```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.fft import fft, fftfreq
from scipy.signal import butter, filtfilt
```

```python
df = pd.read_csv('kalibrasi10-20.csv')
df.head()
```

```
             Timestamp                               Tag ID  Speed
0  2024-06-09 12:45:17  E2-00-20-23-12-05-EE-AA-00-01-00-87    7.0
1  2024-06-09 12:45:17  E2-00-20-23-12-05-EE-AA-00-01-00-87    7.4
2  2024-06-09 12:45:17  E2-00-20-23-12-05-EE-AA-00-01-00-87    8.7
3  2024-06-09 12:45:17  E2-00-20-23-12-05-EE-AA-00-01-00-87   12.2
4  2024-06-09 12:45:18  E2-00-20-23-12-05-EE-AA-00-01-00-87   10.5
```

```python
df2 = pd.read_csv('kalibrasi30-40.csv')
df2.head()
```

```
   2024-06-09 15:21:09  STAND BY  RFID IS READING  0.4
0  2024-06-09 15:21:09  STAND BY  RFID IS READING  0.9
1  2024-06-09 15:21:09  STAND BY  RFID IS READING  0.4
2  2024-06-09 15:21:10  STAND BY  RFID IS READING  0.9
3  2024-06-09 15:21:10  STAND BY  RFID IS READING  1.3
4  2024-06-09 15:21:10  STAND BY  RFID IS READING  0.9
```

```python
df3 = pd.read_csv('D:\TA\KALIBRASI\kalibrasi 40 kedua.csv')
df3.head()

n = 3

series = df3["Tag ID"]

series.head(n = n)
```

```
<>:1: SyntaxWarning: invalid escape sequence '\T'
<>:1: SyntaxWarning: invalid escape sequence '\T'
C:\Users\M ALIF F\AppData\Local\Temp\ipykernel_19848\1240454779.py:1:
SyntaxWarning: invalid escape sequence '\T'
  df3 = pd.read_csv('D:\TA\KALIBRASI\kalibrasi 40 kedua.csv')

0    RFID IS READING
1    RFID IS READING
2    RFID IS READING
Name: Tag ID, dtype: object
```

```python
print(df3.describe())
```

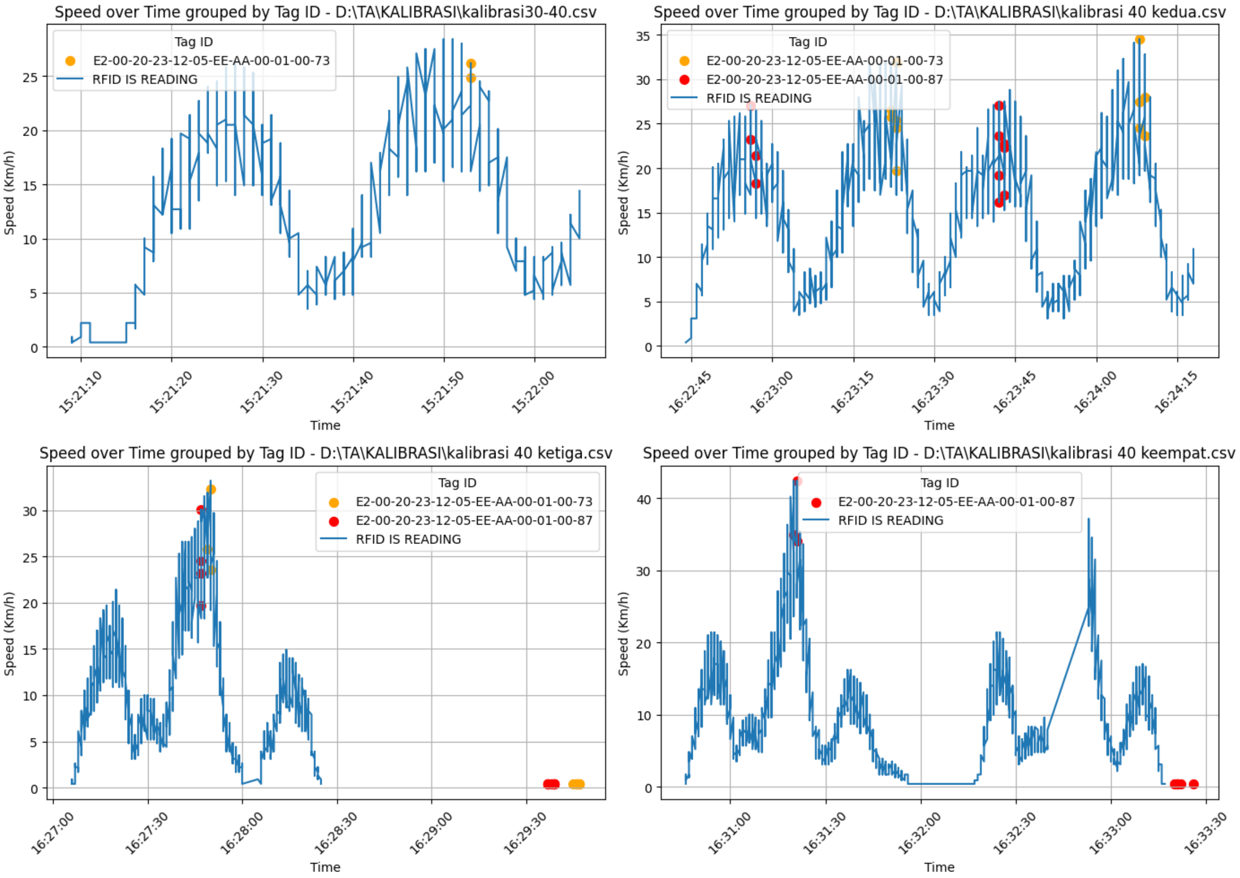
```
             Speed
count  1941.000000
mean     14.258423
std       7.556751
```

```
min        0.400000
25%        7.400000
50%       14.000000
75%       20.500000
max       34.500000
```

KALIBRASI 40 KEDUA

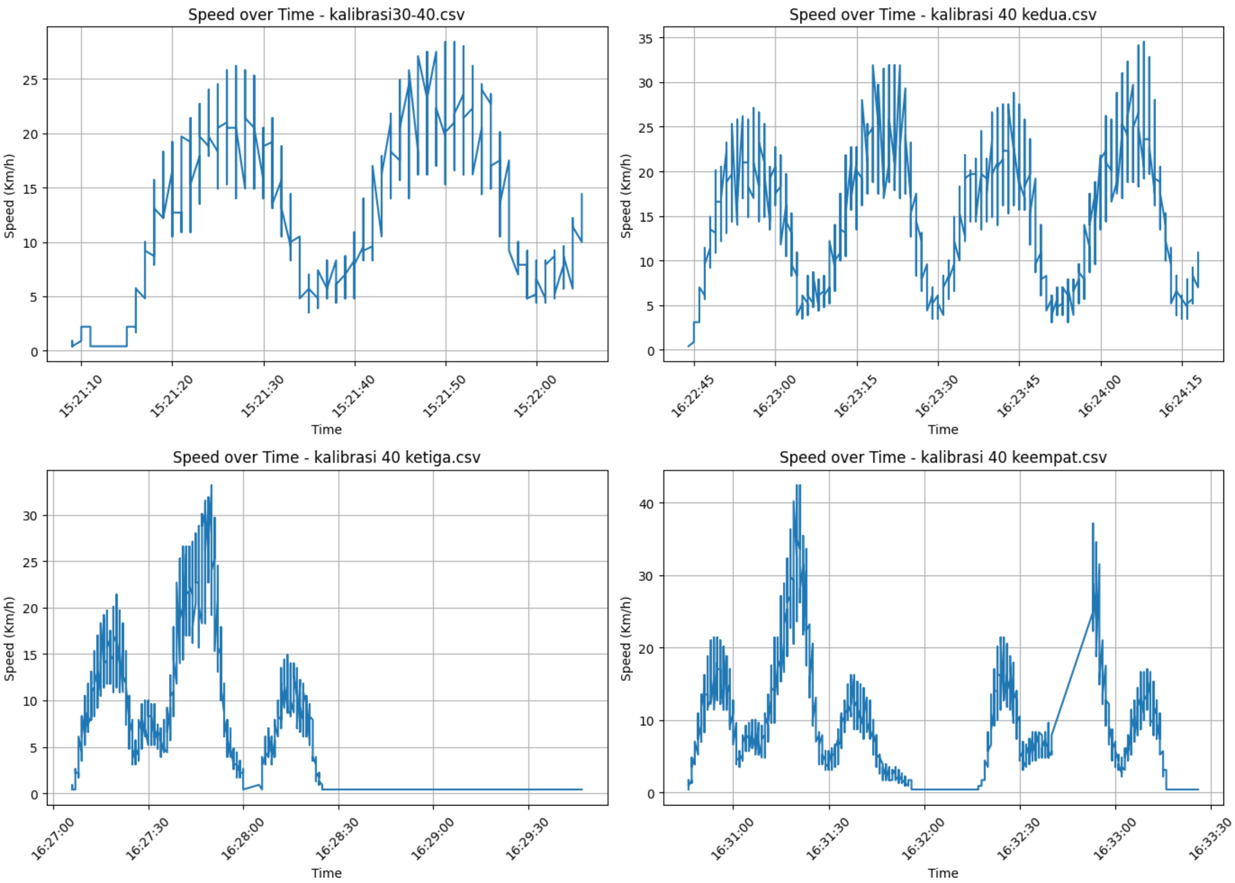
```
def process_and_plot(file_path, ax):
    # Load the data
    df = pd.read_csv(file_path)

    df['Timestamp'] = pd.to_datetime(df['Timestamp'])

    # Drop rows from index 951 to 1371
    df = df.drop(df.index[1707:1943])

    # Ensure the Speed column is numeric
    df['Speed'] = pd.to_numeric(df['Speed'], errors='coerce')

    # Drop rows with NaN values in the Speed column
    df = df.dropna(subset=['Speed'])

    # Group by 'Tag ID'
    tag_groups = df.groupby('Tag ID')

    # Specific Tag IDs to highlight
    highlight_tags = ["E2-00-20-23-12-05-EE-AA-00-01-00-87", "E2-00-
20-23-12-05-EE-AA-00-01-00-73"]

    for tag_id, group in tag_groups:
        if tag_id in highlight_tags:
            color = 'red' if tag_id == highlight_tags[0] else 'orange'
            ax.scatter(group['Timestamp'], group['Speed'],
label=tag_id, s=50, marker='o', color=color)
        else:
            ax.plot(group['Timestamp'], group['Speed'], label=tag_id)

    ax.set_xlabel('Time')
    ax.set_ylabel('Speed (Km/h)')
    ax.set_title(f'Speed over Time grouped by Tag ID - {file_path}')
    ax.legend(title='Tag ID')
    ax.tick_params(axis='x', rotation=45)
    ax.grid()

# Daftar file paths
file_paths = [
    'D:\TA\KALIBRASI\kalibrasi30-40.csv',
    'D:\TA\KALIBRASI\kalibrasi 40 kedua.csv',
    'D:\TA\KALIBRASI\kalibrasi 40 ketiga.csv',
    'D:\TA\KALIBRASI\kalibrasi 40 keempat.csv'
]

# Create a figure and subplots
fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(14, 10))
axes = axes.flatten()

# Process and plot each file
for i, file_path in enumerate(file_paths):
    process_and_plot(file_path, axes[i])

plt.tight_layout()
plt.show()
```

```
<>:38: SyntaxWarning: invalid escape sequence '\T'
<>:39: SyntaxWarning: invalid escape sequence '\T'
<>:40: SyntaxWarning: invalid escape sequence '\T'
<>:41: SyntaxWarning: invalid escape sequence '\T'
<>:38: SyntaxWarning: invalid escape sequence '\T'
<>:39: SyntaxWarning: invalid escape sequence '\T'
<>:40: SyntaxWarning: invalid escape sequence '\T'
<>:41: SyntaxWarning: invalid escape sequence '\T'
C:\Users\M ALIF F\AppData\Local\Temp\ipykernel_19848\3371209051.py:38:
SyntaxWarning: invalid escape sequence '\T'
  'D:\TA\KALIBRASI\kalibrasi30-40.csv',
C:\Users\M ALIF F\AppData\Local\Temp\ipykernel_19848\3371209051.py:39:
SyntaxWarning: invalid escape sequence '\T'
  'D:\TA\KALIBRASI\kalibrasi 40 kedua.csv',
C:\Users\M ALIF F\AppData\Local\Temp\ipykernel_19848\3371209051.py:40:
SyntaxWarning: invalid escape sequence '\T'
  'D:\TA\KALIBRASI\kalibrasi 40 ketiga.csv',
C:\Users\M ALIF F\AppData\Local\Temp\ipykernel_19848\3371209051.py:41:
SyntaxWarning: invalid escape sequence '\T'
  'D:\TA\KALIBRASI\kalibrasi 40 keempat.csv'
```



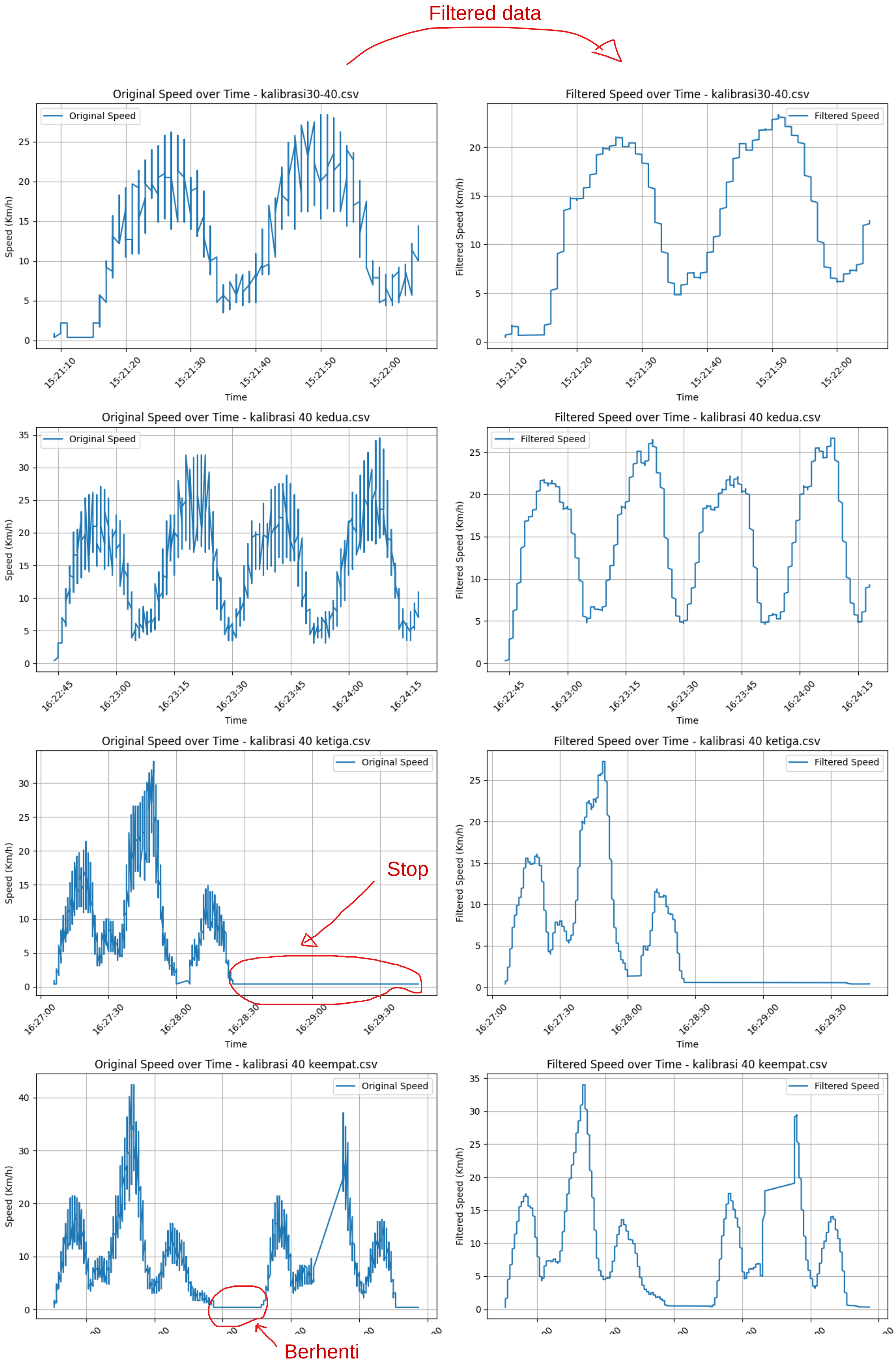
Grafik dari Kecepatan terhadap waktu

```python
def process_and_plot(file_path, ax):
    # Load the data
    df = pd.read_csv(file_path)

    df['Timestamp'] = pd.to_datetime(df['Timestamp'])

    # Drop rows from index 1707 to 1943
    df = df.drop(df.index[1707:1943])

    # Ensure the Speed column is numeric
    df['Speed'] = pd.to_numeric(df['Speed'], errors='coerce')

    # Drop rows with NaN values in the Speed column
    df = df.dropna(subset=['Speed'])

    # Plot the speed over time
    ax.plot(df['Timestamp'], df['Speed'])

    ax.set_xlabel('Time')
    ax.set_ylabel('Speed (Km/h)')
    ax.set_title(f'Speed over Time - {file_path}')
    ax.tick_params(axis='x', rotation=45)
    ax.grid()

# Daftar file paths
file_paths = [
    'kalibrasi30-40.csv',
    'kalibrasi 40 kedua.csv',
    'kalibrasi 40 ketiga.csv',
    'kalibrasi 40 keempat.csv'
]

# Create a figure and subplots
fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(14, 10))
axes = axes.flatten()

# Process and plot each file
for i, file_path in enumerate(file_paths):
    process_and_plot(file_path, axes[i])

plt.tight_layout()
plt.show()
```



```
def process_and_plot(file_path, ax):
    # Load the data
    df = pd.read_csv(file_path)

    df['Timestamp'] = pd.to_datetime(df['Timestamp'])

    # Drop rows from index 1707 to 1943
    df = df.drop(df.index[1707:1943])

    # Ensure the Speed column is numeric
    df['Speed'] = pd.to_numeric(df['Speed'], errors='coerce')

    # Drop rows with NaN values in the Speed column
    df = df.dropna(subset=['Speed'])

    # Plot the original speed over time
    ax[0].plot(df['Timestamp'], df['Speed'], label='Original Speed')
    ax[0].set_xlabel('Time')
    ax[0].set_ylabel('Speed (Km/h)')
    ax[0].set_title(f'Original Speed over Time - {file_path}')
    ax[0].tick_params(axis='x', rotation=45)
    ax[0].legend()
    ax[0].grid()

    # Apply low-pass filter to the speed data
    filtered_speed_data = apply_low_pass_filter(df['Speed'])

    # Plot the filtered speed over time
    ax[1].plot(df['Timestamp'], filtered_speed_data, label='Filtered
Speed')
    ax[1].set_xlabel('Time')
    ax[1].set_ylabel('Filtered Speed (Km/h)')
    ax[1].set_title(f'Filtered Speed over Time - {file_path}')
    ax[1].tick_params(axis='x', rotation=45)
    ax[1].legend()
    ax[1].grid()

def apply_low_pass_filter(speed_data):
    # Define the cutoff frequency and filter order
    cutoff_frequency = 0.05  # Adjust as needed
    order = 4

    # Set up the low-pass filter
    b, a = butter(order, cutoff_frequency, btype='low')

    # Apply the filter to the speed data
    filtered_speed_data = filtfilt(b, a, speed_data)

    return filtered_speed_data

# Daftar file paths
```

```python
file_paths = [
    'kalibrasi30-40.csv',
    'kalibrasi 40 kedua.csv',
    'kalibrasi 40 ketiga.csv',
    'kalibrasi 40 keempat.csv'
]

# Create a figure and subplots
fig, axes = plt.subplots(nrows=4, ncols=2, figsize=(14, 20))
axes = axes.flatten()

# Process and plot each file
for i, file_path in enumerate(file_paths):
    process_and_plot(file_path, axes[i*2:i*2+2])

plt.tight_layout()
plt.show()
```



