_xlabel('time [ms]')
ax[0].set_title('Accelerator healthy')
ax[1].set_title('Accelerator demaged')
ax[2].set_title('Accelerator very demaged')
fig.suptitle('Accelerator in time domain', fontsize=16)
plt.savefig('photos/data_from_acc_1.png')
plt.show()
```



In [ ]:
```python
acc_healthy_split = split_array(acc_healthy[50:len(acc_healthy)-210],150,125)
acc_damaged_split = split_array(acc_damaged[50:len(acc_damaged)-190],150,125)
acc_very_damaged_split = split_array(acc_very_damaged[50:len(acc_very_damaged)-135],150,125)
```

In [ ]:
```python
columns_h = [f'Sample {x}' for x in range(len(acc_healthy_split))]
columns_d = [f'Sample {x}' for x in range(len(acc_damaged_split))]
columns_vd = [f'Sample {x}' for x in range(len(acc_very_damaged_split))]
```

In [ ]:
```python
data_healthy = pd.DataFrame(acc_healthy_split).T
data_damaged = pd.DataFrame(acc_damaged_split).T
data_very_damaged = pd.DataFrame(acc_very_damaged_split).T
data_healthy.columns = columns_h
data_damaged.columns = columns_d
data_very_damaged.columns = columns_vd
```

In [ ]:
```python
