# da_Mushroom_25-05-08_0326-no_stimulation3

May 14, 2025

[1]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import os

# Set file path
file_path = '../data/Mushroom_25-05-08_0326.lvm'

# Check if file exists
if not os.path.exists(file_path):
    print(f"Error: File {file_path} does not exist")
else:
    # Read LVM file
    # LVM files are tab-separated text files without header
    data = pd.read_csv(file_path, sep='\t', header=None)

    # Display basic information about the data
    print(f"Data shape: {data.shape}")
    print("\nFirst 5 rows of data:")
    print(data.head())

    # Based on file content, we need to name the columns
    # Assuming first column is timestamp, others are sensor data
    columns = ['Timestamp'] + [f'Sensor_{i}' for i in range(1, data.shape[1])]
    data.columns = columns

    print("\nData after renaming columns:")
    print(data.head())
```

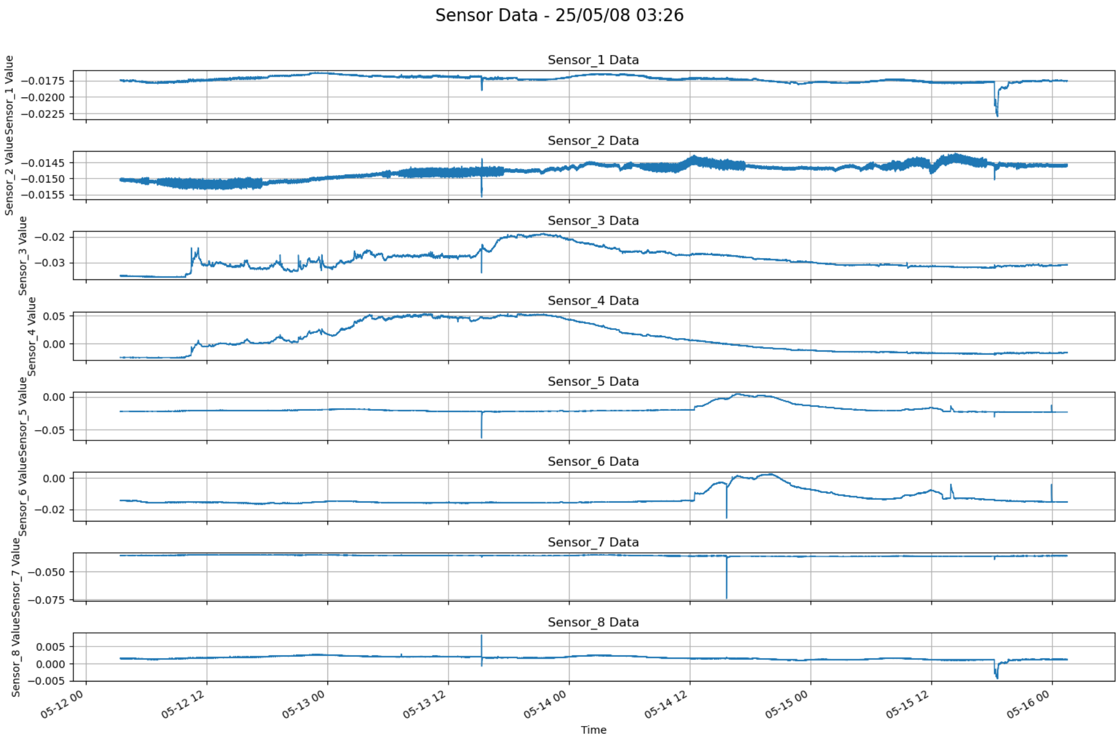
```
Data shape: (1084420, 10)

First 5 rows of data:
              0         1         2         3         4         5         6  \
0  120386.537600 -0.017416 -0.015052 -0.035177 -0.024526 -0.022283 -0.014307
1  120386.714606 -0.017413 -0.015028 -0.035177 -0.024510 -0.022269 -0.014292
2  120386.889620 -0.017420 -0.015043 -0.035157 -0.024524 -0.022270 -0.014293
3  120387.088626 -0.017404 -0.015036 -0.035172 -0.024527 -0.022294 -0.014290
4  120387.273636 -0.017437 -0.015036 -0.035183 -0.024523 -0.022269 -0.014280
```

```
          7         8   9
0 -0.035494  0.001486 NaN
1 -0.035491  0.001480 NaN
2 -0.035494  0.001500 NaN
3 -0.035498  0.001483 NaN
4 -0.035490  0.001495 NaN

Data after renaming columns:
       Timestamp  Sensor_1  Sensor_2  Sensor_3  Sensor_4  Sensor_5  Sensor_6  \
0  120386.537600 -0.017416 -0.015052 -0.035177 -0.024526 -0.022283 -0.014307
1  120386.714606 -0.017413 -0.015028 -0.035177 -0.024510 -0.022269 -0.014292
2  120386.889620 -0.017420 -0.015043 -0.035157 -0.024524 -0.022270 -0.014293
3  120387.088626 -0.017404 -0.015036 -0.035172 -0.024527 -0.022294 -0.014290
4  120387.273636 -0.017437 -0.015036 -0.035183 -0.024523 -0.022269 -0.014280

   Sensor_7  Sensor_8  Sensor_9
0 -0.035494  0.001486       NaN
1 -0.035491  0.001480       NaN
2 -0.035494  0.001500       NaN
3 -0.035498  0.001483       NaN
4 -0.035490  0.001495       NaN
```

[2]:

```
# Extract date and time information from the filename
file_name = os.path.basename(file_path)  # Get the filename
date_time_str = file_name.split('_')[1:3]  # Extract date and time parts
date_str = date_time_str[0].replace('-', '/')  # Format date
time_str = date_time_str[1].replace('.lvm', '')  # Format time
# Parse time string, first two digits are hours, last two are minutes
hour = time_str[:2]
minute = time_str[2:]
formatted_time = f"{hour}:{minute}"

# Use actual timestamps and convert to specific times
actual_time = data['Timestamp']
# Calculate seconds relative to start time
start_time = actual_time.iloc[0]
relative_seconds = actual_time - start_time

# Create specific time labels
from datetime import datetime, timedelta
# Assume data recording started at the date and time specified in the filename
base_time = datetime(2025, 5, 12, int(hour), int(minute))  # Date and time␣
↪parsed from filename
time_labels = [base_time + timedelta(seconds=s) for s in relative_seconds]

# Determine the number of sensors in the dataset
```

```python
num_sensors = len([col for col in data.columns if 'Sensor_' in col]) - 1

# Create a figure with subplots for all sensors
plt.figure(figsize=(15, 10))

# Plot data for all sensors
for i in range(1, num_sensors + 1):
    sensor_name = f'Sensor_{i}'
    plt.subplot(num_sensors, 1, i)
    plt.plot(time_labels, data[sensor_name], linewidth=1)
    plt.title(f'{sensor_name} Data')
    plt.ylabel(f'{sensor_name} Value')
    plt.grid(True)

    # Only add x-label for the bottom subplot
    if i == num_sensors:
        plt.xlabel('Time')

    plt.gcf().autofmt_xdate()  # Automatically format x-axis date labels

# Add a main title for the entire figure
plt.suptitle(f'Sensor Data - {date_str} {formatted_time}', fontsize=16)

# Adjust layout
plt.tight_layout(rect=[0, 0, 1, 0.97])  # Make room for the suptitle

# Display the figure
plt.show()

# Print basic statistics for all sensors
print("Sensor Statistics:")
for i in range(1, num_sensors):
    sensor_name = f'Sensor_{i}'
    print(f"\n{sensor_name}:\n{data[sensor_name].describe()}")
```



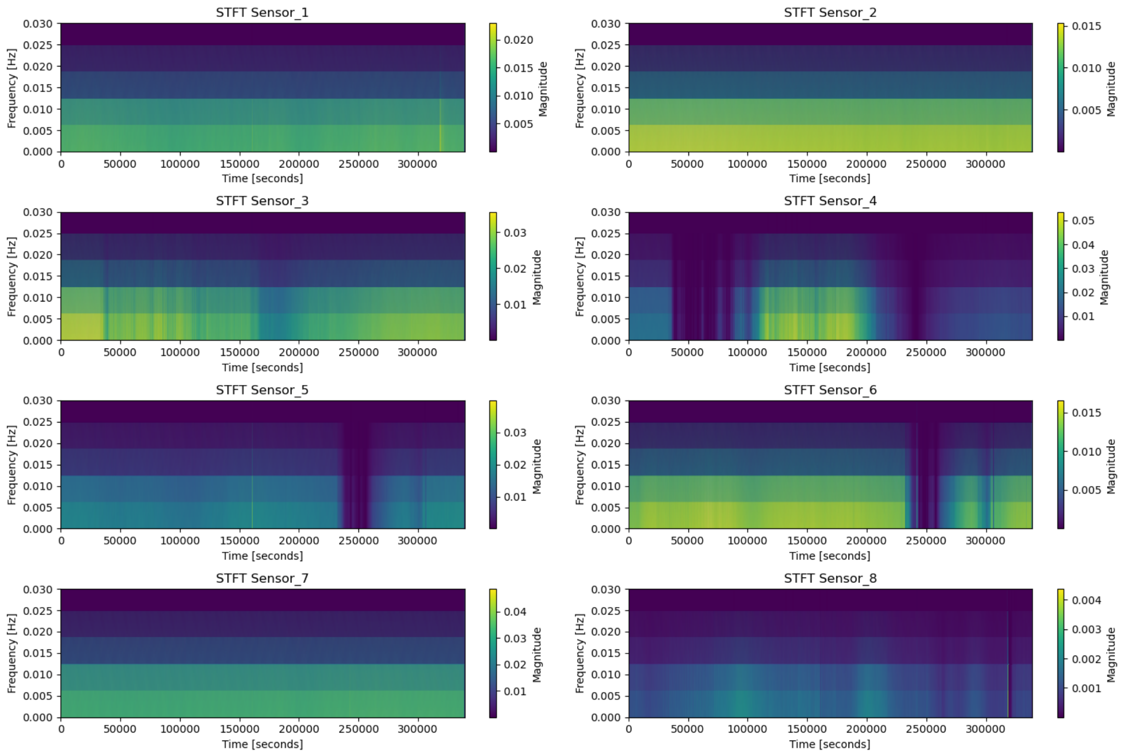

```
Sensor Statistics:

Sensor_1:
count     1.084420e+06
mean     -1.727884e-02
std       5.048720e-04
min      -2.300700e-02
25%      -1.762500e-02
50%      -1.728600e-02
75%      -1.690600e-02
max      -1.632500e-02
Name: Sensor_1, dtype: float64

Sensor_2:
count     1.084420e+06
mean     -1.481143e-02
std       2.283351e-04
min      -1.557400e-02
25%      -1.504800e-02
50%      -1.476700e-02
75%      -1.461900e-02
max      -1.421000e-02
Name: Sensor_2, dtype: float64
```

```
Sensor_3:
count    1.084420e+06
mean    -2.902471e-02
std      4.014688e-03
min     -3.576500e-02
25%     -3.148200e-02
50%     -2.978400e-02
75%     -2.700600e-02
max     -1.869200e-02
Name: Sensor_3, dtype: float64

Sensor_4:
count    1.084420e+06
mean     9.146397e-03
std      2.632990e-02
min     -2.506200e-02
25%     -1.520400e-02
50%      2.054000e-03
75%      3.857700e-02
max      5.387100e-02
Name: Sensor_4, dtype: float64

Sensor_5:
count    1.084420e+06
mean    -1.927404e-02
std      5.577440e-03
min     -6.273700e-02
25%     -2.202700e-02
50%     -2.094700e-02
75%     -1.956500e-02
max      4.554000e-03
Name: Sensor_5, dtype: float64

Sensor_6:
count    1.084420e+06
mean    -1.349133e-02
std      4.149826e-03
min     -2.560900e-02
25%     -1.566500e-02
50%     -1.521000e-02
75%     -1.375100e-02
max      2.503000e-03
Name: Sensor_6, dtype: float64

Sensor_7:
count    1.084420e+06
mean    -3.571238e-02
std      5.212150e-04
```

```
min     -7.406800e-02
25%     -3.605700e-02
50%     -3.561300e-02
75%     -3.531600e-02
max     -3.472400e-02
Name: Sensor_7, dtype: float64
```

[3]:

```python
# Perform Short-Time Fourier Transform (STFT) analysis
from scipy import signal
import matplotlib.pyplot as plt
import numpy as np

# Create a new figure for STFT analysis
plt.figure(figsize=(15, 10))

# Perform STFT on all sensor data
for i in range(1, 9):  # Assuming 8 sensors
    sensor_name = f'Sensor_{i}'

    # Get sensor data
    sensor_data = data[sensor_name].values

    # Calculate sampling rate (based on timestamp differences)
    sampling_rate = 1.0 / np.mean(np.diff(data['Timestamp']))

    # Perform STFT
    f, t, Zxx = signal.stft(sensor_data, fs=sampling_rate, nperseg=256)

    # Plot STFT results
    plt.subplot(4, 2, i)

    plt.pcolormesh(t, f, np.abs(Zxx), shading='gouraud')

    plt.title(f'STFT {sensor_name}')
    plt.ylabel('Frequency [Hz]')
    plt.xlabel('Time [seconds]')
    plt.colorbar(label='Magnitude')
    plt.ylim(0, 0.03)  # Limit y-axis to 0.03Hz

plt.tight_layout()
plt.show()

# Print basic information about the STFT analysis
print(f"STFT analysis completed")
print(f"Sampling rate: {sampling_rate:.2f} Hz")
print(f"Frequency resolution: {f[1]-f[0]:.4f} Hz")
print(f"Time resolution: {t[1]-t[0]:.4f} seconds")
```