Original Speed over Time - kalibrasi30-40.csv

Filtered Speed over Time - kalibrasi30-40.csv

Original Speed over Time - kalibrasi 40 kedua.csv

Filtered Speed over Time - kalibrasi 40 kedua.csv

Original Speed over Time - kalibrasi 40 ketiga.csv

Filtered Speed over Time - kalibrasi 40 ketiga.csv

Original Speed over Time - kalibrasi 40 keempat.csv

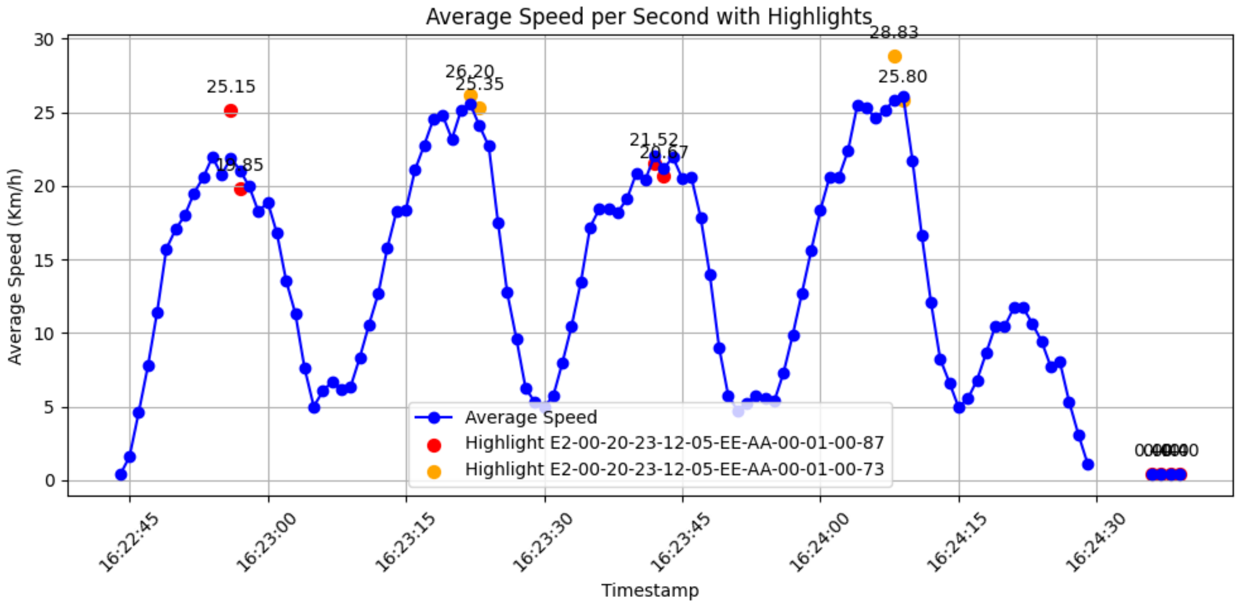
Filtered Speed over Time - kalibrasi 40 keempat.csv

```
# Assuming 'df' is your DataFrame containing the CSV data
df = pd.read_csv("kalibrasi 40 kedua.csv")
df['Timestamp'] = pd.to_datetime(df['Timestamp'])

# Set the 'Timestamp' column as the index
df.set_index('Timestamp', inplace=True)

# Resample the data in 1-second intervals and calculate the mean speed
average_speed_per_second = df['Speed'].resample('1S').mean()

# Reset the index to turn the Timestamp back into a column
average_speed_per_second = average_speed_per_second.reset_index()

# Print the result
print(average_speed_per_second)

# Plotting the average speed per second
plt.figure(figsize=(10, 5))
plt.plot(average_speed_per_second['Timestamp'],
average_speed_per_second['Speed'], marker='o', color='b',
label='Average Speed')

# Highlight specific tags and annotate with speed
highlight_tags = ["E2-00-20-23-12-05-EE-AA-00-01-00-87", "E2-00-20-23-
12-05-EE-AA-00-01-00-73"]
colors = ['red', 'orange']

# Group by 'Tag ID' and plot highlights with annotations
tag_groups = df.groupby('Tag ID')
for tag_id, color in zip(highlight_tags, colors):
    if tag_id in tag_groups.groups:
        group = tag_groups.get_group(tag_id)
        print(group)
        # Resample the group in 1-second intervals and calculate the
mean speed
        group_average = group['Speed'].resample('1S').mean()
        plt.scatter(group_average.index, group_average,
label=f'Highlight {tag_id}', s=50, marker='o', color=color)
        # Annotate each point with its speed value using the correct
method 'items()'
        for time, speed in group_average.items():
            plt.annotate(f'{speed:.2f}', (time, speed),
textcoords="offset points", xytext=(0,10), ha='center')

plt.title('Average Speed per Second with Highlights')
plt.xlabel('Timestamp')
plt.ylabel('Average Speed (Km/h)')
plt.legend()
plt.xticks(rotation=45)
plt.tight_layout()  # Adjust the plot to ensure everything fits
```

```
without overlapping
plt.grid()
plt.show()

C:\Users\M ALIF F\AppData\Local\Temp\ipykernel_19848\4089122439.py:9:
FutureWarning: 'S' is deprecated and will be removed in a future
version, please use 's' instead.
  average_speed_per_second = df['Speed'].resample('1S').mean()
C:\Users\M ALIF F\AppData\Local\Temp\ipykernel_19848\4089122439.py:32:
FutureWarning: 'S' is deprecated and will be removed in a future
version, please use 's' instead.
  group_average = group['Speed'].resample('1S').mean()
C:\Users\M ALIF F\AppData\Local\Temp\ipykernel_19848\4089122439.py:32:
FutureWarning: 'S' is deprecated and will be removed in a future
version, please use 's' instead.
  group_average = group['Speed'].resample('1S').mean()

              Timestamp      Speed
0   2024-06-09 16:22:44   0.400000
1   2024-06-09 16:22:45   1.605882
2   2024-06-09 16:22:46   4.611111
3   2024-06-09 16:22:47   7.772222
4   2024-06-09 16:22:48  11.433333
..                  ...        ...
111 2024-06-09 16:24:35        NaN
112 2024-06-09 16:24:36   0.400000
113 2024-06-09 16:24:37   0.400000
114 2024-06-09 16:24:38   0.400000
115 2024-06-09 16:24:39   0.400000

[116 rows x 2 columns]
                    Tag Status                               Tag ID
Speed
Timestamp

2024-06-09 16:22:56      TAG 4  E2-00-20-23-12-05-EE-AA-00-01-00-87
23.2
2024-06-09 16:22:56      TAG 4  E2-00-20-23-12-05-EE-AA-00-01-00-87
27.1
2024-06-09 16:22:57      TAG 4  E2-00-20-23-12-05-EE-AA-00-01-00-87
18.3
2024-06-09 16:22:57      TAG 4  E2-00-20-23-12-05-EE-AA-00-01-00-87
21.4
2024-06-09 16:23:42      TAG 4  E2-00-20-23-12-05-EE-AA-00-01-00-87
27.1
2024-06-09 16:23:42      TAG 4  E2-00-20-23-12-05-EE-AA-00-01-00-87
23.6
2024-06-09 16:23:42      TAG 4  E2-00-20-23-12-05-EE-AA-00-01-00-87
19.2
2024-06-09 16:23:42      TAG 4  E2-00-20-23-12-05-EE-AA-00-01-00-87
```

```
16.2
2024-06-09 16:23:43      TAG 4  E2-00-20-23-12-05-EE-AA-00-01-00-87
22.3
2024-06-09 16:23:43      TAG 4  E2-00-20-23-12-05-EE-AA-00-01-00-87
17.0
2024-06-09 16:23:43      TAG 4  E2-00-20-23-12-05-EE-AA-00-01-00-87
22.7
2024-06-09 16:24:36      TAG 4  E2-00-20-23-12-05-EE-AA-00-01-00-87
0.4
2024-06-09 16:24:36      TAG 4  E2-00-20-23-12-05-EE-AA-00-01-00-87
0.4
2024-06-09 16:24:36      TAG 4  E2-00-20-23-12-05-EE-AA-00-01-00-87
0.4
2024-06-09 16:24:37      TAG 4  E2-00-20-23-12-05-EE-AA-00-01-00-87
0.4
2024-06-09 16:24:37      TAG 4  E2-00-20-23-12-05-EE-AA-00-01-00-87
0.4
2024-06-09 16:24:37      TAG 4  E2-00-20-23-12-05-EE-AA-00-01-00-87
0.4
2024-06-09 16:24:37      TAG 4  E2-00-20-23-12-05-EE-AA-00-01-00-87
0.4
2024-06-09 16:24:37      TAG 4  E2-00-20-23-12-05-EE-AA-00-01-00-87
0.4
2024-06-09 16:24:37      TAG 4  E2-00-20-23-12-05-EE-AA-00-01-00-87
0.4
2024-06-09 16:24:37      TAG 4  E2-00-20-23-12-05-EE-AA-00-01-00-87
0.4
2024-06-09 16:24:37      TAG 4  E2-00-20-23-12-05-EE-AA-00-01-00-87
0.4
2024-06-09 16:24:38      TAG 4  E2-00-20-23-12-05-EE-AA-00-01-00-87
0.4
2024-06-09 16:24:38      TAG 4  E2-00-20-23-12-05-EE-AA-00-01-00-87
0.4
2024-06-09 16:24:38      TAG 4  E2-00-20-23-12-05-EE-AA-00-01-00-87
0.4
2024-06-09 16:24:38      TAG 4  E2-00-20-23-12-05-EE-AA-00-01-00-87
0.4
2024-06-09 16:24:38      TAG 4  E2-00-20-23-12-05-EE-AA-00-01-00-87
0.4
2024-06-09 16:24:38      TAG 4  E2-00-20-23-12-05-EE-AA-00-01-00-87
0.4
2024-06-09 16:24:39      TAG 4  E2-00-20-23-12-05-EE-AA-00-01-00-87
0.4
2024-06-09 16:24:39      TAG 4  E2-00-20-23-12-05-EE-AA-00-01-00-87
0.4
2024-06-09 16:24:39      TAG 4  E2-00-20-23-12-05-EE-AA-00-01-00-87
0.4
                    Tag Status                               Tag ID
Speed
```

```
Timestamp

2024-06-09 16:23:22      TAG 1  E2-00-20-23-12-05-EE-AA-00-01-00-73
26.6
2024-06-09 16:23:22      TAG 1  E2-00-20-23-12-05-EE-AA-00-01-00-73
25.8
2024-06-09 16:23:23      TAG 1  E2-00-20-23-12-05-EE-AA-00-01-00-73
31.9
2024-06-09 16:23:23      TAG 1  E2-00-20-23-12-05-EE-AA-00-01-00-73
25.3
2024-06-09 16:23:23      TAG 1  E2-00-20-23-12-05-EE-AA-00-01-00-73
19.7
2024-06-09 16:23:23      TAG 1  E2-00-20-23-12-05-EE-AA-00-01-00-73
24.5
2024-06-09 16:24:08      TAG 1  E2-00-20-23-12-05-EE-AA-00-01-00-73
34.5
2024-06-09 16:24:08      TAG 1  E2-00-20-23-12-05-EE-AA-00-01-00-73
24.5
2024-06-09 16:24:08      TAG 1  E2-00-20-23-12-05-EE-AA-00-01-00-73
27.5
2024-06-09 16:24:09      TAG 1  E2-00-20-23-12-05-EE-AA-00-01-00-73
23.6
2024-06-09 16:24:09      TAG 1  E2-00-20-23-12-05-EE-AA-00-01-00-73
28.0
```



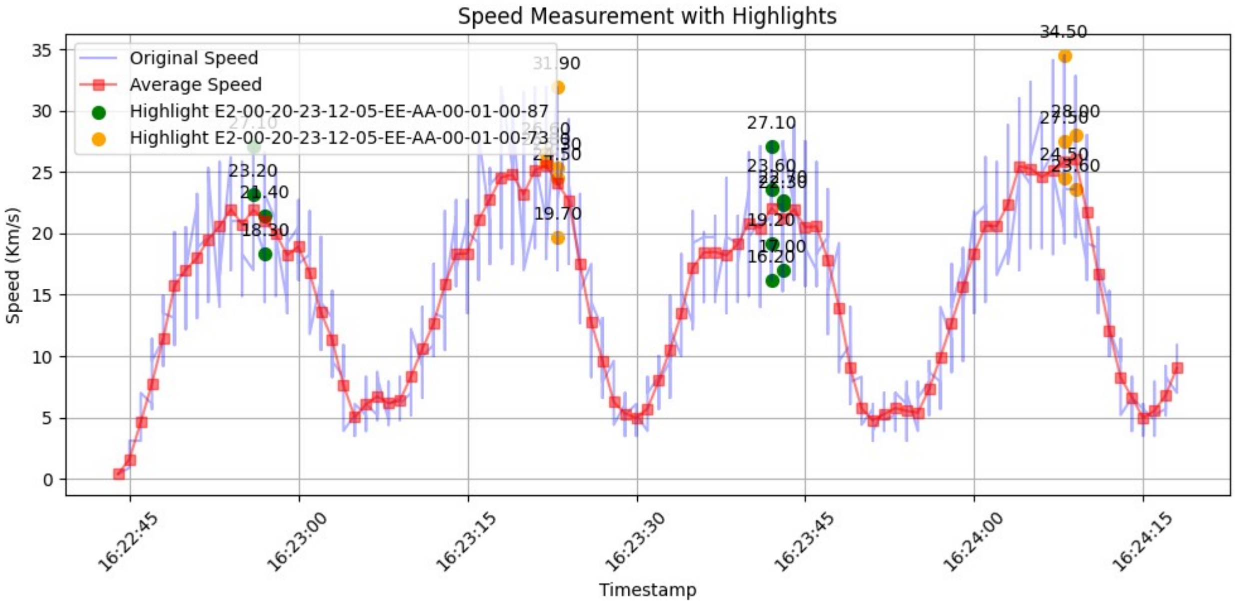
Kalibrasi 40 Kedua

```
df = pd.read_csv('kalibrasi 40 kedua.csv')

df['Timestamp'] = pd.to_datetime(df['Timestamp'])
```

```
df = df.drop(df.index[1707:1943])

df.set_index('Timestamp', inplace=True)

plt.figure(figsize=(10, 5))
plt.plot(df.index, df['Speed'], label='Original Speed', color='blue',
linestyle='-',alpha=0.3)

average_speed_per_second = df['Speed'].resample('1S').mean()

plt.plot(average_speed_per_second.index, average_speed_per_second,
marker='s', color='red', label='Average Speed',alpha = 0.5)

highlight_tags = ["E2-00-20-23-12-05-EE-AA-00-01-00-87", "E2-00-20-23-
12-05-EE-AA-00-01-00-73"]
colors = ['green', 'orange']

tag_groups = df.groupby('Tag ID')
for tag_id, color in zip(highlight_tags, colors):
    if tag_id in tag_groups.groups:
        group = tag_groups.get_group(tag_id)
        print(group)
        plt.scatter(group.index, group['Speed'], label=f'Highlight
{tag_id}', s=50, marker='o', color=color)

        for time, speed in group['Speed'].items():
            plt.annotate(f'{speed:.2f}', (time, speed),
textcoords="offset points", xytext=(0,10), ha='center')

plt.title('Speed Measurement with Highlights')
plt.xlabel('Timestamp')
plt.ylabel('Speed (Km/s)')
plt.legend()
plt.xticks(rotation=45)
plt.tight_layout()
plt.grid(True)
```

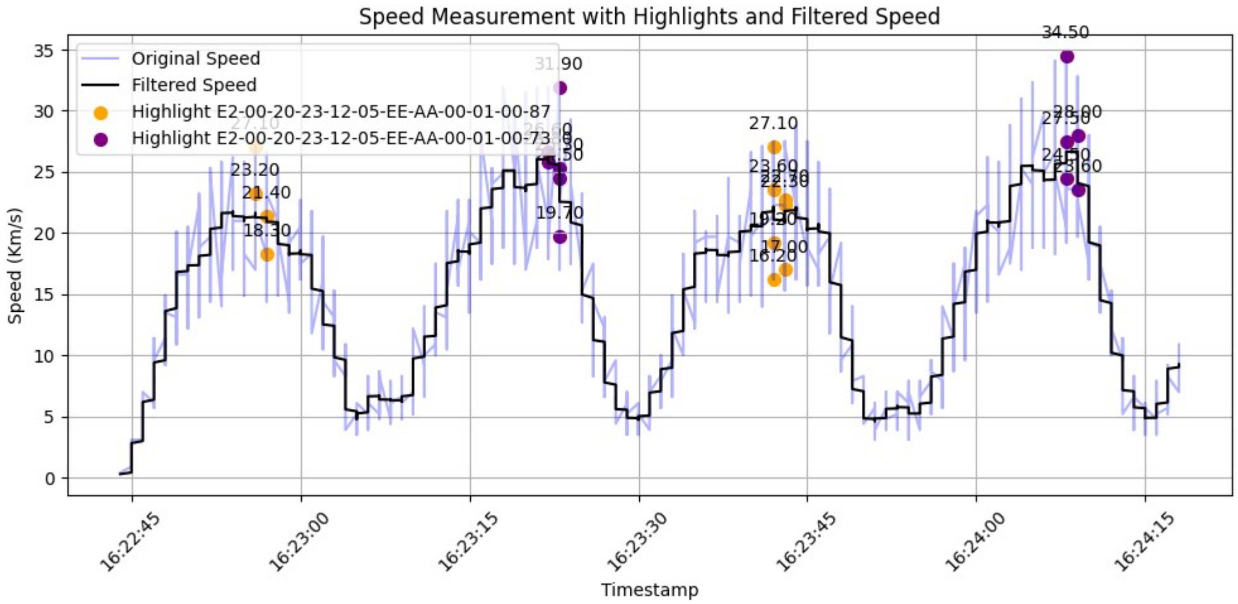
```
C:\Users\M ALIF F\AppData\Local\Temp\ipykernel_19848\3781400935.py:11:
FutureWarning: 'S' is deprecated and will be removed in a future
version, please use 's' instead.
  average_speed_per_second = df['Speed'].resample('1S').mean()

                    Tag Status                                  Tag ID
Speed
Timestamp

2024-06-09 16:22:56      TAG 4  E2-00-20-23-12-05-EE-AA-00-01-00-87
23.2
2024-06-09 16:22:56      TAG 4  E2-00-20-23-12-05-EE-AA-00-01-00-87
27.1
```

```
2024-06-09 16:22:57      TAG 4  E2-00-20-23-12-05-EE-AA-00-01-00-87
18.3
2024-06-09 16:22:57      TAG 4  E2-00-20-23-12-05-EE-AA-00-01-00-87
21.4
2024-06-09 16:23:42      TAG 4  E2-00-20-23-12-05-EE-AA-00-01-00-87
27.1
2024-06-09 16:23:42      TAG 4  E2-00-20-23-12-05-EE-AA-00-01-00-87
23.6
2024-06-09 16:23:42      TAG 4  E2-00-20-23-12-05-EE-AA-00-01-00-87
19.2
2024-06-09 16:23:42      TAG 4  E2-00-20-23-12-05-EE-AA-00-01-00-87
16.2
2024-06-09 16:23:43      TAG 4  E2-00-20-23-12-05-EE-AA-00-01-00-87
22.3
2024-06-09 16:23:43      TAG 4  E2-00-20-23-12-05-EE-AA-00-01-00-87
17.0
2024-06-09 16:23:43      TAG 4  E2-00-20-23-12-05-EE-AA-00-01-00-87
22.7
                    Tag Status                               Tag ID
Speed
Timestamp

2024-06-09 16:23:22      TAG 1  E2-00-20-23-12-05-EE-AA-00-01-00-73
26.6
2024-06-09 16:23:22      TAG 1  E2-00-20-23-12-05-EE-AA-00-01-00-73
25.8
2024-06-09 16:23:23      TAG 1  E2-00-20-23-12-05-EE-AA-00-01-00-73
31.9
2024-06-09 16:23:23      TAG 1  E2-00-20-23-12-05-EE-AA-00-01-00-73
25.3
2024-06-09 16:23:23      TAG 1  E2-00-20-23-12-05-EE-AA-00-01-00-73
19.7
2024-06-09 16:23:23      TAG 1  E2-00-20-23-12-05-EE-AA-00-01-00-73
24.5
2024-06-09 16:24:08      TAG 1  E2-00-20-23-12-05-EE-AA-00-01-00-73
34.5
2024-06-09 16:24:08      TAG 1  E2-00-20-23-12-05-EE-AA-00-01-00-73
24.5
2024-06-09 16:24:08      TAG 1  E2-00-20-23-12-05-EE-AA-00-01-00-73
27.5
2024-06-09 16:24:09      TAG 1  E2-00-20-23-12-05-EE-AA-00-01-00-73
23.6
2024-06-09 16:24:09      TAG 1  E2-00-20-23-12-05-EE-AA-00-01-00-73
28.0
```



```
def apply_low_pass_filter(speed_data):
    cutoff_frequency = 0.05
    order = 4

    # Set up the low-pass filter
    b, a = butter(order, cutoff_frequency, btype='low')

    # Apply the filter to the speed data
    filtered_speed_data = filtfilt(b, a, speed_data)

    return filtered_speed_data

df = pd.read_csv('kalibrasi 40 kedua.csv')

df['Timestamp'] = pd.to_datetime(df['Timestamp'])
df = df.drop(df.index[1707:1943])

df.set_index('Timestamp', inplace=True)

# Apply low-pass filter to the speed data
df['Filtered Speed'] = apply_low_pass_filter(df['Speed'])

plt.figure(figsize=(10, 5))

plt.plot(df.index, df['Speed'], label='Original Speed', color='blue',
linestyle='-', alpha=0.3)
plt.plot(df.index, df['Filtered Speed'], label='Filtered
Speed',color='black', linestyle='-', alpha=1,)

highlight_tags = ["E2-00-20-23-12-05-EE-AA-00-01-00-87", "E2-00-20-23-
12-05-EE-AA-00-01-00-73"]
```

```
colors = ['orange', 'purple']

tag_groups = df.groupby('Tag ID')
for tag_id, color in zip(highlight_tags, colors):
    if tag_id in tag_groups.groups:
        group = tag_groups.get_group(tag_id)
        plt.scatter(group.index, group['Speed'], label=f'Highlight
{tag_id}', s=50, marker='o', color=color)

        for time, speed in group['Speed'].items():
            plt.annotate(f'{speed:.2f}', (time, speed),
textcoords="offset points", xytext=(0,10), ha='center')

plt.title('Speed Measurement with Highlights and Filtered Speed')
plt.xlabel('Timestamp')
plt.ylabel('Speed (Km/s)')
plt.legend()
plt.xticks(rotation=45)
plt.tight_layout()
plt.grid(True)

plt.show()
```



