Biosignal Processing

▼ Introduction

// in progress

▼ 1. Dataset

Read Data and Extract the channels data

First of all, we analyze directory tree and role of each file.

```
sub-NORB00027
    └── ses-1
        ├── eeg
        │   ├── sub-NORB00027_ses-1_coordsystem.json
        │   ├── sub-NORB00027_ses-1_electrodes.tsv
        │   ├── sub-NORB00027_ses-1_task-EEG_channels.tsv
        │   ├── sub-NORB00027_ses-1_task-EEG_eeg.edf
        │   ├── sub-NORB00027_ses-1_task-EEG_eeg.json
        │   └── sub-NORB00027_ses-1_task-EEG_events.tsv
        └── sub-NORB00027_ses-1_scans.tsv
```

In `ses-1/eeg/`

▼ The file `sub-NORB00027_ses-1_coordsystem.json` contains metadata about the coordinate system used for EEG data.

```
{
  "EEGCoordinateSystem": "MNI305",
```

```
  "EEGCoordinateUnits": "mm"
}
```

- `"EEGCoordinateSystem": "MNI305"`: This indicates that the EEG data coordinates are referenced to the **MNI305** space. MNI305 refers to an average brain template, also known as the **Montreal Neurological Institute (MNI) 305** reference system. It's a standardized space used for neuroimaging data that aligns individual brain anatomy to a common framework, facilitating comparison and analysis across subjects.
- `"EEGCoordinateUnits": "mm"`: This specifies that the units of measurement for the EEG data coordinates are in **millimeters (mm)**. This is important for precise localization of EEG electrodes or sources within the brain.

▼ The file `sub-NORB00027_ses-1_electrodes.tsv` is a **tab-separated values (TSV)** file that contains the **3D spatial coordinates** and **material information** for EEG electrodes placed on a subject's scalp

```
name   x    y   z type material
 Fp1 -32   77  19  cup  Ag/AgCl
 Fp2  32   77  19  cup  Ag/AgCl
  F3 -52   35  57  cup  Ag/AgCl
  F4  52   35  57  cup  Ag/AgCl
  C3 -69  -29  69  cup  Ag/AgCl
  C4  69  -29  69  cup  Ag/AgCl
  P3 -54  -92  44  cup  Ag/AgCl
  P4  54  -92  44  cup  Ag/AgCl
  O1 -31 -116  -7  cup  Ag/AgCl
  O2  31 -116  -7  cup  Ag/AgCl
  F7 -72   32   5  cup  Ag/AgCl
  F8  72   32   5  cup  Ag/AgCl
  T3 -83  -21  -4  cup  Ag/AgCl
  T4  83  -21  -4  cup  Ag/AgCl
  T5 -73  -74  -8  cup  Ag/AgCl
  T6  73  -74  -8  cup  Ag/AgCl
  FZ   0   37  84  cup  Ag/AgCl
```

```
CZ   0  -34 105  cup  Ag/AgCl
PZ   0  -98  66  cup  Ag/AgCl
```

- **name**: This column lists the **standardized names** of the EEG electrodes. For example, “Fp1” refers to the electrode positioned on the left side of the forehead.
- **x, y, z**: These columns provide the **3D Cartesian coordinates** of each electrode in millimeters. The coordinates are based on the MNI305 reference space, which is a standardized brain template. The **x-coordinate** indicates the position left (+) or right (-) of the midline, the **y-coordinate** indicates the position anterior (+) or posterior (-) relative to a central point, and the **z-coordinate** indicates the position superior (+) or inferior (-) relative to a horizontal plane.
- **type**: This indicates the **type of electrode** used, which in this case is “cup”. Cup electrodes are small, round, and typically made of a conductive material that can hold conductive gel to improve signal quality.
- **material**: This specifies the **material composition** of the electrodes, which is “Ag/AgCl” for all listed electrodes. Ag/AgCl stands for **silver/silver chloride**, a common material used for EEG electrodes due to its good conductivity and stable electrochemical properties

▼ The file `sub-NORB00027_ses-1_task-EEG_channels.tsv` is a **tab-separated values (TSV)** file that describes the characteristics and settings of EEG channels used during a recording session.

```
name    type    units   description sampling_frequency  l
ow_cutoff   high_cutoff notch   status
Fp1 EEG uV  electrode   200 0.5 100 n/a good
Fp2 EEG uV  electrode   200 0.5 100 n/a good
F3  EEG uV  electrode   200 0.5 100 n/a good
F4  EEG uV  electrode   200 0.5 100 n/a good
C3  EEG uV  electrode   200 0.5 100 n/a good
C4  EEG uV  electrode   200 0.5 100 n/a good
P3  EEG uV  electrode   200 0.5 100 n/a good
```

```
P4  EEG uV  electrode   200 0.5 100 n/a good
O1  EEG uV  electrode   200 0.5 100 n/a good
O2  EEG uV  electrode   200 0.5 100 n/a good
F7  EEG uV  electrode   200 0.5 100 n/a good
F8  EEG uV  electrode   200 0.5 100 n/a good
T3  EEG uV  electrode   200 0.5 100 n/a good
T4  EEG uV  electrode   200 0.5 100 n/a good
T5  EEG uV  electrode   200 0.5 100 n/a good
T6  EEG uV  electrode   200 0.5 100 n/a good
FZ  EEG uV  electrode   200 0.5 100 n/a good
CZ  EEG uV  electrode   200 0.5 100 n/a good
PZ  EEG uV  electrode   200 0.5 100 n/a good
```

- **name**: This column lists the **names of the EEG channels**, which correspond to the electrode placemen.
- **type**: Indicates the **type of data** recorded by the channel, which is EEG (Electroencephalography) in this case.
- **units**: Specifies the **units of measurement** for the EEG signal, which is microvolts (uV).
- **description**: Provides a **brief description** of the channel, here simply noted as “electrode”.
- **sampling_frequency**: The **rate at which the EEG data is sampled**, measured in Hertz (Hz). In this file, it’s **200 Hz**, meaning the EEG signal is recorded 200 times per second.
- **low_cutoff**: The **low-frequency cutoff** for filtering the EEG signal, measured in Hz. It’s **0.5 Hz** here, indicating that **frequencies below 0.5 Hz are filtered out to remove slow drifts or artifacts.**
- **high_cutoff**: The **high-frequency cutoff** for filtering the EEG signal, measured in Hz. It’s **100 Hz** here, meaning that **frequencies above 100 Hz are filtered out to remove high-frequency noise.**
- **notch**: Indicates whether a **notch filter** is applied to remove power line noise. “n/a” suggests that no notch filter is applied or it’s not applicable.