```
STFT analysis completed
Sampling rate: 3.20 Hz
Frequency resolution: 0.0125 Hz
Time resolution: 39.9933 seconds
```

[4]:

```python
# Calculate the recording end time based on the timestamp
import datetime
# Extract start time from the filename (Mushroom_25-05-08_0326)
filename = file_path.split('/')[-1]
date_part = filename.split('_')[1]   # '25-05-08'
time_part = filename.split('_')[2]   # '0326'

# Handle potential file extension in time_part
if '.' in time_part:
    time_part = time_part.split('.')[0]  # Remove file extension if present

year = 2000 + int(date_part.split('-')[0])  # '25' -> 2025
month = int(date_part.split('-')[1])  # '05' -> 5
day = int(date_part.split('-')[2])  # '08' -> 8
hour = int(time_part[:2])  # '03' -> 3
minute = int(time_part[2:])  # '26' -> 26

start_time = datetime.datetime(year, month, day, hour, minute)
```

```
# Get the first and last timestamp
first_timestamp = data['Timestamp'].iloc[0]
last_timestamp = data['Timestamp'].iloc[-1]

# Calculate the duration in seconds
duration_seconds = last_timestamp - first_timestamp

# Calculate the end time
end_time = start_time + datetime.timedelta(seconds=duration_seconds)

# Format and print the results
print(f"Recording start time: {start_time.strftime('%Y-%m-%d %H:%M:%S')}")
print(f"Recording end time: {end_time.strftime('%Y-%m-%d %H:%M:%S')}")
print(f"Total recording duration: {duration_seconds:.2f} seconds␣
↪({duration_seconds/60:.2f} minutes)")
```

```
Recording start time: 2025-05-08 03:26:00
Recording end time: 2025-05-12 01:33:04
Total recording duration: 338824.48 seconds (5647.07 minutes)
```

[5]:

```
# Function to find the closest timestamp in the data to a given event time
import pytz
import datetime

# Parse the event time string
event_time_str = "2025-05-11T10:13:23.544Z"
event_time = datetime.datetime.strptime(event_time_str, "%Y-%m-%dT%H:%M:%S.%fZ")
event_time = event_time.replace(tzinfo=pytz.UTC)  # Make it timezone-aware

# Make start_time timezone-aware as well
start_time = start_time.replace(tzinfo=pytz.UTC)

# Calculate seconds elapsed since recording start
elapsed_seconds = (event_time - start_time).total_seconds()

print(f"Event time: {event_time_str}")
print(f"Recording start time: {start_time.strftime('%Y-%m-%d %H:%M:%S %Z')}")
print(f"Seconds elapsed since recording start: {elapsed_seconds:.2f} seconds")

# Get the first timestamp from the data
first_timestamp = data['Timestamp'].iloc[0]

# Calculate the target timestamp by adding elapsed seconds to the first␣
↪timestamp
target_timestamp = first_timestamp + elapsed_seconds
```

```python
# Find the closest timestamp in the data
closest_idx = (data['Timestamp'] - target_timestamp).abs().idxmin()
closest_timestamp = data['Timestamp'].iloc[closest_idx]
closest_time_diff = abs(closest_timestamp - target_timestamp)

print(f"First data timestamp: {first_timestamp:.2f} seconds")
print(f"Target timestamp: {target_timestamp:.2f} seconds")
print(f"Closest data timestamp: {closest_timestamp:.2f} seconds")
print(f"Difference from target: {closest_time_diff:.2f} seconds")

# Extract the data at the closest timestamp
event_data = data.iloc[closest_idx]
print("\nSensor readings at event time:")
for column in data.columns:
    if column != 'Timestamp':
        print(f"{column}: {event_data[column]}")
```

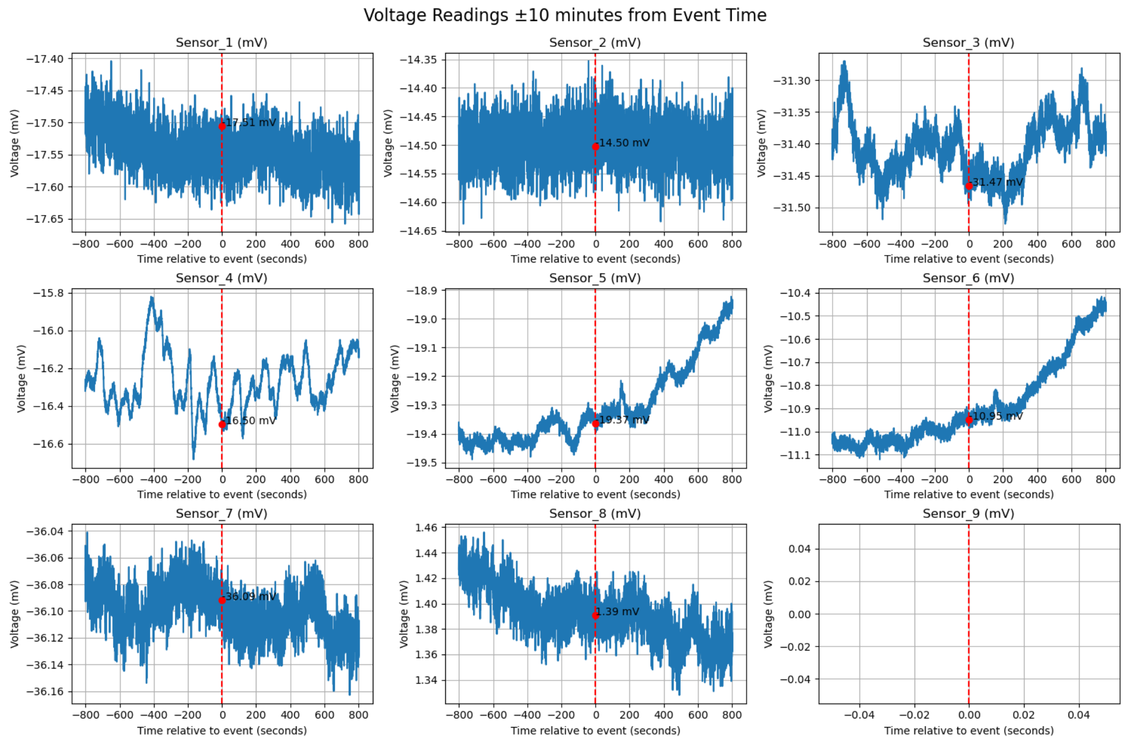
```
Event time: 2025-05-11T10:13:23.544Z
Recording start time: 2025-05-08 03:26:00 UTC
Seconds elapsed since recording start: 283643.54 seconds
First data timestamp: 120386.54 seconds
Target timestamp: 404030.08 seconds
Closest data timestamp: 404030.04 seconds
Difference from target: 0.04 seconds

Sensor readings at event time:
Sensor_1: -0.017506
Sensor_2: -0.014503
Sensor_3: -0.031466
Sensor_4: -0.016499
Sensor_5: -0.019365
Sensor_6: -0.010949
Sensor_7: -0.036092
Sensor_8: 0.001391
Sensor_9: nan
```

[6]:

```python
# Plot voltage data for 10 minutes before and after the event time
import matplotlib.pyplot as plt
import numpy as np

# Define the time window (10 minutes before and after the event)
window_minutes = 10
window_seconds = window_minutes * 60  # Convert minutes to seconds
event_idx = closest_idx
start_idx = max(0, event_idx - int(window_seconds * data['Timestamp'].diff().
 ↪median() ** -1))
```

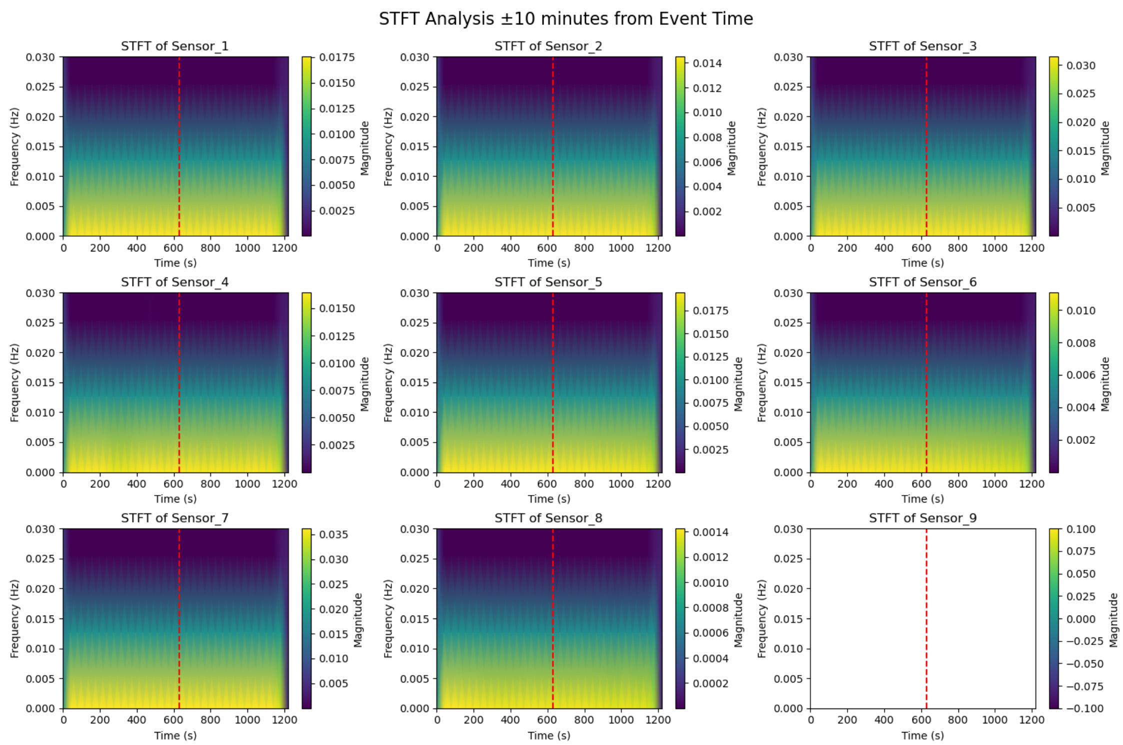
```
end_idx = min(len(data) - 1, event_idx + int(window_seconds * data['Timestamp'].
    ↪diff().median() ** -1))

# Extract the data for the time window
window_data = data.iloc[start_idx:end_idx+1]

# Calculate time relative to the event (in seconds)
relative_time = window_data['Timestamp'] - closest_timestamp

# Create a figure with subplots for each voltage channel
plt.figure(figsize=(15, 10))
voltage_columns = [col for col in data.columns if col != 'Timestamp']

for i, column in enumerate(voltage_columns):
    plt.subplot(3, 3, i+1)

    # Convert voltage to millivolts
    voltage_mv = window_data[column] * 1000  # Convert to mV

    plt.plot(relative_time, voltage_mv)
    plt.axvline(x=0, color='r', linestyle='--', label='Event time')
    plt.title(f'{column} (mV)')
    plt.xlabel('Time relative to event (seconds)')
    plt.ylabel('Voltage (mV)')
    plt.grid(True)

    # Add a red dot at the event time point
    event_value_mv = event_data[column] * 1000  # Convert to mV
    plt.plot(0, event_value_mv, 'ro', markersize=6)  # Red dot at event time
    plt.text(1, event_value_mv, f'{event_value_mv:.2f} mV')  # Text label␣
    ↪without arrow

plt.tight_layout()
plt.suptitle('Voltage Readings ±10 minutes from Event Time', fontsize=16)
plt.subplots_adjust(top=0.92)
plt.show()
```

```
posx and posy should be finite values
posx and posy should be finite values
```




[7]:
```
# Perform Short-Time Fourier Transform (STFT) analysis for each voltage channel
import matplotlib.pyplot as plt
from scipy import signal
import numpy as np

# Create a figure with subplots for STFT of each voltage channel
plt.figure(figsize=(15, 10))
voltage_columns = [col for col in data.columns if col != 'Timestamp']

# Calculate sampling frequency
sampling_freq = 1.0 / data['Timestamp'].diff().median()

for i, column in enumerate(voltage_columns):
    plt.subplot(3, 3, i+1)

    # Get voltage data for this channel
    voltage_data = window_data[column].values

    # Perform STFT
    f, t, Zxx = signal.stft(voltage_data, fs=sampling_freq, nperseg=256)

    # Plot the STFT magnitude (in dB)
    plt.pcolormesh(t, f, np.abs(Zxx), shading='gouraud')
```

```
    # Mark the event time
    plt.axvline(x=t[len(t)//2], color='r', linestyle='--', label='Event time')

    plt.title(f'STFT of {column}')
    plt.ylabel('Frequency (Hz)')
    plt.xlabel('Time (s)')
    plt.colorbar(label='Magnitude')
    plt.ylim(0, 0.03)

plt.tight_layout()
plt.suptitle('STFT Analysis ±10 minutes from Event Time', fontsize=16)
plt.subplots_adjust(top=0.92)
plt.show()
```




[8]:
```
# Analyze the 0.02Hz frequency band before and after event for each sensor
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import os
import datetime

# Get dataset name from the notebook filename
```

```python
notebook_name = os.path.basename(__file__) if '__file__' in globals() else␣
↪'Mushroom_25-05-08_0326'
if notebook_name.endswith('.ipynb'):
    notebook_name = notebook_name[:-6]  # Remove .ipynb extension
if notebook_name.startswith('da_'):
    notebook_name = notebook_name[3:]  # Remove da_ prefix

# Create a directory to save CSV files with dataset name
csv_dir = f"significant_changes_csv_{notebook_name}"
if not os.path.exists(csv_dir):
    os.makedirs(csv_dir)
    print(f"Created directory: {csv_dir}")