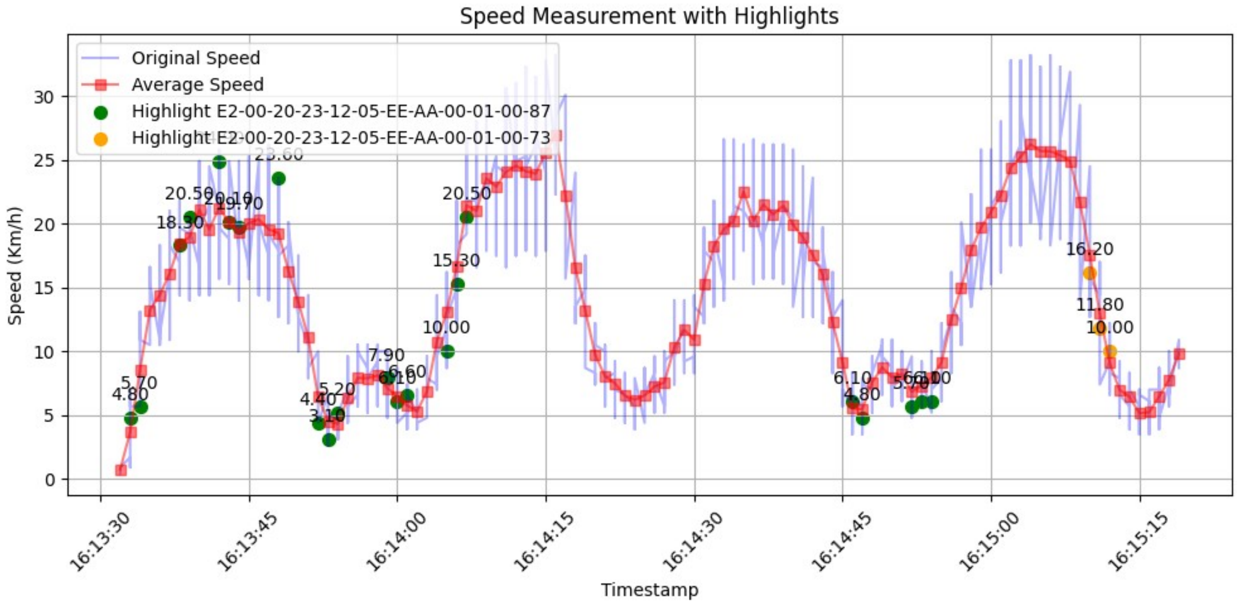
KALIBRASI 40 (kemungkinan error)

```
df = pd.read_csv('kalibrasi 40.csv')
df['Timestamp'] = pd.to_datetime(df['Timestamp'])
df = df.drop(df.index[1986:2174])
df = df.drop(df.index[0:34])
df.set_index('Timestamp', inplace=True)
```

```python
plt.figure(figsize=(10, 5))
plt.plot(df.index, df['Speed'], label='Original Speed', color='blue',
linestyle='-',alpha=0.3)

average_speed_per_second = df['Speed'].resample('1S').mean() # rata-
rata kecepatan tiap detik

plt.plot(average_speed_per_second.index, average_speed_per_second,
marker='s', color='red', label='Average Speed',alpha = 0.5)

# Menandai Tag ID yang terbaca pada grafik
highlight_tags = ["E2-00-20-23-12-05-EE-AA-00-01-00-87", "E2-00-20-23-
12-05-EE-AA-00-01-00-73"]
colors = ['green', 'orange']

# Menambahkan kerterangan pada Tag ID yang udah di tandai
tag_groups = df.groupby('Tag ID')
for tag_id, color in zip(highlight_tags, colors):
    if tag_id in tag_groups.groups:
        group = tag_groups.get_group(tag_id)
        print(group)
        plt.scatter(group.index, group['Speed'], label=f'Highlight
{tag_id}', s=50, marker='o', color=color)
        for time, speed in group['Speed'].items():
            plt.annotate(f'{speed:.2f}', (time, speed),
textcoords="offset points", xytext=(0,10), ha='center')

plt.title('Speed Measurement with Highlights')
plt.xlabel('Timestamp')
plt.ylabel('Speed (Km/h)')
plt.legend()
plt.xticks(rotation=45)
plt.tight_layout()
plt.grid(True)
```

```
C:\Users\M ALIF F\AppData\Local\Temp\ipykernel_19848\1905089993.py:10:
FutureWarning: 'S' is deprecated and will be removed in a future
version, please use 's' instead.
  average_speed_per_second = df['Speed'].resample('1S').mean() # rata-
rata kecepatan tiap detik

                    Tag Status                               Tag ID
Speed
Timestamp

2024-06-09 16:13:33      TAG 4  E2-00-20-23-12-05-EE-AA-00-01-00-87
4.8
2024-06-09 16:13:34      TAG 4  E2-00-20-23-12-05-EE-AA-00-01-00-87
5.7
2024-06-09 16:13:38      TAG 4  E2-00-20-23-12-05-EE-AA-00-01-00-87
```

```
18.3
2024-06-09 16:13:39      TAG 4  E2-00-20-23-12-05-EE-AA-00-01-00-87
20.5
2024-06-09 16:13:42      TAG 4  E2-00-20-23-12-05-EE-AA-00-01-00-87
24.9
2024-06-09 16:13:43      TAG 4  E2-00-20-23-12-05-EE-AA-00-01-00-87
20.1
2024-06-09 16:13:44      TAG 4  E2-00-20-23-12-05-EE-AA-00-01-00-87
19.7
2024-06-09 16:13:48      TAG 4  E2-00-20-23-12-05-EE-AA-00-01-00-87
23.6
2024-06-09 16:13:52      TAG 4  E2-00-20-23-12-05-EE-AA-00-01-00-87
4.4
2024-06-09 16:13:53      TAG 4  E2-00-20-23-12-05-EE-AA-00-01-00-87
3.1
2024-06-09 16:13:54      TAG 4  E2-00-20-23-12-05-EE-AA-00-01-00-87
5.2
2024-06-09 16:13:59      TAG 4  E2-00-20-23-12-05-EE-AA-00-01-00-87
7.9
2024-06-09 16:14:00      TAG 4  E2-00-20-23-12-05-EE-AA-00-01-00-87
6.1
2024-06-09 16:14:01      TAG 4  E2-00-20-23-12-05-EE-AA-00-01-00-87
6.6
2024-06-09 16:14:05      TAG 4  E2-00-20-23-12-05-EE-AA-00-01-00-87
10.0
2024-06-09 16:14:06      TAG 4  E2-00-20-23-12-05-EE-AA-00-01-00-87
15.3
2024-06-09 16:14:07      TAG 4  E2-00-20-23-12-05-EE-AA-00-01-00-87
20.5
2024-06-09 16:14:46      TAG 4  E2-00-20-23-12-05-EE-AA-00-01-00-87
6.1
2024-06-09 16:14:47      TAG 4  E2-00-20-23-12-05-EE-AA-00-01-00-87
4.8
2024-06-09 16:14:52      TAG 4  E2-00-20-23-12-05-EE-AA-00-01-00-87
5.7
2024-06-09 16:14:53      TAG 4  E2-00-20-23-12-05-EE-AA-00-01-00-87
6.1
2024-06-09 16:14:54      TAG 4  E2-00-20-23-12-05-EE-AA-00-01-00-87
6.1
                    Tag Status                               Tag ID
Speed
Timestamp

2024-06-09 16:15:10      TAG 1  E2-00-20-23-12-05-EE-AA-00-01-00-73
16.2
2024-06-09 16:15:11      TAG 1  E2-00-20-23-12-05-EE-AA-00-01-00-73
11.8
2024-06-09 16:15:12      TAG 1  E2-00-20-23-12-05-EE-AA-00-01-00-73
10.0
```

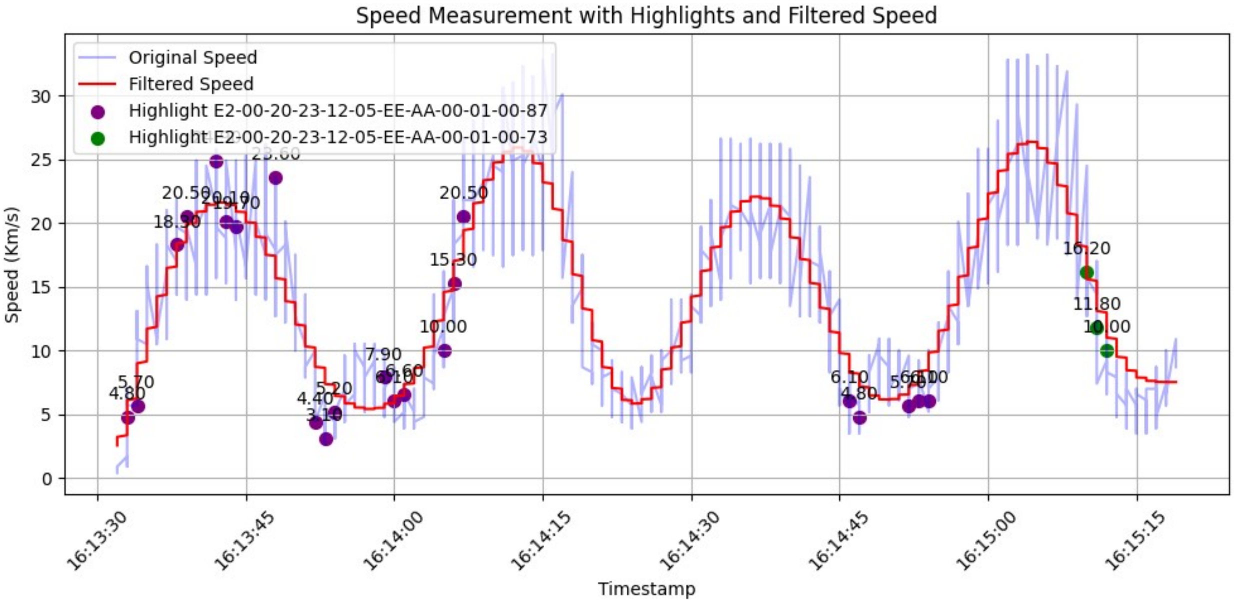


```
def apply_low_pass_filter(speed_data):
    cutoff_frequency = 0.008
    order = 4

    # Set up the low-pass filter
    b, a = butter(order, cutoff_frequency, btype='low')

    # Apply the filter to the speed data
    filtered_speed_data = filtfilt(b, a, speed_data)

    return filtered_speed_data

df = pd.read_csv('kalibrasi 40.csv')

df['Timestamp'] = pd.to_datetime(df['Timestamp'])
df = df.drop(df.index[1986:2174])
df = df.drop(df.index[0:34])

df.set_index('Timestamp', inplace=True)

# Apply low-pass filter to the speed data
df['Filtered Speed'] = apply_low_pass_filter(df['Speed'])

plt.figure(figsize=(10, 5))

plt.plot(df.index, df['Speed'], label='Original Speed', color='blue',
linestyle='-', alpha=0.3)
plt.plot(df.index, df['Filtered Speed'], label='Filtered
Speed',color='red', linestyle='-', alpha=1,)

highlight_tags = ["E2-00-20-23-12-05-EE-AA-00-01-00-87", "E2-00-20-23-
12-05-EE-AA-00-01-00-73"]
colors = ['purple', 'green']

tag_groups = df.groupby('Tag ID')
for tag_id, color in zip(highlight_tags, colors):
    if tag_id in tag_groups.groups:
        group = tag_groups.get_group(tag_id)
        print(group)
        plt.scatter(group.index, group['Speed'], label=f'Highlight 
{tag_id}', s=50, marker='o', color=color)

        for time, speed in group['Speed'].items():
            plt.annotate(f'{speed:.2f}', (time, speed), 
textcoords="offset points", xytext=(0,10), ha='center')

plt.title('Speed Measurement with Highlights and Filtered Speed')
plt.xlabel('Timestamp')
plt.ylabel('Speed (Km/s)')
plt.legend()
plt.xticks(rotation=45)
plt.tight_layout()
plt.grid(True)

plt.show()
```

```
                    Tag Status                               Tag ID
Speed  \
Timestamp

2024-06-09 16:13:33      TAG 4  E2-00-20-23-12-05-EE-AA-00-01-00-87
4.8
2024-06-09 16:13:34      TAG 4  E2-00-20-23-12-05-EE-AA-00-01-00-87
5.7
2024-06-09 16:13:38      TAG 4  E2-00-20-23-12-05-EE-AA-00-01-00-87
18.3
2024-06-09 16:13:39      TAG 4  E2-00-20-23-12-05-EE-AA-00-01-00-87
20.5
2024-06-09 16:13:42      TAG 4  E2-00-20-23-12-05-EE-AA-00-01-00-87
24.9
2024-06-09 16:13:43      TAG 4  E2-00-20-23-12-05-EE-AA-00-01-00-87
20.1
2024-06-09 16:13:44      TAG 4  E2-00-20-23-12-05-EE-AA-00-01-00-87
19.7
2024-06-09 16:13:48      TAG 4  E2-00-20-23-12-05-EE-AA-00-01-00-87
23.6
2024-06-09 16:13:52      TAG 4  E2-00-20-23-12-05-EE-AA-00-01-00-87
4.4
2024-06-09 16:13:53      TAG 4  E2-00-20-23-12-05-EE-AA-00-01-00-87
3.1
2024-06-09 16:13:54      TAG 4  E2-00-20-23-12-05-EE-AA-00-01-00-87
```

```
5.2
2024-06-09 16:13:59      TAG 4  E2-00-20-23-12-05-EE-AA-00-01-00-87
7.9
2024-06-09 16:14:00      TAG 4  E2-00-20-23-12-05-EE-AA-00-01-00-87
6.1
2024-06-09 16:14:01      TAG 4  E2-00-20-23-12-05-EE-AA-00-01-00-87
6.6
2024-06-09 16:14:05      TAG 4  E2-00-20-23-12-05-EE-AA-00-01-00-87
10.0
2024-06-09 16:14:06      TAG 4  E2-00-20-23-12-05-EE-AA-00-01-00-87
15.3
2024-06-09 16:14:07      TAG 4  E2-00-20-23-12-05-EE-AA-00-01-00-87
20.5
2024-06-09 16:14:46      TAG 4  E2-00-20-23-12-05-EE-AA-00-01-00-87
6.1
2024-06-09 16:14:47      TAG 4  E2-00-20-23-12-05-EE-AA-00-01-00-87
4.8
2024-06-09 16:14:52      TAG 4  E2-00-20-23-12-05-EE-AA-00-01-00-87
5.7
2024-06-09 16:14:53      TAG 4  E2-00-20-23-12-05-EE-AA-00-01-00-87
6.1
2024-06-09 16:14:54      TAG 4  E2-00-20-23-12-05-EE-AA-00-01-00-87
6.1

                     Filtered Speed
Timestamp
2024-06-09 16:13:33        4.914367
2024-06-09 16:13:34        7.768324
2024-06-09 16:13:38       17.749106
2024-06-09 16:13:39       19.389110
2024-06-09 16:13:42       21.580881
2024-06-09 16:13:43       21.540918
2024-06-09 16:13:44       21.133652
2024-06-09 16:13:48       16.437545
2024-06-09 16:13:52        9.279432
2024-06-09 16:13:53        7.867341
2024-06-09 16:13:54        6.746497
2024-06-09 16:13:59        5.677525
2024-06-09 16:14:00        6.165895
2024-06-09 16:14:01        6.973358
2024-06-09 16:14:05       13.711759
2024-06-09 16:14:06       16.124950
2024-06-09 16:14:07       18.691667
2024-06-09 16:14:46        8.679530
2024-06-09 16:14:47        7.466021
2024-06-09 16:14:52        7.018703
2024-06-09 16:14:53        7.999741
2024-06-09 16:14:54        9.427265
                    Tag Status                               Tag ID
```

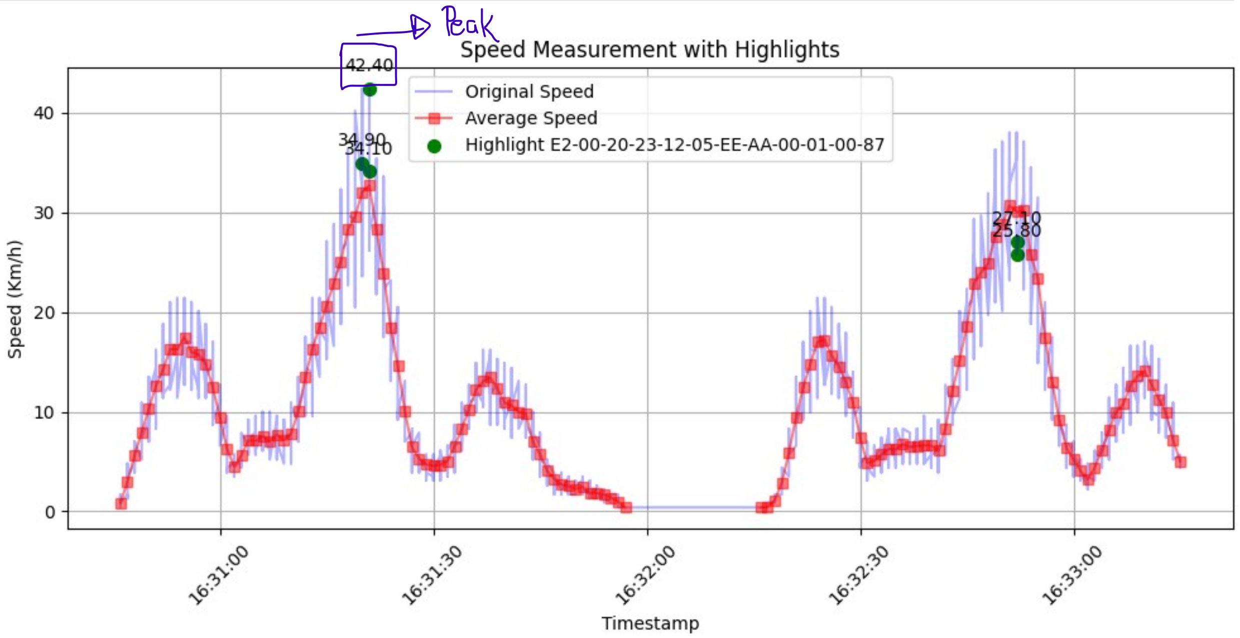
```
Speed  \
Timestamp

2024-06-09 16:15:10      TAG 1  E2-00-20-23-12-05-EE-AA-00-01-00-73
16.2
2024-06-09 16:15:11      TAG 1  E2-00-20-23-12-05-EE-AA-00-01-00-73
11.8
2024-06-09 16:15:12      TAG 1  E2-00-20-23-12-05-EE-AA-00-01-00-73
10.0

                     Filtered Speed
Timestamp
2024-06-09 16:15:10       16.297215
2024-06-09 16:15:11       13.730821
2024-06-09 16:15:12       11.516914
```



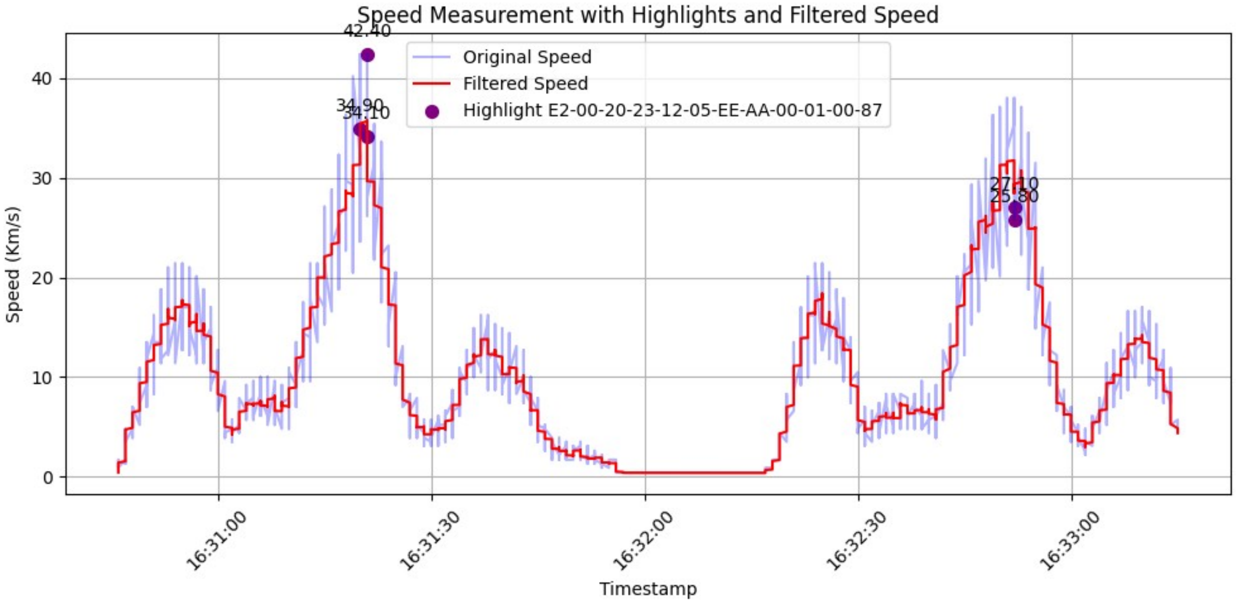
Kalibrasi 40 Keempat

```
df = pd.read_csv('kalibrasi 40 keempat.csv')

df['Timestamp'] = pd.to_datetime(df['Timestamp'])
df = df.drop(df.index[2340:2382])
df.set_index('Timestamp', inplace=True)

plt.figure(figsize=(10, 5))
plt.plot(df.index, df['Speed'], label='Original Speed', color='blue',
linestyle='-',alpha=0.3)

average_speed_per_second = df['Speed'].resample('1S').mean()

plt.plot(average_speed_per_second.index, average_speed_per_second,
marker='s', color='red', label='Average Speed',alpha = 0.5)


highlight_tags = ["E2-00-20-23-12-05-EE-AA-00-01-00-87", "E2-00-20-23-
12-05-EE-AA-00-01-00-73"]
colors = ['green', 'orange']

tag_groups = df.groupby('Tag ID')
for tag_id, color in zip(highlight_tags, colors):
    if tag_id in tag_groups.groups:
        group = tag_groups.get_group(tag_id)
        plt.scatter(group.index, group['Speed'], label=f'Highlight
{tag_id}', s=50, marker='o', color=color)

        for time, speed in group['Speed'].items():
            plt.annotate(f'{speed:.2f}', (time, speed),
textcoords="offset points", xytext=(0,10), ha='center')

plt.title('Speed Measurement with Highlights')
plt.xlabel('Timestamp')
plt.ylabel('Speed (Km/h)')
plt.legend()
plt.xticks(rotation=45)
plt.tight_layout()
plt.grid(True)
```

```
C:\Users\M ALIF F\AppData\Local\Temp\ipykernel_19848\1482900658.py:13:
FutureWarning: 'S' is deprecated and will be removed in a future
version, please use 's' instead.
  average_speed_per_second = df['Speed'].resample('1S').mean()
```



