EXPERIMENT 5

In [ ]:

```
import numpy as np
from matplotlib import pyplot as plt
from scipy.stats import norm
import warnings
warnings.filterwarnings("ignore")
```

In [ ]:

```
# Loading the data
B3_C11 = np.loadtxt('C11_1024_Packets_B3.out')
B3_C12 = np.loadtxt('C12_1024_Packets_B3.out')
B5_C11 = np.loadtxt('C11_1024_Packets_B5.out')
B5_C12 = np.loadtxt('C12_1024_Packets_B5.out')
```

In [ ]:

```
# Setting Constants
delta_t = 2.5e-9
delta_f = 4e+8

time_samples = np.arange(0, len(B3_C11)) * delta_t
```

1. Plots of Voltage vs Time

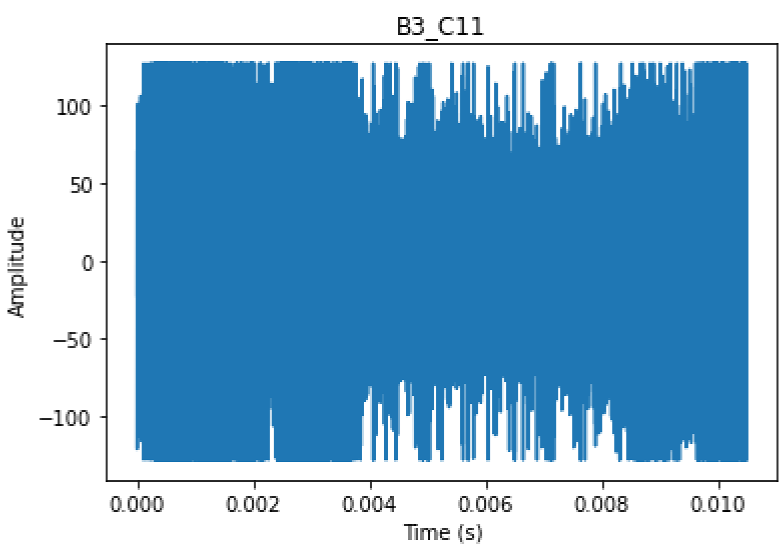
1.1 Voltage vs Time for Band 3, Antennae C11

In [ ]:

```
plt.plot(time_samples, B3_C11)
plt.title('B3_C11')
plt.xlabel('Time (s)')
plt.ylabel('Amplitude')
```

Out[ ]:

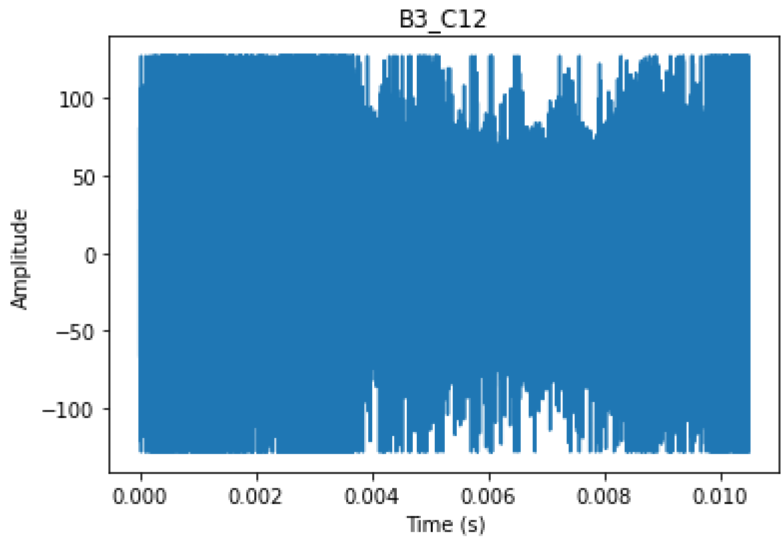
```
Text(0, 0.5, 'Amplitude')
```



1.2 Voltage vs Time for Band 3, Antennae C12

In [ ]:
```
plt.plot(time_samples, B3_C12)
plt.title('B3_C12')
plt.xlabel('Time (s)')
plt.ylabel('Amplitude')
```

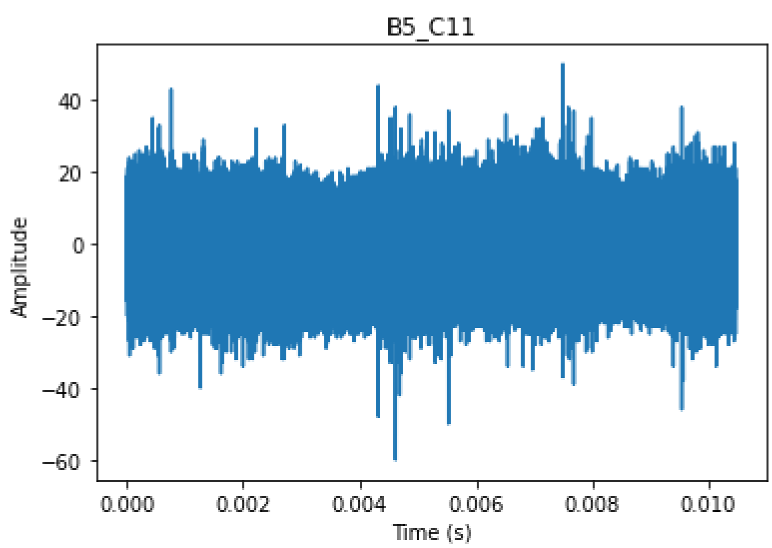
Out[ ]:
```
Text(0, 0.5, 'Amplitude')
```