# Calculate sampling frequency
sampling_freq = 1.0 / data['Timestamp'].diff().median()

# Find the event time (assuming it's at the center of the filtered data)
event_time = window_data['Timestamp'].mean()

# Loop through each voltage channel
for channel_to_analyze in voltage_columns:
    print(f"\n=== Analysis for {channel_to_analyze} ===")
    voltage_data = window_data[channel_to_analyze].values

    # Perform STFT for the selected channel
    f, t, Zxx = signal.stft(voltage_data, fs=sampling_freq, nperseg=256)

    # Find the closest frequency to 0.02Hz in the STFT results
    target_freq = 0.02
    freq_idx = np.argmin(np.abs(f - target_freq))
    actual_freq = f[freq_idx]
    print(f"Analyzing frequency: {actual_freq:.4f} Hz (closest to 0.02 Hz)")

    # Extract the magnitude data for this frequency
    freq_magnitude = np.abs(Zxx[freq_idx, :])

    # Create a time axis in minutes for better visualization
    time_min = t / 60

    # Plot the magnitude of the 0.02Hz component over time
    plt.figure(figsize=(15, 6))

    # Plot the magnitude
    plt.plot(time_min, freq_magnitude, 'b-', linewidth=2, label=f'{actual_freq:.
↪4f} Hz Component')

    # Convert event time to minutes
```

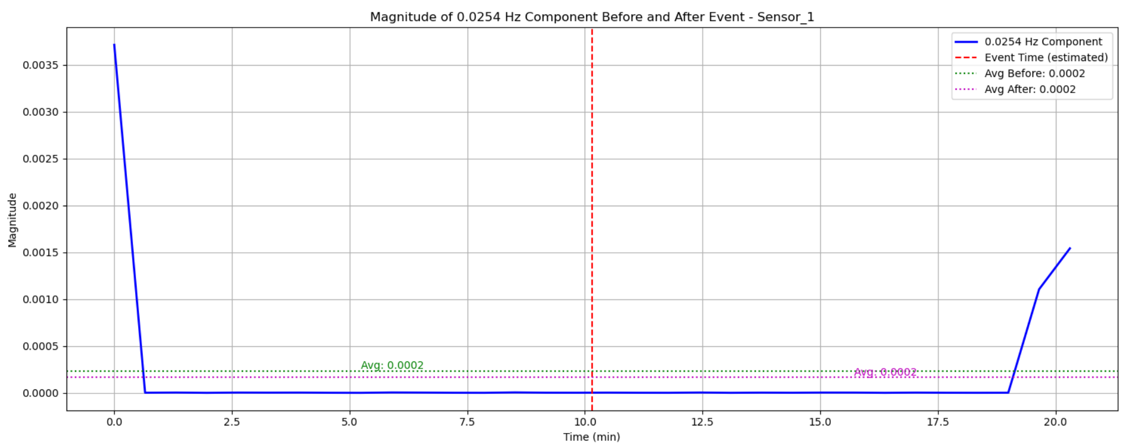
```
    event_time_min = t.mean() / 60
    plt.axvline(x=event_time_min, color='r', linestyle='--', label='Event Time␣
↪(estimated)')

    # Calculate average magnitude before and after event
    before_mask = t < t.mean()
    after_mask = t >= t.mean()

    avg_before = np.mean(freq_magnitude[before_mask])
    avg_after = np.mean(freq_magnitude[after_mask])

    print(f"Average magnitude before event: {avg_before:.4f}")
    print(f"Average magnitude after event: {avg_after:.4f}")
    print(f"Change: {(avg_after - avg_before):.4f} ({(avg_after - avg_before)/
↪avg_before*100:.2f}%)")

    # Add horizontal lines showing the average values
    plt.axhline(y=avg_before, color='g', linestyle=':', label=f'Avg Before:␣
↪{avg_before:.4f}')
    plt.axhline(y=avg_after, color='m', linestyle=':', label=f'Avg After:␣
↪{avg_after:.4f}')

    # Add annotations
    plt.annotate(f"Avg: {avg_before:.4f}", xy=(time_min[len(time_min)//4],␣
↪avg_before),
                 xytext=(time_min[len(time_min)//4], avg_before*1.1), color='g')
    plt.annotate(f"Avg: {avg_after:.4f}", xy=(time_min[3*len(time_min)//4],␣
↪avg_after),
                 xytext=(time_min[3*len(time_min)//4], avg_after*1.1),␣
↪color='m')

    # Set axis labels and title
    plt.xlabel('Time (min)')
    plt.ylabel('Magnitude')
    plt.title(f'Magnitude of {actual_freq:.4f} Hz Component Before and After␣
↪Event - {channel_to_analyze}')
    plt.grid(True)
    plt.legend()
    plt.tight_layout()
    plt.show()

    # Calculate energy (integral of magnitude squared) before and after event
    energy_before = np.sum(freq_magnitude[before_mask]**2)
    energy_after = np.sum(freq_magnitude[after_mask]**2)

    # Normalize by the number of samples to get average energy
```

```
    num_samples_before = np.sum(before_mask)
    num_samples_after = np.sum(after_mask)
    avg_energy_before = energy_before / num_samples_before
    avg_energy_after = energy_after / num_samples_after

    print("\nEnergy Analysis:")
    print(f"Total energy before event: {energy_before:.4f}")
    print(f"Total energy after event: {energy_after:.4f}")
    print(f"Average energy before event: {avg_energy_before:.4f}")
    print(f"Average energy after event: {avg_energy_after:.4f}")
    print(f"Energy change: {(avg_energy_after - avg_energy_before):.4f} ({(avg_energy_after - avg_energy_before)/avg_energy_before*100:.2f}%)")

    # Power Spectral Density (PSD) Analysis
    # Calculate power (magnitude squared)
    power_matrix = np.abs(Zxx) ** 2

    # Convert time to minutes for consistency with previous plots
    time_min = t / 60

    # Define the event time point (assuming same as before)
    event_time_min = time_min[len(time_min) // 2]  # Middle point as event time

    # Create masks for before and after event
    before_mask_time = time_min < event_time_min
    after_mask_time = time_min > event_time_min

    # Calculate average PSD before and after event
    avg_psd_before = np.mean(power_matrix[:, before_mask_time], axis=1)
    avg_psd_after = np.mean(power_matrix[:, after_mask_time], axis=1)

    # Plot the power spectral density comparison
    plt.figure(figsize=(15, 6))
    plt.plot(f, avg_psd_before, 'g-', label='Before Event')
    plt.plot(f, avg_psd_after, 'm-', label='After Event')

    # Calculate and display the difference
    psd_diff = avg_psd_after - avg_psd_before
    plt.plot(f, psd_diff, 'b--', label='Difference (After - Before)')

    # Set axis labels and title
    plt.xlabel('Frequency (Hz)')
    plt.xlim(0, 0.2)  # Limit x-axis to show only frequencies below 0.2 Hz
    plt.ylabel('Power Spectral Density')
    plt.title(f'Power Spectral Density Comparison Before and After Event - {channel_to_analyze}')
    plt.grid(True)
```

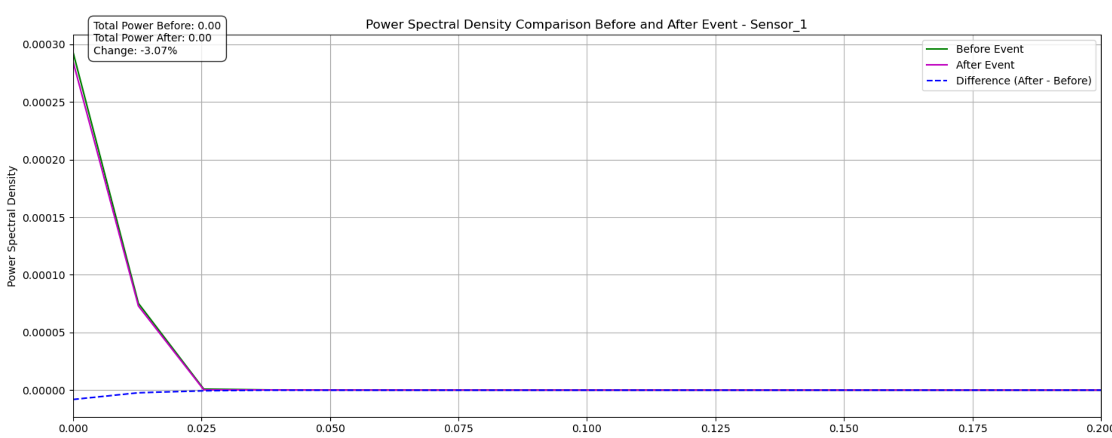
```
    plt.legend()

    # Add text box with summary statistics
    total_power_before = np.sum(avg_psd_before)
    total_power_after = np.sum(avg_psd_after)
    power_change = (total_power_after - total_power_before) /␣
↪total_power_before * 100

    stats_text = f"Total Power Before: {total_power_before:.2f}\n"
    stats_text += f"Total Power After: {total_power_after:.2f}\n"
    stats_text += f"Change: {power_change:.2f}%"

    plt.annotate(stats_text, xy=(0.02, 0.95), xycoords='axes fraction',
                 bbox=dict(boxstyle="round,pad=0.5", fc="white", alpha=0.8))

    plt.tight_layout()
    plt.show()

    # Print detailed statistics
    print("\nPower Spectral Density Analysis:")
    print(f"Total power before event: {total_power_before:.4f}")
    print(f"Total power after event: {total_power_after:.4f}")
    print(f"Absolute power change: {total_power_after - total_power_before:.
↪4f}")
    print(f"Relative power change: {power_change:.2f}%")

    # Find frequency bands with the most significant changes
    freq_change_percent = (avg_psd_after - avg_psd_before) / (avg_psd_before +␣
↪1e-10) * 100  # Avoid division by zero
    significant_changes = pd.DataFrame({
        'Frequency': f,
        'Before': avg_psd_before,
        'After': avg_psd_after,
        'Absolute_Change': avg_psd_after - avg_psd_before,
        'Percent_Change': freq_change_percent
    })

    # Save the significant_changes DataFrame to CSV
    csv_filename = os.path.join(csv_dir,␣
↪f"{channel_to_analyze}_significant_changes.csv")
    significant_changes.to_csv(csv_filename, index=False)
    print(f"Saved significant changes data to: {csv_filename}")

    # Display top 5 frequencies with largest increase and decrease
    print("\nTop 5 frequencies with largest power increase:")
    print(significant_changes.sort_values('Percent_Change', ascending=False).
↪head(5))
```

```
    print("\nTop 5 frequencies with largest power decrease:")
    print(significant_changes.sort_values('Percent_Change', ascending=True).
 ↪head(5))
```

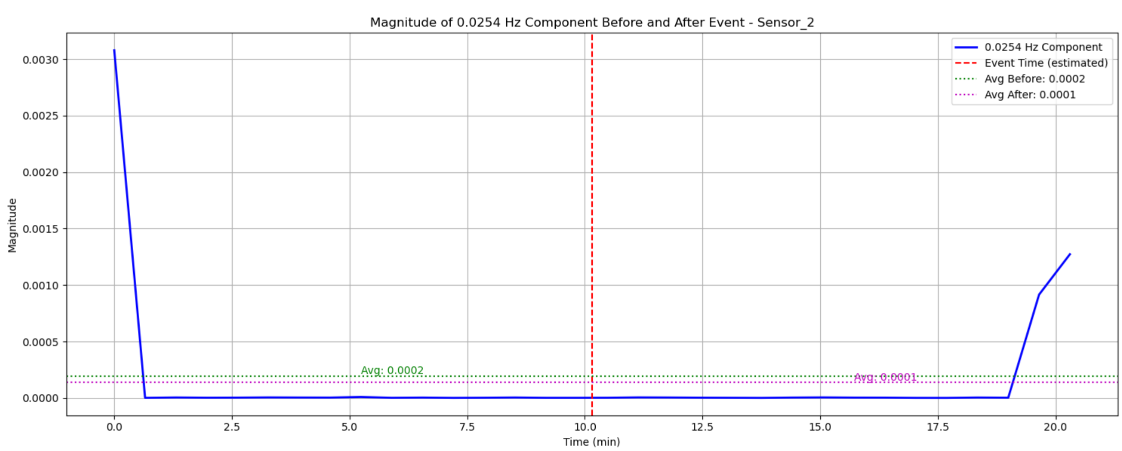
```
=== Analysis for Sensor_1 ===
Analyzing frequency: 0.0254 Hz (closest to 0.02 Hz)
Average magnitude before event: 0.0002
Average magnitude after event: 0.0002
Change: -0.0001 (-28.62%)
```



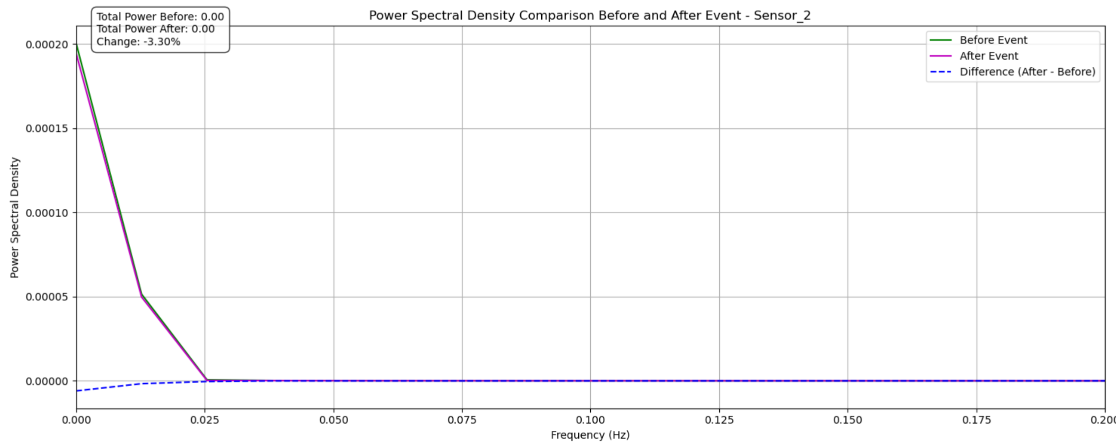