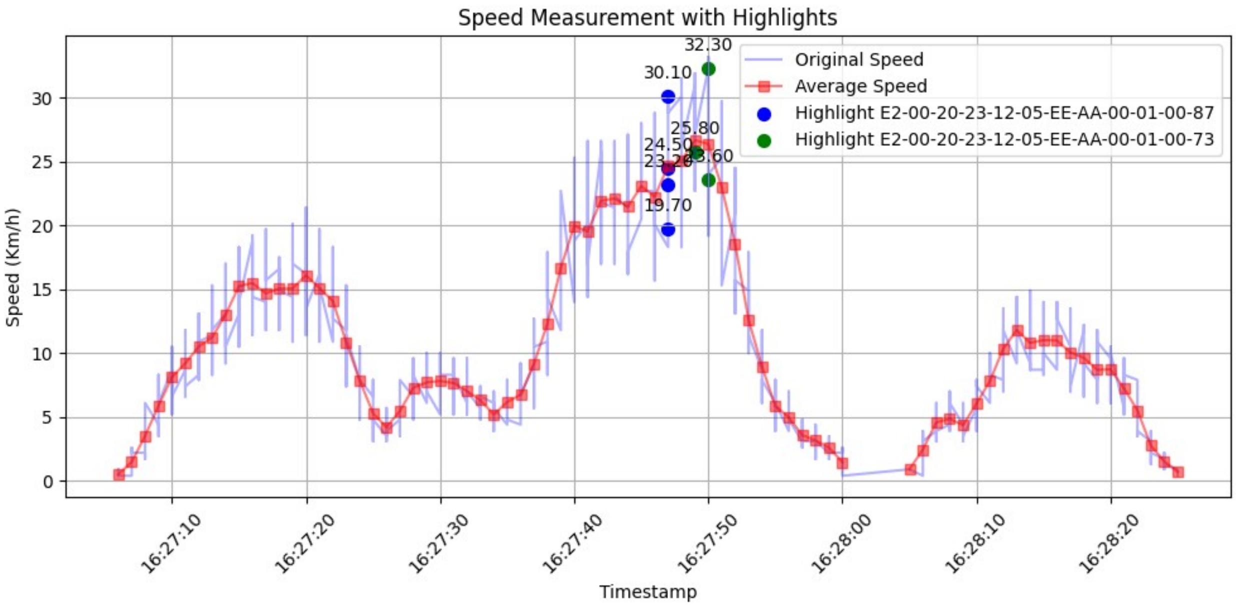
```python
def apply_low_pass_filter(speed_data):
    cutoff_frequency = 0.1
    order = 4

    # Set up the low-pass filter
    b, a = butter(order, cutoff_frequency, btype='low')

    # Apply the filter to the speed data
    filtered_speed_data = filtfilt(b, a, speed_data)

    return filtered_speed_data

df = pd.read_csv('kalibrasi 40 keempat.csv')

df['Timestamp'] = pd.to_datetime(df['Timestamp'])
df = df.drop(df.index[2340:2382])

df.set_index('Timestamp', inplace=True)

# Apply low-pass filter to the speed data
df['Filtered Speed'] = apply_low_pass_filter(df['Speed'])

plt.figure(figsize=(10, 5))

plt.plot(df.index, df['Speed'], label='Original Speed', color='blue', linestyle='-', alpha=0.3)
plt.plot(df.index, df['Filtered Speed'], label='Filtered Speed',color='red', linestyle='-', alpha=1,)

highlight_tags = ["E2-00-20-23-12-05-EE-AA-00-01-00-87", "E2-00-20-23-12-05-EE-AA-00-01-00-73"]
colors = ['purple', 'purple']

tag_groups = df.groupby('Tag ID')
for tag_id, color in zip(highlight_tags, colors):
    if tag_id in tag_groups.groups:
        group = tag_groups.get_group(tag_id)
        plt.scatter(group.index, group['Speed'], label=f'Highlight {tag_id}', s=50, marker='o', color=color)

        for time, speed in group['Speed'].items():
            plt.annotate(f'{speed:.2f}', (time, speed), textcoords="offset points", xytext=(0,10), ha='center')

plt.title('Speed Measurement with Highlights and Filtered Speed')
plt.xlabel('Timestamp')
plt.ylabel('Speed (Km/s)')
plt.legend()
plt.xticks(rotation=45)
plt.tight_layout()
plt.grid(True)
plt.show()
```



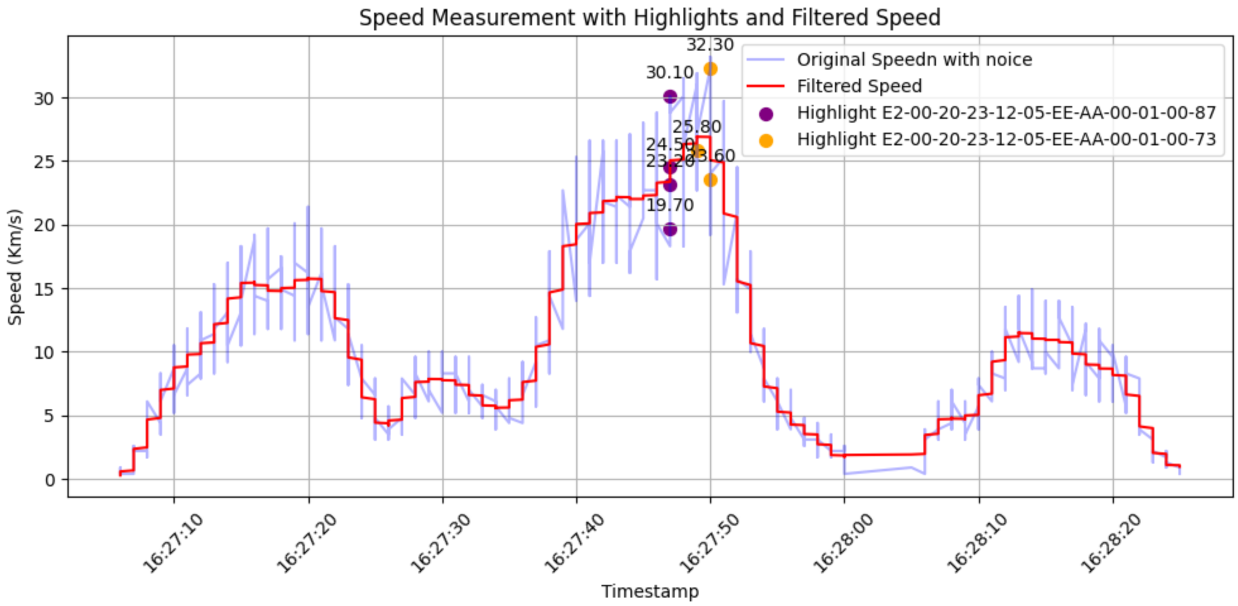
KALIBRASI DATA 40 KETIGA

```
df = pd.read_csv('kalibrasi 40 ketiga.csv')

df['Timestamp'] = pd.to_datetime(df['Timestamp'])
df = df.drop(df.index[1345:1370])

df.set_index('Timestamp', inplace=True)


plt.figure(figsize=(10, 5))
plt.plot(df.index, df['Speed'], label='Original Speed', color='blue',
linestyle='-',alpha=0.3)


average_speed_per_second = df['Speed'].resample('1S').mean()


plt.plot(average_speed_per_second.index, average_speed_per_second,
marker='s', color='red', label='Average Speed',alpha=0.5)


highlight_tags = ["E2-00-20-23-12-05-EE-AA-00-01-00-87", "E2-00-20-23-
12-05-EE-AA-00-01-00-73"]
colors = ['blue', 'green']

tag_groups = df.groupby('Tag ID')
for tag_id, color in zip(highlight_tags, colors):
    if tag_id in tag_groups.groups:
        group = tag_groups.get_group(tag_id)
        plt.scatter(group.index, group['Speed'], label=f'Highlight
{tag_id}', s=50, marker='o', color=color)

        for time, speed in group['Speed'].items():
            plt.annotate(f'{speed:.2f}', (time, speed),
textcoords="offset points", xytext=(0,10), ha='center')

plt.title('Speed Measurement with Highlights')
plt.xlabel('Timestamp')
plt.ylabel('Speed (Km/h)')
plt.legend()
plt.xticks(rotation=45)
plt.tight_layout()
plt.grid(True)
```

```
C:\Users\M ALIF F\AppData\Local\Temp\ipykernel_19848\2088860628.py:13:
FutureWarning: 'S' is deprecated and will be removed in a future
version, please use 's' instead.
  average_speed_per_second = df['Speed'].resample('1S').mean()
```



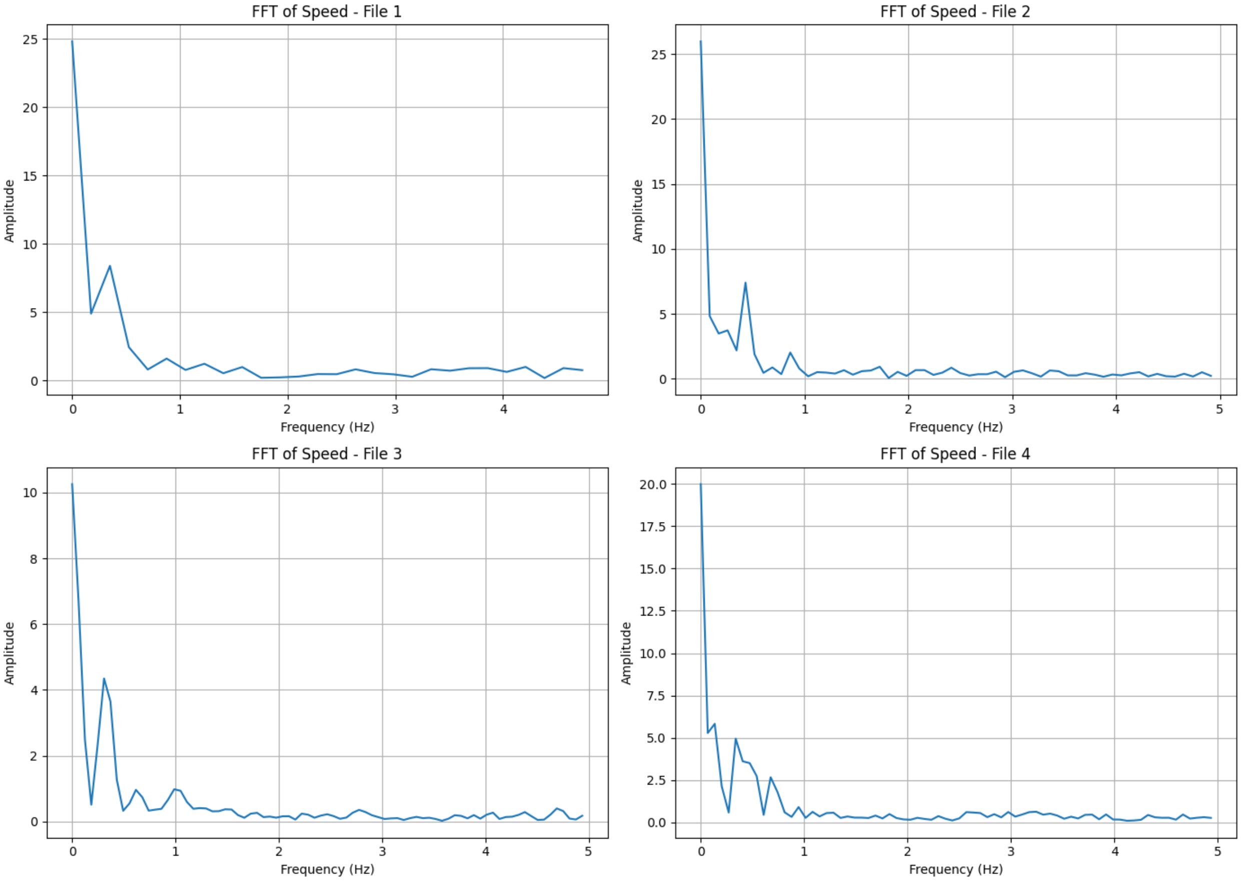
```
def apply_low_pass_filter(speed_data):
    cutoff_frequency = 0.03
    order = 4

    # Set up the low-pass filter
    b, a = butter(order, cutoff_frequency, btype='low')

    # Apply the filter to the speed data
    filtered_speed_data = filtfilt(b, a, speed_data)

    return filtered_speed_data

df = pd.read_csv('kalibrasi 40 ketiga.csv')

df['Timestamp'] = pd.to_datetime(df['Timestamp'])
df = df.drop(df.index[1345:1370])

df.set_index('Timestamp', inplace=True)

# Apply low-pass filter to the speed data
df['Filtered Speed'] = apply_low_pass_filter(df['Speed'])

plt.figure(figsize=(10, 5))

plt.plot(df.index, df['Speed'], label='Original Speedn with noice',
color='blue', linestyle='-', alpha=0.3)
plt.plot(df.index, df['Filtered Speed'], label='Filtered
Speed',color='red', linestyle='-', alpha=1,)

highlight_tags = ["E2-00-20-23-12-05-EE-AA-00-01-00-87", "E2-00-20-23-
12-05-EE-AA-00-01-00-73"]
colors = ['purple', 'orange']

tag_groups = df.groupby('Tag ID')
for tag_id, color in zip(highlight_tags, colors):
    if tag_id in tag_groups.groups:
        group = tag_groups.get_group(tag_id)
        plt.scatter(group.index, group['Speed'], label=f'Highlight
{tag_id}', s=50, marker='o', color=color)

        for time, speed in group['Speed'].items():
            plt.annotate(f'{speed:.2f}', (time, speed),
textcoords="offset points", xytext=(0,10), ha='center')

plt.title('Speed Measurement with Highlights and Filtered Speed')
plt.xlabel('Timestamp')
plt.ylabel('Speed (Km/s)')
plt.legend()
plt.xticks(rotation=45)
plt.tight_layout()
plt.grid(True)

plt.show()
```



```
def process_file(file_path):
    # Membaca data dari file CSV
    df = pd.read_csv(file_path)

    # Mengonversi kolom Timestamp ke tipe datetime
    df['Timestamp'] = pd.to_datetime(df['Timestamp'])

    # Menghapus data yang tidak diperlukan
    df = df.drop(df.index[2305:2383])

    # Mengonversi kolom Speed ke numerik
    df['Speed'] = pd.to_numeric(df['Speed'], errors='coerce')

    # Menghapus baris dengan nilai Speed yang hilang
    df = df.dropna(subset=['Speed'])

    # Set the index to the Timestamp column
    df.set_index('Timestamp', inplace=True)

    # Remove duplicate index values
    df = df[~df.index.duplicated(keep='first')]

    # Resampling dengan interval waktu yang tetap, misalnya 1 detik
    resampled_df = df.resample('1S').interpolate('linear')

    # Mengambil data kecepatan dan timestamp dari data yang telah di-
resample
    speeds = resampled_df['Speed'].values

    # Interval sampling setelah resampling
    T = 0.1  # Karena kita resample dengan interval 1 detik
    N = len(speeds)

    # Menghitung FFT
    yf = fft(speeds)
    xf = fftfreq(N, T)[:N//2]

    return xf, yf, N

# Daftar file paths
file_paths = [
    'kalibrasi30-40.csv',
    'kalibrasi 40 kedua.csv',
    'kalibrasi 40 ketiga.csv',
    'kalibrasi 40 keempat.csv'
]

# Plot hasil FFT untuk setiap file
plt.figure(figsize=(14, 10))

for i, file_path in enumerate(file_paths):
    xf, yf, N = process_file(file_path)
    plt.subplot(2, 2, i+1)
    plt.plot(xf, 2.0/N * np.abs(yf[:N//2]))
    plt.title(f'FFT of Speed - File {i+1}')
    plt.xlabel('Frequency (Hz)')
    plt.ylabel('Amplitude')
    plt.grid()

plt.tight_layout()
plt.show()
```

```
C:\Users\M ALIF F\AppData\Local\Temp\ipykernel_19848\925964425.py:24:
FutureWarning: 'S' is deprecated and will be removed in a future
version, please use 's' instead.
  resampled_df = df.resample('1S').interpolate('linear')
C:\Users\M ALIF F\AppData\Local\Temp\ipykernel_19848\925964425.py:24:
FutureWarning: DataFrame.interpolate with object dtype is deprecated
and will raise in a future version. Call obj.infer_objects(copy=False)
before interpolating instead.
  resampled_df = df.resample('1S').interpolate('linear')
C:\Users\M ALIF F\AppData\Local\Temp\ipykernel_19848\925964425.py:24:
FutureWarning: 'S' is deprecated and will be removed in a future
version, please use 's' instead.
  resampled_df = df.resample('1S').interpolate('linear')
C:\Users\M ALIF F\AppData\Local\Temp\ipykernel_19848\925964425.py:24:
FutureWarning: DataFrame.interpolate with object dtype is deprecated
and will raise in a future version. Call obj.infer_objects(copy=False)
before interpolating instead.
  resampled_df = df.resample('1S').interpolate('linear')
C:\Users\M ALIF F\AppData\Local\Temp\ipykernel_19848\925964425.py:24:
```

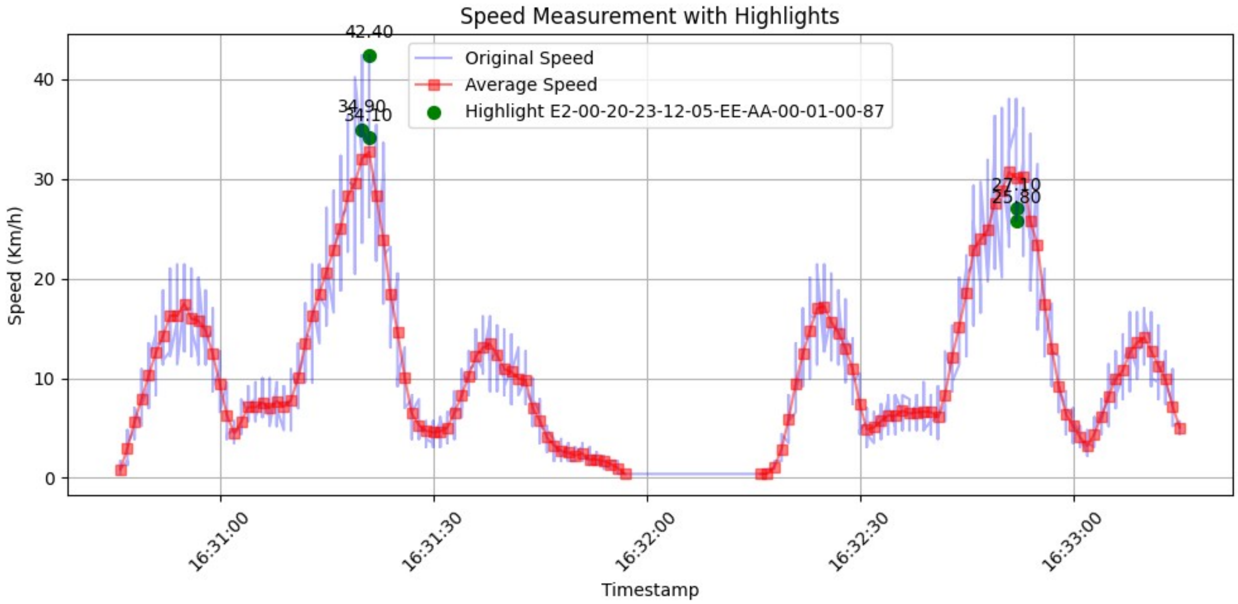
```
FutureWarning: 'S' is deprecated and will be removed in a future
version, please use 's' instead.
  resampled_df = df.resample('1S').interpolate('linear')
C:\Users\M ALIF F\AppData\Local\Temp\ipykernel_19848\925964425.py:24:
FutureWarning: DataFrame.interpolate with object dtype is deprecated
and will raise in a future version. Call obj.infer_objects(copy=False)
before interpolating instead.
  resampled_df = df.resample('1S').interpolate('linear')
C:\Users\M ALIF F\AppData\Local\Temp\ipykernel_19848\925964425.py:24:
FutureWarning: 'S' is deprecated and will be removed in a future
version, please use 's' instead.
  resampled_df = df.resample('1S').interpolate('linear')
C:\Users\M ALIF F\AppData\Local\Temp\ipykernel_19848\925964425.py:24:
FutureWarning: DataFrame.interpolate with object dtype is deprecated
and will raise in a future version. Call obj.infer_objects(copy=False)
before interpolating instead.
  resampled_df = df.resample('1S').interpolate('linear')
```



```
df = pd.read_csv('kalibrasi 40 keempat.csv')

df['Timestamp'] = pd.to_datetime(df['Timestamp'])
df = df.drop(df.index[2340:2382])
```

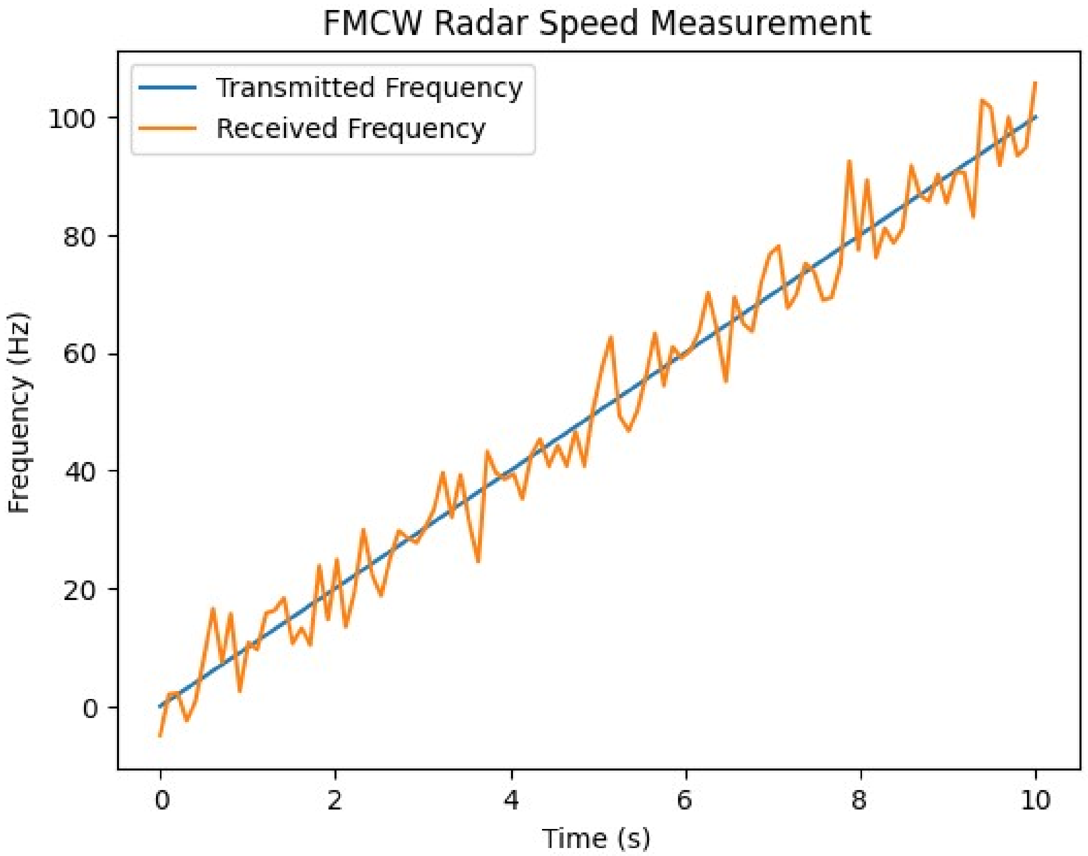
```
df.set_index('Timestamp', inplace=True)


plt.figure(figsize=(10, 5))
plt.plot(df.index, df['Speed'], label='Original Speed', color='blue',
linestyle='-',alpha=0.3)


average_speed_per_second = df['Speed'].resample('1S').mean()


plt.plot(average_speed_per_second.index, average_speed_per_second,
marker='s', color='red', label='Average Speed',alpha = 0.5)


highlight_tags = ["E2-00-20-23-12-05-EE-AA-00-01-00-87", "E2-00-20-23-
12-05-EE-AA-00-01-00-73"]
colors = ['green', 'orange']

tag_groups = df.groupby('Tag ID')
for tag_id, color in zip(highlight_tags, colors):
    if tag_id in tag_groups.groups:
        group = tag_groups.get_group(tag_id)
        plt.scatter(group.index, group['Speed'], label=f'Highlight
{tag_id}', s=50, marker='o', color=color)

        for time, speed in group['Speed'].items():
            plt.annotate(f'{speed:.2f}', (time, speed),
textcoords="offset points", xytext=(0,10), ha='center')

plt.title('Speed Measurement with Highlights')
plt.xlabel('Timestamp')
plt.ylabel('Speed (Km/h)')
plt.legend()
plt.xticks(rotation=45)
plt.tight_layout()
plt.grid(True)
```

```
C:\Users\M ALIF F\AppData\Local\Temp\ipykernel_19848\1482900658.py:13:
FutureWarning: 'S' is deprecated and will be removed in a future
version, please use 's' instead.
  average_speed_per_second = df['Speed'].resample('1S').mean()
```