1.3 Voltage vs Time for Band 5, Antennae C11

In [ ]:
```
plt.plot(time_samples, B5_C11)
plt.title('B5_C11')
plt.xlabel('Time (s)')
plt.ylabel('Amplitude')
```

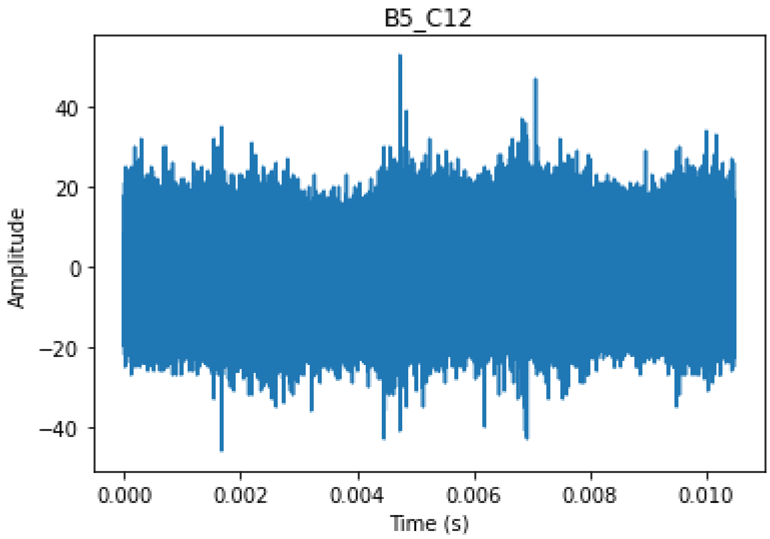
Out[ ]:
```
Text(0, 0.5, 'Amplitude')
```



1.4 Voltage vs Time for Band 5, Antennae C12

In [ ]:
```
plt.plot(time_samples, B5_C12)
plt.title('B5_C12')
plt.xlabel('Time (s)')
plt.ylabel('Amplitude')
```

Out[ ]: Text(0, 0.5, 'Amplitude')



2. Fitting a Gaussian to the data

In [ ]:

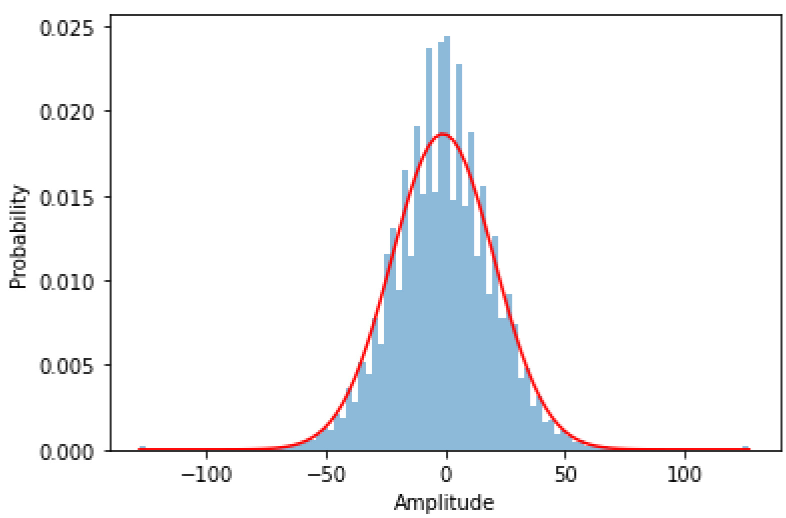
```
def fit_normal_distribution(data, name):
    mean, std = norm.fit(data)
    print(f'{name}: mean = {mean}, std = {std}')
    x_values = np.linspace(min(data), max(data), 100)
    data_pdf = (1/np.sqrt(2*np.pi*std**2)) * np.exp(-(x_values-mean)**2/(2*std**2))
    plt.xlabel('Amplitude')
    plt.ylabel('Probability')
    plt.hist(data, bins=100, density=True, alpha=0.5)
    plt.plot(x_values, data_pdf, 'r', label= name + '_pdf')
    plt.show()
    dc_offset_removed = data - mean
    return mean, std, dc_offset_removed
```

2.1 Gaussian Fit for Band 3, Antennae C11

In [ ]:

```
B3C11_mean, B3C11_std, B3_C11_dc_removed = fit_normal_distribution(B3_C11, 'B3_C11')
```

B3_C11: mean = -1.0569393634796143, std = 21.385557014213084

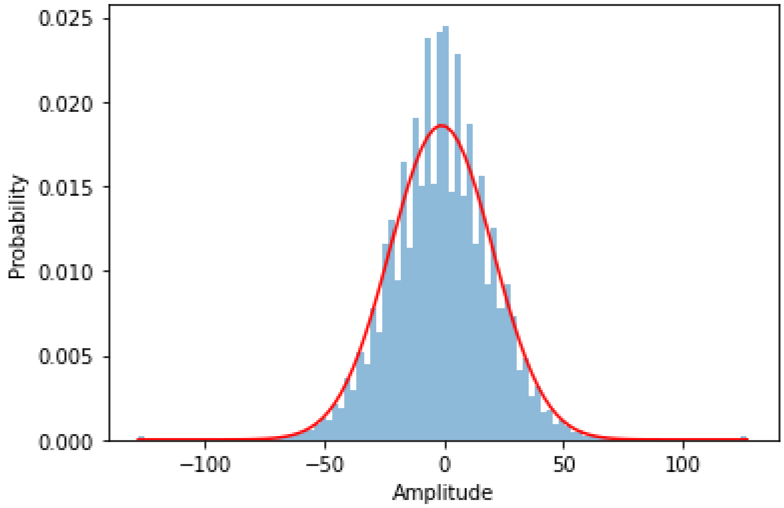


2.2 Gaussian Fit for Band 3, Antennae C12

In [ ]:

```
B3C12_mean, B3C12_std, B3_C12_dc_removed  = fit_normal_distribution(B3_C12, 'B3_C12')
```

```
B3_C12: mean = -1.058096170425415, std = 21.462586466437152
```