- **status**: Describes the **quality or condition** of the channel. “good” indicates that the channel is functioning properly and the data quality is acceptable.

▼ The file `sub-NORB00027_ses-1_task-EEG_eeg.edf` is the actual **EEG data file** in the **European Data Format (EDF)**. This format is widely used for storing and exchanging recordings of neurophysiological signals, such as EEG, ECG, EMG, and other biosignals.

▼ The `sub-NORB00027_ses-1_task-EEG_eeg.json` file contains metadata about an EEG recording session.


```
{
  "TaskName": "EEG",
  "TaskDescription": "Resting EEG",
  "SamplingFrequency": 200,
  "EEGChannelCount": 19,
  "EOGChannelCount": 0,
  "ECGChannelCount": 0,
  "EMGChannelCount": 0,
  "MiscChannelCount":  0,
  "TriggerChannelCount":  0,
  "EEGPlacementScheme": "10-10",
  "EEGReference": "common",
  "PowerLineFrequency": 60,
  "SoftwareFilters": "n/a",
  "RecordingDuration": 596.24,
  "RecordingType": "continuous"
}
```


- **“TaskName”: “EEG”**: This indicates the name of the task during which the EEG data was recorded. In this case, the task is simply named “EEG”.
- **“TaskDescription”: “Resting EEG”**: Provides a description of the task. “Resting EEG” suggests that the EEG data was recorded while the subject was at rest, not performing any specific cognitive task.

- **“SamplingFrequency”: 200**: The rate at which EEG data is sampled, measured in Hertz (Hz). Here, it’s **200 Hz**, meaning the EEG signal is recorded 200 times per second.
- **“EEGChannelCount”: 19**: The number of EEG channels used in the recording. This dataset used 19 channels.
- **“EOGChannelCount”: 0**: The number of EOG (Electrooculography) channels. EOG records eye movements and is often used to track blinks and saccades. Zero indicates that no EOG channels were used.
- **“ECGChannelCount”: 0**: The number of ECG (Electrocardiography) channels. ECG records the electrical activity of the heart. Zero indicates that no ECG channels were used.
- **“EMGChannelCount”: 0**: The number of EMG (Electromyography) channels. EMG records the electrical activity of muscles. Zero indicates that no EMG channels were used.
- **“MiscChannelCount”: 0**: The number of miscellaneous channels that might be used for other types of data. Zero indicates none were used.
- **“TriggerChannelCount”: 0**: The number of trigger channels used to mark events or stimuli presentation. Zero indicates no such channels were used.
- **“EEGPlacementScheme”: “10-10”**: Indicates the electrode placement scheme used. The **“10-10” system** is an internationally recognized method to describe the location of scalp electrodes and is an extended version of the 10-20 system, providing more detailed electrode placement.
- **“EEGReference”: “common”**: Describes the reference electrode used in the EEG recording. *“**common” suggests a common average reference, where the signals from all electrodes are averaged and this average is used as the reference for each channel.***
- **“PowerLineFrequency”: 60**: The frequency of the power line, measured in Hz. This is important for identifying and filtering out electrical noise from the power supply, which can contaminate the EEG signal.

- **“SoftwareFilters”: “n/a”**: Indicates what software filters, if any, were applied to the EEG data. “n/a” suggests that no software filters were applied or the information is not available.
- **“RecordingDuration”: 596.24**: The total duration of the EEG recording, measured in seconds. Here, the recording lasted approximately 596.24 seconds.
- **“RecordingType”: “continuous”**: Describes the type of EEG recording. “continuous” means the EEG was recorded in a single, uninterrupted session.

▼ The file `sub-NORB00027_ses-1_task-EEG_events.tsv` is a **tab-separated values (TSV)** file that logs events occurring during an EEG recording session.

```
onset  duration    trial_type  value  sample
  0.005         0 discontinuity      2       1
  0.005         0   eyes_closed     65       1
 41.305         0 discontinuity      2    8261
 48.455         0 discontinuity      2    9691
135.175         0 discontinuity      2   27035
197.245         0 discontinuity      2   39449
296.025         0 discontinuity      2   59205
299.395         0 discontinuity      2   59879
375.725         0 discontinuity      2   75145
```

- **onset**: The time at which an event starts, relative to the beginning of the EEG recording, measured in seconds. For example, the first event occurred at **0.005 seconds** after the recording started.
- **duration**: The length of time the event lasts, measured in seconds. In this file, all events have a duration of **0 seconds**, indicating they are instantaneous events.
- **trial_type**: A description of the type of event. There are two types listed:
  - **discontinuity**: This likely indicates a break or interruption in the EEG data recording or an artifact that caused a discontinuity in the signal.

    - **eyes_closed**: This signifies that the subject closed their eyes, which is a common condition in resting-state EEG studies to standardize the state of visual input.

- **value**: A numerical code assigned to each trial_type for easy identification. “2” is associated with discontinuities, and “65” with the eyes_closed event.
- **sample**: The index of the EEG data sample at which the event occurred. Since the sampling frequency is **200 Hz**, the sample number can be used to calculate the exact time of the event. For instance, the event at sample “8261” occurred at **41.305 seconds** into the recording.

▼ The file `sub-NORB00027_ses-1_scans.tsv` is a **tab-separated values (TSV)** provides information about the EEG data file and its acquisition timing.

```
                                filename  age_acq_time
eeg/sub-NORB00027_ses-1_task-EEG_eeg.edf        0.2541
```

- **filename**: This column lists the **path and name of the EEG data file**. The file `eeg/sub-NORB00027_ses-1_task-EEG_eeg.edf` is an **EDF (European Data Format)** file, which is a standard file format for storing neurophysiological data.
- **age_acq_time**: Represents the **age of the subject at the time of acquisition**, measured in years. However, the value “0.2541” seems unusually low for a typical age and might represent a different metric or be recorded incorrectly. It could potentially be the age in years calculated from the date of birth to the acquisition date, which would make sense if the subject is an infant (**approximately 3 months old**), or it could be a different time-related metric specific to the study.