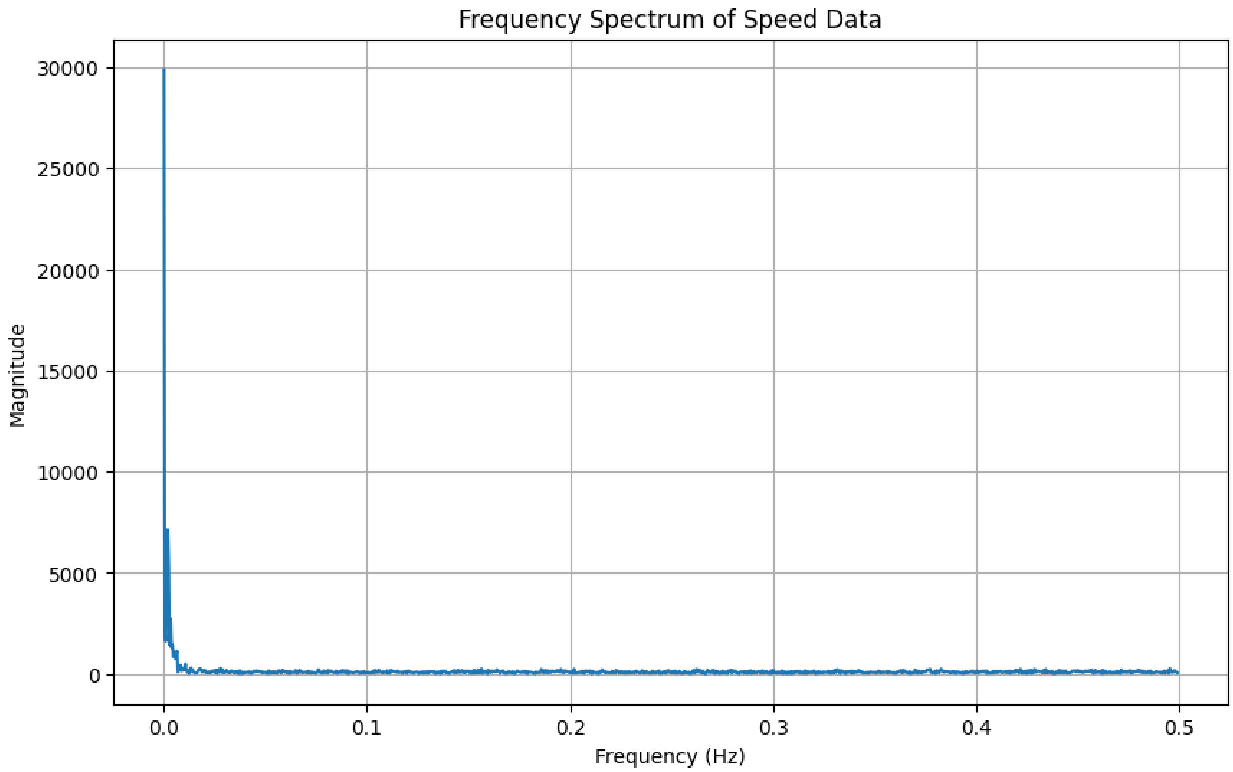


CONTOH DATA FMCW

```
# Example data: replace with your actual radar data
time = np.linspace(0, 10, 100)  # Time in seconds
frequency_transmitted = np.linspace(0, 100, 100)  # Frequency in Hz
frequency_received = frequency_transmitted + np.random.normal(0, 5,
100)  # Simulated Doppler shift

plt.figure()
plt.title('FMCW Radar Speed Measurement')
plt.xlabel('Time (s)')
plt.ylabel('Frequency (Hz)')
plt.plot(time, frequency_transmitted, label='Transmitted Frequency')
plt.plot(time, frequency_received, label='Received Frequency')
plt.legend()
plt.show()
```

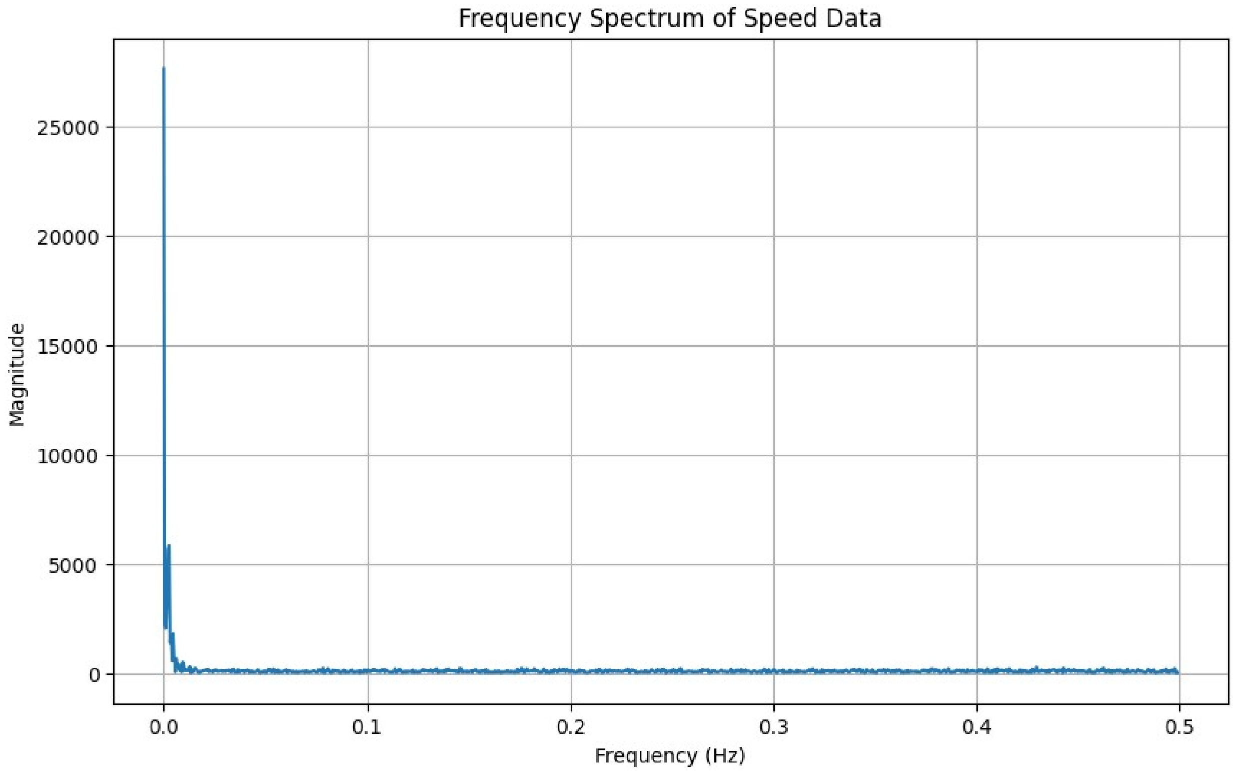


```
csv_file_path = 'kalibrasi 40.csv'  # Replace with your CSV file path
data = pd.read_csv(csv_file_path)
speed_data = data['Speed'].values

# Step 2: Perform FFT
sampling_rate = 1  # Replace with your actual sampling rate in Hz
n = len(speed_data)
fft_result = np.fft.fft(speed_data)
fft_magnitude = np.abs(fft_result)
frequencies = np.fft.fftfreq(n, d=1/sampling_rate)

# Step 3: Plot the Frequency Spectrum
plt.figure(figsize=(10, 6))
plt.plot(frequencies[:n//2], fft_magnitude[:n//2])  # Plot only the
positive frequencies
plt.title('Frequency Spectrum of Speed Data')
plt.xlabel('Frequency (Hz)')
plt.ylabel('Magnitude')
plt.grid()
plt.show()
```



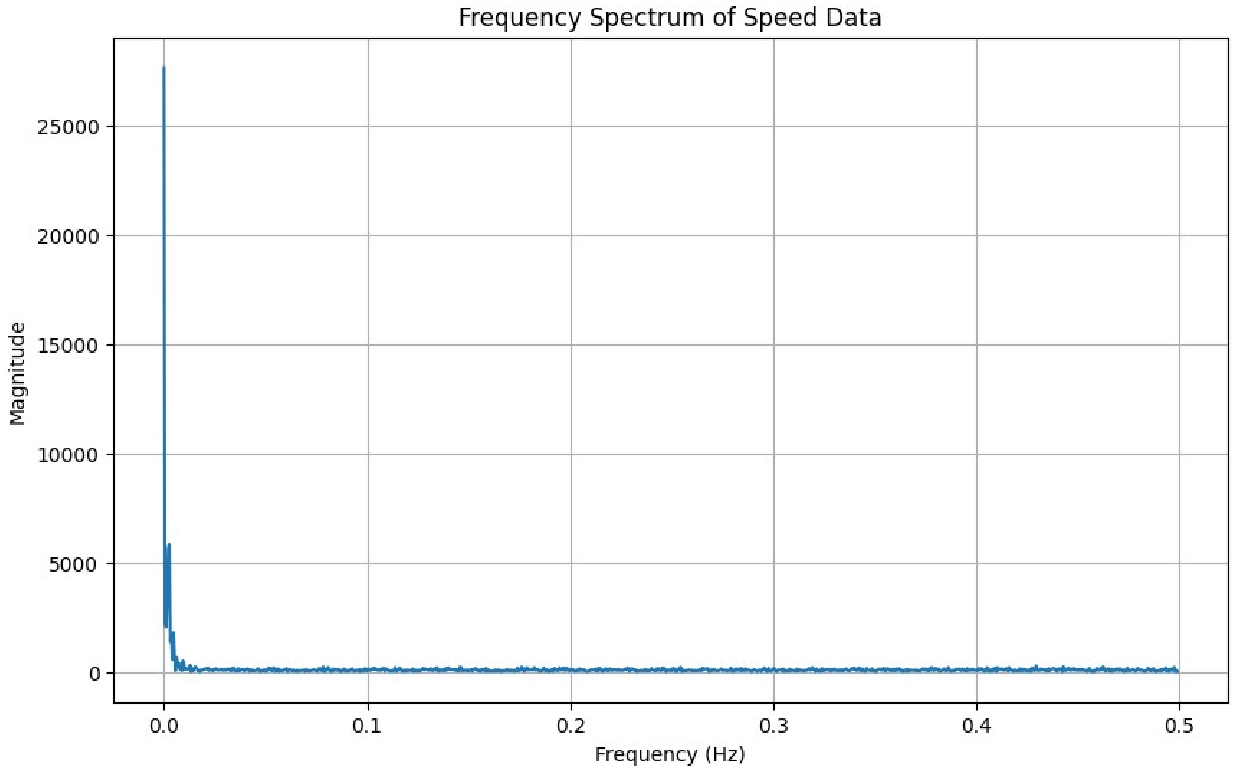
```
csv_file_path = 'kalibrasi 40 kedua.csv'  # Replace with your CSV file
path
data = pd.read_csv(csv_file_path)
speed_data = data['Speed'].values

# Step 2: Perform FFT
sampling_rate = 1  # Replace with your actual sampling rate in Hz
n = len(speed_data)
fft_result = np.fft.fft(speed_data)
fft_magnitude = np.abs(fft_result)
frequencies = np.fft.fftfreq(n, d=1/sampling_rate)

print('nilai fft',fft_result)
print('nilai frequencies: ',frequencies)
print('nilai fft_magnitude: ',fft_magnitude)
# Step 3: Plot the Frequency Spectrum
plt.figure(figsize=(10, 6))
plt.plot(frequencies[:n//2], fft_magnitude[:n//2])  # Plot only the
positive frequencies
plt.title('Frequency Spectrum of Speed Data')
plt.xlabel('Frequency (Hz)')
plt.ylabel('Magnitude')
plt.grid()
plt.show()
```

```
nilai fft [27675.6       +6.57252031e-13j -2169.59838308-
5.82951120e+02j
 -2047.30094932-2.15628613e+02j ... -1095.93559711+3.13096023e+03j
 -2047.30094932+2.15628613e+02j -2169.59838308+5.82951120e+02j]
nilai frequencies:  [ 0.         0.0005152  0.0010304 ... -0.0015456 -
0.0010304 -0.0005152]
nilai fft_magnitude:  [27675.6         2246.55050075
2058.62499636 ...  3317.22577162
  2058.62499636  2246.55050075]
```