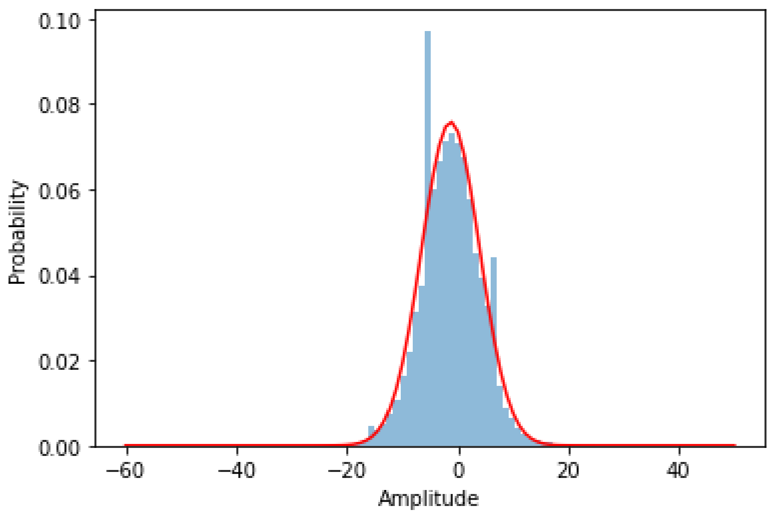


2.3 Gaussian Fit for Band 5, Antennae C11

In [ ]:

```
B5C11_mean, B5C11_std, B5_C11_dc_removed  = fit_normal_distribution(B5_C11, 'B5_C11')
```

```
B5_C11: mean = -1.3008010387420654, std = 5.265730068815026
```

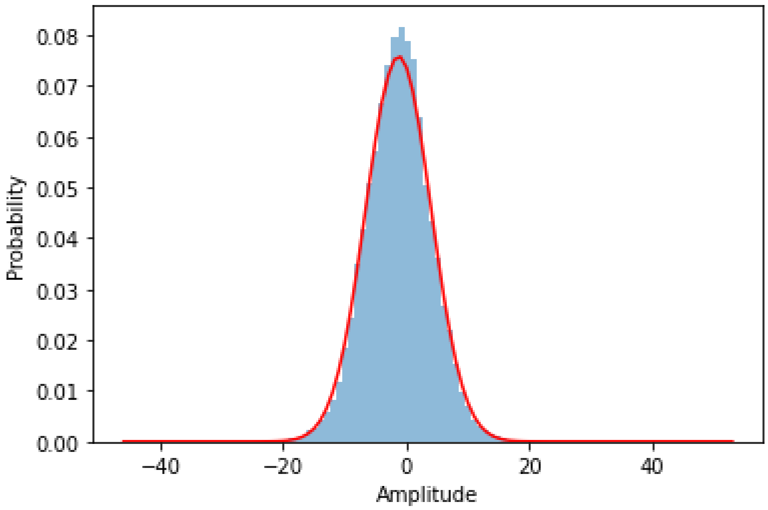


2.4 Gaussian Fit for Band 5, Antennae C12

In [ ]:

```
B5C12_mean, B5C12_std, B5_C12_dc_removed  = fit_normal_distribution(B5_C12, 'B5_C12')
```

```
B5_C12: mean = -1.3019917011260986, std = 5.259691845641863
```



3. Properties of Power

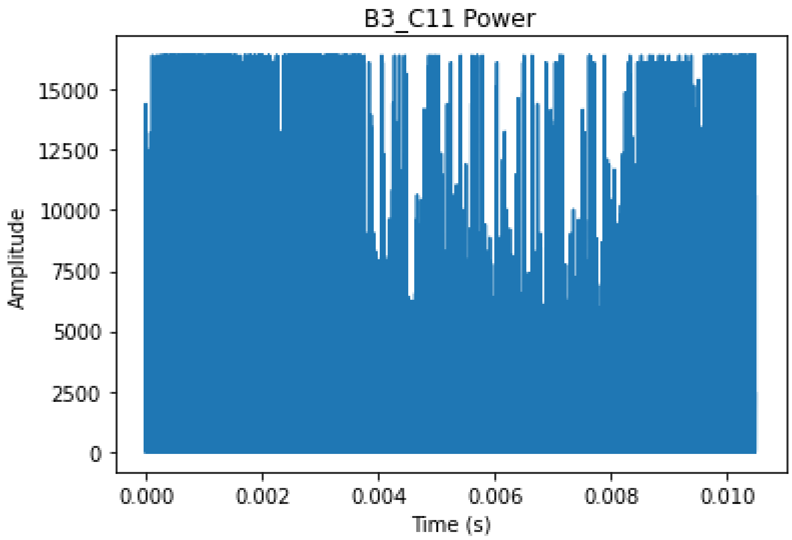
In [ ]:

```
def power_fit_normal_distribution(data, name):
    mean, std = norm.fit(data)
    print(f'{name}: mean = {mean}, std = {std}')
    x_values = np.linspace(min(data), max(data), 100)
    data_pdf = (1/np.sqrt(2*np.pi*std**2)) * np.exp(-(x_values-mean)**2/(2*std**2))
    plt.xlabel('Amplitude')
    plt.ylabel('Probability')
    plt.hist(data, bins=100, density=True, alpha=0.5)
    plt.plot(x_values, data_pdf, 'r', label= name + '_pdf')
    plt.show()

def plot_power_vs_time(data, name):
    plt.plot(time_samples, data)
    plt.title(name)
    plt.xlabel('Time (s)')
    plt.ylabel('Amplitude')
    plt.show()
```