▼ 2. Load channels data

- First, we need to load the EEG data from the provided files. The `sub-NORB00027_ses-1_task-EEG_eeg.edf` file contains the raw EEG data.
- We’ll use a library like `mne` (MNE-Python) to read the data. If you haven’t already installed it, you can do so using `pip install mne`.

▼ **About `mne` python library**

The Python MNE library is an **open-source package** designed for exploring, visualizing, and analyzing human neurophysiological data. It's specifically tailored for working with data from modalities like **MEG (Magnetoencephalography)**, **EEG (Electroencephalography)**, **sEEG (stereotactic EEG)**, **ECoG (Electrocorticography)**, and more [1][2].

MNE is particularly useful for researchers and professionals in neuroscience, cognitive science, and related fields who work with electrophysiological data. Its comprehensive set of tools and active community make it a go-to library for neurophysiological data analysis [1][2].

[1] MNE — MNE 1.7.0 documentation - Identity Digital.
https://mne.tools/stable/index.html.
[2] mne · PyPI.
https://pypi.org/project/mne/.

- To read raw `.edf` data, we use `mne.io.read_raw_edf()` function which takes data path as input. Another parameter is `preload=True`, which means preload data into memory for data manipulation and faster indexing. If True, the data will be preloaded into memory (fast, requires large amount of memory).

  `mne.io.read_raw_edf()` function returns an instance of Raw object which is simply an entity keep data with additional internal functions (methods) and attributes (which act on its own data).

```
import mne
import matplotlib.pyplot as plt

# Load the EEG data
data_path = '/workspaces/biosignal-processing/final/sub-N
ORB00027/ses-1/eeg/sub-NORB00027_ses-1_task-EEG_eeg.edf'
raw = mne.io.read_raw_edf(data_path, preload=True)
```

  We evenly can convert Raw object to dataframe (a kind of table/matrix representation our data) by `raw.to_data_frame()` . For example,

```
d = raw.to_data_frame()
d
```

[24] ✓ 0.0s

	time	Fp1	Fp2	F3	F4	C3	C4	P3	P4	O1
0	0.000	37.794060	38.802208	35.275967	41.127494	32.365882	67.594096	31.063564	48.486317	17.823710
1	0.005	40.119658	42.150950	36.210545	44.238100	33.409961	71.876571	33.488515	53.148359	16.666285
2	0.010	39.887098	43.190215	33.874101	45.966214	31.437811	67.362611	31.871881	49.418725	16.434800
3	0.015	37.794060	41.920002	31.888124	46.542252	29.465662	65.047760	30.370721	48.719419	16.897770
4	0.020	34.538222	38.802208	30.603080	45.274968	30.509741	63.658850	31.409985	50.817338	17.129255
...	...	...	...	...	...	...	...	...	...	...
118395	591.975	36.398701	34.067780	40.182499	33.869414	40.370489	35.533410	38.684839	32.985027	17.823710
118396	591.980	33.840542	32.682093	36.444189	32.026092	38.514348	34.838955	37.530100	33.334680	19.675591
118397	591.985	30.003305	30.026194	32.004946	30.528393	35.034084	32.524104	35.451571	31.120210	19.907076
118398	591.990	27.328867	28.525034	29.318036	29.721939	32.249873	32.408361	33.834936	30.537455	19.907076
118399	591.995	26.980027	29.102404	28.967570	29.606732	30.509741	33.334301	30.717142	31.353312	21.064502

118400 rows × 20 columns

▼ **Additional**

Summarize statistic on each channel

```
d.describe()
```

✓ 0.2s

	time	Fp1	Fp2	F3	F4	C3	C4	P3	P4	O1
count	118400.000000	118400.000000	118400.000000	118400.000000	118400.000000	118400.000000	118400.000000	118400.000000	118400.000000	118400.000000
mean	295.997500	7.270163	6.427694	6.844579	6.264259	7.354548	6.816291	6.967986	7.751543	5.453543
std	170.896401	32.345300	32.354484	31.915020	32.390927	29.892493	30.749590	29.739084	30.370865	30.914219
min	0.000000	-230.580000	-229.790000	-232.130000	-229.380000	-229.930000	-229.980000	-229.330000	-230.770000	-231.370000
25%	147.998750	-5.578351	-6.925442	-5.845440	-7.029292	-4.176888	-4.976483	-4.040491	-3.495452	-5.787770
50%	295.997500	7.212440	6.354052	6.771356	6.104377	7.423991	6.829257	6.929526	7.693449	5.323515
75%	443.996250	20.003232	19.518073	19.621795	19.583669	18.792853	18.403512	18.015017	18.998901	16.550542
max	591.995000	245.470000	242.960000	246.140000	242.280000	245.010000	243.870000	243.420000	246.390000	242.480000