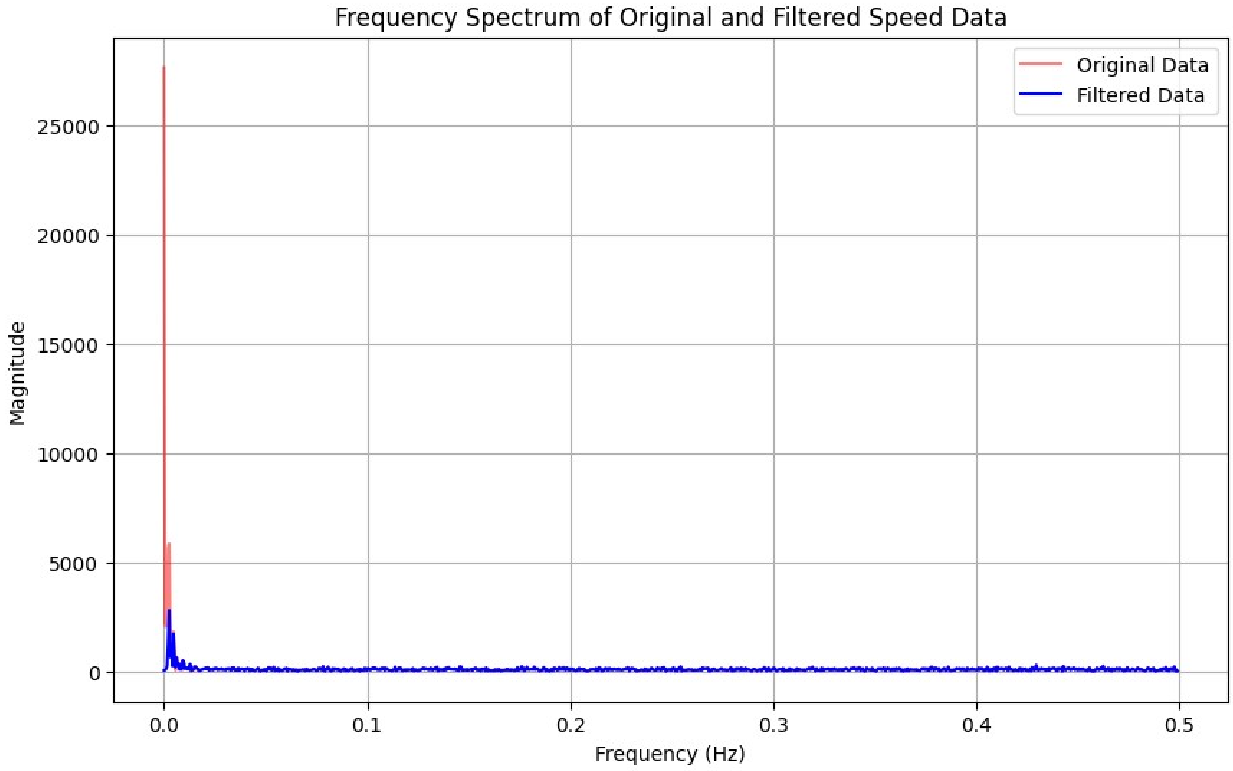
```
plt.figure(figsize=(10, 6))
positive_freqs = frequencies[:len(frequencies)//2]
positive_magnitudes = fft_magnitude[:len(fft_magnitude)//2]

print("Value of positive_freqs: \n",positive_freqs)
print("Value of positive_magnitudes: \n",positive_magnitudes)
plt.plot(positive_freqs, positive_magnitudes)
plt.title('Frequency Spectrum of Speed Data')
plt.xlabel('Frequency (Hz)')
plt.ylabel('Magnitude')
plt.grid()
plt.show()
```

```
Value of positive_freqs:
 [0.         0.0005152  0.0010304  0.0015456  0.00206079 0.00257599
 0.00309119 0.00360639 0.00412159 0.00463679 0.00515198 0.00566718
 0.00618238 0.00669758 0.00721278 0.00772798 0.00824317 0.00875837
 0.00927357 0.00978877 0.01030397 0.01081917 0.01133436 0.01184956
 0.01236476 0.01287996 0.01339516 0.01391036 0.01442555 0.01494075
 0.01545595 0.01597115 0.01648635 0.01700155 0.01751674 0.01803194
 0.01854714 0.01906234 0.01957754 0.02009274 0.02060793 0.02112313
 0.02163833 0.02215353 0.02266873 0.02318393 0.02369912 0.02421432
 0.02472952 0.02524472 0.02575992 0.02627512 0.02679031 0.02730551
 0.02782071 0.02833591 0.02885111 0.02936631 0.0298815  0.0303967
 0.0309119  0.0314271  0.0319423  0.0324575  0.03297269 0.03348789
 0.03400309 0.03451829 0.03503349 0.03554869 0.03606388 0.03657908
 0.03709428 0.03760948 0.03812468 0.03863988 0.03915507 0.03967027
 0.04018547 0.04070067 0.04121587 0.04173107 0.04224626 0.04276146
 0.04327666 0.04379186 0.04430706 0.04482226 0.04533745 0.04585265
 0.04636785 0.04688305 0.04739825 0.04791345 0.04842865 0.04894384
 0.04945904 0.04997424 0.05048944 0.05100464 0.05151984 0.05203503
 0.05255023 0.05306543 0.05358063 0.05409583 0.05461103 0.05512622
 0.05564142 0.05615662 0.05667182 0.05718702 0.05770222 0.05821741
 0.05873261 0.05924781 0.05976301 0.06027821 0.06079341 0.0613086
 0.0618238  0.062339   0.0628542  0.0633694  0.0638846  0.06439979
 0.06491499 0.06543019 0.06594539 0.06646059 0.06697579 0.06749098
 0.06800618 0.06852138 0.06903658 0.06955178 0.07006698 0.07058217
 0.07109737 0.07161257 0.07212777 0.07264297 0.07315817 0.07367336
 0.07418856 0.07470376 0.07521896 0.07573416 0.07624936 0.07676455
 0.07727975 0.07779495 0.07831015 0.07882535 0.07934055 0.07985574
 0.08037094 0.08088614 0.08140134 0.08191654 0.08243174 0.08294693
 0.08346213 0.08397733 0.08449253 0.08500773 0.08552293 0.08603812
 0.08655332 0.08706852 0.08758372 0.08809892 0.08861412 0.08912931
 0.08964451 0.09015971 0.09067491 0.09119011 0.09170531 0.0922205
 0.0927357  0.0932509  0.0937661  0.0942813  0.0947965  0.0953117
 0.09582689 0.09634209 0.09685729 0.09737249 0.09788769 0.09840289
 0.09891808 0.09943328 0.09994848 0.10046368 0.10097888 0.10149408
 0.10200927 0.10252447 0.10303967 0.10355487 0.10407007 0.10458527
 0.10510046 0.10561566 0.10613086 0.10664606 0.10716126 0.10767646
 0.10819165 0.10870685 0.10922205 0.10973725 0.11025245 0.11076765
 0.11128284 0.11179804 0.11231324 0.11282844 0.11334364 0.11385884
 0.11437403 0.11488923 0.11540443 0.11591963 0.11643483 0.11695003
 0.11746522 0.11798042 0.11849562 0.11901082 0.11952602 0.12004122
 0.12055641 0.12107161 0.12158681 0.12210201 0.12261721 0.12313241
 0.1236476  0.1241628  0.124678   0.1251932  0.1257084  0.1262236
 0.12673879 0.12725399 0.12776919 0.12828439 0.12879959 0.12931479
 0.12982998 0.13034518 0.13086038 0.13137558 0.13189078 0.13240598
 0.13292117 0.13343637 0.13395157 0.13446677 0.13498197 0.13549717
 0.13601236 0.13652756 0.13704276 0.13755796 0.13807316 0.13858836
 0.13910355 0.13961875 0.14013395 0.14064915 0.14116435 0.14167955
 0.14219474 0.14270994 0.14322514 0.14374034 0.14425554 0.14477074
 0.14528594 0.14580113 0.14631633 0.14683153 0.14734673 0.14786193
 0.14837713 0.14889232 0.14940752 0.14992272 0.15043792 0.15095312
 0.15146832 0.15198351 0.15249871 0.15301391 0.15352911 0.15404431
 0.15455951 0.1550747  0.1555899  0.1561051  0.1566203  0.1571355
 0.1576507  0.15816589 0.15868109 0.15919629 0.15971149 0.16022669
 0.16074189 0.16125708 0.16177228 0.16228748 0.16280268 0.16331788
 0.16383308 0.16434827 0.16486347 0.16537867 0.16589387 0.16640907
 0.16692427 0.16743946 0.16795466 0.16846986 0.16898506 0.16950026
 0.17001546 0.17053065 0.17104585 0.17156105 0.17207625 0.17259145
 0.17310665 0.17362184 0.17413704 0.17465224 0.17516744 0.17568264
 0.17619784 0.17671303 0.17722823 0.17774343 0.17825863 0.17877383
 0.17928903 0.17980422 0.18031942 0.18083462 0.18134982 0.18186502
 0.18238022 0.18289541 0.18341061 0.18392581 0.18444101 0.18495621
 0.18547141 0.1859866  0.1865018  0.187017   0.1875322  0.1880474
 0.1885626  0.18907779 0.18959299 0.19010819 0.19062339 0.19113859
 0.19165379 0.19216899 0.19268418 0.19319938 0.19371458 0.19422978
 0.19474498 0.19526018 0.19577537 0.19629057 0.19680577 0.19732097
 0.19783617 0.19835137 0.19886656 0.19938176 0.19989696 0.20041216
 0.20092736 0.20144256 0.20195775 0.20247295 0.20298815 0.20350335
 0.20401855 0.20453375 0.20504894 0.20556414 0.20607934 0.20659454
 0.20710974 0.20762494 0.20814013 0.20865533 0.20917053 0.20968573
 0.21020093 0.21071613 0.21123132 0.21174652 0.21226172 0.21277692
 0.21329212 0.21380732 0.21432251 0.21483771 0.21535291 0.21586811
 0.21638331 0.21689851 0.2174137  0.2179289  0.2184441  0.2189593
 0.2194745  0.2199897  0.22050489 0.22102009 0.22153529 0.22205049
 0.22256569 0.22308089 0.22359608 0.22411128 0.22462648 0.22514168
 0.22565688 0.22617208 0.22668727 0.22720247 0.22771767 0.22823287
 0.22874807 0.22926327 0.22977846 0.23029366 0.23080886 0.23132406
 0.23183926 0.23235446 0.23286965 0.23338485 0.23390005 0.23441525
 0.23493045 0.23544565 0.23596084 0.23647604 0.23699124 0.23750644
 0.23802164 0.23853684 0.23905204 0.23956723 0.24008243 0.24059763
 0.24111283 0.24162803 0.24214323 0.24265842 0.24317362 0.24368882
 0.24420402 0.24471922 0.24523442 0.24574961 0.24626481 0.24678001
 0.24729521 0.24781041 0.24832561 0.2488408  0.249356   0.2498712
 0.2503864  0.2509016  0.2514168  0.25193199 0.25244719 0.25296239
 0.25347759 0.25399279 0.25450799 0.25502318 0.25553838 0.25605358
 0.25656878 0.25708398 0.25759918 0.25811437 0.25862957 0.25914477
 0.25965997 0.26017517 0.26069037 0.26120556 0.26172076 0.26223596
 0.26275116 0.26326636 0.26378156 0.26429675 0.26481195 0.26532715
 0.26584235 0.26635755 0.26687275 0.26738794 0.26790314 0.26841834
 0.26893354 0.26944874 0.26996394 0.27047913 0.27099433 0.27150953
 0.27202473 0.27253993 0.27305513 0.27357032 0.27408552 0.27460072
 0.27511592 0.27563112 0.27614632 0.27666151 0.27717671 0.27769191
 0.27820711 0.27872231 0.27923751 0.2797527  0.2802679  0.2807831
 0.2812983  0.2818135  0.2823287  0.28284389 0.28335909 0.28387429
 0.28438949 0.28490469 0.28541989 0.28593509 0.28645028 0.28696548
 0.28748068 0.28799588 0.28851108 0.28902628 0.28954147 0.29005667
 0.29057187 0.29108707 0.29160227 0.29211747 0.29263266 0.29314786
 0.29366306 0.29417826 0.29469346 0.29520866 0.29572385 0.29623905
 0.29675425 0.29726945 0.29778465 0.29829985 0.29881504 0.29933024
 0.29984544 0.30036064 0.30087584 0.30139104 0.30190623 0.30242143
 0.30293663 0.30345183 0.30396703 0.30448223 0.30499742 0.30551262
 0.30602782 0.30654302 0.30705822 0.30757342 0.30808861 0.30860381
 0.30911901 0.30963421 0.31014941 0.31066461 0.3111798  0.311695
 0.3122102  0.3127254  0.3132406  0.3137558  0.31427099 0.31478619
 0.31530139 0.31581659 0.31633179 0.31684699 0.31736218 0.31787738
 0.31839258 0.31890778 0.31942298 0.31993818 0.32045337 0.32096857
 0.32148377 0.32199897 0.32251417 0.32302937 0.32354456 0.32405976
 0.32457496 0.32509016 0.32560536 0.32612056 0.32663575 0.32715095
 0.32766615 0.32818135 0.32869655 0.32921175 0.32972694 0.33024214
 0.33075734 0.33127254 0.33178774 0.33230294 0.33281813 0.33333333
 0.33384853 0.33436373 0.33487893 0.33539413 0.33590933 0.33642452
 0.33693972 0.33745492 0.33797012 0.33848532 0.33900052 0.33951571
 0.34003091 0.34054611 0.34106131 0.34157651 0.34209171 0.3426069
 0.3431221  0.3436373  0.3441525  0.3446677  0.3451829  0.34569809
 0.34621329 0.34672849 0.34724369 0.34775889 0.34827409 0.34878928
 0.34930448 0.34981968 0.35033488 0.35085008 0.35136528 0.35188047
 0.35239567 0.35291087 0.35342607 0.35394127 0.35445647 0.35497166
 0.35548686 0.35600206 0.35651726 0.35703246 0.35754766 0.35806285
 0.35857805 0.35909325 0.35960845 0.36012365 0.36063885 0.36115404
 0.36166924 0.36218444 0.36269964 0.36321484 0.36373004 0.36424523
 0.36476043 0.36527563 0.36579083 0.36630603 0.36682123 0.36733642
 0.36785162 0.36836682 0.36888202 0.36939722 0.36991242 0.37042761
 0.37094281 0.37145801 0.37197321 0.37248841 0.37300361 0.3735188
 0.374034   0.3745492  0.3750644  0.3755796  0.3760948  0.37660999
 0.37712519 0.37764039 0.37815559 0.37867079 0.37918599 0.37970118
 0.38021638 0.38073158 0.38124678 0.38176198 0.38227718 0.38279238
 0.38330757 0.38382277 0.38433797 0.38485317 0.38536837 0.38588357
 0.38639876 0.38691396 0.38742916 0.38794436 0.38845956 0.38897476
 0.38948995 0.39000515 0.39052035 0.39103555 0.39155075 0.39206595
 0.39258114 0.39309634 0.39361154 0.39412674 0.39464194 0.39515714
 0.39567233 0.39618753 0.39670273 0.39721793 0.39773313 0.39824833
 0.39876352 0.39927872 0.39979392 0.40030912 0.40082432 0.40133952
 0.40185471 0.40236991 0.40288511 0.40340031 0.40391551 0.40443071
 0.4049459  0.4054611  0.4059763  0.4064915  0.4070067  0.4075219
 0.40803709 0.40855229 0.40906749 0.40958269 0.41009789 0.41061309
 0.41112828 0.41164348 0.41215868 0.41267388 0.41318908 0.41370428
 0.41421947 0.41473467 0.41524987 0.41576507 0.41628027 0.41679547
 0.41731066 0.41782586 0.41834106 0.41885626 0.41937146 0.41988666
 0.42040185 0.42091705 0.42143225 0.42194745 0.42246265 0.42297785
 0.42349304 0.42400824 0.42452344 0.42503864 0.42555384 0.42606904
 0.42658423 0.42709943 0.42761463 0.42812983 0.42864503 0.42916023
 0.42967543 0.43019062 0.43070582 0.43122102 0.43173622 0.43225142
 0.43276662 0.43328181 0.43379701 0.43431221 0.43482741 0.43534261
 0.43585781 0.436373   0.4368882  0.4374034  0.4379186  0.4384338
 0.438949   0.43946419 0.43997939 0.44049459 0.44100979 0.44152499
 0.44204019 0.44255538 0.44307058 0.44358578 0.44410098 0.44461618
 0.44513138 0.44564657 0.44616177 0.44667697 0.44719217 0.44770737
 0.44822257 0.44873776 0.44925296 0.44976816 0.45028336 0.45079856
 0.45131376 0.45182895 0.45234415 0.45285935 0.45337455 0.45388975
 0.45440495 0.45492014 0.45543534 0.45595054 0.45646574 0.45698094
 0.45749614 0.45801133 0.45852653 0.45904173 0.45955693 0.46007213
 0.46058733 0.46110252 0.46161772 0.46213292 0.46264812 0.46316332
 0.46367852 0.46419371 0.46470891 0.46522411 0.46573931 0.46625451
 0.46676971 0.4672849  0.4678001  0.4683153  0.4688305  0.4693457
 0.4698609  0.47037609 0.47089129 0.47140649 0.47192169 0.47243689
 0.47295209 0.47346728 0.47398248 0.47449768 0.47501288 0.47552808
 0.47604328 0.47655848 0.47707367 0.47758887 0.47810407 0.47861927
 0.47913447 0.47964967 0.48016486 0.48068006 0.48119526 0.48171046
 0.48222566 0.48274086 0.48325605 0.48377125 0.48428645 0.48480165
 0.48531685 0.48583205 0.48634724 0.48686244 0.48737764 0.48789284
 0.48840804 0.48892324 0.48943843 0.48995363 0.49046883 0.49098403
 0.49149923 0.49201443 0.49252962 0.49304482 0.49356002 0.49407522
 0.49459042 0.49510562 0.49562081 0.49613601 0.49665121 0.49716641
 0.49768161 0.49819681 0.498712   0.4992272 ]
Value of positive_magnitudes:
 [2.76756000e+04 2.24655050e+03 2.05862500e+03 3.31722577e+03
 5.65887970e+03 5.85230221e+03 1.40108779e+03 1.67989273e+03
 5.64572221e+02 1.82331992e+03 7.23674125e+02 5.35563044e+01
 6.71365330e+02 4.21852882e+02 4.00377547e+02 1.39009587e+02
 3.48903448e+02 5.59244740e+01 5.11143037e+02 3.98467367e+02
 1.12086760e+02 1.70771351e+02 1.59063087e+02 1.08394488e+02
 1.62600326e+02 3.16712431e+02 9.63960411e+00 1.36438888e+02
 6.24085292e+01 2.15287231e+02 2.44759955e+02 1.89329870e+02
 5.71770362e+01 7.38927254e+01 1.02731008e+01 9.46353512e+01
 7.85189562e+01 9.77301680e+01 1.29944475e+02 1.22312561e+02
 1.56516482e+02 1.35211450e+02 1.91204871e+02 6.47918958e+01
 9.77747121e+01 9.96346721e+01 9.05047367e+01 4.37760036e+01
 1.40762502e+02 1.24155360e+02 1.61862502e+02 8.09580190e+01
 8.29431866e+01 1.03309481e+02 1.07047755e+02 1.30918050e+02
 6.79232066e+01 1.18541733e+02 9.94065615e+01 7.15436367e+01
 1.29598795e+02 1.15054435e+02 1.45515721e+02 7.52287358e+01
 4.30158341e+01 1.76324025e+02 9.56266640e+01 1.90667651e+02
 4.17816424e+01 5.94640968e+01 3.72631414e+01 1.70771471e+02
 4.30029069e+01 7.03205950e+01 1.15143878e+02 1.15652364e+02
 1.20446925e+02 3.01498934e+01 1.12233471e+02 1.08158593e+02
 1.58932235e+02 7.97272569e+01 3.03851836e+01 1.08244696e+02
 6.56211180e+01 8.67716614e+01 4.17053829e+01 3.95106139e+01
 8.83006930e+01 9.33517578e+01 1.65841325e+02 7.53988301e+01
 1.34462039e+01 4.18514151e+01 5.81057319e+01 9.35472172e+01
 5.31591358e+01 9.88302100e+01 2.11263511e+01 7.83265042e+01
 1.91085650e+02 9.84247989e+01 1.71752091e+02 4.85373181e+01
 1.36055304e+02 2.03695526e+02 3.05754059e+01 9.74491458e+01
 4.96096312e+01 1.51880792e+02 2.14555918e+01 9.37189070e+01
 7.53857209e+01 3.64063167e+01 7.31435437e+01 1.18247372e+02
 9.31601007e+01 1.69418419e+02 1.57820924e+01 1.16606494e+02
 1.02623043e+02 1.08860771e+02 8.03681735e+01 3.50613285e+01
 9.72482441e+01 5.59480382e+01 2.56247313e+01 8.84938440e+01
 1.58698853e+01 6.31113525e+01 6.94935406e+01 6.18533110e+01
 8.28974693e+01 9.60423690e+01 5.89423895e+01 2.73569643e+01
 1.20295677e+02 6.02608965e+01 7.26769135e+01 3.83501837e+01
 5.72291703e+01 6.23695415e+01 9.65193389e+01 1.17787360e+02
 9.57680730e+01 4.03383737e+01 4.68069294e+01 4.11237223e+01
 1.62430484e+02 1.60260718e+02 8.37204465e+01 5.46331021e+01
 2.45102693e+02 4.34446496e+01 7.87790783e+00 1.18656762e+02
 1.16332387e+02 2.19799753e+02 6.76919788e+01 4.12626699e+01
 1.60428899e+01 1.43103777e+02 5.29529612e+01 1.38536263e+02
 1.12292798e+02 7.68725538e+01 1.22593460e+02 6.52292582e+01
 8.93488172e+01 9.01266686e+00 6.50143176e+01 1.05387767e+02
 1.41041618e+02 2.80298863e+01 1.00152525e+02 4.85838631e+01
 2.54480889e+01 7.65606581e+01 5.52368803e+01 1.18752148e+02
 1.49939656e+02 5.72133642e+01 2.52321439e+01 7.08455016e+01
 5.88555457e+01 8.59451584e+01 1.01817878e+02 8.18982293e+00
 1.13165394e+02 4.34555274e+01 7.54020350e+01 4.13596600e+01
 1.59078842e+02 9.90602396e+01 4.45146761e+01 1.56263079e+02
 1.12817547e+02 6.76364742e+01 8.21752851e+01 1.47392134e+02
 1.80458635e+02 1.10792515e+02 1.76507980e+02 1.17951671e+02
 8.38261266e+01 1.04233237e+02 7.90490603e+01 6.79074905e+01
 1.60804520e+02 1.26478500e+02 1.66947582e+02 4.07349946e+01
 1.68833830e+02 1.33873604e+02 9.02878781e+01 5.42551996e+01
 5.65908342e+01 5.34632182e+01 2.70976281e+01 5.66532196e+01
 1.10407161e+02 2.19615481e+02 2.75062067e+01 4.88443840e+01
 1.13158182e+02 1.81036134e+02 1.77109510e+02 5.49333777e+01
 2.34443448e+01 8.03854330e+01 8.70443319e+01 3.58727386e+01
 9.09774936e+01 4.73395273e+01 9.11821600e+01 8.49989383e+01
 7.58856427e+01 4.77981521e+01 1.48037975e+02 2.43153259e+01
 4.58635032e+01 9.56340700e+01 5.26346908e+01 3.10418170e+01
 8.62917335e+01 5.98864392e+01 8.43749708e+01 1.38614406e+02
 1.79482217e+02 9.15997506e+01 1.43701433e+02 1.14171635e+02
 8.94838505e+01 1.15837780e+02 7.43350094e+01 1.68339711e+02
 8.94136727e+01 5.35365133e+01 1.02016435e+02 1.63396897e+02
 9.19445375e+01 2.13820176e+02 2.44799655e+01 5.85723974e+01
 9.92664701e+01 7.47821970e+01 9.62733884e+01 5.50676557e+01
 1.32217319e+02 5.45035180e+01 1.74934936e+02 9.53462714e+01
 2.01491349e+02 1.33717827e+02 1.01564086e+02 1.20968544e+02
 1.05678063e+02 9.13300148e+01 1.33288724e+02 8.23113758e+01
 1.06422044e+02 4.60622851e+01 4.28929258e+01 2.38449755e+02
 2.21027640e+02 5.59089032e+01 4.86652255e+01 9.13921660e+01
 7.45197341e+01 6.16231579e+01 1.03212222e+02 4.66478738e+01
 7.20875421e+01 9.91446658e+01 1.17747789e+02 3.95670583e+01
 1.23553249e+02 1.23424952e+02 1.74689717e+02 6.80404062e+01
 7.85034651e+01 5.87921164e+01 9.74404509e+01 2.50933227e+01
 1.12557700e+02 1.24993753e+02 9.41722507e+01 3.11768283e+01
 1.84538719e+02 3.46409306e+01 3.17533359e+01 7.87821445e+01
 3.35254952e+01 3.52639691e+01 3.61689925e+01 6.36934540e+01
 2.26775119e+01 1.05446583e+02 5.17868836e+01 3.48850005e+01
 1.11217277e+02 8.27770133e+01 1.80432449e+01 2.98264703e+01
 6.93190739e+01 5.03780506e+01 6.49279698e+01 6.34106060e+01
 1.21596239e+02 6.12669282e+01 7.57234061e+01 4.85739496e+01
 9.27769222e+01 7.43473177e+01 7.83004617e+01 2.61886527e+01
 7.60853091e+01 6.33793970e+01 1.93332253e+02 4.98068989e+01
 1.00155682e+02 8.83125013e+01 2.41094581e+02 1.86895223e+02
 3.74701903e+01 3.45755980e+01 1.00949276e+02 2.13101512e+02
 1.07444945e+02 8.53287843e+01 9.44934279e+01 1.14840076e+02
 1.17097176e+02 9.24904349e+01 2.14730846e+02 8.35057418e+01
 1.16334170e+02 1.31928532e+02 8.70218471e+01 2.17592993e+01
 6.02902479e+01 3.52792962e+01 1.04851088e+02 1.94432474e+02
 8.48208350e+01 1.99571716e+01 1.15501630e+02 6.91283419e+01
 1.81436620e+01 1.71768133e+01 1.02710098e+02 1.13468896e+02
 2.68258895e+01 1.69909755e+02 1.53609762e+02 1.03263190e+02
 4.64210814e+01 8.28258803e+01 2.02449936e+02 8.23998380e+01
 1.44482447e+02 9.57864560e+01 1.94787092e+02 8.91252241e+01
 5.51544653e+01 1.84400687e+02 1.19946407e+02 5.77854390e+01
 1.26523996e+02 7.39748362e+01 7.73935659e+01 1.28529899e+02
 6.76493818e+01 4.07714942e+01 3.13142789e+01 8.41716144e+01
 6.83410373e+01 1.58839291e+02 6.48048393e+01 1.15736663e+02
 1.07228414e+02 7.24713007e+01 1.10854672e+02 3.47395869e+01
 3.09928369e+01 1.74579345e+01 3.25143816e+01 5.44321313e+01
 1.08380112e+02 9.81509188e+01 1.05973910e+02 8.54079267e+01
 6.99095486e+01 1.49107787e+02 4.39525173e+01 4.45069978e+01
 1.92236371e+02 1.66543150e+02 5.87903283e+01 1.61924865e+01
 9.88134745e+01 6.44517330e+01 1.50171835e+02 1.44266251e+02
 5.47222200e+01 5.85316078e+01 1.57987039e+02 8.55190521e+01
 1.07497692e+02 6.92544422e+01 1.26341745e+02 1.13519874e+02
 7.97389425e+01 9.71357193e+01 6.53982483e+01 1.26126599e+02
 9.94166485e+01 5.45198618e+01 6.22763467e+01 1.02680436e+02
 9.42764224e+01 1.25730547e+02 9.94756554e+01 1.94338938e+02
 1.68136994e+02 2.26288305e+01 7.22569979e+01 1.57284825e+02
 1.50196907e+02 1.26338290e+02 6.48104266e+01 1.02763168e+02
 1.54498230e+01 5.07838790e+01 3.91806793e+01 5.00945994e+01
 8.41988066e+01 8.06724671e+01 4.74664798e+01 4.25890779e+01
 1.28713914e+02 9.76285964e+01 2.93435054e+01 8.47931574e+01
 4.90878318e+01 6.77451947e+01 1.67474546e+02 1.11851879e+02
 9.46951864e+01 4.64441299e+01 6.19290268e+01 5.57513825e+01
 1.87181027e+02 1.15557993e+02 7.56379385e+01 4.99810378e+01
 1.86373240e+02 9.43429171e+01 1.95727413e+02 1.31683930e+02
 8.90519780e+00 8.56843875e+01 6.21230919e+01 9.74164003e+01
 1.06595999e+02 4.34192235e+01 9.36576102e+01 1.86650272e+02
 1.02367541e+02 2.51235161e+01 1.03586240e+02 2.30133647e+01
 1.50746863e+02 1.17690494e+02 2.30673552e+02 1.77779679e+01
 8.54060473e+01 6.18856647e+01 6.68096918e+01 3.97181675e+01
 8.42247428e+01 7.17007633e+01 1.00622700e+02 8.54309901e+01
 7.11853450e+01 9.40756489e+01 4.63029892e+01 4.66887332e+01
 7.97261921e+01 4.88928477e+01 5.47229883e+01 7.79716632e+01
 1.27457066e+02 6.36525942e+01 5.20422688e+01 1.84423122e+02
 1.03165090e+02 1.40081050e+02 1.37681453e+02 1.39724557e+02
 1.71148661e+02 1.71834841e+01 7.54113296e+01 3.00295404e+01
 1.01577169e+02 8.66439831e+01 1.27415889e+02 7.79840028e+01
 5.30488746e+01 5.65098589e+01 5.84698436e+01 8.61910845e+01
 9.77733029e+01 1.49716423e+02 6.71532095e+01 7.45993203e+01
 7.49746451e+01 9.36245859e+01 4.38367217e+01 4.63459925e+01
 8.07363734e+01 1.01269345e+02 1.90712928e+02 2.37116282e+01
 1.07354040e+02 2.99338294e+01 8.10208805e+01 1.48931905e+02
 6.82288286e+01 1.43560664e+02 1.17935260e+02 6.02136523e+01
 1.76145959e+02 6.42482929e+01 1.05785266e+02 7.22931249e+01
 1.64606817e+02 5.89526960e+01 9.87818484e+01 1.03751014e+02
 8.24629284e+01 5.13321181e+01 9.59088792e+00 8.49945312e+01
 1.13406436e+02 1.00292521e+02 6.34146453e+01 1.50771475e+01
 7.41208930e+01 8.81670873e+01 9.72356753e+01 2.13920402e+01
 9.82960146e+01 9.50671516e+01 1.95756721e+02 1.11723334e+02
 1.75347757e+01 5.16352227e+01 1.76491543e+02 7.89199412e+01
 2.78549141e+01 6.74666989e+01 1.18286793e+02 1.73291881e+02
 1.37468805e+02 3.87025717e+01 1.41662815e+01 8.85503362e+01
 9.69850597e+01 1.07786191e+02 1.11237650e+02 4.81683280e+01
 1.53316013e+02 2.48069741e+01 5.62864807e+01 7.48279405e+01
 3.41295960e+01 4.33281077e+01 1.38475748e+02 4.02217242e+01
 9.25748874e+01 9.00289847e+01 6.60580903e+01 5.23039244e+01
 3.84740325e+01 1.39224960e+02 6.01141916e+01 1.15688515e+02
 6.72791977e+01 8.02965143e+01 6.61012660e+01 5.93503983e+01
 8.62701546e+01 3.46421047e+01 6.82530374e+01 8.95413044e+01
 1.32270250e+02 1.73598857e+02 8.06817223e+01 1.22685718e+02
 2.09866843e+02 1.13669036e+02 8.00348116e+01 1.91700400e+01
 1.46555349e+02 4.88639760e+01 5.93156831e+01 6.48658881e+01
 2.96823326e+01 1.72337704e+02 5.74182270e+01 3.57910034e+01
 1.19374918e+02 1.07153290e+02 9.99301121e+01 1.36072898e+02
 7.36237094e+01 6.01247890e+01 1.22484359e+02 6.67149071e+01
 7.31534254e+01 1.48030562e+02 7.79566798e+01 6.48953382e+01
 8.74833986e+01 5.03527099e+01 1.00338301e+02 9.76772235e+01
 4.84151341e+01 1.00541010e+02 6.11369177e+01 9.40935618e+01
 1.52000741e+02 1.13399451e+02 5.30256830e+01 4.59530159e+01
 6.65136147e+01 1.59260373e+02 1.36507221e+02 1.20773716e+02
 1.93111821e+02 1.42933770e+02 7.08664699e+01 7.53607017e+01
 1.79038608e+02 1.91059317e+02 3.82269199e+01 9.62926755e+01
 1.45305761e+02 5.91422614e+01 9.34963276e+00 4.43815241e+01
 1.74169450e+02 1.11232014e+02 7.92475326e+01 6.36323522e+01
 5.23624344e+01 1.63434083e+02 7.22833247e+01 3.88797340e+01
 6.18713135e+01 8.50403514e+00 6.66337642e+01 1.40395191e+02
 1.14631479e+02 1.42105187e+02 7.65251667e+01 1.01657027e+02
 1.83835253e+02 8.29096763e+01 3.06624942e+01 1.31503641e+02
 1.69798402e+02 1.33962817e+02 8.12774249e+01 3.33678130e+01
 1.10262212e+02 1.34016186e+02 5.33119504e+01 5.66525917e+01
 1.54039021e+02 8.03652205e+01 7.95405369e+01 1.61353700e+02
 1.10072107e+02 3.00348983e+01 8.25720425e+01 1.15183311e+01
 7.46541748e+01 9.25716029e+01 3.56467434e+01 1.26571761e+02
 2.95575951e+01 6.17014924e+01 1.03745852e+02 8.82330452e+01
 6.71669057e+01 8.11215785e+01 9.00620286e+01 5.23934539e+01
 9.07006520e+01 4.76763357e+01 5.65878863e+01 2.97755264e+01
 9.22328424e+01 1.42872887e+02 9.57453488e+01 1.53498452e+02
 7.96747699e+01 1.36258093e+02 6.72995889e+01 1.36875770e+02
 4.48607641e+01 1.86053420e+02 2.14519090e+02 1.85076520e+02
 5.10228295e+01 9.23896014e+01 1.80358862e+02 1.31916503e+02
 7.30091844e+01 1.39317389e+02 1.87283015e+01 5.01179347e+01
 7.26447301e+01 1.20809425e+02 1.57215073e+02 4.06495223e+01
 4.72123524e+01 7.19343296e+01 2.17137644e+02 1.04105573e+02
 2.98971231e+01 3.80992717e+01 6.27746124e+01 7.90400395e+01
 4.19138365e+01 1.11690957e+02 1.54064477e+02 1.10236745e+02
 1.15673486e+02 1.46914338e+02 5.86699061e+01 8.95902350e+01
 1.43675996e+02 6.27882804e+01 5.88008962e+01 4.35369077e+01
 1.67927459e+02 8.25383498e+01 6.54888266e+01 9.62323834e+01
 8.25302452e+01 1.38687518e+02 8.15789814e+01 8.35088729e+01
 8.86755887e+01 5.22319341e+01 7.77500338e+01 1.76998345e+02
 7.08254497e+01 1.29178382e+02 1.60185614e+02 1.01130515e+02
 3.23481586e+01 1.00171515e+02 4.33834788e+01 2.07791124e+02
 1.60790258e+02 2.36004464e+01 6.22048649e+01 1.07864129e+02
 3.10262262e+01 1.43250649e+02 7.86817231e+01 1.92569388e+02
 1.88361761e+01 2.11263688e+02 9.00578942e+01 1.39352906e+02
 1.03998656e+02 1.51969222e+02 1.27292662e+02 7.06892552e+01
 1.11384020e+02 1.67189304e+02 1.33917262e+02 3.95858680e+01
 2.16207357e+02 8.96760124e+01 1.09626948e+02 8.74501329e+01
 2.67193936e+01 6.09543501e+01 1.06432573e+02 5.14968675e+01
 5.36238745e+01 3.12903630e+01 1.52360205e+02 1.47420017e+02
 2.66635414e+01 1.31521555e+02 9.24133476e+01 7.27440976e+01
 1.27989704e+02 3.67905123e+01 6.80205225e+01 2.76370402e+01
 1.10317536e+02 2.42087567e+01 1.96921794e+02 1.55609140e+02
 4.76161032e+01 8.67370832e+01 2.91768348e+02 1.31853065e+02
 4.95760026e+01 7.48232186e+01 8.97418892e+01 1.56012847e+02
 8.83718817e+01 1.14679068e+02 1.20936742e+02 4.01098503e+01
 1.03885557e+02 5.27380771e+01 1.54343063e+02 5.70257647e+01
 1.30608116e+02 1.21336979e+02 1.60477676e+02 1.76748781e+02
 1.33511304e+02 5.22874655e+01 2.50616268e+01 1.55229123e+02
 9.81976828e+01 1.42194157e+02 1.44913059e+02 8.51785505e+00
 2.49271006e+02 1.33320120e+02 5.52842715e+01 1.37868529e+02
 1.50938769e+02 1.87152555e+02 8.95096193e+01 6.63737405e+01
 1.35474806e+02 1.20082499e+02 1.78058013e+02 7.95497741e+01
 1.30464692e+02 9.41655957e+01 5.92160704e+01 1.30729113e+02
 1.02706511e+02 6.39821198e+01 1.23736318e+02 1.24787035e+02
 6.08997414e+01 6.38964863e+01 7.69494173e+01 2.25210240e+01
 1.05944168e+02 1.77071960e+02 4.69376315e+01 3.25942469e+01
 4.88716291e+00 5.91960685e+01 2.22348714e+01 1.02631137e+02
 1.73581958e+02 1.02130004e+02 6.89881938e+01 1.01636532e+02
 1.87284075e+02 1.02949679e+02 2.56207909e+02 1.03667290e+02
 3.74424431e+01 9.87938409e+01 8.59872868e+01 1.68183894e+02
 4.88119692e+01 1.20282756e+02 1.87136106e+01 1.30564956e+02
 1.07709069e+02 5.71344721e+01 7.57803186e+01 1.59714536e+02
 1.16947465e+02 6.01673841e+01 1.59571465e+02 9.51447329e+01
 7.61628059e+01 9.90186935e+01 1.68853234e+01 8.59869413e+01
 4.48569112e+01 1.92291889e+02 7.06267507e+01 6.56065738e+01
 3.91654650e+01 7.82629526e+01 1.28193125e+02 1.09222801e+02
 7.89346928e+01 3.16403050e+01 8.06049994e+01 1.75545049e+02
 1.07189422e+02 7.33548780e+01 1.09144258e+02 9.99254046e+01
 4.86261695e+01 5.34278962e+01 1.69405186e+01 4.55046170e+01
 9.46854769e+01 1.87240196e+02 9.83656985e+01 5.07637900e+01
 1.13255285e+02 7.55848307e+01 1.62959339e+02 5.56721413e+01
 3.57086263e+00 4.40287246e+01 4.58253605e+01 4.98198973e+01
 8.63619642e+01 1.02932808e+02 4.91502387e+01 1.30422788e+02
 1.34685675e+02 7.37680862e+01 2.00244606e+02 4.71433910e+01
 9.71773768e+01 2.26482981e+01 1.09191105e+02 1.42996536e+02
 6.30175795e+01 1.03982985e+02 2.26379411e+02 1.63259378e+01
 7.85741884e+01 2.23194086e+01]
```



```python
from scipy.signal import butter, filtfilt

cutoff_frequency = 0.005  # This is a normalized frequency, where 1.0 corresponds to Nyquist frequency (sampling_rate / 2)
csv_file_path = 'kalibrasi 40 kedua.csv'  # Replace with your CSV file path
data = pd.read_csv(csv_file_path)
speed_data = data['Speed'].values
# Set up the filter
order = 4  # Order of the filter (adjust as needed)
b, a = butter(order, cutoff_frequency, btype='high')

# Apply the filter to your speed data
filtered_speed_data = filtfilt(b, a, speed_data)

print('filteret data :',filtered_speed_data)
# Plot original and filtered speed data
```

```
plt.figure(figsize=(10, 6))
plt.plot(positive_freqs, positive_magnitudes, label='Original
Data',alpha = 0.5,color='red')
plt.plot(positive_freqs, np.abs(np.fft.fft(filtered_speed_data))
[:len(positive_magnitudes)], label='Filtered Data',color = 'blue')
plt.title('Frequency Spectrum of Original and Filtered Speed Data')
plt.xlabel('Frequency (Hz)')
plt.ylabel('Magnitude')
plt.legend()
plt.grid()
plt.show()

filteret data : [-3.902595   -3.50535428 -3.60866987 ... -1.10438594 -
1.03819578
 -0.97208241]
```