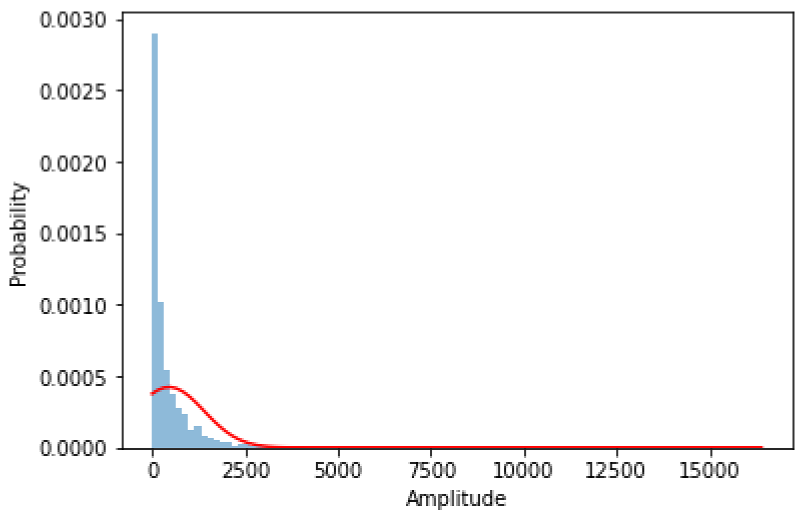
3.1 Gaussian Fit for Power at Band 3, Antennae C11

In [ ]:

```
plot_power_vs_time(B3_C11_dc_removed**2, 'B3_C11 Power')
power_fit_normal_distribution(B3_C11_dc_removed**2, 'B3_C11 Power')
```



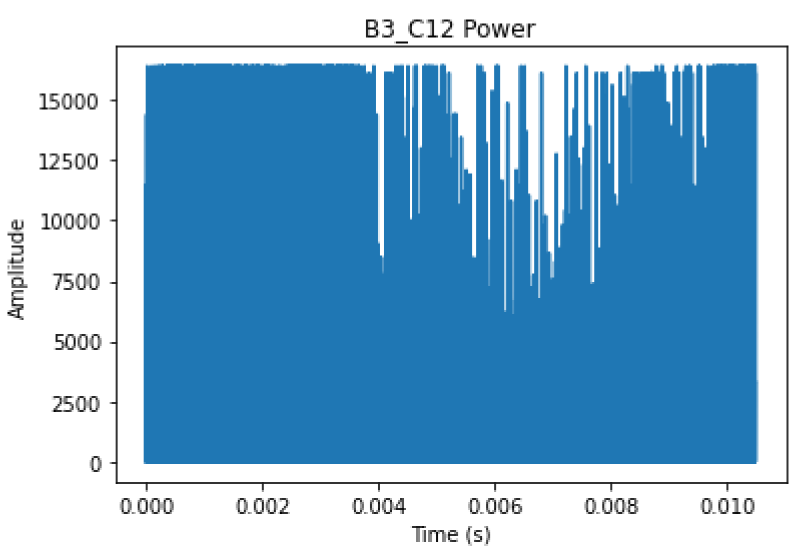
B3_C11 Power: mean = 457.34204880815844, std = 940.0560541920667



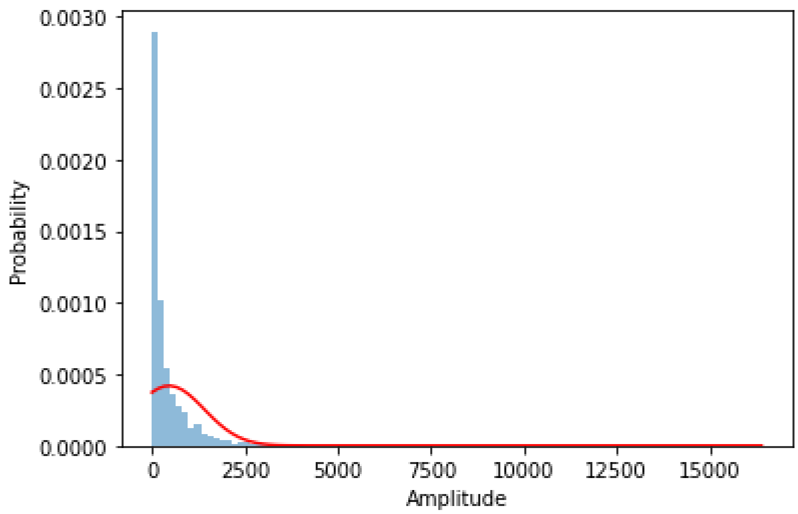
3.2 Gaussian Fit for Power at Band 3, Antennae C12

In [ ]:

```
plot_power_vs_time(B3_C12_dc_removed**2, 'B3_C12 Power')
power_fit_normal_distribution(B3_C12_dc_removed**2, 'B3_C12 Power')
```