```
Energy Analysis:
Total energy before event: 0.0000
Total energy after event: 0.0000
Average energy before event: 0.0000
Average energy after event: 0.0000
Energy change: -0.0000 (-73.90%)
```



```
Power Spectral Density Analysis:
Total power before event: 0.0004
Total power after event: 0.0004
Absolute power change: -0.0000
Relative power change: -3.07%
Saved significant changes data to:
significant_changes_csv_Mushroom_25-05-08_0326\Sensor_1_significant_changes.csv

Top 5 frequencies with largest power increase:
     Frequency        Before         After  Absolute_Change  Percent_Change
0     0.000000  2.930519e-04  2.849437e-04    -8.108225e-06       -2.766821
1     0.012722  7.521722e-05  7.291841e-05    -2.298816e-06       -3.056232
126   1.602950  2.753338e-10  1.715979e-10    -1.037358e-10      -27.638294
115   1.463010  2.906607e-10  1.790208e-10    -1.116399e-10      -28.577211
117   1.488454  2.873529e-10  1.750505e-10    -1.123024e-10      -28.992260

Top 5 frequencies with largest power decrease:
    Frequency        Before         After  Absolute_Change  Percent_Change
2    0.025444  8.619753e-07  2.399969e-07    -6.219784e-07      -72.148961
6    0.076331  5.684861e-08  2.889211e-08    -2.795650e-08      -49.090757
10   0.127218  1.999247e-08  1.072618e-08    -9.266280e-09      -46.118186
11   0.139940  1.621282e-08  8.785580e-09    -7.427238e-09      -45.530072
35   0.445264  1.718891e-09  8.928409e-10    -8.260497e-10      -45.415030

=== Analysis for Sensor_2 ===
Analyzing frequency: 0.0254 Hz (closest to 0.02 Hz)
Average magnitude before event: 0.0002
Average magnitude after event: 0.0001
Change: -0.0001 (-28.67%)
```

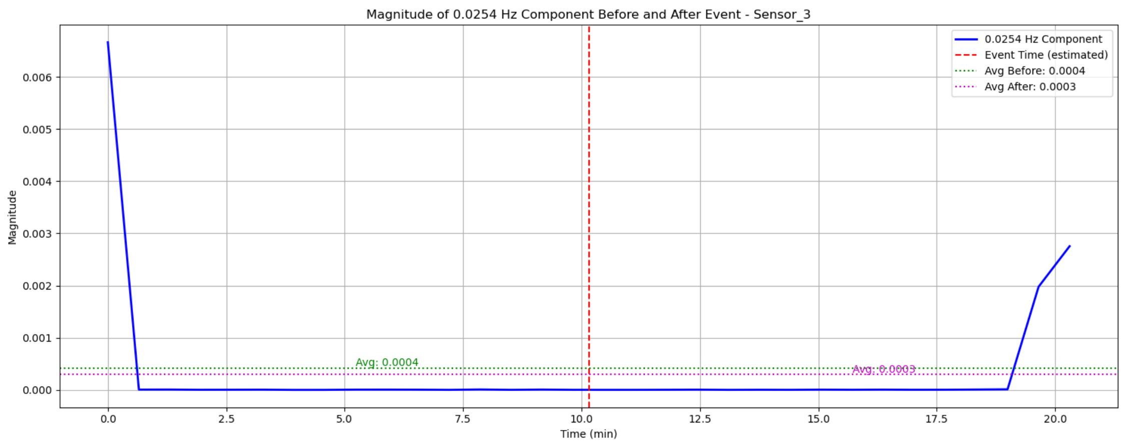


```
Energy Analysis:
Total energy before event: 0.0000
Total energy after event: 0.0000
Average energy before event: 0.0000
Average energy after event: 0.0000
Energy change: -0.0000 (-74.05%)
```

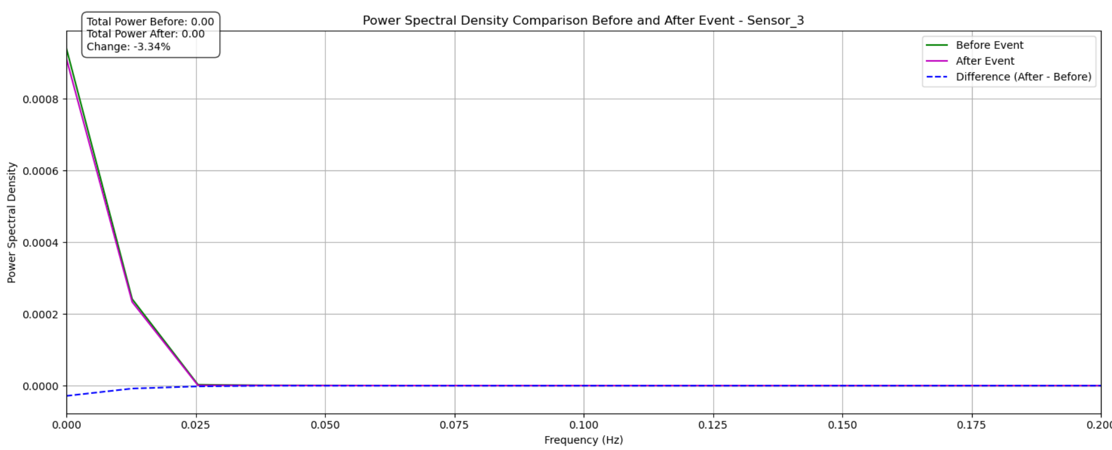



```
Power Spectral Density Analysis:
Total power before event: 0.0003
Total power after event: 0.0002
Absolute power change: -0.0000
Relative power change: -3.30%
Saved significant changes data to:
significant_changes_csv_Mushroom_25-05-08_0326\Sensor_2_significant_changes.csv

Top 5 frequencies with largest power increase:
     Frequency        Before         After  Absolute_Change  Percent_Change
0     0.000000  2.003581e-04  1.943700e-04    -5.988121e-06       -2.988708
1     0.012722  5.144071e-05  4.974024e-05    -1.700460e-06       -3.305665
125   1.590228  2.063294e-10  1.306203e-10    -7.570903e-11      -24.714912
124   1.577507  1.999986e-10  1.258485e-10    -7.415006e-11      -24.716803
99    1.259461  2.200611e-10  1.407460e-10    -7.931517e-11      -24.781254

Top 5 frequencies with largest power decrease:
    Frequency        Before         After  Absolute_Change  Percent_Change
2    0.025444  5.921369e-07  1.639020e-07    -4.282349e-07      -72.308040
6    0.076331  3.927829e-08  1.969443e-08    -1.958386e-08      -49.732628
10   0.127218  1.356712e-08  7.077358e-09    -6.489758e-09      -47.484475
22   0.279880  2.875108e-09  1.479130e-09    -1.395978e-09      -46.921938
```

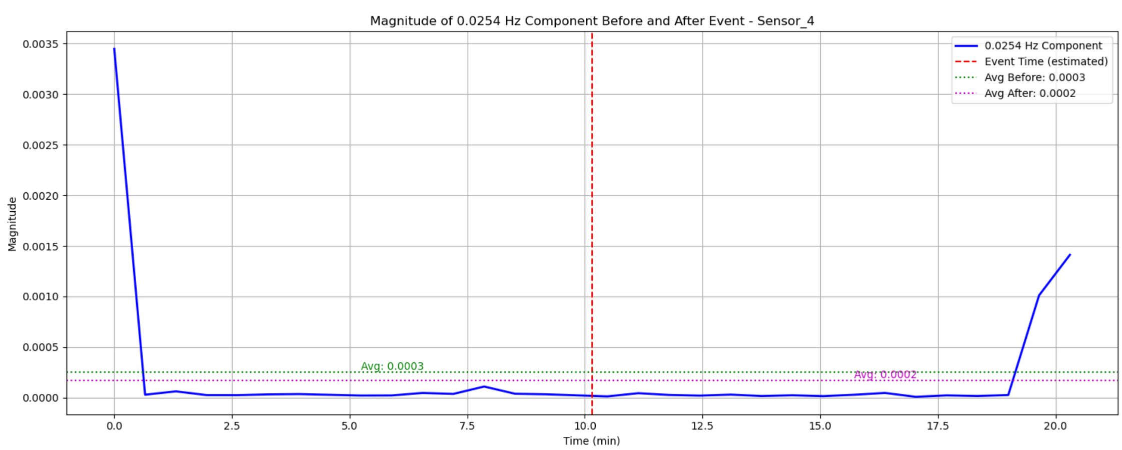
```
12   0.152662  9.582926e-09  5.048442e-09    -4.534484e-09      -46.829689

=== Analysis for Sensor_3 ===
Analyzing frequency: 0.0254 Hz (closest to 0.02 Hz)
Average magnitude before event: 0.0004
Average magnitude after event: 0.0003
Change: -0.0001 (-28.72%)
```




```
Energy Analysis:
Total energy before event: 0.0000
Total energy after event: 0.0000
Average energy before event: 0.0000
Average energy after event: 0.0000
Energy change: -0.0000 (-74.09%)
```

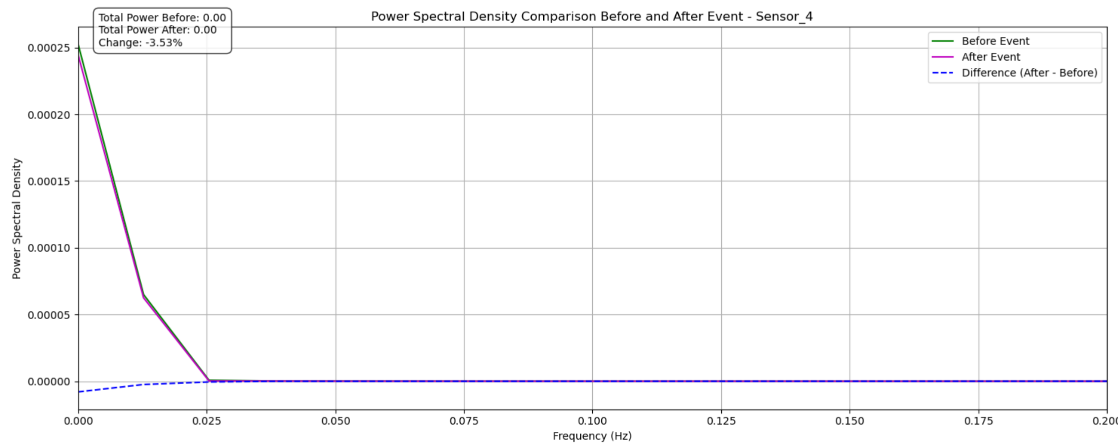


```
Power Spectral Density Analysis:
Total power before event: 0.0012
Total power after event: 0.0011
Absolute power change: -0.0000
Relative power change: -3.34%
Saved significant changes data to:
significant_changes_csv_Mushroom_25-05-08_0326\Sensor_3_significant_changes.csv

Top 5 frequencies with largest power increase:
     Frequency        Before         After  Absolute_Change  Percent_Change
0     0.000000  9.404428e-04  9.118956e-04    -2.854718e-05       -3.035504
1     0.012722  2.414134e-04  2.333462e-04    -8.067178e-06       -3.341643
3     0.038165  6.932727e-07  4.685567e-07    -2.247160e-07      -32.409127
109   1.386679  9.965393e-10  5.568232e-10    -4.397161e-10      -40.100354
114   1.450288  9.618013e-10  5.289763e-10    -4.328250e-10      -40.763278

Top 5 frequencies with largest power decrease:
    Frequency        Before         After  Absolute_Change  Percent_Change
2    0.025444  2.773258e-06  7.665843e-07    -2.006674e-06      -72.355378
6    0.076331  1.843669e-07  9.244078e-08    -9.192611e-08      -49.833391
10   0.127218  6.386627e-08  3.409758e-08    -2.976869e-08      -46.538107
16   0.203549  2.490065e-08  1.334463e-08    -1.155602e-08      -46.222895
12   0.152662  4.423609e-08  2.375229e-08    -2.048381e-08      -46.201198

=== Analysis for Sensor_4 ===
Analyzing frequency: 0.0254 Hz (closest to 0.02 Hz)
Average magnitude before event: 0.0003
Average magnitude after event: 0.0002
Change: -0.0001 (-31.24%)
```