▼ 3. Visualizing each channel

Plot the original data in time-series with the full of title, units

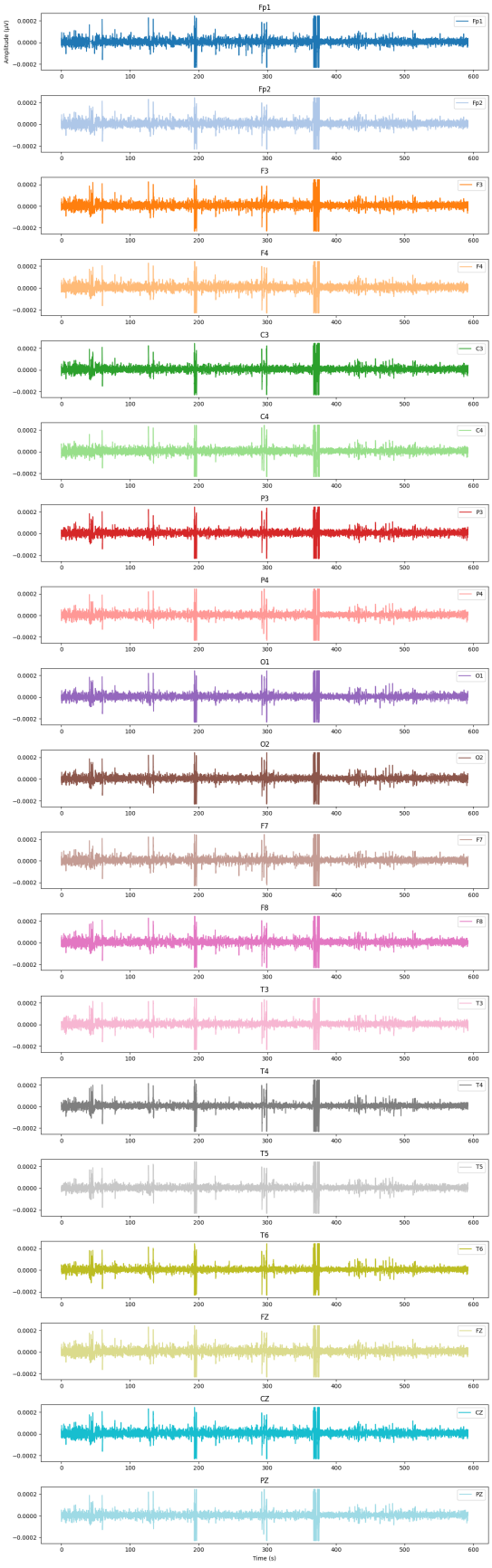
▼ **Plot manually with matplotlib**

▼ **Script code**

```
# ========== Visualizing each channel purely with nump
y ==============
# data and channel names
data, times = raw[:]
ch_names = raw.ch_names
```

```
n_channels = len(ch_names) # num channel

# Subplots
fig, axes = plt.subplots(n_channels, 1, figsize=(12, 2
*n_channels))

# Plot each channel in a separate subplot
colors = plt.cm.get_cmap('tab20', n_channels) # each c
hannel has its own color

for i in range(n_channels):
    axes[i].plot(times, data[i], label=ch_names[i], co
lor=colors(i))
    axes[i].set_title(ch_names[i])
    axes[i].legend(loc='upper right')

# Set labels for the x-axis and y-axis
axes[-1].set_xlabel('Time (s)')
axes[0].set_ylabel('Amplitude (μV)')

# Display the plot
plt.tight_layout()
plt.show()
```

▼ **Result Figure(s)**

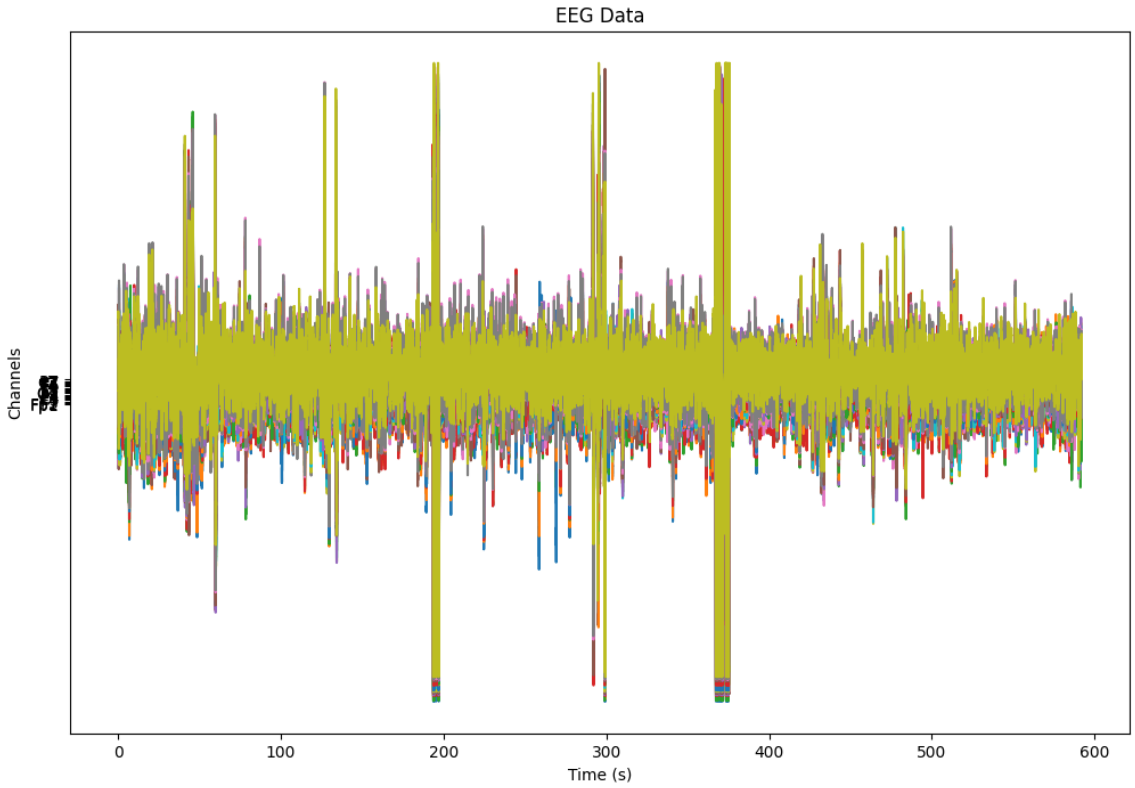

Fp1
Fp2
F3
F4
C3
C4
P3
P4
O1
O2
F7
F8
T3
T4
T5
T6
FZ
CZ
PZ
Amplitude (μV)
0.0002
0.0000
−0.0002
0
100
200
300
400
500
600
Time (s)

▼ **Comment:**

The general pattern may appear consistent across channels, indicating synchronized neural activity

Although the visualizations of each channel look similar, there are tiny differences between them. The following figure represents all signals in one plot.

▼ **Plot mannually (with matplotlib)**



Another way to plotting original data is levering the power of `mne` library which return a interactive interface to play with data. (to

▼ **Plot with `mne`**

▼ **Script code**

```
import matplotlib
matplotlib.use('Qt5Agg')
```

```
import mne
import matplotlib.pyplot as plt

# Load the EEG data
raw = mne.io.read_raw_edf(data_path, preload=True)

# ==== Visualize with MNE =========
# Plot the original data in time-series with the full
title, units
raw.plot(title='Raw EEG data', show_options=True, bloc
k=True)
```

The provided Python code imports the `matplotlib` and `mne` libraries, sets the 'Qt5Agg' backend for `matplotlib`, and loads EEG data from a file using `mne.io.read_raw_edf`. The data is preloaded into memory and then plotted using `raw.plot`, with an interactive plot titled 'Raw EEG data'. This serves as a starting point for EEG data analysis, with potential next steps including preprocessing tasks like filtering and epoching, followed by specific analyses such as ERP or time-frequency analysis.