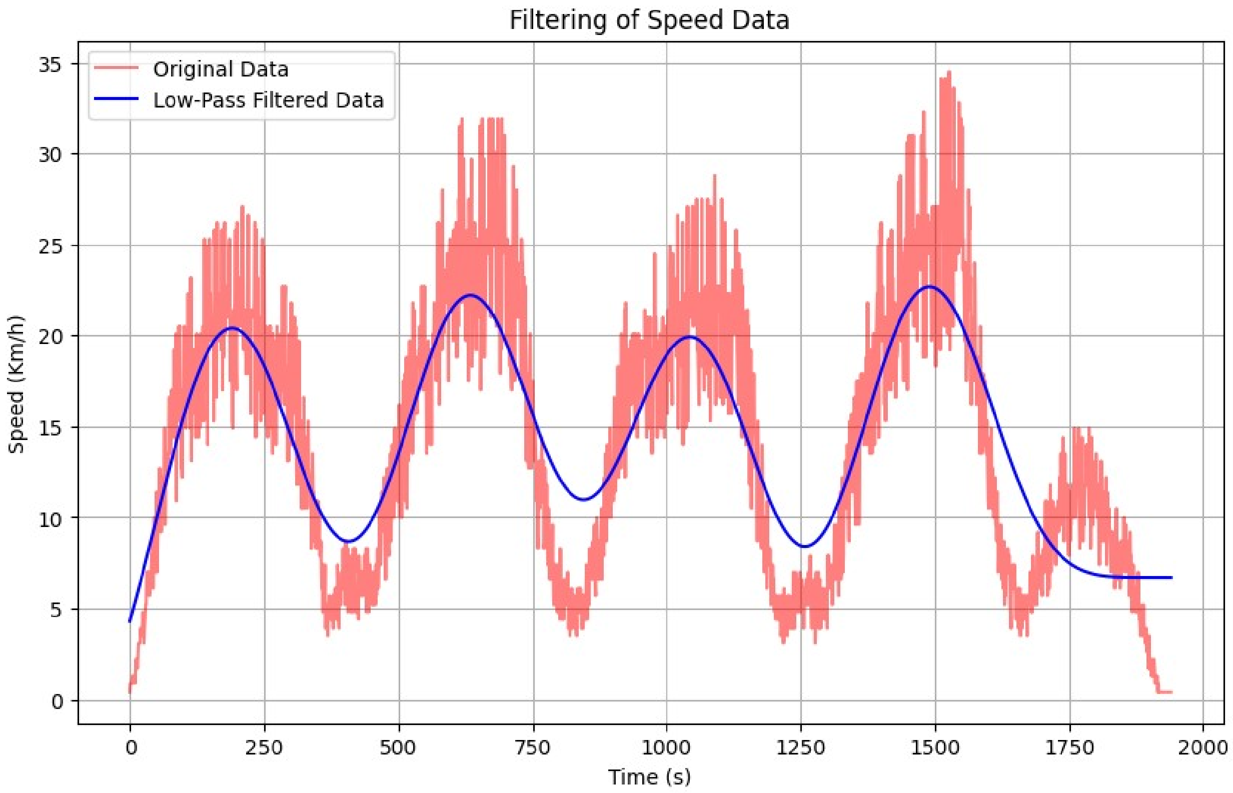


PENERAPAN NOICE PADA KALIBRASI 40 KMPH dengan

```
low_cutoff_frequency = 0.005  # Low-pass filter cutoff frequency
high_cutoff_frequency = 0.1  # High-pass filter cutoff frequency

# Read data from CSV file
csv_file_path = 'kalibrasi 40 kedua.csv'  # Replace with your CSV file
path
data = pd.read_csv(csv_file_path)
speed_data = data['Speed'].values
```

```
# Set up the low-pass filter
low_order = 4  # Order of the low-pass filter (adjust as needed)
low_b, low_a = butter(low_order, low_cutoff_frequency, btype='low')

# Apply the low-pass filter to the speed data
low_filtered_speed_data = filtfilt(low_b, low_a, speed_data)

# Set up the high-pass filter
high_order = 4  # Order of the high-pass filter (adjust as needed)
high_b, high_a = butter(high_order, high_cutoff_frequency,
btype='high')

# Apply the high-pass filter to the low-pass filtered data
high_filtered_speed_data = filtfilt(high_b, high_a,
low_filtered_speed_data)

print("Filter BPF : ",low_filtered_speed_data)
print("Flter HPF: ",high_filtered_speed_data)
# Plot original, low-pass filtered, and high-pass filtered speed data
plt.figure(figsize=(10, 6))
plt.plot(data.index, speed_data, label='Original Data', alpha=0.5,
color='red')
plt.plot(data.index, low_filtered_speed_data, label='Low-Pass Filtered
Data', color='blue')
# plt.plot(data.index, high_filtered_speed_data, label='High-Pass
Filtered Data', color='green')
plt.title('Filtering of Speed Data')
plt.xlabel('Time (s)')
plt.ylabel('Speed (Km/h)')
plt.legend()
plt.grid()
plt.show()
```

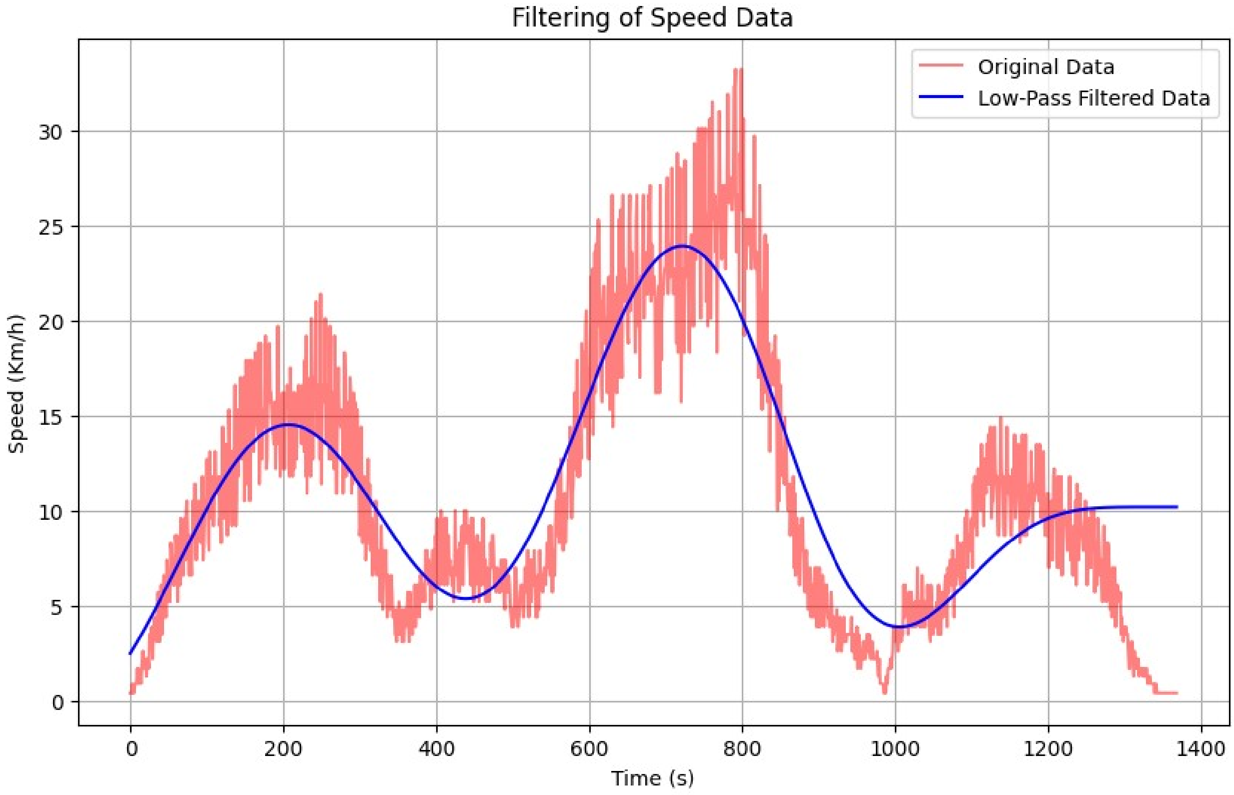
```
Filter BPF :  [4.30256783 4.40532679 4.50864207 ... 6.70396315
6.70396309 6.70396304]
Flter HPF:  [-1.03556516e-02 -1.21893130e-02 -1.23608925e-02 ... -
1.59586140e-08
 -1.04001261e-08  3.10964980e-09]
```



```
low_cutoff_frequency = 0.005  # Low-pass filter cutoff frequency
high_cutoff_frequency = 0.1  # High-pass filter cutoff frequency

# Read data from CSV file
csv_file_path = 'kalibrasi 40 ketiga.csv'  # Replace with your CSV
file path
data = pd.read_csv(csv_file_path)
speed_data = data['Speed'].values

# Set up the low-pass filter
low_order = 4  # Order of the low-pass filter (adjust as needed)
low_b, low_a = butter(low_order, low_cutoff_frequency, btype='low')

# Apply the low-pass filter to the speed data
low_filtered_speed_data = filtfilt(low_b, low_a, speed_data)

# Set up the high-pass filter
high_order = 4  # Order of the high-pass filter (adjust as needed)
high_b, high_a = butter(high_order, high_cutoff_frequency,
btype='high')

# Apply the high-pass filter to the low-pass filtered data
high_filtered_speed_data = filtfilt(high_b, high_a,
low_filtered_speed_data)
```