```
Energy Analysis:
```

Total energy before event: 0.0000
Total energy after event: 0.0000
Average energy before event: 0.0000
Average energy after event: 0.0000
Energy change: -0.0000 (-74.60%)




Power Spectral Density Analysis:
Total power before event: 0.0003
Total power after event: 0.0003
Absolute power change: -0.0000
Relative power change: -3.53%
Saved significant changes data to:
significant_changes_csv_Mushroom_25-05-08_0326\Sensor_4_significant_changes.csv

Top 5 frequencies with largest power increase:

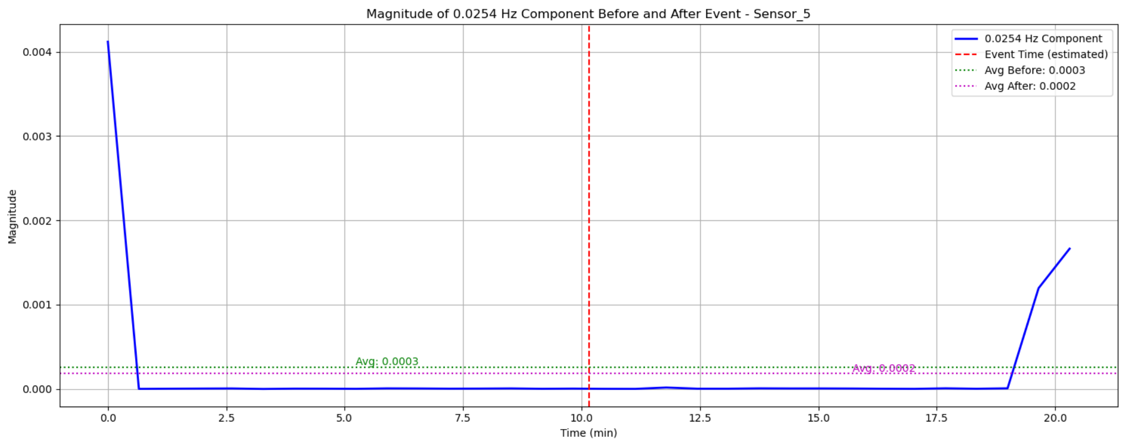
| | Frequency | Before | After | Absolute_Change | Percent_Change |
|---|---|---|---|---|---|
| 0 | 0.000000 | 2.524682e-04 | 2.444765e-04 | -7.991726e-06 | -3.165437 |
| 1 | 0.012722 | 6.492484e-05 | 6.248789e-05 | -2.436951e-06 | -3.753490 |
| 125 | 1.590228 | 2.429389e-10 | 1.342816e-10 | -1.086573e-10 | -31.684161 |
| 126 | 1.602950 | 2.473670e-10 | 1.367951e-10 | -1.105719e-10 | -31.831431 |
| 124 | 1.577507 | 2.453329e-10 | 1.334297e-10 | -1.119032e-10 | -32.404437 |

Top 5 frequencies with largest power decrease:

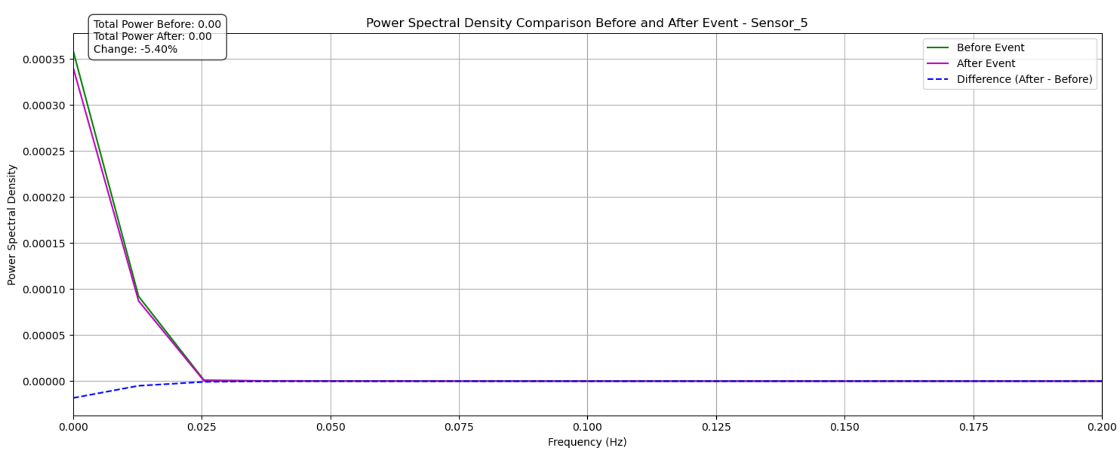
| | Frequency | Before | After | Absolute_Change | Percent_Change |
|---|---|---|---|---|---|
| 2 | 0.025444 | 7.448357e-07 | 2.017619e-07 | -5.430737e-07 | -72.902097 |
| 6 | 0.076331 | 4.930823e-08 | 2.444802e-08 | -2.486022e-08 | -50.315938 |
| 10 | 0.127218 | 1.715493e-08 | 8.950291e-09 | -8.204640e-09 | -47.549538 |
| 18 | 0.228993 | 5.352607e-09 | 2.785294e-09 | -2.567313e-09 | -47.084137 |
| 14 | 0.178106 | 8.753509e-09 | 4.602480e-09 | -4.151028e-09 | -46.885687 |

=== Analysis for Sensor_5 ===

```
Analyzing frequency: 0.0254 Hz (closest to 0.02 Hz)
Average magnitude before event: 0.0003
Average magnitude after event: 0.0002
Change: -0.0001 (-29.73%)
```




```
Energy Analysis:
Total energy before event: 0.0000
Total energy after event: 0.0000
Average energy before event: 0.0000
Average energy after event: 0.0000
Energy change: -0.0000 (-75.23%)
```