⚠ **Note:** You need to run locally rather than on Google Colab, Kaggle if you want to have interactive interface.

▼ **Result Figure(s)**

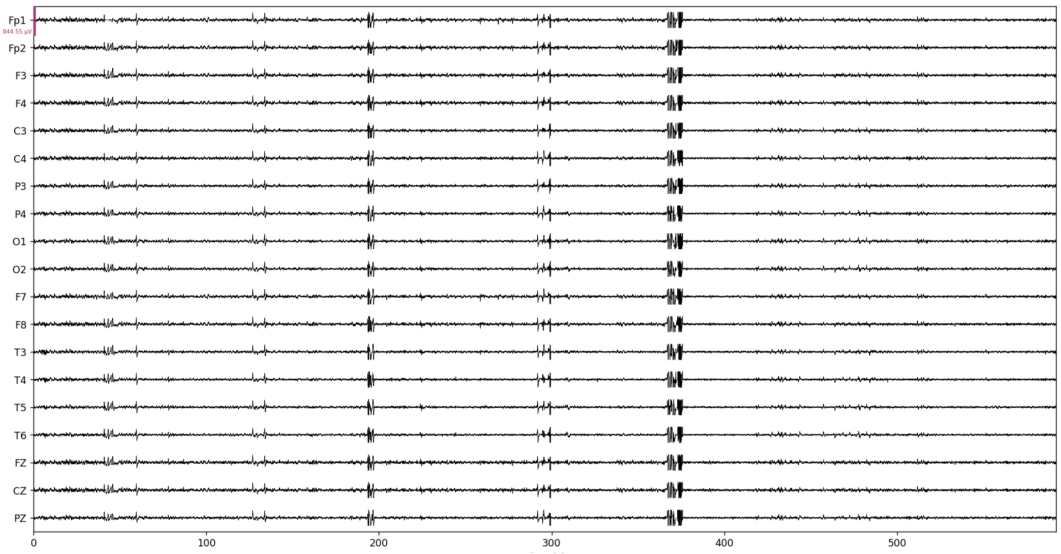


Fig. 1. Plot each channel seperately

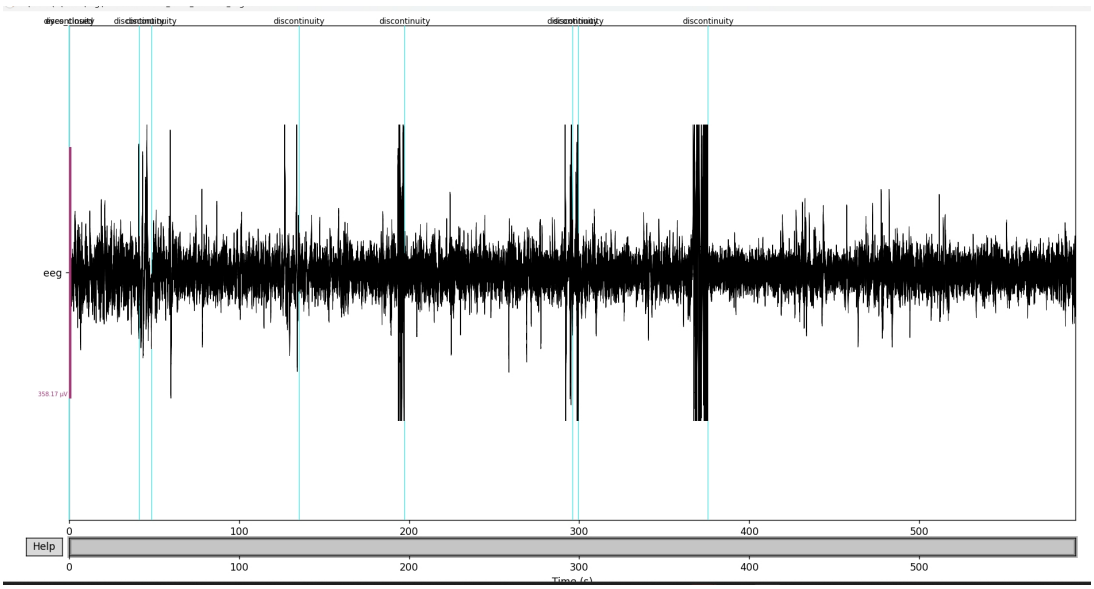


Fig. All channel in one image

Let’s plot the raw data again, but add event markers in this time

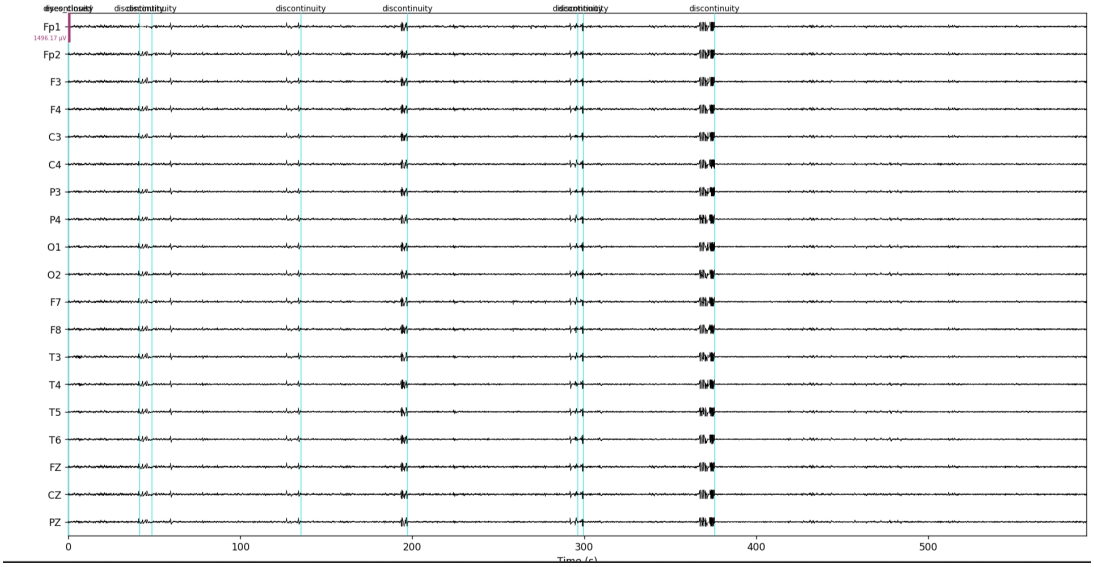


Fig. Plot each channel seperately with event markers

▼ 4. Analyzing sources of noise

Analyze the original data signal and identify noises

To detect the noise of the signal we project them in different view and see their properties.