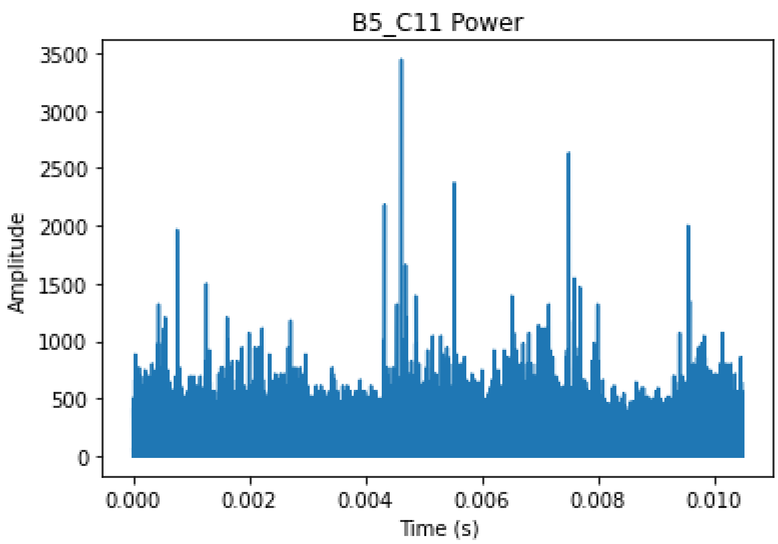
B3_C12 Power: mean = 460.64261782929117, std = 950.7699618623751



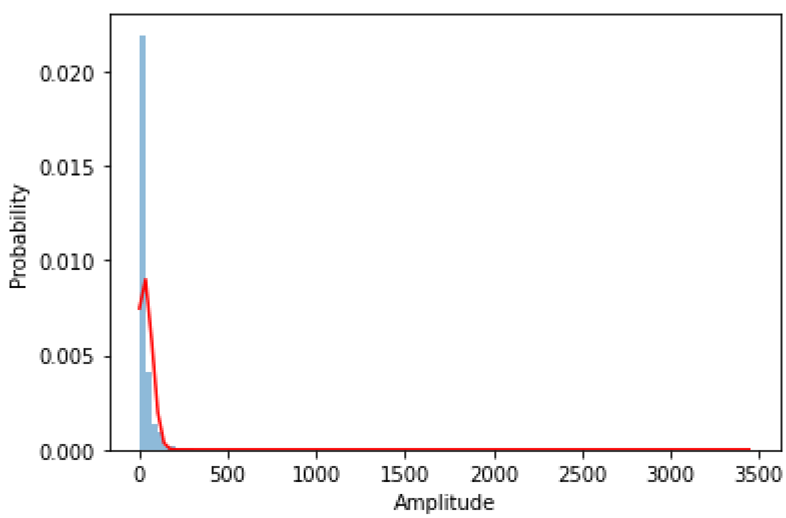
3.3 Gaussian Fit for Power at Band 5, Antennae C11

In [ ]:

```
plot_power_vs_time(B5_C11_dc_removed**2, 'B5_C11 Power')
power_fit_normal_distribution(B5_C11_dc_removed**2, 'B5_C11 Power')
```



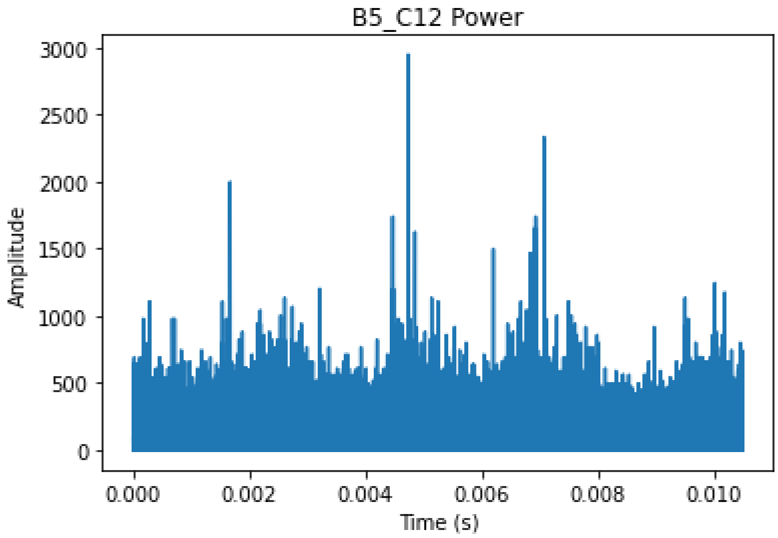

```
B5_C11 Power: mean = 27.7279131576227, std = 43.70293044221829
```



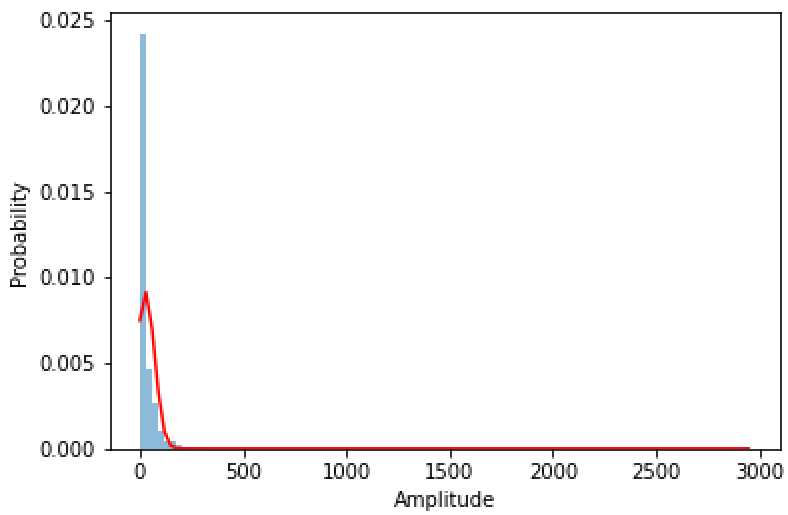
3.4 Gaussian Fit for Power at Band 5, Antennae C12