data_healthy.head()
```

Out[ ]:

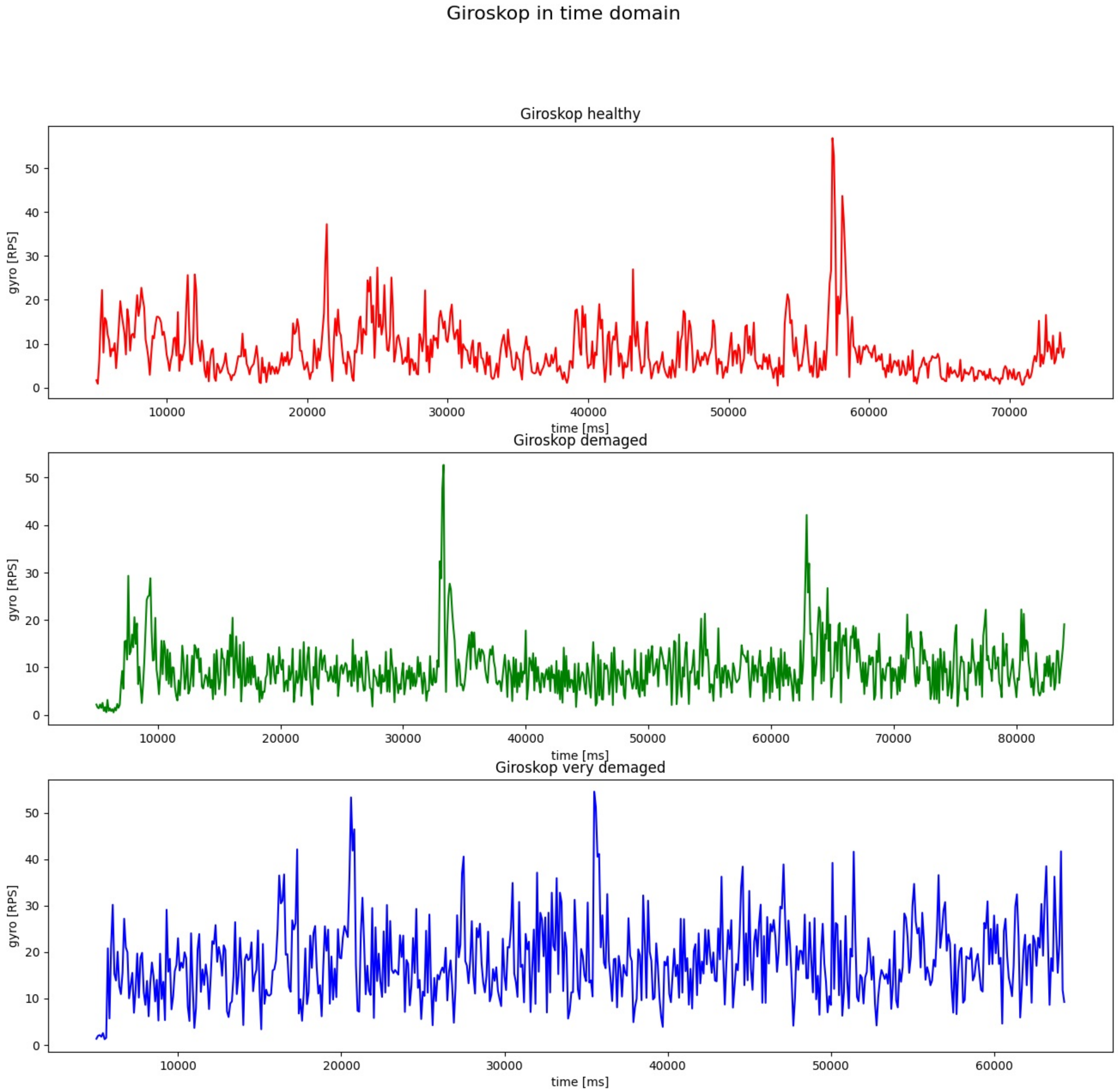
	Sample 0	Sample 1	Sample 2	Sample 3	Sample 4	Sample 5	Sample 6	Sample 7	Sample 8	Sample 9	...
0	9.698080	9.696647	10.539860	10.171881	9.815653	10.082813	9.568929	10.000623	10.185560	10.003679	...
1	9.983599	8.667918	10.829459	10.178237	9.908080	9.509679	9.292522	9.602241	9.526098	9.330222	...
2	10.014586	9.406136	10.811474	9.746929	9.964858	9.806895	9.592371	9.655449	9.946701	10.260179	...
3	10.560446	8.947886	10.365710	10.069560	10.357260	9.984034	9.520640	9.468885	9.174200	9.459368	...
4	11.926492	9.368466	10.433258	9.702129	10.616596	9.949000	9.939603	9.763273	10.721262	10.194759	...

5 rows × 22 columns

Save accelerometer data in the time domain.

In [ ]:
```python
data_healthy.to_csv('data_preprocesed/acc_healthy_samples.csv')
data_damaged.to_csv('data_preprocesed/acc_damaged_samples.csv')
data_very_damaged.to_csv('data_preprocesed/acc_very_damaged_samples.csv')
```

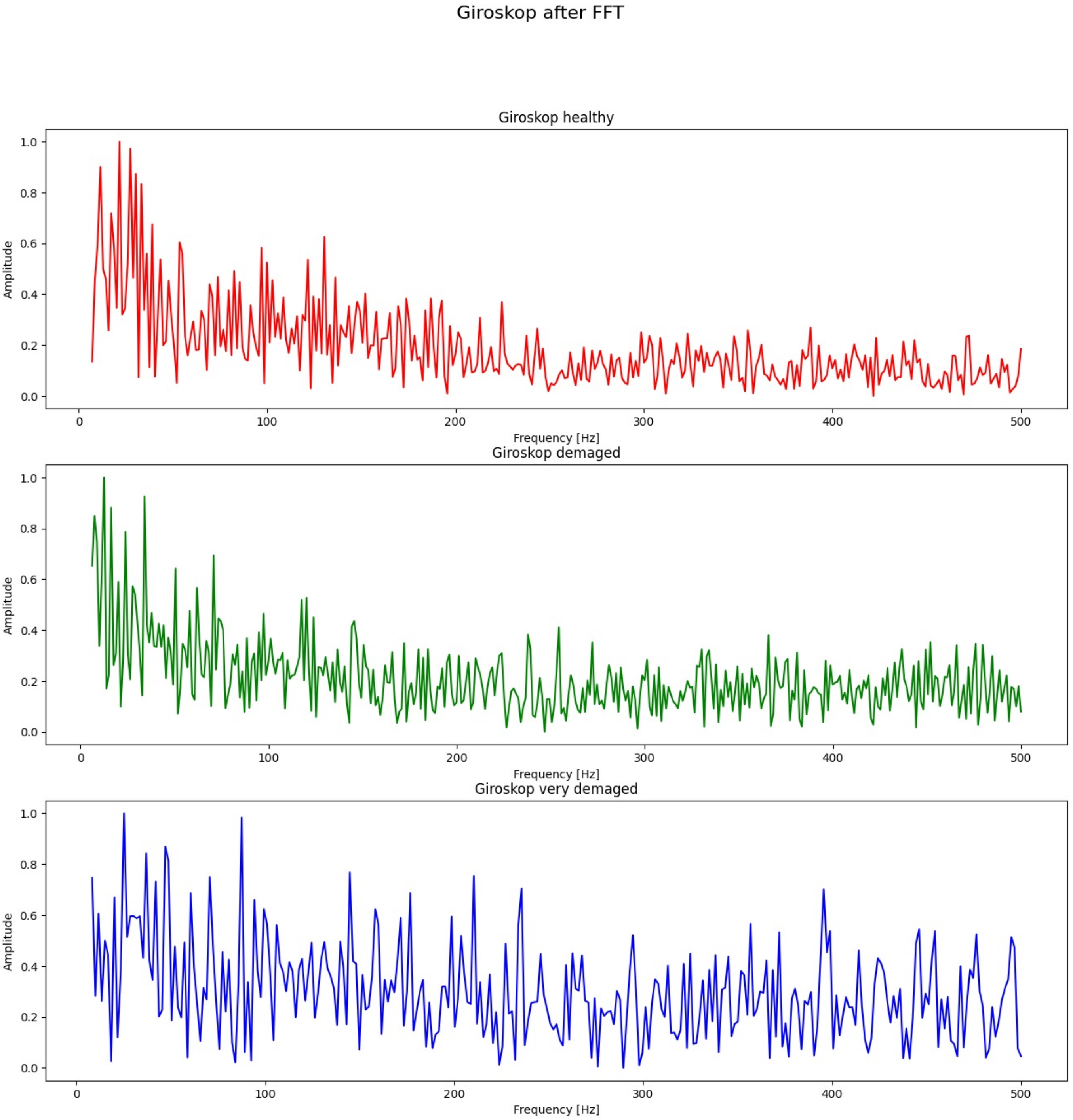
In [ ]:
```
fig,ax = plt.subplots(3,1,figsize=[16,14])