▼ Analyzing Metadata

- **Electrode Configuration**: From information in file `sub-NORB00027_ses-1_task-EEG_channels.tsv`, we can see that the EEG data was collected using a **common reference** with a **10-10 placement scheme**, which is standard for high-resolution EEG recordings. All channels are reported as good, which suggests that the electrodes themselves are not the source of noise.
- **Sampling and Filtering**: Based on the information of the file `ub-NORB00027_ses-1_task-EEG_eeg.json`, the EEG data has a **sampling frequency of 200 Hz**, with a **low cutoff of 0.5 Hz** and a **high cutoff of 100 Hz**. This range is adequate to capture the EEG signals while filtering out low-frequency drift and high-frequency noise.
    - Low-frequency cutoff (0.5 Hz): slow drifts or artifacts (noise) typically below 0.5 Hz

- High-frequency cutoff (100 Hz): because most of the EEG activity of interest is below 100 Hz

▼ ❓ **Maybe you know:**

1. **EEG activity below 100 Hz**: The different EEG frequency bands include Delta (0.5-4 Hz), Theta (4-8 Hz), Alpha (8-13 Hz), Beta (14-30 Hz), and Gamma (30-100 Hz) [1]. These bands cover the range of frequencies that are typically of interest in EEG studies. Moreover, a study titled "Human EEG responses to 1–100 Hz flicker: resonance phenomena in visual cortex and their potential correlation to cognitive phenomena" also supports this fact [2].
2. **Slow drifts or artifacts below 0.5 Hz**: Artifacts in EEG recordings are an unavoidable aspect and can have various physiological and non-physiological origins [3]. Common cutoffs in EEG are between 0.1 or 0.5 Hz to reduce drifts such as body sway, or skin potentials. Likewise, they dramatically reduce offsets as very slow oscillations in the data. Some studies report prominent oscillations with frequency around 0.1 Hz or 0.02 Hz[5].

(1) Normal EEG Waves: Understanding Brain Waves and Their Characteristics. https://www.nhnscr.org/blog/normal-eeg-waves-understanding-brain-waves-and-their-characteristics/.

(2) Human EEG responses to 1–100 Hz flicker: resonance ... - Springer. https://link.springer.com/article/10.1007/s002210100682.

(3) Getting to know EEG artifacts and how to handle them in BrainVision .... https://pressrelease.brainproducts.com/eeg-artifacts-handling-in-analyzer/.

(4) The enigma of infra-slow fluctuations in the human EEG. https://www.frontiersin.org/articles/10.3389/fnhum.2022.928410/full.

▼ **Analyzing Time Domain**

// in analyzing progress

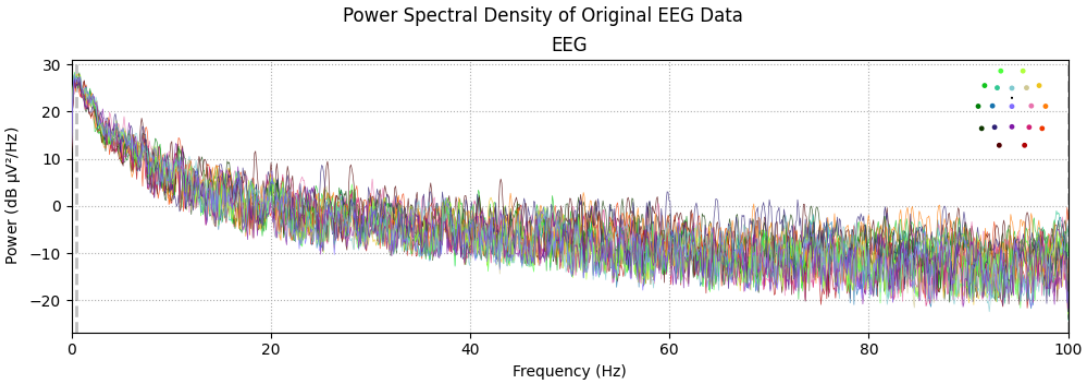
▼ **Frequency Domain Analysis**

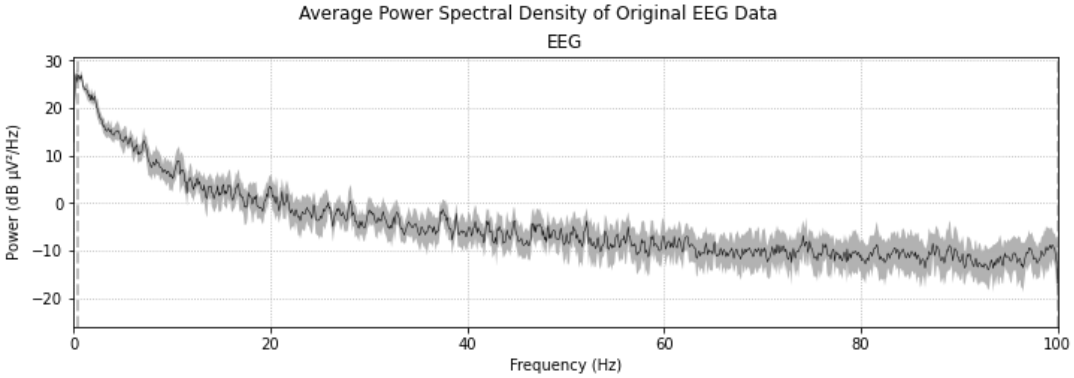
▼ **Power Spectral Density (PSD)**

First, let look at the Power Spectral Density (PSD) of all channel in the same image.

▼ **Scripts**

```
montage_dict = {row['name']: [row['x'], row['y'], r
ow['z']] for _, row in electrodes.iterrows()}
montage = mne.channels.make_dig_montage(ch_pos=mont
age_dict, coord_frame ='head')
raw.set_montage(montage)

# Analyze the data to identify signal and noises
psds = raw.compute_psd() # power spectral density
psd_plot = psds.plot()
psd_plot.suptitle("Power Spectral Density of Origin
al EEG Data")
```





The PSD graph shows the power distribution across frequencies. Since the power line frequency is **60 Hz** and there are no significant spikes at this frequency in the graph, it's unlikely that electrical interference from power lines is the source of noise.