In [ ]:

```
plot_power_vs_time(B5_C12_dc_removed**2, 'B5_C12 Power')
power_fit_normal_distribution(B5_C12_dc_removed**2, 'B5_C12 Power')
```




```
B5_C12 Power: mean = 27.664358311111513, std = 43.57146254736617
```



4. Obtaining the Power Spectrum

In [ ]:

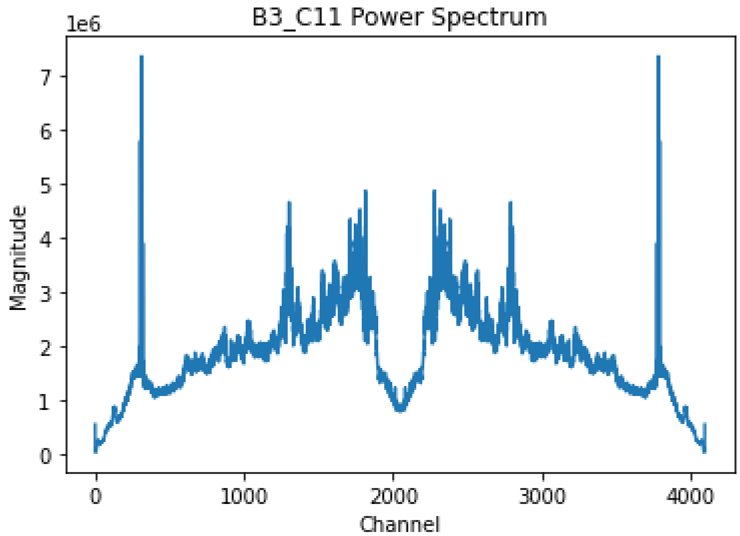
```
def figure_out_power_spectrum(data):
    fft_data = np.fft.fft(data)
    fft_conjugate = np.conj(fft_data)
    power_spectrum = fft_data * fft_conjugate
    return power_spectrum

def mean_power_spectrum(data):
    s = figure_out_power_spectrum(data[0:4096])
    for i in range(1, 1024):
        s += figure_out_power_spectrum(data[i*4096:(i+1)*4096])
    return s/1024

def plot_mean_power_spectrum(data, text):
    plt.plot(mean_power_spectrum(data)[1:])
    plt.xlabel('Channel')
    plt.ylabel('Magnitude')
    plt.title(text)
```

```python
def plot_dynamic_spectrum(data, text):
    pass
```

4.1 Mean Power Spectrums for 4 cases

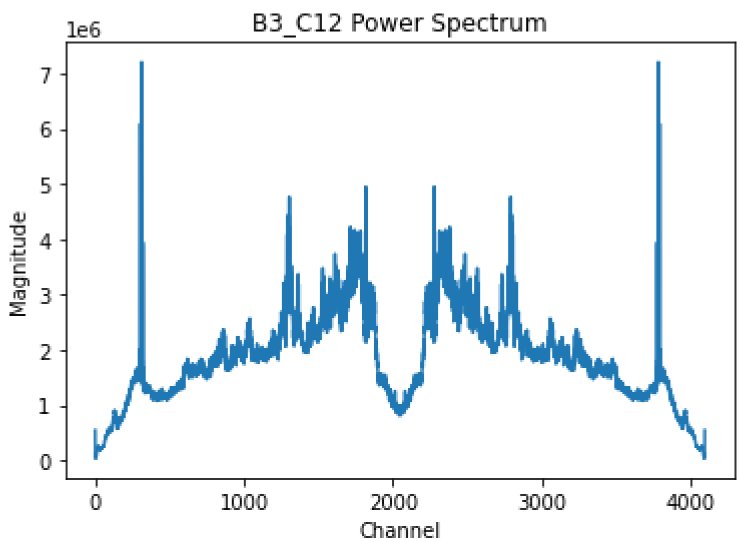
In [ ]:

```python
plot_mean_power_spectrum(B3_C11_dc_removed, 'B3_C11 Power Spectrum')
```



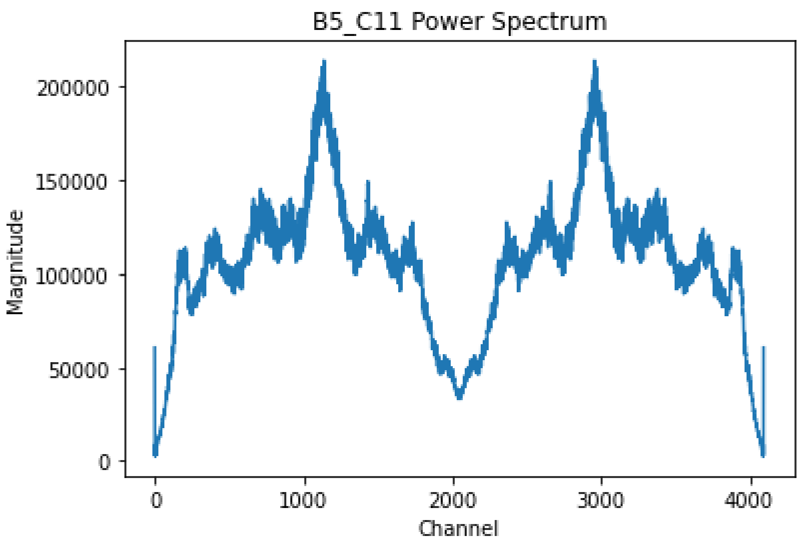
In [ ]:

```python
plot_mean_power_spectrum(B3_C12_dc_removed, 'B3_C12 Power Spectrum')
```