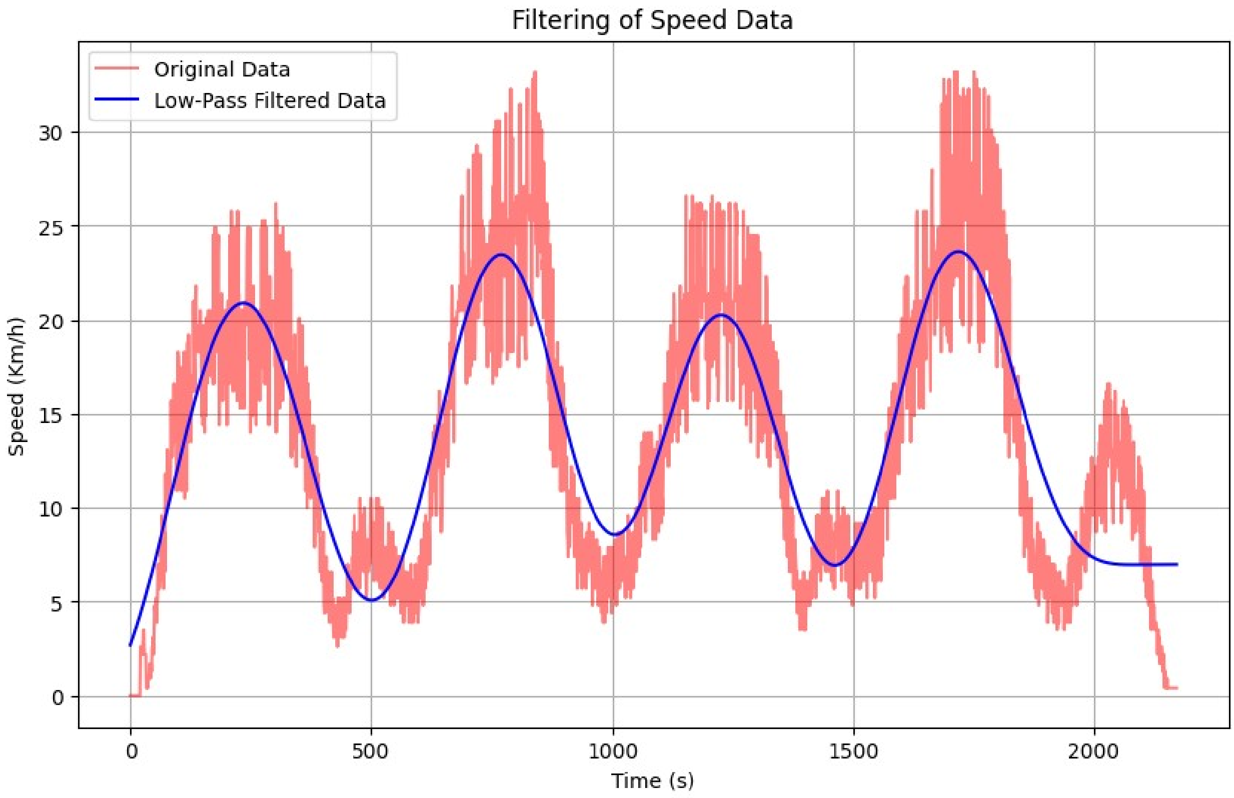
```
print("Filter BPF : ",low_filtered_speed_data)
print("Flter HPF: ",high_filtered_speed_data)
# Plot original, low-pass filtered, and high-pass filtered speed data
plt.figure(figsize=(10, 6))
plt.plot(data.index, speed_data, label='Original Data', alpha=0.5,
color='red')
plt.plot(data.index, low_filtered_speed_data, label='Low-Pass Filtered
Data', color='blue')
# plt.plot(data.index, high_filtered_speed_data, label='High-Pass
Filtered Data', color='green')
plt.title('Filtering of Speed Data')
plt.xlabel('Time (s)')
plt.ylabel('Speed (Km/h)')
plt.legend()
plt.grid()
plt.show()
```

```
Filter BPF :  [ 2.49073572  2.55630464  2.62229806 ... 10.18225079
10.18224932
 10.18224814]
Flter HPF:  [-6.73881740e-03 -8.02080562e-03 -8.17529827e-03 ... -
5.35270705e-07
 -3.70294979e-07 -1.70130227e-08]
```



```
low_cutoff_frequency = 0.005  # Low-pass filter cutoff frequency
high_cutoff_frequency = 0.1  # High-pass filter cutoff frequency

# Read data from CSV file
csv_file_path = 'kalibrasi 40.csv'  # Replace with your CSV file path
data = pd.read_csv(csv_file_path)
speed_data = data['Speed'].values

# Set up the low-pass filter
low_order = 4  # Order of the low-pass filter (adjust as needed)
low_b, low_a = butter(low_order, low_cutoff_frequency, btype='low')

# Apply the low-pass filter to the speed data
low_filtered_speed_data = filtfilt(low_b, low_a, speed_data)

# Set up the high-pass filter
high_order = 4  # Order of the high-pass filter (adjust as needed)
high_b, high_a = butter(high_order, high_cutoff_frequency,
btype='high')

# Apply the high-pass filter to the low-pass filtered data
high_filtered_speed_data = filtfilt(high_b, high_a,
low_filtered_speed_data)

print("Filter BPF : ",low_filtered_speed_data)
print("Flter HPF: ",high_filtered_speed_data)
# Plot original, low-pass filtered, and high-pass filtered speed data
plt.figure(figsize=(10, 6))
plt.plot(data.index, speed_data, label='Original Data', alpha=0.5,
color='red')
plt.plot(data.index, low_filtered_speed_data, label='Low-Pass Filtered
Data', color='blue')
# plt.plot(data.index, high_filtered_speed_data, label='High-Pass
Filtered Data', color='green')
plt.title('Filtering of Speed Data')
plt.xlabel('Time (s)')
plt.ylabel('Speed (Km/h)')
plt.legend()
plt.grid()
plt.show()
```

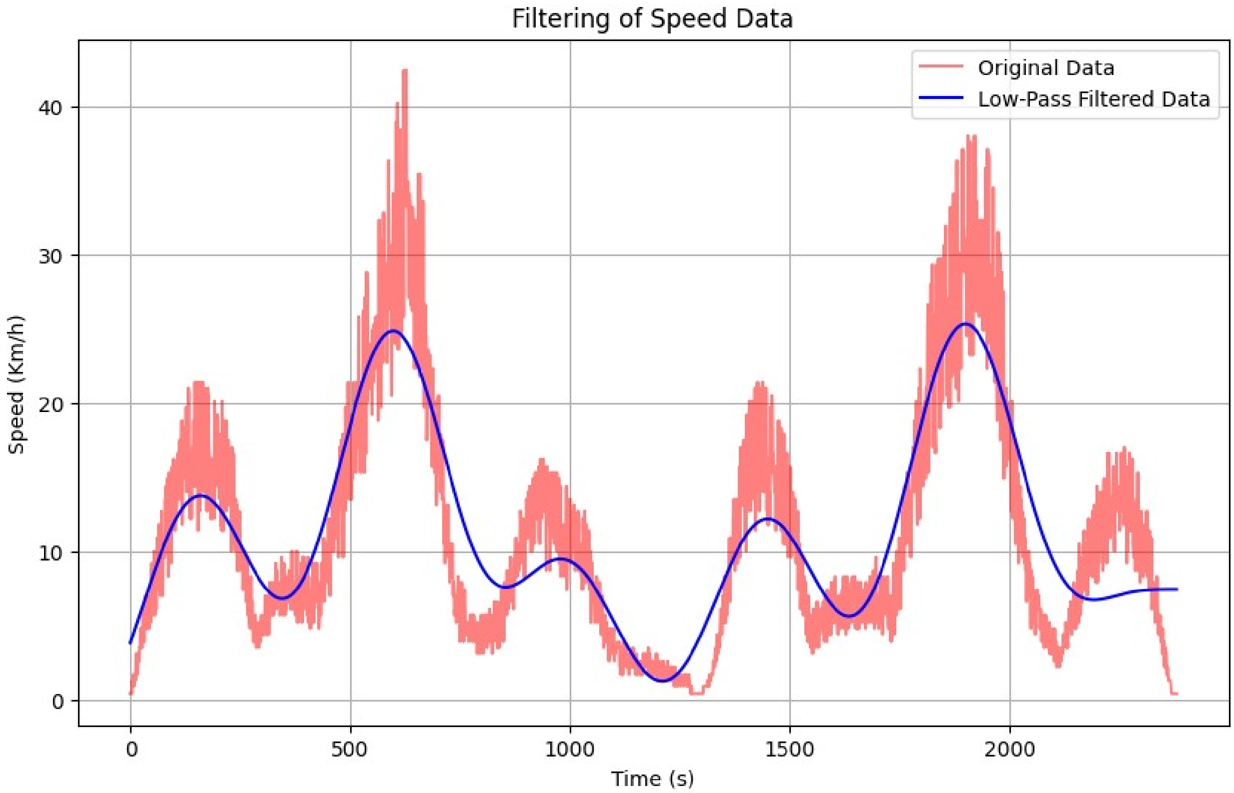
```
Filter BPF :  [2.69242908 2.76566314 2.83956848 ... 6.988108
6.98810982 6.98811126]
Flter HPF:  [-7.88478976e-03 -9.63653230e-03 -9.93936664e-03 ...
6.32845460e-07
  4.45044467e-07  1.93194901e-08]
```



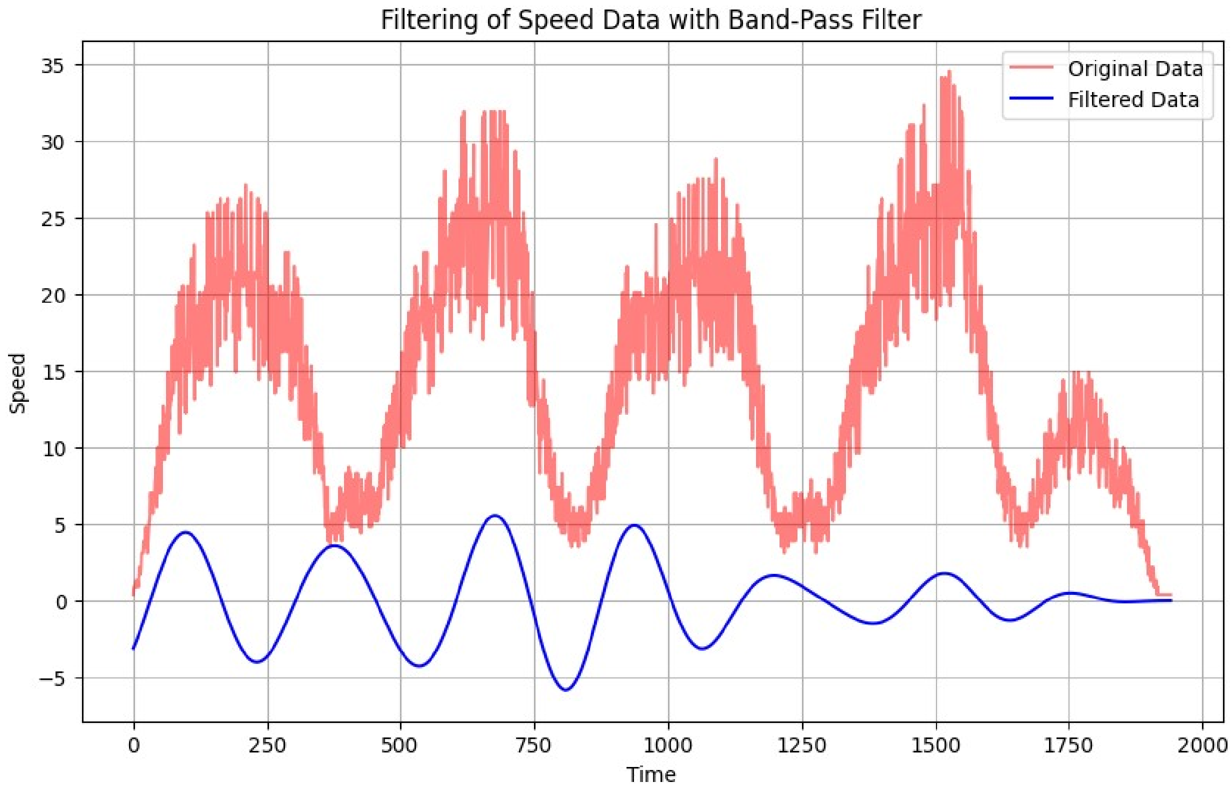
```
low_cutoff_frequency = 0.005  # Low-pass filter cutoff frequency
high_cutoff_frequency = 0.1  # High-pass filter cutoff frequency

# Read data from CSV file
csv_file_path = 'kalibrasi 40 keempat.csv'  # Replace with your CSV
file path
data = pd.read_csv(csv_file_path)
speed_data = data['Speed'].values

# Set up the low-pass filter
low_order = 4  # Order of the low-pass filter (adjust as needed)
low_b, low_a = butter(low_order, low_cutoff_frequency, btype='low')

# Apply the low-pass filter to the speed data
low_filtered_speed_data = filtfilt(low_b, low_a, speed_data)

# Set up the high-pass filter
high_order = 4  # Order of the high-pass filter (adjust as needed)
high_b, high_a = butter(high_order, high_cutoff_frequency,
btype='high')

# Apply the high-pass filter to the low-pass filtered data
high_filtered_speed_data = filtfilt(high_b, high_a,
low_filtered_speed_data)
```

```python
print("Filter BPF : ",low_filtered_speed_data)
print("Flter HPF: ",high_filtered_speed_data)
# Plot original, low-pass filtered, and high-pass filtered speed data
plt.figure(figsize=(10, 6))
plt.plot(data.index, speed_data, label='Original Data', alpha=0.5,
color='red')
plt.plot(data.index, low_filtered_speed_data, label='Low-Pass Filtered
Data', color='blue')
# plt.plot(data.index, high_filtered_speed_data, label='High-Pass
Filtered Data', color='green')
plt.title('Filtering of Speed Data')
plt.xlabel('Time (s)')
plt.ylabel('Speed (Km/h)')
plt.legend()
plt.grid()
plt.show()
```

```
Filter BPF :  [3.82553906 3.9114263  3.99758231 ... 7.43153215
7.43153929 7.43154491]
Flter HPF:  [-8.29174616e-03 -9.50730739e-03 -9.52248520e-03 ...
2.44678894e-06
  1.72086374e-06  5.99849476e-08]
```



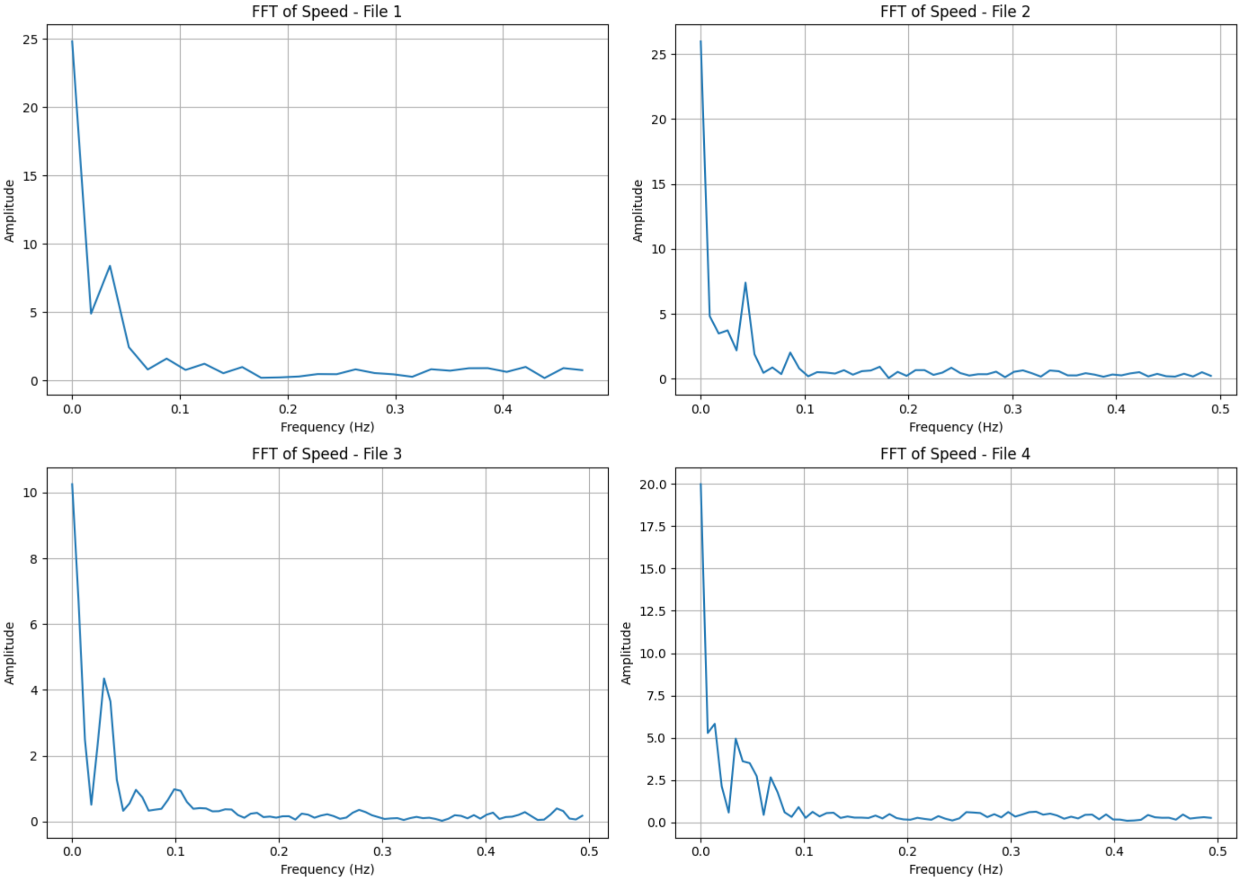
```python
# Define cutoff frequencies for the band-pass filter
low_cutoff_frequency = 0.005  # Low-pass filter cutoff frequency
high_cutoff_frequency = 0.01  # High-pass filter cutoff frequency

# Read data from CSV file
csv_file_path = 'kalibrasi 40 kedua.csv'  # Replace with your CSV file
path
data = pd.read_csv(csv_file_path)
speed_data = data['Speed'].values

# Set up the band-pass filter
order = 4  # Order of the band-pass filter (adjust as needed)
b, a = butter(order, [low_cutoff_frequency, high_cutoff_frequency],
btype='band')

# Apply the band-pass filter to the speed data
filtered_speed_data = filtfilt(b, a, speed_data)

# Plot original and filtered speed data
plt.figure(figsize=(10, 6))
plt.plot(data.index, speed_data, label='Original Data', alpha=0.5,
color='red')
plt.plot(data.index, filtered_speed_data, label='Filtered Data',
color='blue')
plt.title('Filtering of Speed Data with Band-Pass Filter')
plt.xlabel('Time')
plt.ylabel('Speed')
plt.legend()
plt.grid()
plt.show()
```



```python
def process_file(file_path):
    # Membaca data dari file CSV
    df = pd.read_csv(file_path)

    # Mengonversi kolom Timestamp ke tipe datetime
    df['Timestamp'] = pd.to_datetime(df['Timestamp'])

    # Menghapus data yang tidak diperlukan
    df = df.drop(df.index[2305:2383])

    # Mengonversi kolom Speed ke numerik
    df['Speed'] = pd.to_numeric(df['Speed'], errors='coerce')

    # Menghapus baris dengan nilai Speed yang hilang
    df = df.dropna(subset=['Speed'])

    # Set the index to the Timestamp column
    df.set_index('Timestamp', inplace=True)

    # Remove duplicate index values
    df = df[~df.index.duplicated(keep='first')]

    # Resampling dengan interval waktu yang tetap, misalnya 1 detik
    resampled_df = df.resample('1S').interpolate('linear')

    # Mengambil data kecepatan dari data yang telah di-resample
    speeds = resampled_df['Speed'].values

    # Interval sampling setelah resampling (1 detik)
    sampling_rate = 1.0  # Sampling interval in seconds
    N = len(speeds)

    # Menghitung FFT
    fft_result = np.fft.fft(speeds)
    fft_magnitude = np.abs(fft_result) / N  # Normalize the FFT output
    frequencies = np.fft.fftfreq(N, d=1/sampling_rate)

    # Only return the positive frequencies and their magnitudes
    positive_frequencies = frequencies[:N//2]
    positive_magnitudes = fft_magnitude[:N//2] * 2

    return positive_frequencies, positive_magnitudes

# Daftar file paths
file_paths = [
    'kalibrasi30-40.csv',
    'kalibrasi 40 kedua.csv',
    'kalibrasi 40 ketiga.csv',
    'kalibrasi 40 keempat.csv',
]

# Plot hasil FFT untuk setiap file
plt.figure(figsize=(14, 10))

for i, file_path in enumerate(file_paths):
    frequencies, magnitudes = process_file(file_path)
    plt.subplot(2, 2, i+1)
    plt.plot(frequencies, magnitudes)
    plt.title(f'FFT of Speed - File {i+1}')
    plt.xlabel('Frequency (Hz)')
    plt.ylabel('Amplitude')
    plt.grid()

plt.tight_layout()
plt.show()
```

```
C:\Users\M ALIF F\AppData\Local\Temp\ipykernel_19848\749855980.py:24:
FutureWarning: 'S' is deprecated and will be removed in a future
version, please use 's' instead.
  resampled_df = df.resample('1S').interpolate('linear')
C:\Users\M ALIF F\AppData\Local\Temp\ipykernel_19848\749855980.py:24:
FutureWarning: DataFrame.interpolate with object dtype is deprecated
and will raise in a future version. Call obj.infer_objects(copy=False)
before interpolating instead.
  resampled_df = df.resample('1S').interpolate('linear')
C:\Users\M ALIF F\AppData\Local\Temp\ipykernel_19848\749855980.py:24:
```

```
FutureWarning: 'S' is deprecated and will be removed in a future
version, please use 's' instead.
  resampled_df = df.resample('1S').interpolate('linear')
C:\Users\M ALIF F\AppData\Local\Temp\ipykernel_19848\749855980.py:24:
FutureWarning: DataFrame.interpolate with object dtype is deprecated
and will raise in a future version. Call obj.infer_objects(copy=False)
before interpolating instead.
  resampled_df = df.resample('1S').interpolate('linear')
C:\Users\M ALIF F\AppData\Local\Temp\ipykernel_19848\749855980.py:24:
FutureWarning: 'S' is deprecated and will be removed in a future
version, please use 's' instead.
  resampled_df = df.resample('1S').interpolate('linear')
C:\Users\M ALIF F\AppData\Local\Temp\ipykernel_19848\749855980.py:24:
FutureWarning: DataFrame.interpolate with object dtype is deprecated
and will raise in a future version. Call obj.infer_objects(copy=False)
before interpolating instead.
  resampled_df = df.resample('1S').interpolate('linear')
C:\Users\M ALIF F\AppData\Local\Temp\ipykernel_19848\749855980.py:24:
FutureWarning: 'S' is deprecated and will be removed in a future
version, please use 's' instead.
  resampled_df = df.resample('1S').interpolate('linear')
C:\Users\M ALIF F\AppData\Local\Temp\ipykernel_19848\749855980.py:24:
FutureWarning: DataFrame.interpolate with object dtype is deprecated
and will raise in a future version. Call obj.infer_objects(copy=False)
before interpolating instead.
  resampled_df = df.resample('1S').interpolate('linear')
```


FFT of Speed - File 1
Amplitude
25
20
15
10
5
0
0.0
0.1
0.2
0.3
0.4
Frequency (Hz)
FFT of Speed - File 2
Amplitude
25
20
15
10
5
0
0.0
0.1
0.2
0.3
0.4
0.5
Frequency (Hz)
FFT of Speed - File 3
Amplitude
10
8
6
4
2
0
0.0
0.1
0.2
0.3
0.4
0.5
Frequency (Hz)
FFT of Speed - File 4
Amplitude
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0.0
0.1
0.2
0.3
0.4
0.5
Frequency (Hz)