```
Power Spectral Density Analysis:
Total power before event: 0.0005
Total power after event: 0.0004
```

Absolute power change: -0.0000
Relative power change: -5.40%
Saved significant changes data to:
significant_changes_csv_Mushroom_25-05-08_0326\Sensor_5_significant_changes.csv

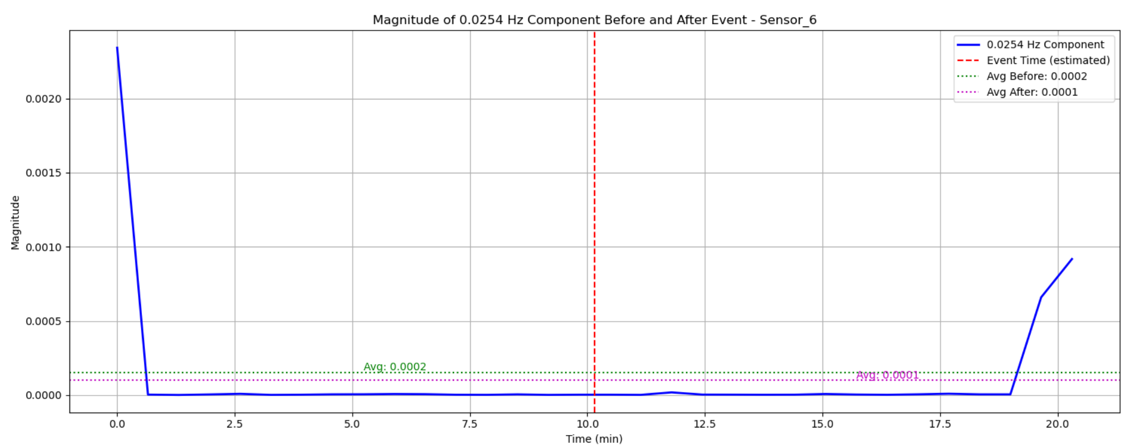
Top 5 frequencies with largest power increase:

| | Frequency | Before | After | Absolute_Change | Percent_Change |
|---|---|---|---|---|---|
| 0 | 0.000000 | 3.589704e-04 | 3.407053e-04 | -1.826503e-05 | -5.088170 |
| 1 | 0.012722 | 9.215228e-05 | 8.714426e-05 | -5.008016e-06 | -5.434495 |
| 3 | 0.038165 | 2.648924e-07 | 1.709285e-07 | -9.396388e-08 | -35.459084 |
| 119 | 1.513898 | 3.605312e-10 | 1.916252e-10 | -1.689060e-10 | -36.676341 |
| 120 | 1.526619 | 3.624475e-10 | 1.905126e-10 | -1.719349e-10 | -37.179325 |

Top 5 frequencies with largest power decrease:

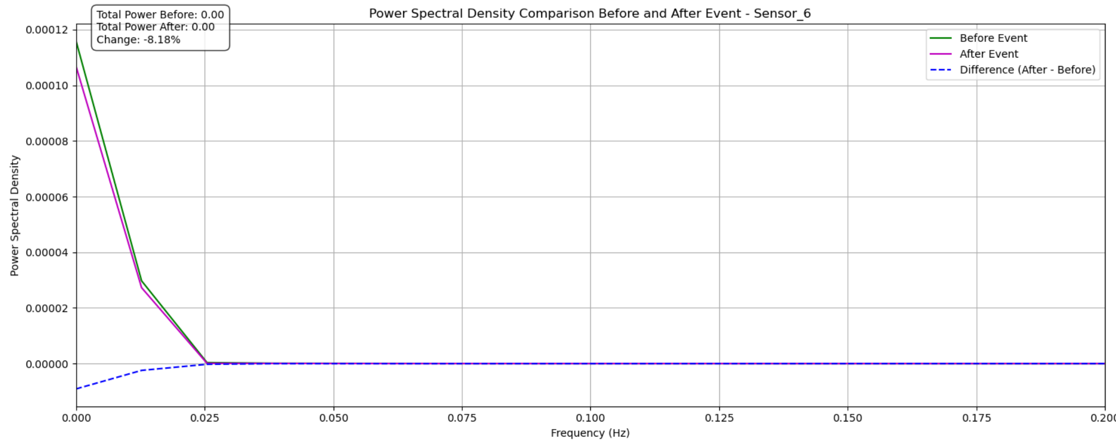
| | Frequency | Before | After | Absolute_Change | Percent_Change |
|---|---|---|---|---|---|
| 2 | 0.025444 | 1.059779e-06 | 2.799578e-07 | -7.798208e-07 | -73.576422 |
| 6 | 0.076331 | 7.015034e-08 | 3.367824e-08 | -3.647211e-08 | -51.917335 |
| 10 | 0.127218 | 2.442514e-08 | 1.240989e-08 | -1.201525e-08 | -48.991571 |
| 12 | 0.152662 | 1.688214e-08 | 8.636898e-09 | -8.245238e-09 | -48.552420 |
| 14 | 0.178106 | 1.240089e-08 | 6.354230e-09 | -6.046657e-09 | -48.369826 |

=== Analysis for Sensor_6 ===
Analyzing frequency: 0.0254 Hz (closest to 0.02 Hz)
Average magnitude before event: 0.0002
Average magnitude after event: 0.0001
Change: -0.0000 (-31.34%)




Energy Analysis:
Total energy before event: 0.0000
Total energy after event: 0.0000
Average energy before event: 0.0000

Average energy after event: 0.0000
Energy change: -0.0000 (-76.72%)




Power Spectral Density Analysis:
Total power before event: 0.0001
Total power after event: 0.0001
Absolute power change: -0.0000
Relative power change: -8.18%
Saved significant changes data to:
significant_changes_csv_Mushroom_25-05-08_0326\Sensor_6_significant_changes.csv

Top 5 frequencies with largest power increase:

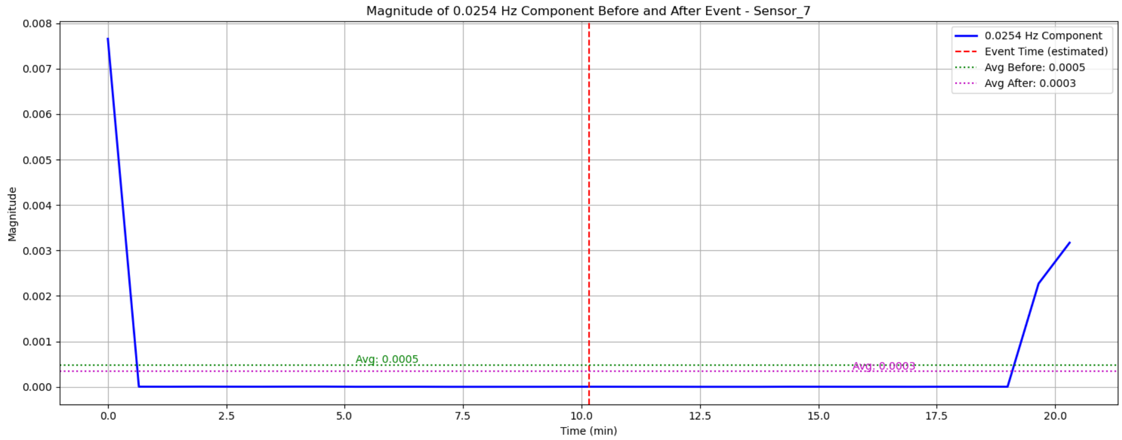
| | Frequency | Before | After | Absolute_Change | Percent_Change |
|---|---|---|---|---|---|
| 0 | 0.000000 | 1.158188e-04 | 1.067152e-04 | -9.103595e-06 | -7.860200 |
| 1 | 0.012722 | 2.973380e-05 | 2.727949e-05 | -2.454311e-06 | -8.254250 |
| 116 | 1.475732 | 1.138004e-10 | 5.861724e-11 | -5.518319e-11 | -25.810608 |
| 103 | 1.310348 | 1.243575e-10 | 6.625059e-11 | -5.810694e-11 | -25.899261 |
| 113 | 1.437567 | 1.185812e-10 | 6.192212e-11 | -5.665904e-11 | -25.921283 |

Top 5 frequencies with largest power decrease:

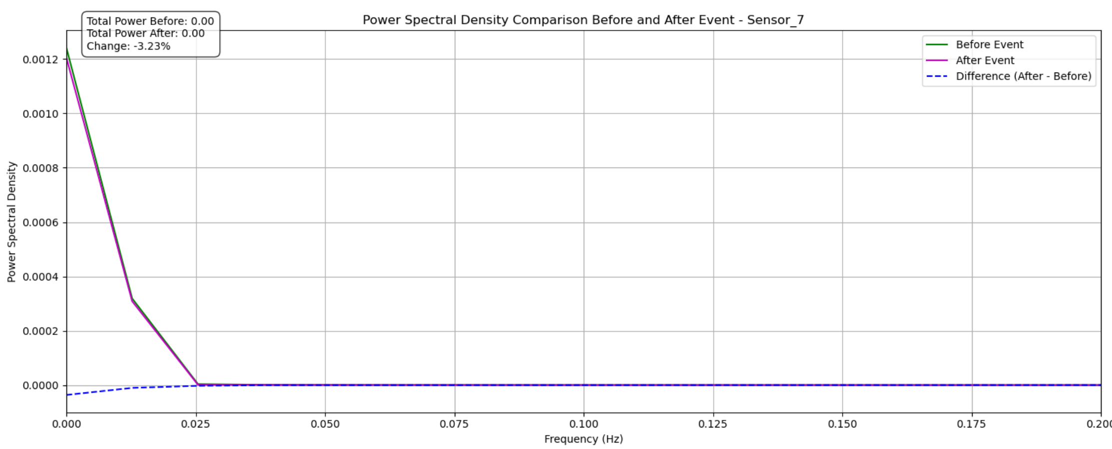
| | Frequency | Before | After | Absolute_Change | Percent_Change |
|---|---|---|---|---|---|
| 2 | 0.025444 | 3.428297e-07 | 8.514534e-08 | -2.576844e-07 | -75.142039 |
| 6 | 0.076331 | 2.259049e-08 | 1.027952e-08 | -1.231097e-08 | -54.256069 |
| 10 | 0.127218 | 7.843084e-09 | 3.779859e-09 | -4.063225e-09 | -51.154248 |
| 14 | 0.178106 | 4.012981e-09 | 1.925220e-09 | -2.087761e-09 | -50.760287 |
| 8 | 0.101775 | 1.245246e-08 | 6.085727e-09 | -6.366729e-09 | -50.720982 |

=== Analysis for Sensor_7 ===
Analyzing frequency: 0.0254 Hz (closest to 0.02 Hz)
Average magnitude before event: 0.0005
Average magnitude after event: 0.0003

Change: -0.0001 (-28.95%)




Energy Analysis:
Total energy before event: 0.0001
Total energy after event: 0.0000
Average energy before event: 0.0000
Average energy after event: 0.0000
Energy change: -0.0000 (-74.04%)




Power Spectral Density Analysis:
Total power before event: 0.0016
Total power after event: 0.0015
Absolute power change: -0.0001
Relative power change: -3.23%
Saved significant changes data to:

significant_changes_csv_Mushroom_25-05-08_0326\Sensor_7_significant_changes.csv

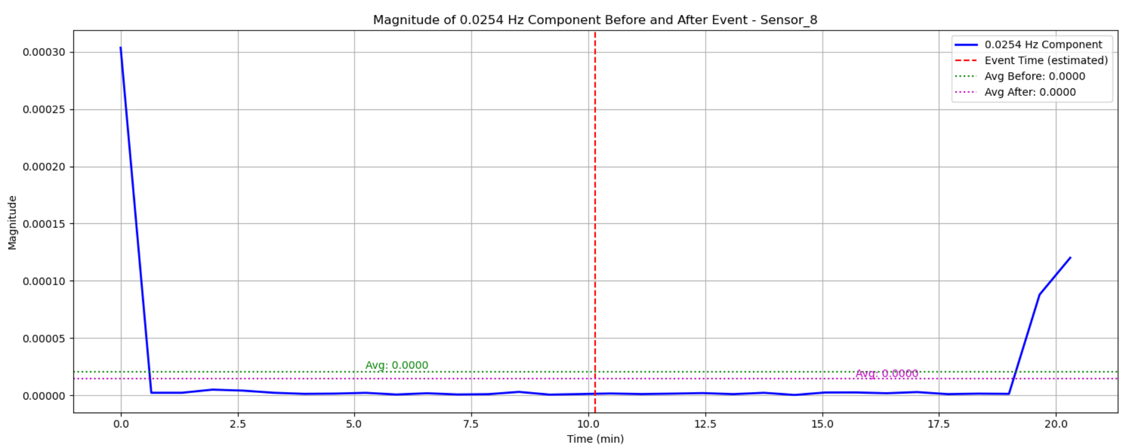
Top 5 frequencies with largest power increase:

| | Frequency | Before | After | Absolute_Change | Percent_Change |
|---|---|---|---|---|---|
| 0 | 0.000000 | 1.241843e-03 | 1.205521e-03 | -3.632197e-05 | -2.924843 |
| 1 | 0.012722 | 3.187832e-04 | 3.084939e-04 | -1.028924e-05 | -3.227659 |
| 3 | 0.038165 | 9.158139e-07 | 6.213960e-07 | -2.944178e-07 | -32.144707 |
| 108 | 1.373957 | 1.322142e-09 | 7.285206e-10 | -5.936212e-10 | -41.741353 |
| 120 | 1.526619 | 1.255898e-09 | 6.873418e-10 | -5.685566e-10 | -41.932093 |

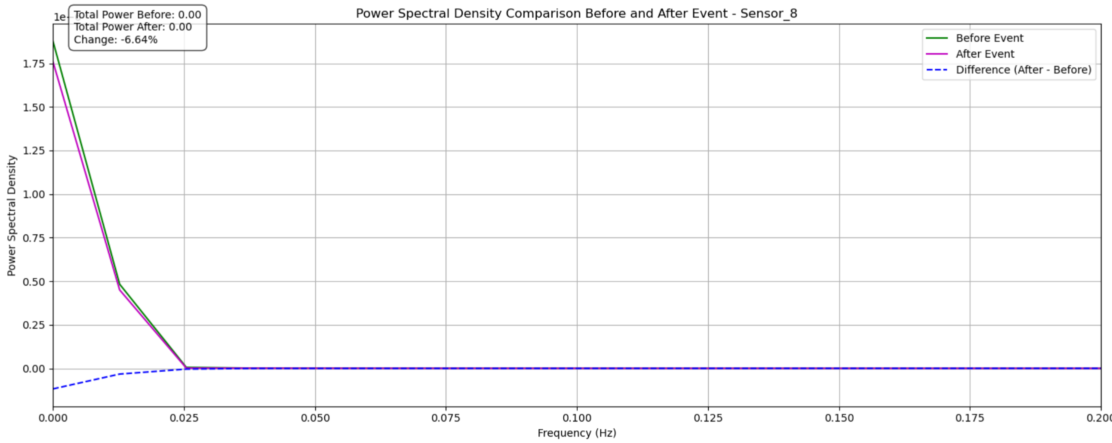
Top 5 frequencies with largest power decrease:

| | Frequency | Before | After | Absolute_Change | Percent_Change |
|---|---|---|---|---|---|
| 2 | 0.025444 | 3.662332e-06 | 1.014217e-06 | -2.648115e-06 | -72.304825 |
| 6 | 0.076331 | 2.426162e-07 | 1.223967e-07 | -1.202195e-07 | -49.530903 |
| 10 | 0.127218 | 8.438285e-08 | 4.492592e-08 | -3.945694e-08 | -46.704076 |
| 16 | 0.203549 | 3.291473e-08 | 1.766753e-08 | -1.524719e-08 | -46.183010 |
| 14 | 0.178106 | 4.296656e-08 | 2.313103e-08 | -1.983553e-08 | -46.057846 |

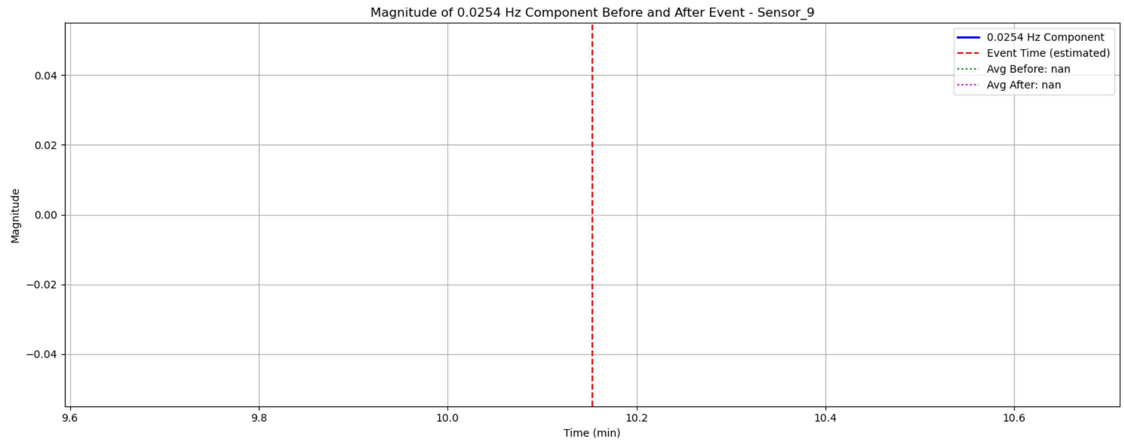
=== Analysis for Sensor_8 ===
Analyzing frequency: 0.0254 Hz (closest to 0.02 Hz)
Average magnitude before event: 0.0000
Average magnitude after event: 0.0000
Change: -0.0000 (-30.74%)



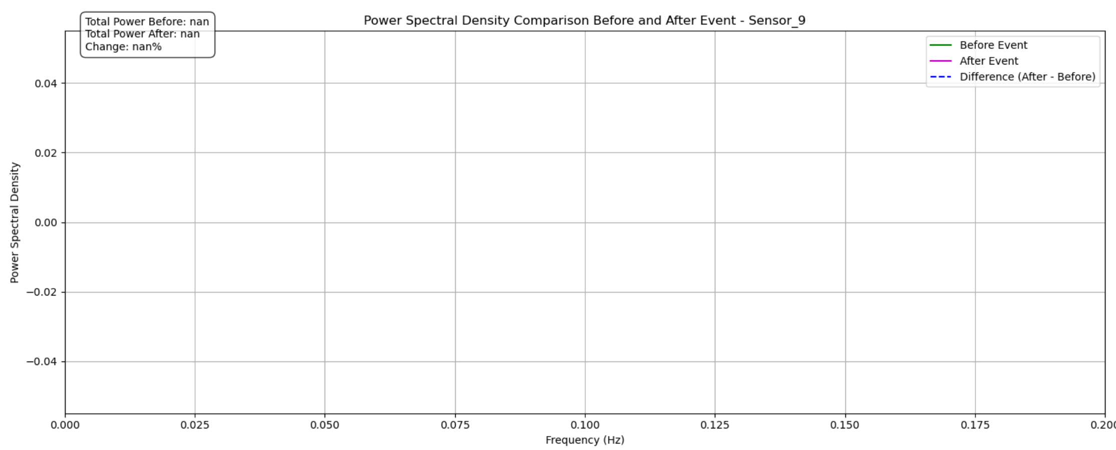

Energy Analysis:
Total energy before event: 0.0000
Total energy after event: 0.0000
Average energy before event: 0.0000
Average energy after event: 0.0000
Energy change: -0.0000 (-75.94%)




```
Power Spectral Density Analysis:
Total power before event: 0.0000
Total power after event: 0.0000
Absolute power change: -0.0000
Relative power change: -6.64%
Saved significant changes data to:
significant_changes_csv_Mushroom_25-05-08_0326\Sensor_8_significant_changes.csv

Top 5 frequencies with largest power increase:
     Frequency        Before         After  Absolute_Change  Percent_Change
112   1.424845  2.306109e-12  1.996446e-12    -3.096627e-13       -0.302682
113   1.437567  2.500586e-12  1.989339e-12    -5.112475e-13       -0.498775
124   1.577507  2.046662e-12  1.530468e-12    -5.161941e-13       -0.505841
100   1.272183  2.581782e-12  2.047705e-12    -5.340773e-13       -0.520636
115   1.463010  2.846482e-12  2.187398e-12    -6.590835e-13       -0.640842

Top 5 frequencies with largest power decrease:
   Frequency        Before         After  Absolute_Change  Percent_Change
2   0.025444  5.764520e-09  1.478961e-09    -4.285559e-09      -73.076040
6   0.076331  3.853444e-10  1.756238e-10    -2.097206e-10      -43.210685
4   0.050887  9.172181e-10  4.814072e-10    -4.358109e-10      -42.843406
5   0.063609  5.211042e-10  2.609417e-10    -2.601625e-10      -41.887097
3   0.038165  1.432377e-09  8.824809e-10    -5.498962e-10      -35.885175

=== Analysis for Sensor_9 ===
Analyzing frequency: 0.0254 Hz (closest to 0.02 Hz)
Average magnitude before event: nan
Average magnitude after event: nan
Change: nan (nan%)
```




Energy Analysis:
Total energy before event: nan
Total energy after event: nan
Average energy before event: nan
Average energy after event: nan
Energy change: nan (nan%)




Power Spectral Density Analysis:
Total power before event: nan
Total power after event: nan
Absolute power change: nan
Relative power change: nan%
Saved significant changes data to:
significant_changes_csv_Mushroom_25-05-08_0326\Sensor_9_significant_changes.csv

```
Top 5 frequencies with largest power increase:
   Frequency  Before  After  Absolute_Change  Percent_Change
0   0.000000     NaN    NaN              NaN             NaN
1   0.012722     NaN    NaN              NaN             NaN
2   0.025444     NaN    NaN              NaN             NaN
3   0.038165     NaN    NaN              NaN             NaN
4   0.050887     NaN    NaN              NaN             NaN

Top 5 frequencies with largest power decrease:
   Frequency  Before  After  Absolute_Change  Percent_Change
0   0.000000     NaN    NaN              NaN             NaN
1   0.012722     NaN    NaN              NaN             NaN
2   0.025444     NaN    NaN              NaN             NaN
3   0.038165     NaN    NaN              NaN             NaN
4   0.050887     NaN    NaN              NaN             NaN
```

[9]:

```python
# Delete data for the 9th sensor in the␣
 ↪significant_changes_csv_Mushroom_25-05-08_0326 directory

import os
import shutil

# Define the directory containing the CSV files
csv_dir_path = "significant_changes_csv_Mushroom_25-05-08_0326"

# Get all CSV files in the directory
csv_files = [f for f in os.listdir(csv_dir_path) if f.
 ↪endswith('_significant_changes.csv')]