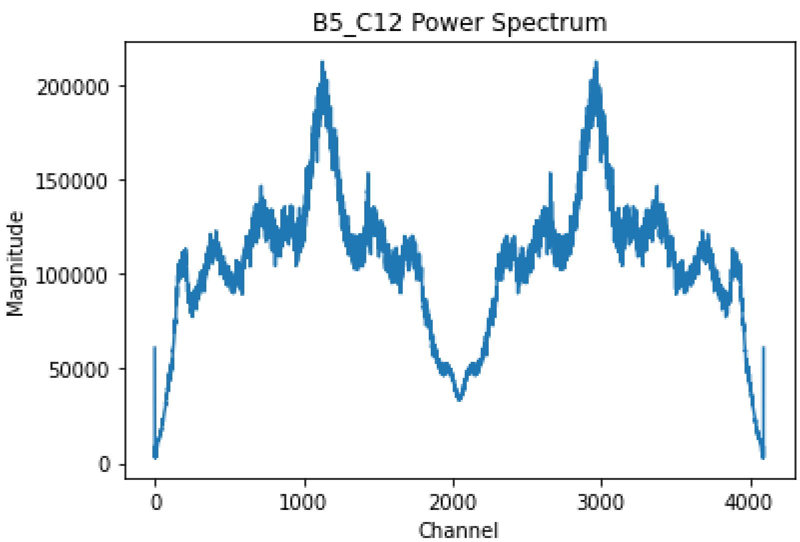
In [ ]:

```python
plot_mean_power_spectrum(B5_C11_dc_removed, 'B5_C11 Power Spectrum')
```



In [ ]:

```
plot_mean_power_spectrum(B5_C12_dc_removed, 'B5_C12 Power Spectrum')
```



5. Finding the RFI

In [ ]:

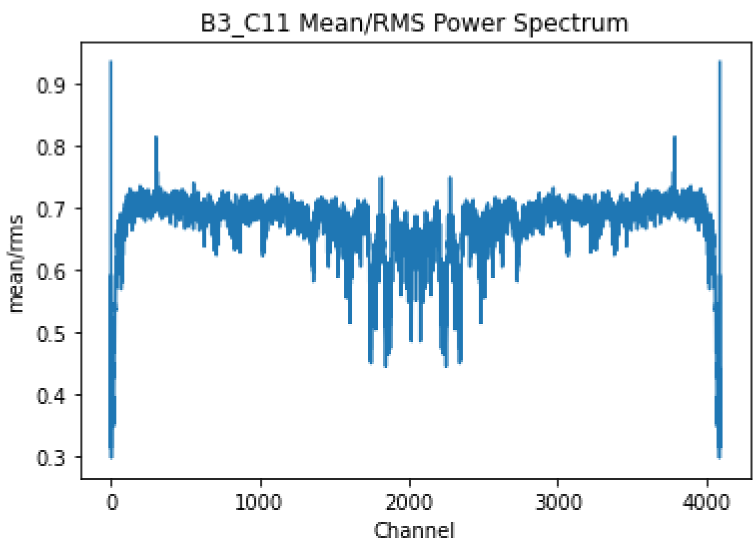
```
def mean_to_rms_ratio_power_spectrum(data):
    s = figure_out_power_spectrum(data[0:4096])
    sq = s**2
    for i in range(1, 1024):
        spec = figure_out_power_spectrum(data[i*4096:(i+1)*4096])
        s += spec
        sq += spec**2

    mean_power_spectrum = s/1024
    rms_power_spectrum = np.sqrt(sq/1024)
    return mean_power_spectrum/rms_power_spectrum

def plot_mean_to_rms_ratio_spectrum(data, text):
    plt.plot(mean_to_rms_ratio_power_spectrum(data)[1:])
    plt.xlabel('Channel')
    plt.ylabel('mean/rms')
    plt.title(text)
```

```python
def find_rfi_channels(mean_to_rms_spectrum, mean_power_spectrum):
    rslt = []
    for i in range(4096):
        mean_near_points = np.mean(mean_to_rms_spectrum[i-5:i+5])
        if np.abs(mean_to_rms_spectrum[i] - mean_near_points) > 0.1:
            power = mean_power_spectrum[i]
            print(f'Channel {i} is RFI, with power of {np.abs(power)}')
            rslt.append((i, power))

    return rslt
```

5.1 Mean/RMS Spectrums for 4 cases

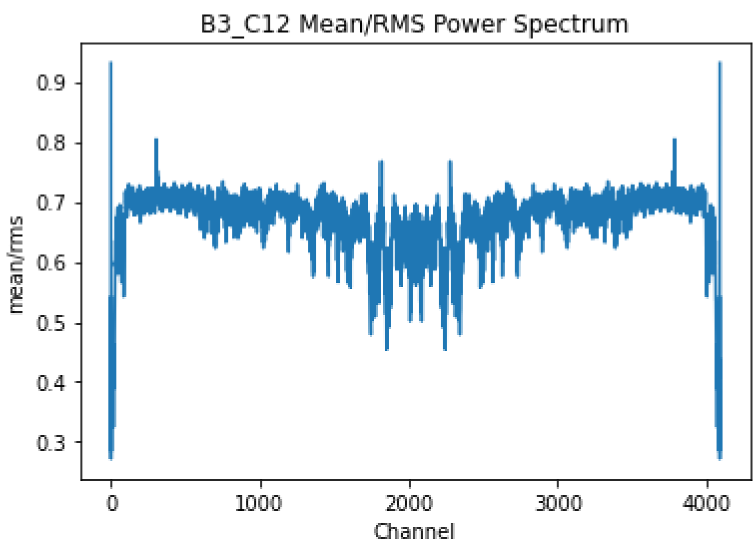
In [ ]:

```python
plot_mean_to_rms_ratio_spectrum(B3_C11_dc_removed, 'B3_C11 Mean/RMS Power Spectrum')
```