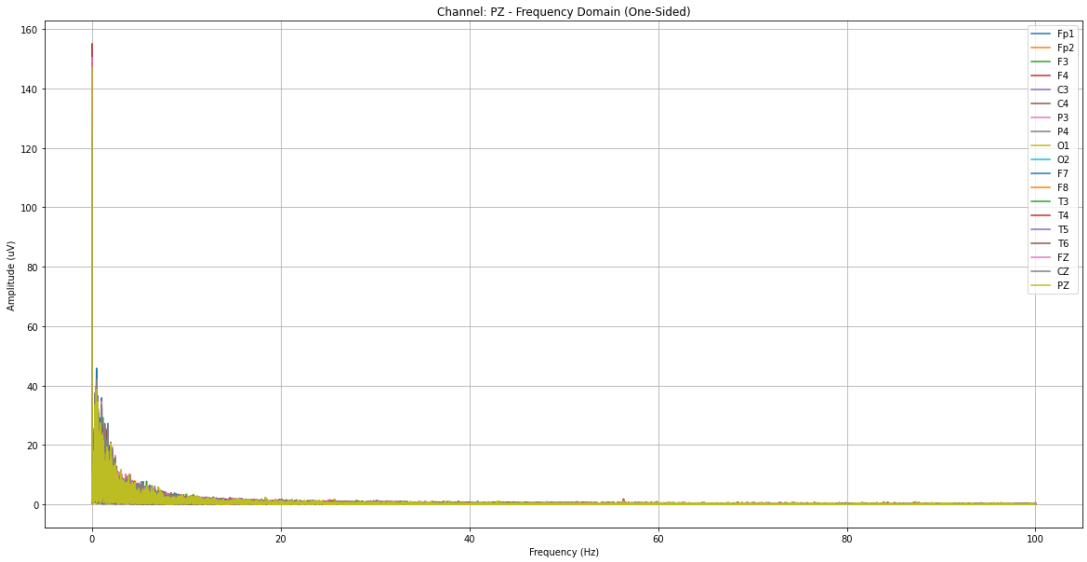
- **Potential Noise Sources**: Given the absence of power line noise, other potential sources could be physiological, such as muscle artifacts or eye movements (although no EOG channels are reported), or environmental, such as radio frequency interference or movement artifacts.

▼ **Fourier Transform**

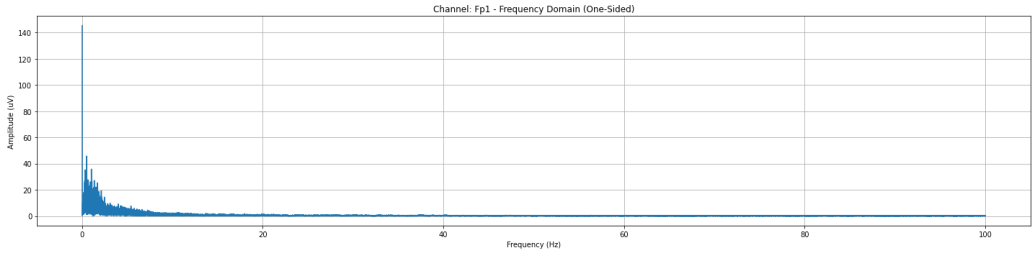
▼ **Scripts**

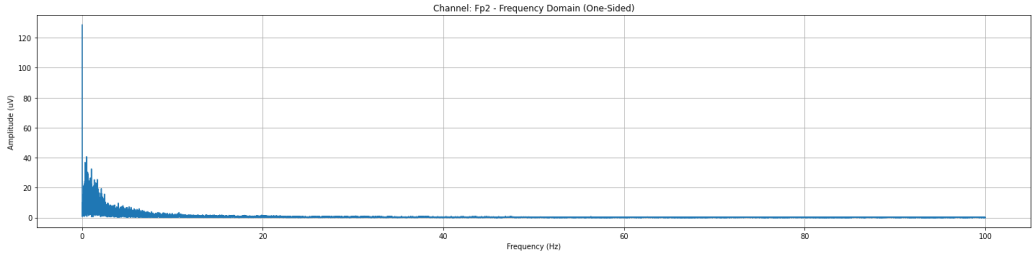
```
%matplotlib inline
import matplotlib.pyplot as plt
from scipy.fft import fft, fftfreq

def fft_plot(raw_mne):
    '''Performce Fourier Transform and plot from Ra
w obj of MNE.
    '''
    # extract
    ch_names = raw_mne.info['ch_names']
    s_freq = raw_mne.info['sfreq']
    signal_data, _ = raw_mne[:]
    signal_data = signal_data * 10e6 # V to uV
    n_samples = signal_data.shape[1]

    # plot
    fig, ax = plt.subplots(figsize=(20, 10))

    # fft for each channel
    for ch_name, signal in zip(ch_names, signal_dat
a):
        yf = fft(signal)
        xf = fftfreq(n_samples, 1/s_freq)
```

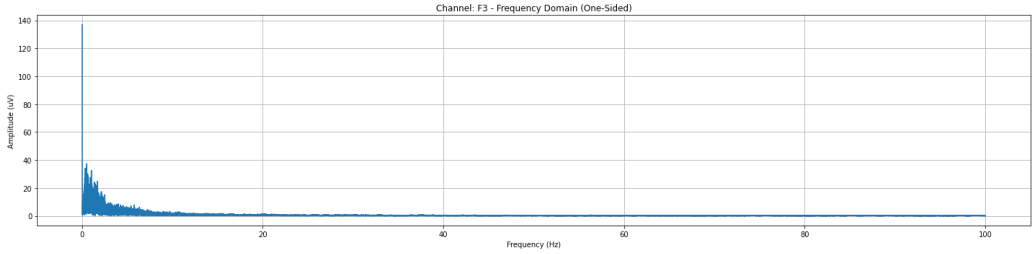
```
        half_len = n_samples//2
        ax.plot(xf[:half_len], 2.0/n_samples * np.a
bs(yf[0:half_len]), label=ch_name)  # symmetric of
FFT

    ax.set_xlabel('Frequency (Hz)')
    ax.set_ylabel('Amplitude (uV)')
    ax.set_title(f'Channel: {ch_name} - Frequency D
omain (One-Sided)')
    plt.legend()
    plt.grid()
    plt.show()
    # break

fft_plot(raw)
```