# Sort the files to ensure consistent ordering
csv_files.sort()

# Check if we have at least 9 sensors
if len(csv_files) >= 9:
    # Get the 9th sensor's filename (index 8 since zero-based)
    ninth_sensor_file = csv_files[8]
    ninth_sensor_path = os.path.join(csv_dir_path, ninth_sensor_file)

    # Print information about the file being deleted
    print(f"Deleting data for the 9th sensor: {ninth_sensor_file}")

    # Option 1: Delete the file
    os.remove(ninth_sensor_path)
    print(f"File {ninth_sensor_file} has been deleted.")

    # Alternative option (commented out): Create a backup instead of deleting
    # backup_path = ninth_sensor_path + ".backup"
```

```
    # shutil.copy2(ninth_sensor_path, backup_path)
    # os.remove(ninth_sensor_path)
    # print(f"File {ninth_sensor_file} has been deleted. Backup created at␣
 ↪{backup_path}")
else:
    print(f"There are only {len(csv_files)} sensor files in the directory,␣
 ↪cannot delete the 9th sensor.")
```

```
Deleting data for the 9th sensor: Sensor_9_significant_changes.csv
File Sensor_9_significant_changes.csv has been deleted.
```

[10]:

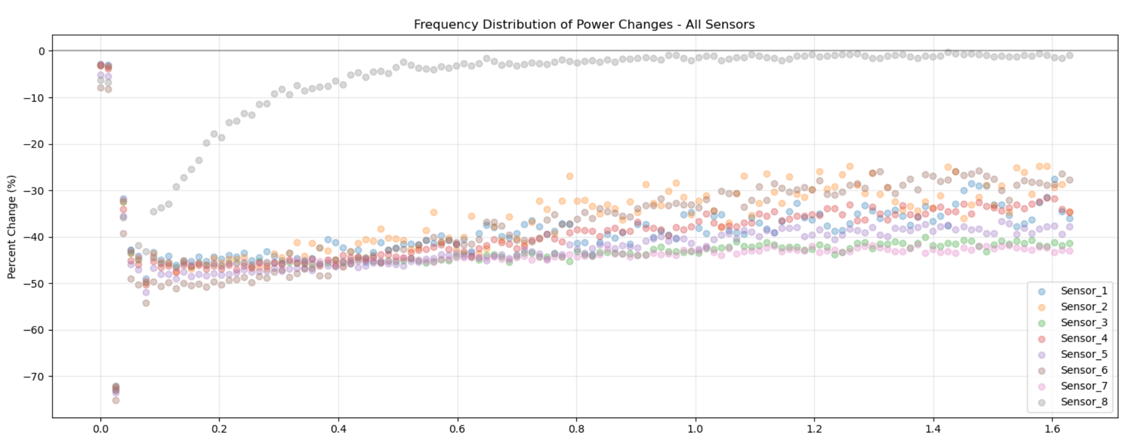
```
import seaborn as sns

# Analyze significant changes across all sensors
print("\nAnalyzing significant changes across all sensors...")

# Define the directory containing the CSV files
csv_dir_path = "significant_changes_csv_Mushroom_25-05-08_0326"

# Get all CSV files in the directory
csv_files = [f for f in os.listdir(csv_dir_path) if f.
 ↪endswith('_significant_changes.csv')]

# Initialize lists to store summary data
sensor_names = []
top_increase_freqs = []
top_decrease_freqs = []
all_sensor_data = {}

# Create a figure for comparing all sensors
plt.figure(figsize=(15, 6))

# Process each sensor's data
for csv_file in csv_files:
    # Extract sensor name from filename
    sensor_name = csv_file.split('_significant_changes.csv')[0]
    sensor_names.append(sensor_name)

    # Load the CSV data
    csv_path = os.path.join(csv_dir_path, csv_file)
    sensor_data = pd.read_csv(csv_path)
    all_sensor_data[sensor_name] = sensor_data

    # Sort by absolute percent change
    sensor_data['Abs_Percent_Change'] = np.abs(sensor_data['Percent_Change'])

    # Get top increases and decreases
```

```
    top_increases = sensor_data.sort_values('Percent_Change', ascending=False).
↪head(20)
    top_increase_freqs.append(top_increases['Frequency'].tolist())

    top_decreases = sensor_data.sort_values('Percent_Change', ascending=True).
↪head(20)
    top_decrease_freqs.append(top_decreases['Frequency'].tolist())

    # Plot frequency vs percent change for this sensor
    plt.scatter(sensor_data['Frequency'], sensor_data['Percent_Change'],
                alpha=0.3, label=sensor_name)

# Add plot details
plt.axhline(y=0, color='k', linestyle='-', alpha=0.3)
plt.xlabel('Frequency (Hz)')
plt.ylabel('Percent Change (%)')
plt.title('Frequency Distribution of Power Changes - All Sensors')
plt.legend()
plt.grid(True, alpha=0.3)
plt.tight_layout()
plt.show()

# Analyze patterns in top increases and decreases
print("\nAnalyzing patterns in top increases and decreases...")

# Create figures for top increases and decreases
plt.figure(figsize=(15, 6))
plt.subplot(1, 2, 1)
for i, sensor_name in enumerate(sensor_names):
    sensor_data = all_sensor_data[sensor_name]
    top_increases = sensor_data.sort_values('Percent_Change', ascending=False).
↪head(10)
    plt.scatter(top_increases['Frequency'], top_increases['Percent_Change'],
                label=sensor_name, s=100, alpha=0.7)

    # Removed annotation of frequencies to avoid overlapping text

plt.title('Top 10 Frequencies with Largest Increases')
plt.xlabel('Frequency (Hz)')
plt.ylabel('Percent Change (%)')
plt.legend()
plt.grid(True, alpha=0.3)

plt.subplot(1, 2, 2)
for i, sensor_name in enumerate(sensor_names):
    sensor_data = all_sensor_data[sensor_name]
```

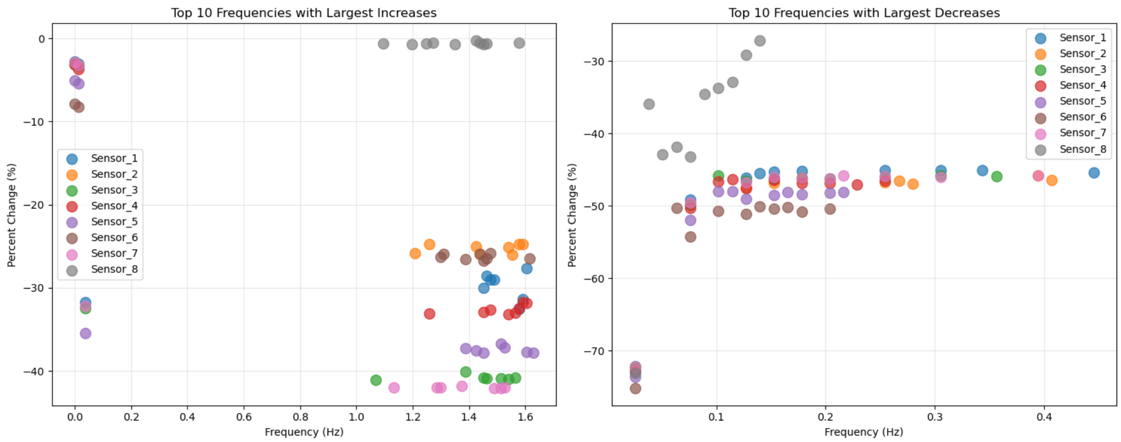
```
    top_decreases = sensor_data.sort_values('Percent_Change', ascending=True).
↪head(10)
    plt.scatter(top_decreases['Frequency'], top_decreases['Percent_Change'],
                label=sensor_name, s=100, alpha=0.7)

    # Removed annotation of frequencies to avoid overlapping text

plt.title('Top 10 Frequencies with Largest Decreases')
plt.xlabel('Frequency (Hz)')
plt.ylabel('Percent Change (%)')
plt.legend()
plt.grid(True, alpha=0.3)

plt.tight_layout()
plt.show()

# Analyze frequency overlap between sensors for top increases and decreases
print("\nAnalyzing frequency overlap between sensors...")

# For increases
increase_overlap = set(top_increase_freqs[0])
for freqs in top_increase_freqs[1:]:
    increase_overlap = increase_overlap.intersection(set(freqs))

# For decreases
decrease_overlap = set(top_decrease_freqs[0])
for freqs in top_decrease_freqs[1:]:
    decrease_overlap = decrease_overlap.intersection(set(freqs))

print(f"Common frequencies showing increases across all sensors:␣
↪{sorted(list(increase_overlap))}")
print(f"Common frequencies showing decreases across all sensors:␣
↪{sorted(list(decrease_overlap))}")

# Analyze the distribution of top changes by frequency range
for sensor_name in sensor_names:
    sensor_data = all_sensor_data[sensor_name]

    # Define frequency bands
    sensor_data['Frequency_Band'] = pd.cut(sensor_data['Frequency'],
                                          bins=[0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.
↪6, 0.7],
                                          labels=['0-0.1', '0.1-0.2', '0.2-0.
↪3', '0.3-0.4', '0.4-0.5', '0.5-0.6', '0.6-0.7'])

    # Count top increases and decreases by frequency band
```

```
    top_increases = sensor_data.sort_values('Percent_Change', ascending=False).
↪head(20)
    top_decreases = sensor_data.sort_values('Percent_Change', ascending=True).
↪head(20)

    increase_band_counts = top_increases['Frequency_Band'].value_counts().
↪sort_index()
    decrease_band_counts = top_decreases['Frequency_Band'].value_counts().
↪sort_index()

    # Plot distribution of top changes by frequency band
    plt.figure(figsize=(15, 6))
    plt.subplot(1, 2, 1)
    increase_band_counts.plot(kind='bar', color='green', alpha=0.7)
    plt.title(f'{sensor_name}: Distribution of Top 20 Increases by Frequency␣
↪Band')
    plt.xlabel('Frequency Band (Hz)')
    plt.ylabel('Count')
    plt.grid(True, alpha=0.3)

    plt.subplot(1, 2, 2)
    decrease_band_counts.plot(kind='bar', color='red', alpha=0.7)
    plt.title(f'{sensor_name}: Distribution of Top 20 Decreases by Frequency␣
↪Band')
    plt.xlabel('Frequency Band (Hz)')
    plt.ylabel('Count')
    plt.grid(True, alpha=0.3)

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

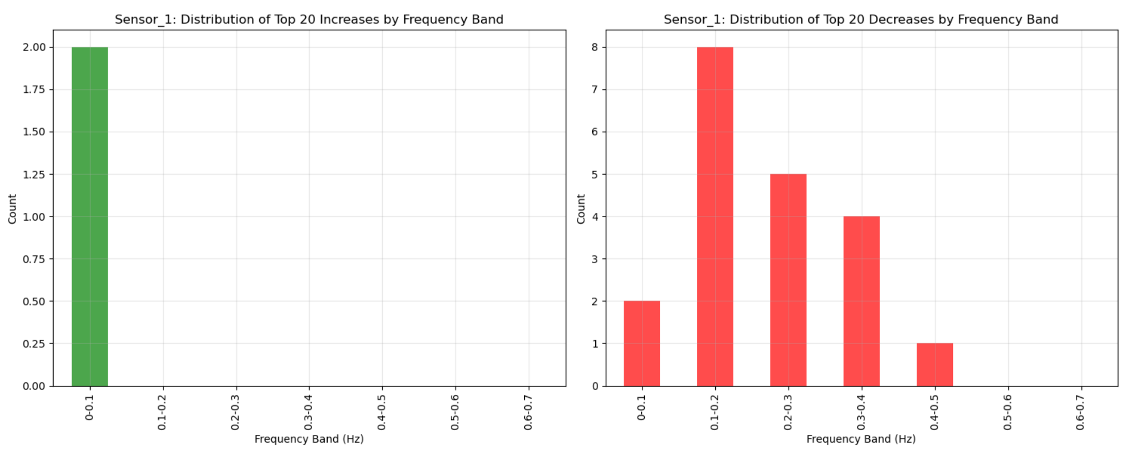
```
Analyzing significant changes across all sensors…
```

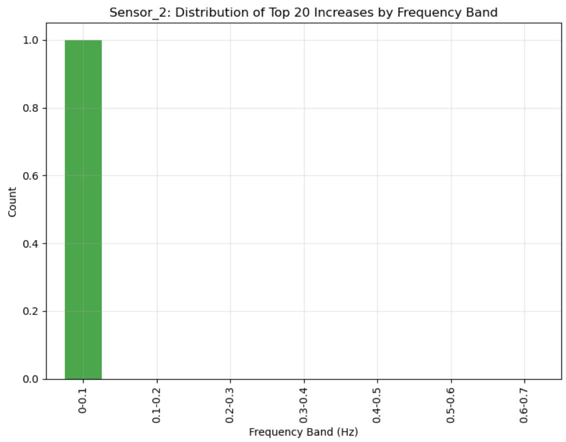


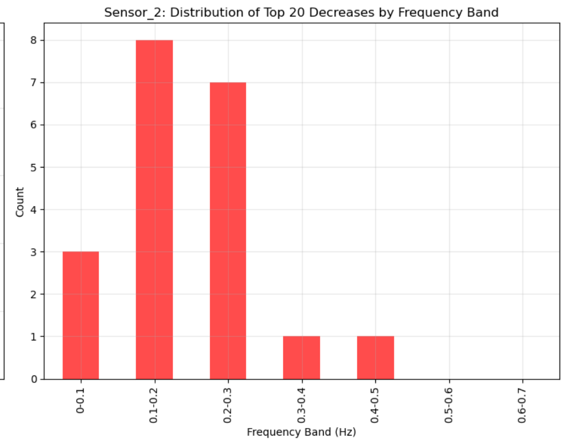
Analyzing patterns in top increases and decreases…