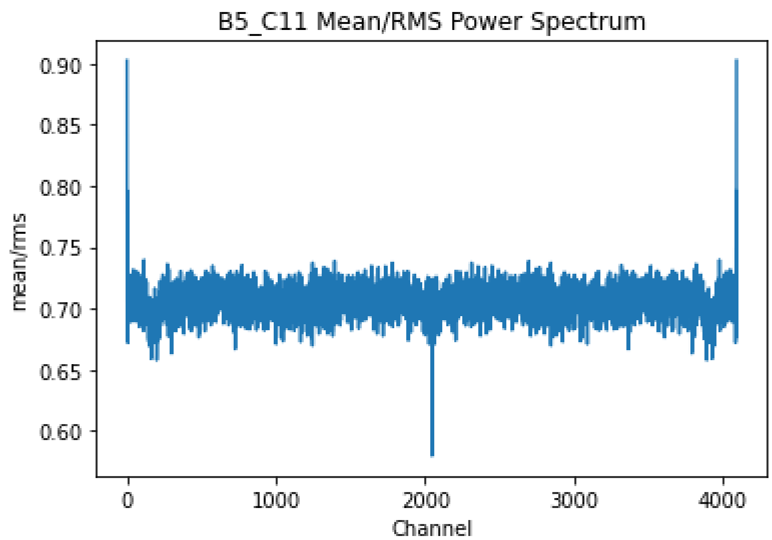
In [ ]:

```python
plot_mean_to_rms_ratio_spectrum(B3_C12_dc_removed, 'B3_C12 Mean/RMS Power Spectrum')
```



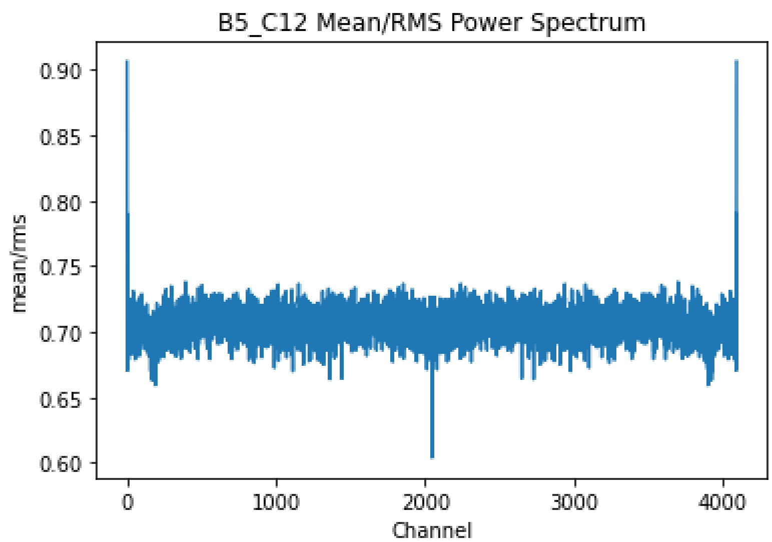
In [ ]:

```python
plot_mean_to_rms_ratio_spectrum(B5_C11_dc_removed, 'B5_C11 Mean/RMS Power Spectrum')
```



In [ ]:

```
plot_mean_to_rms_ratio_spectrum(B5_C12_dc_removed, 'B5_C12 Mean/RMS Power Spectrum')
```



5.2 Finding the RFI channels for 4 cases

In [ ]:

```
# RFIs for B3_C11
RFIs_B3C11 = find_rfi_channels(mean_to_rms_ratio_power_spectrum(B3_C11_dc_removed), mea
```

```
Channel 2018 is RFI, with power of 998104.7194856696
Channel 2078 is RFI, with power of 998104.7194856694
Channel 4092 is RFI, with power of 58906.51467313905
Channel 4093 is RFI, with power of 109135.44735023391
Channel 4094 is RFI, with power of 50869.46387530973
Channel 4095 is RFI, with power of 557209.5273902193
```

In [ ]:

```
# RFIs for B3_C12
RFIs_B3C12 = find_rfi_channels(mean_to_rms_ratio_power_spectrum(B3_C12_dc_removed),
                               mean_power_spectrum(B3_C12_dc_removed))
```

```
Channel 2009 is RFI, with power of 1084137.0803059288
Channel 2048 is RFI, with power of 884059.6376953125
Channel 2087 is RFI, with power of 1084137.0803059284
```

```
Channel 4092 is RFI, with power of 50418.10938389165
Channel 4094 is RFI, with power of 48276.90067039216
Channel 4095 is RFI, with power of 548489.1568930054
```

In [ ]:

```python
# RFIs for B5_C11
RFIs_B5C11 = find_rfi_channels(mean_to_rms_ratio_power_spectrum(B5_C11_dc_removed),
                               mean_power_spectrum(B5_C11_dc_removed))
```

```
Channel 2048 is RFI, with power of 33787.2041015625
Channel 4095 is RFI, with power of 60669.93759622133
```

In [ ]:

```python
# RFIs for B5_C12
RFIs_B5C11 = find_rfi_channels(mean_to_rms_ratio_power_spectrum(B5_C12_dc_removed),
                               mean_power_spectrum(B5_C12_dc_removed))
```

```
Channel 4095 is RFI, with power of 61076.061428678506
```