ax[0].plot(time_healthy[50:len(time_healthy)-210], gyro_healthy[50:len(time_healthy)-210], label='gyro_healthy'
ax[1].plot(time_damaged[50:len(time_damaged)-190], gyro_damaged[50:len(time_damaged)-190], label='gyro_damaged'
ax[2].plot(time_very_damaged[50:len(time_very_damaged)-135],gyro_very_damaged[50:len(time_very_damaged)-135],lal
ax[0].set_title('Giroskop healthy')
ax[1].set_title('Giroskop demaged')
ax[2].set_title('Giroskop very demaged')
fig.suptitle('Giroskop in time domain', fontsize=16)

ax[0].set_ylabel('gyro [RPS]')
ax[1].set_ylabel('gyro [RPS]')
ax[2].set_ylabel('gyro [RPS]')
ax[0].set_xlabel('time [ms]')
ax[1].set_xlabel('time [ms]')
ax[2].set_xlabel('time [ms]')
plt.savefig('photos/gyro_3.png')
plt.show()
```



Performing Fourier transform to obtain gyroscope data in the frequency domain.

In [ ]:
```
fft_val_h = abs(rfft(gyro_healthy[50:len(gyro_healthy)-210]))
fft_freq_h = rfftfreq(len(gyro_healthy[50:len(gyro_healthy)-210]),d=1/1000)

fft_val_d = abs(rfft(gyro_damaged[50:len(gyro_damaged)-190]))
fft_freq_d = rfftfreq(len(gyro_damaged[50:len(gyro_damaged)-190]),d=1/1000)

fft_val_vd = abs(rfft(gyro_very_damaged[50:len(gyro_very_damaged)-135]))
fft_freq_vd = rfftfreq(len(gyro_very_damaged[50:len(gyro_very_damaged)-135]),d=1/1000)
```