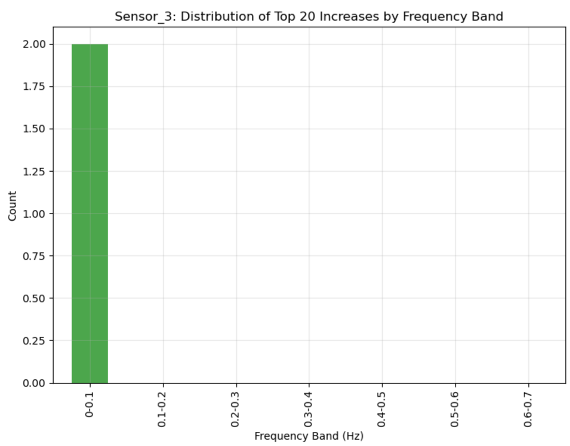


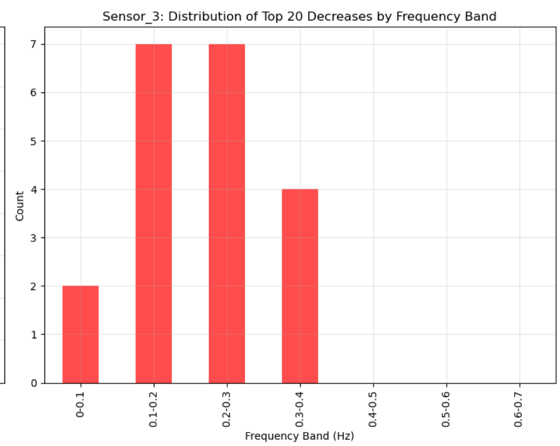

Analyzing frequency overlap between sensors…
Common frequencies showing increases across all sensors: []
Common frequencies showing decreases across all sensors: [0.02544365594141187, 0.0763309678242562, 0.101774623765675, 0.1144964517363844, 0.1272182797070937, 0.1526619356485125, 0.1653837636192219, 0.1781055915899312, 0.1908274195606406, 0.20354924753135, 0.2289929034727688, 0.2544365594141875]

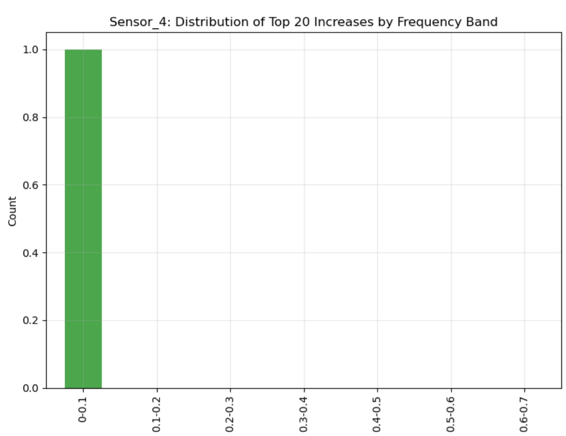


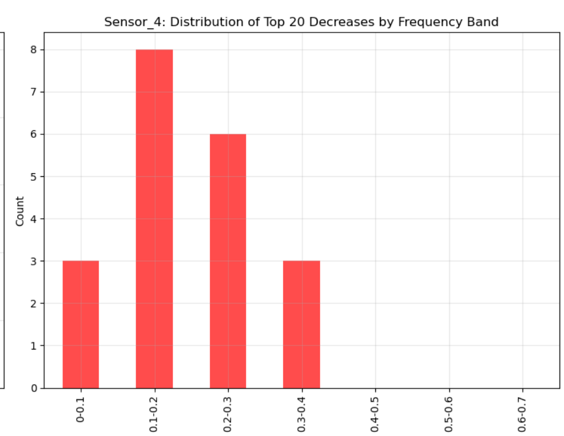

Sensor_2: Distribution of Top 20 Increases by Frequency Band
Count
Frequency Band (Hz)
0-0.1
0.1-0.2
0.2-0.3
0.3-0.4
0.4-0.5
0.5-0.6
0.6-0.7

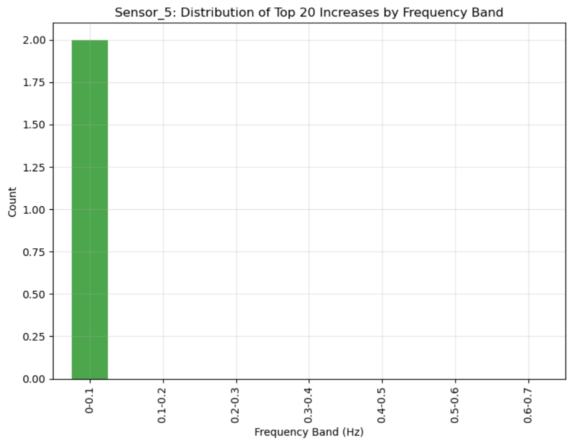
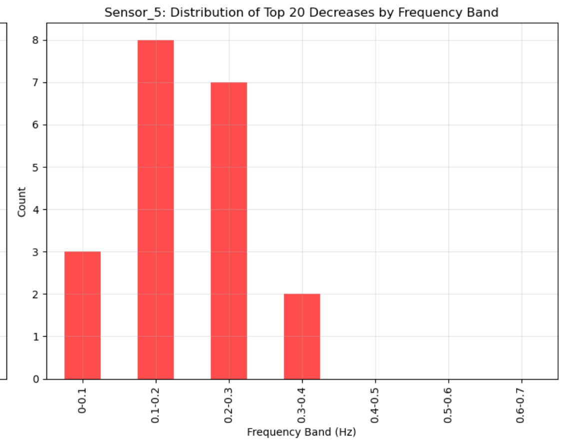
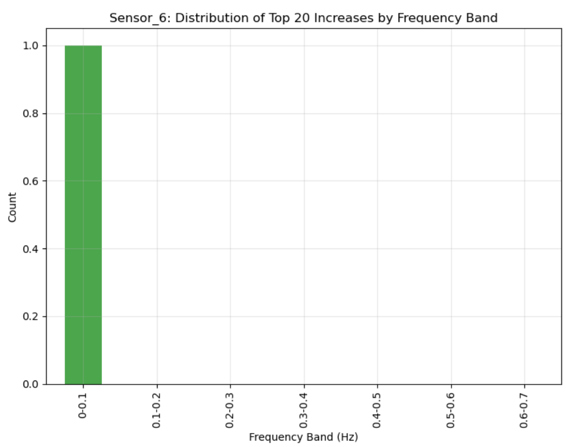
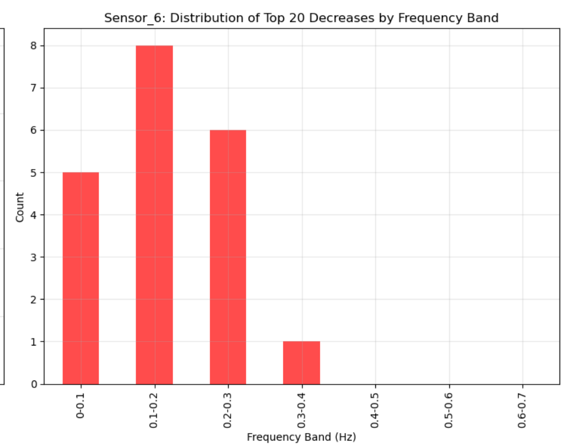
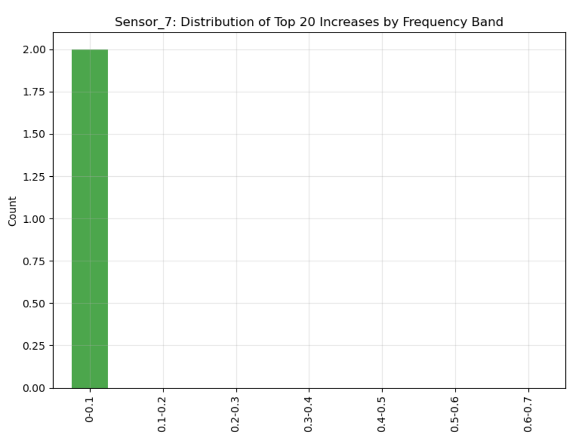
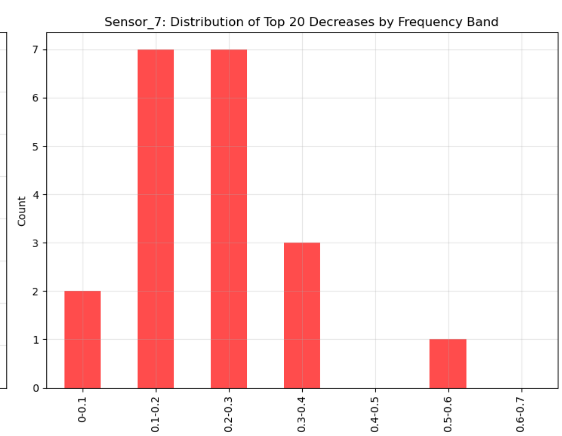

Sensor_2: Distribution of Top 20 Decreases by Frequency Band
Count
Frequency Band (Hz)
0-0.1
0.1-0.2
0.2-0.3
0.3-0.4
0.4-0.5
0.5-0.6
0.6-0.7

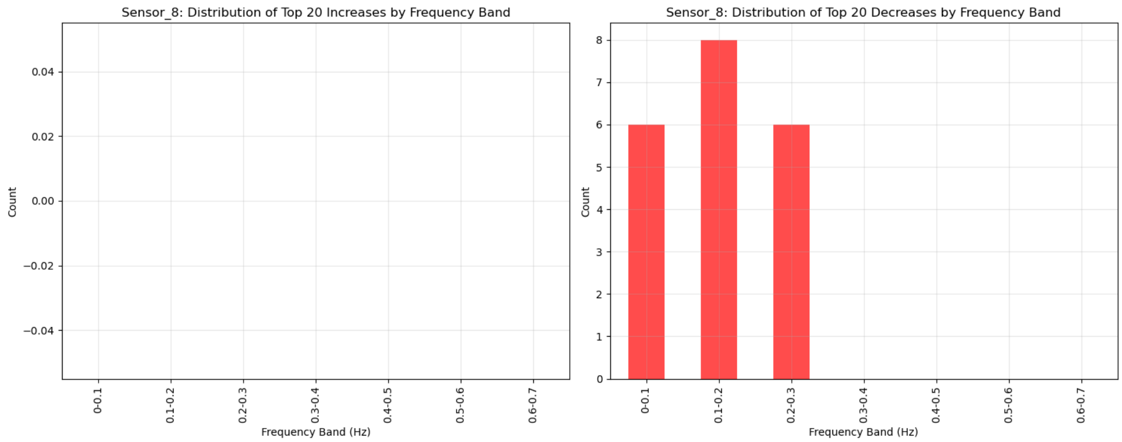

Sensor_3: Distribution of Top 20 Increases by Frequency Band
Count
Frequency Band (Hz)
0-0.1
0.1-0.2
0.2-0.3
0.3-0.4
0.4-0.5
0.5-0.6
0.6-0.7


Sensor_3: Distribution of Top 20 Decreases by Frequency Band
Count
Frequency Band (Hz)
0-0.1
0.1-0.2
0.2-0.3
0.3-0.4
0.4-0.5
0.5-0.6
0.6-0.7


Sensor_4: Distribution of Top 20 Increases by Frequency Band
Count
Frequency Band (Hz)
0-0.1
0.1-0.2
0.2-0.3
0.3-0.4
0.4-0.5
0.5-0.6
0.6-0.7


Sensor_4: Distribution of Top 20 Decreases by Frequency Band
Count
Frequency Band (Hz)
0-0.1
0.1-0.2
0.2-0.3
0.3-0.4
0.4-0.5
0.5-0.6
0.6-0.7


Sensor_5: Distribution of Top 20 Increases by Frequency Band
Count
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
0-0.1
0.1-0.2
0.2-0.3
0.3-0.4
0.4-0.5
0.5-0.6
0.6-0.7
Frequency Band (Hz)

Sensor_5: Distribution of Top 20 Decreases by Frequency Band
Count
0
1
2
3
4
5
6
7
8
0-0.1
0.1-0.2
0.2-0.3
0.3-0.4
0.4-0.5
0.5-0.6
0.6-0.7
Frequency Band (Hz)

Sensor_6: Distribution of Top 20 Increases by Frequency Band
Count
0.0
0.2
0.4
0.6
0.8
1.0
0-0.1
0.1-0.2
0.2-0.3
0.3-0.4
0.4-0.5
0.5-0.6
0.6-0.7
Frequency Band (Hz)

Sensor_6: Distribution of Top 20 Decreases by Frequency Band
Count
0
1
2
3
4
5
6
7
8
0-0.1
0.1-0.2
0.2-0.3
0.3-0.4
0.4-0.5
0.5-0.6
0.6-0.7
Frequency Band (Hz)

Sensor_7: Distribution of Top 20 Increases by Frequency Band
Count
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
0-0.1
0.1-0.2
0.2-0.3
0.3-0.4
0.4-0.5
0.5-0.6
0.6-0.7
Frequency Band (Hz)

Sensor_7: Distribution of Top 20 Decreases by Frequency Band
Count
0
1
2
3
4
5
6
7
0-0.1
0.1-0.2
0.2-0.3
0.3-0.4
0.4-0.5
0.5-0.6
0.6-0.7
Frequency Band (Hz)


Sensor_8: Distribution of Top 20 Increases by Frequency Band
Count
Frequency Band (Hz)
0-0.1
0.1-0.2
0.2-0.3
0.3-0.4
0.4-0.5
0.5-0.6
0.6-0.7
Sensor_8: Distribution of Top 20 Decreases by Frequency Band

```
[ ]:
```