```
fig,ax = plt.subplots(3,1,figsize=[16,15])
ax[0].set_title('Giroskop after FFT')
ax[0].plot(fft_freq_h[5:], normalization(fft_val_h[5:]), label='gyro_healthy',color='red')
ax[1].plot(fft_freq_d[5:], normalization(fft_val_d[5:]), label='gyro_damaged',color='green')
ax[2].plot(fft_freq_vd[5:],normalization(fft_val_vd[5:]),label='gyro_very_damaged',color='blue')
ax[0].set_title('Giroskop healthy')
ax[1].set_title('Giroskop demaged')
ax[2].set_title('Giroskop very demaged')
fig.suptitle('Giroskop after FFT', fontsize=16)
ax[0].set_ylabel('Amplitude')
ax[1].set_ylabel('Amplitude')
ax[2].set_ylabel('Amplitude')

ax[0].set_xlabel('Frequency [Hz]')
ax[1].set_xlabel('Frequency [Hz]')
ax[2].set_xlabel('Frequency [Hz]')
plt.show()
```



In [ ]:
```
gyro_healthy_split = split_array(gyro_healthy[50:len(gyro_healthy)-210],150,125)
gyro_damaged_split = split_array(gyro_damaged[50:len(gyro_damaged)-190],150,125)
gyro_very_damaged_split = split_array(gyro_very_damaged[50:len(gyro_very_damaged)-135],150,125)
```

Save gyroscope data in the frequency domain.

In [ ]:
```
df_gyro_healthy = FFT_data(gyro_healthy_split)
```

```
df_gyro_damaged = FFT_data(gyro_damaged_split)
df_gyro_very_damaged = FFT_data(gyro_very_damaged_split)

df_gyro_healthy.to_csv('data_preprocesed/gyro_healthy_samples.csv')
df_gyro_damaged.to_csv('data_preprocesed/gyro_damaged_samples.csv')
df_gyro_very_damaged.to_csv('data_preprocesed/gyro_very_damaged_samples.csv')
```

In [ ]:
```
df_gyro_healthy.head()
```

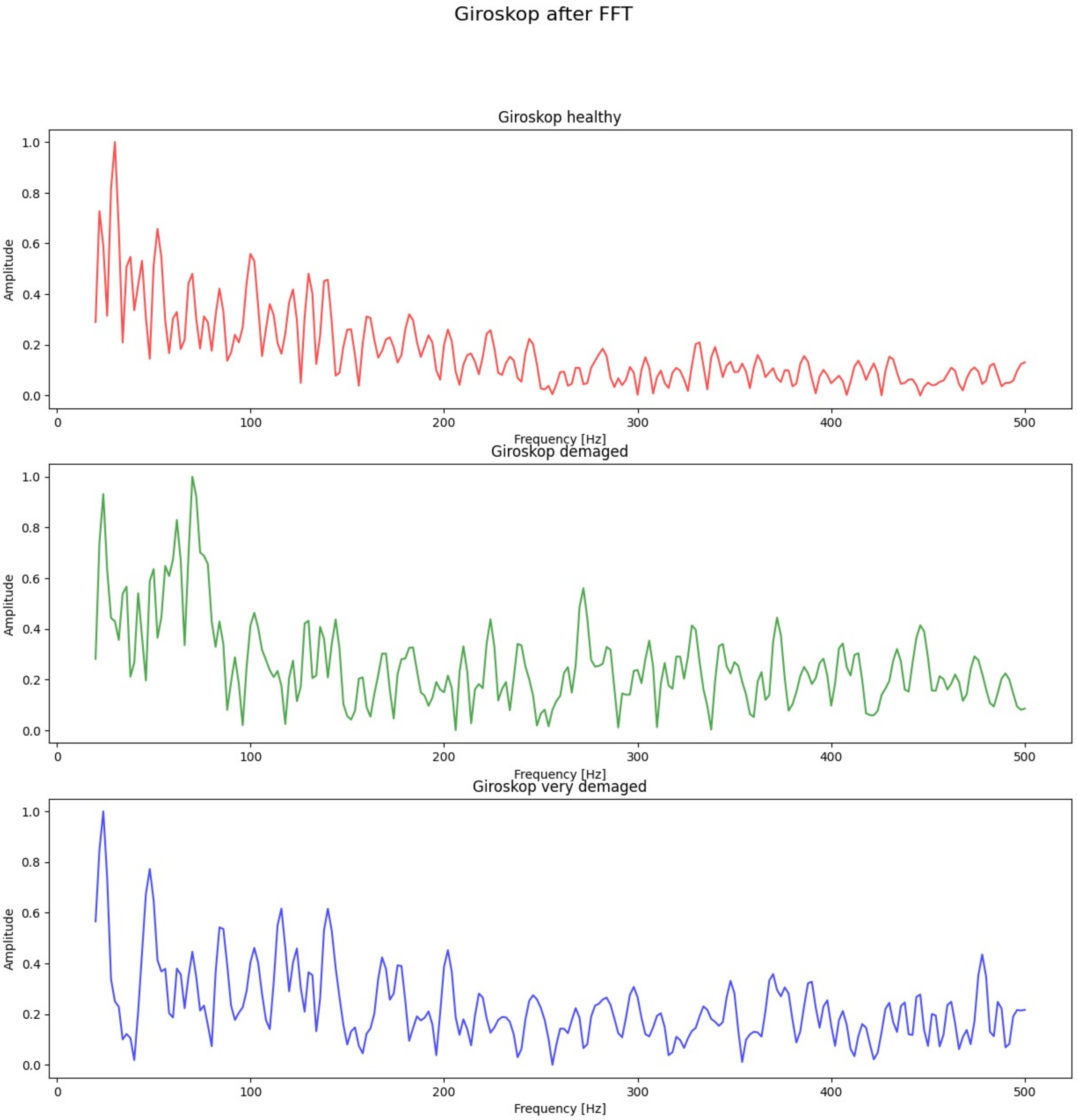
Out[ ]:

	Frequencies	Sample 1	Sample 2	Sample 3	Sample 4	Sample 5	Sample 6	Sample 7	Sample 8	Sample 9	...	Sam
0	20.0	0.291723	0.290193	0.330553	0.776665	0.739334	1.000000	0.664665	1.000000	0.756309	...	0.1055
1	22.0	0.587173	0.727546	0.907014	0.648258	1.000000	0.692805	1.000000	0.641000	1.000000	...	0.5727
2	24.0	0.775388	0.590265	0.912820	0.686112	0.869195	0.602269	0.978140	0.401656	0.701249	...	0.7721
3	26.0	0.740052	0.314294	0.309763	0.568411	0.698073	0.643005	0.621538	0.236465	0.210012	...	0.6784
4	28.0	0.935007	0.815733	0.455119	0.449102	0.834688	0.749689	0.351808	0.051400	0.499509	...	0.5691

5 rows × 23 columns

In [ ]:
```
freq = df_gyro_healthy['Frequencies']
samples_1 = [x for x in df_gyro_healthy.columns if 'Sample' in x]
samples_2 = [x for x in df_gyro_damaged.columns if 'Sample' in x]
samples_3 = [x for x in df_gyro_very_damaged.columns if 'Sample' in x]

fig,ax = plt.subplots(3,1,figsize=[15,14])
for col in samples_1[1:2]:
    ax[0].plot(freq,df_gyro_healthy[col],alpha = 0.7, color='red')
    ax[0].set_title('Giroskop healthy')
for col in samples_2[1:2]:
    ax[1].plot(freq,df_gyro_damaged[col],alpha = 0.7, color='green')
    ax[1].set_title('Giroskop demaged')
for col in samples_3[1:2]:
    ax[2].plot(freq,df_gyro_very_damaged[col],alpha = 0.7, color='blue')
    ax[2].set_title('Giroskop very demaged')

ax[0].set_ylabel('Amplitude')
ax[1].set_ylabel('Amplitude')
ax[2].set_ylabel('Amplitude')

ax[0].set_xlabel('Frequency [Hz]', fontsize=10)
ax[1].set_xlabel('Frequency [Hz]', fontsize=10)
ax[2].set_xlabel('Frequency [Hz]', fontsize=10)
fig.suptitle('Giroskop after FFT', fontsize=16)

plt.plot()
```

Out[ ]:
```
[]
```



In [ ]:
```
gyro_agg_healthy_split = split_aggregation(gyro_healthy[50:len(gyro_healthy)-210],125,25)
gyro_agg_damaged_split = split_aggregation(gyro_damaged[50:len(gyro_damaged)-190],125,25)
gyro_agg_very_damaged_split = split_aggregation(gyro_very_damaged[50:len(gyro_very_damaged)-135],125,25)
```

Save gyroscope data in the frequency domain, but it has aggregated data from previous.

In [ ]:
```
df_gyro_agg_healthy = FFT_data(gyro_agg_healthy_split)
df_gyro_agg_damaged = FFT_data(gyro_agg_damaged_split)
df_gyro_agg_very_damaged = FFT_data(gyro_agg_very_damaged_split)
df_gyro_agg_healthy.to_csv('data_preprocesed/gyro_agg_healthy_samples.csv')
df_gyro_agg_damaged.to_csv('data_preprocesed/gyro_agg_damaged_samples.csv')
df_gyro_agg_very_damaged.to_csv('data_preprocesed/gyro_agg_very_damaged_samples.csv')
```

In [ ]:
```
df_gyro_agg_healthy.head()
```

Out[ ]:

	Frequencies	Sample 1	Sample 2	Sample 3	Sample 4	Sample 5	Sample 6	Sample 7	Sample 8	Sample 9	...	Sam
0	20.0	1.000000	0.291723	0.990856	0.154706	0.892491	0.332129	0.817784	0.486697	0.785491	...	0.3741
1	22.0	0.898316	0.587173	0.701461	0.599560	1.000000	1.000000	1.000000	1.000000	1.000000	...	1.0000
2	24.0	0.591674	0.775388	0.111563	0.527578	0.363623	0.679586	0.373966	0.469347	0.416258	...	0.0895
3	26.0	0.449963	0.740052	0.212512	0.291204	0.208262	0.487992	0.204459	0.271046	0.272988	...	0.1681
4	28.0	0.574103	0.935007	0.544762	0.055050	0.477765	0.649506	0.418330	0.279336	0.431768	...	0.4171

5 rows × 24 columns

In [ ]:

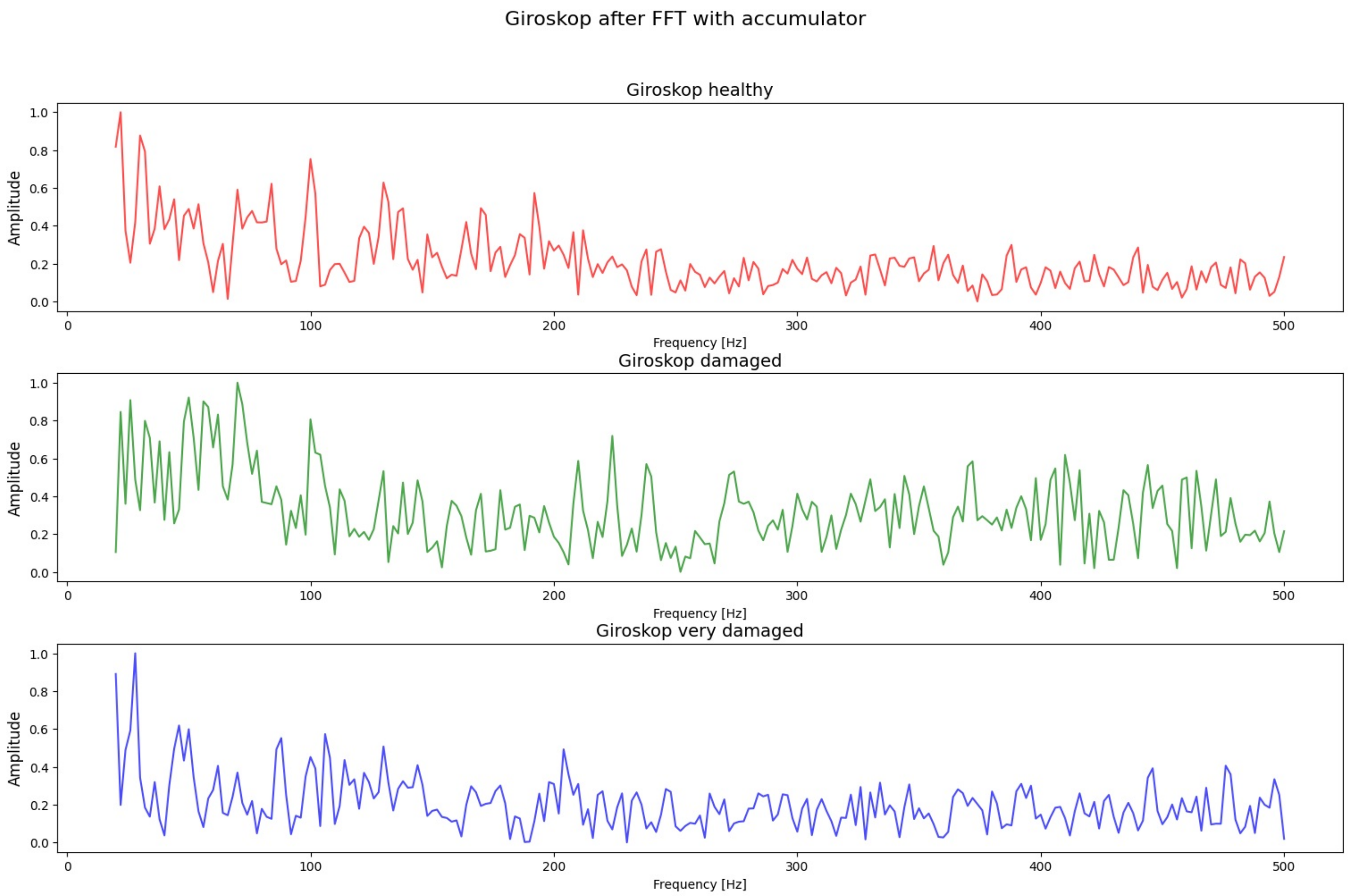
```
df_gyro_agg_healthy.get('Frequencies')
```

Out[ ]:

```
0       20.0
1       22.0
2       24.0
3       26.0
4       28.0
       ...
236    492.0
237    494.0
238    496.0
239    498.0
240    500.0
Name: Frequencies, Length: 241, dtype: float64
```

In [ ]:

```
freq = df_gyro_agg_healthy['Frequencies']

samples_1 = [x for x in df_gyro_agg_healthy.keys() if 'Sample' in x]
samples_2 = [x for x in df_gyro_agg_damaged.keys() if 'Sample' in x]
samples_3 = [x for x in df_gyro_agg_very_damaged.keys() if 'Sample' in x]


fig, ax = plt.subplots(3, 1, figsize=[15, 10])

for col in samples_1[6:7]:
    ax[0].plot(freq, df_gyro_agg_healthy[col], alpha=0.7, color='red')
    ax[0].set_title('Giroskop healthy', fontsize=14)

for col in samples_2[6:7]:
    ax[1].plot(freq, df_gyro_agg_damaged[col], alpha=0.7, color='green')
    ax[1].set_title('Giroskop damaged', fontsize=14)

for col in samples_3[6:7]:
    ax[2].plot(freq, df_gyro_agg_very_damaged[col], alpha=0.7, color='blue')
    ax[2].set_title('Giroskop very damaged', fontsize=14)

fig.suptitle('Giroskop after FFT with accumulator', fontsize=16)

fig.tight_layout(rect=[0, 0, 1, 0.95])
fig.subplots_adjust(hspace=0.3)

ax[0].set_ylabel('Amplitude', fontsize=12)
ax[1].set_ylabel('Amplitude', fontsize=12)
ax[2].set_ylabel('Amplitude', fontsize=12)

ax[0].set_xlabel('Frequency [Hz]', fontsize=10)
ax[1].set_xlabel('Frequency [Hz]', fontsize=10)
ax[2].set_xlabel('Frequency [Hz]', fontsize=10)
plt.savefig('photos/gyro_fft_500_1.png')
plt.show()
```


Giroskop after FFT with accumulator
Giroskop healthy
Amplitude
1.0
0.8
0.6
0.4
0.2
0.0
0
100
200
300
400
500
Frequency [Hz]
Giroskop damaged
Amplitude
1.0
0.8
0.6
0.4
0.2
0.0
0
100
200
300
400
500
Frequency [Hz]
Giroskop very damaged
Amplitude
1.0
0.8
0.6
0.4
0.2
0.0
0
100
200
300
400
500
Frequency [Hz]