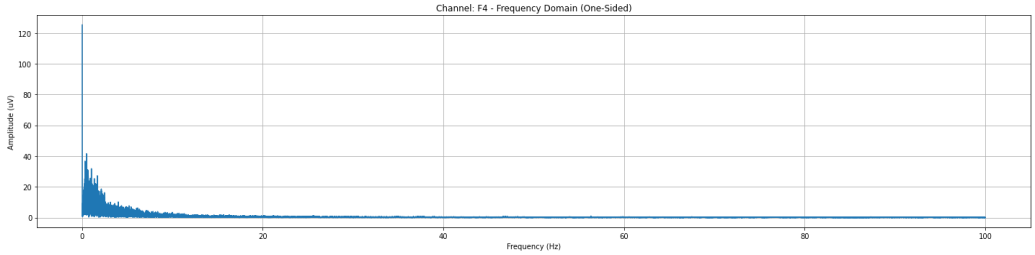


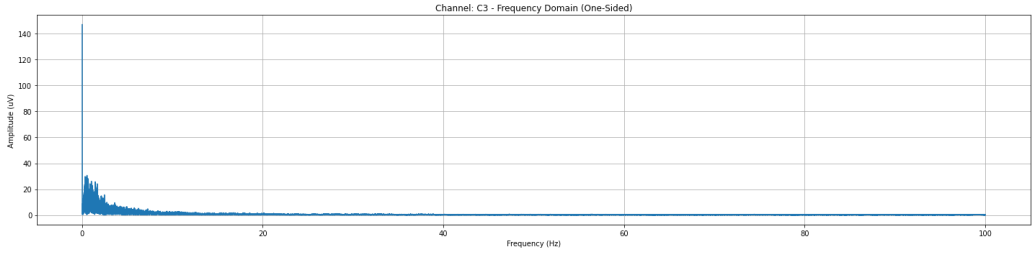
▼ **Seperated each channel (Additional / Appendix)**

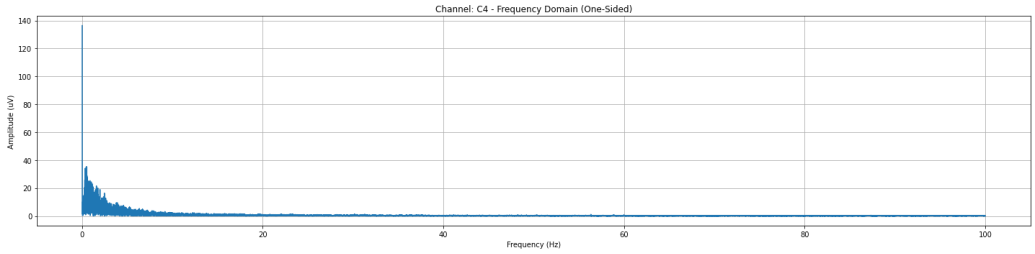

Channel: Fp1 - Frequency Domain (One-Sided)
Amplitude (uV)
Frequency (Hz)

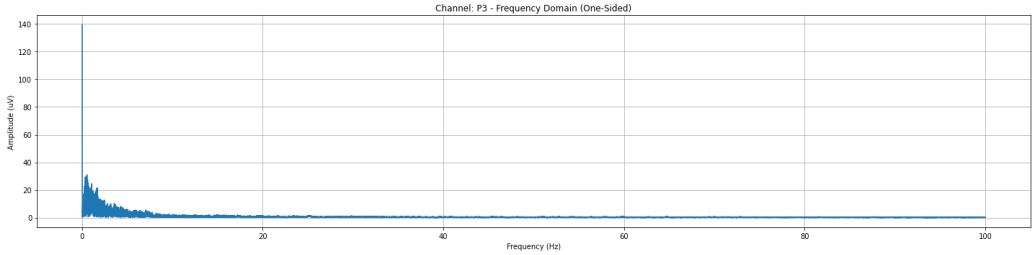

Channel: Fp2 - Frequency Domain (One-Sided)
Amplitude (uV)
Frequency (Hz)

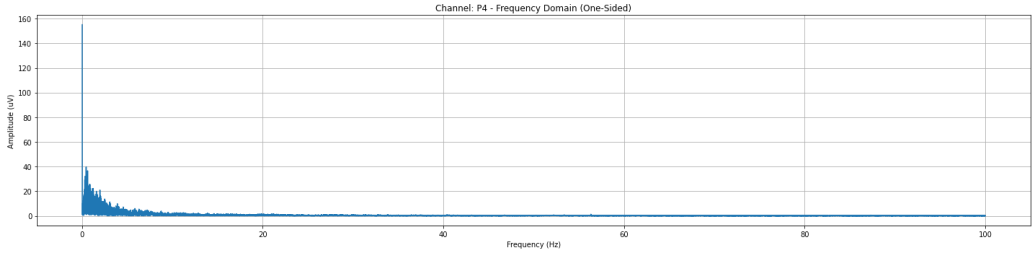

Channel: F3 - Frequency Domain (One-Sided)
Amplitude (uV)
Frequency (Hz)

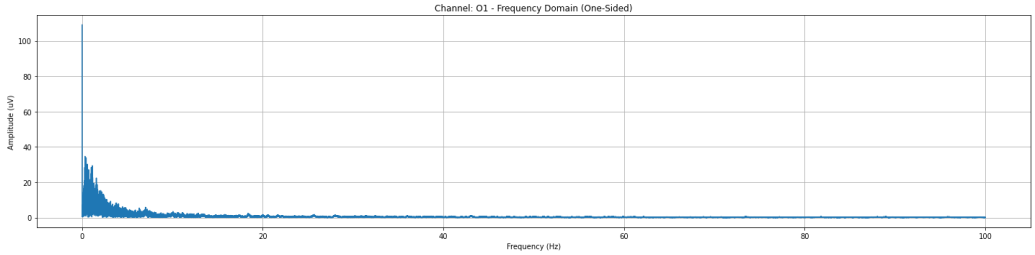

Channel: F4 - Frequency Domain (One-Sided)
Amplitude (uV)
Frequency (Hz)

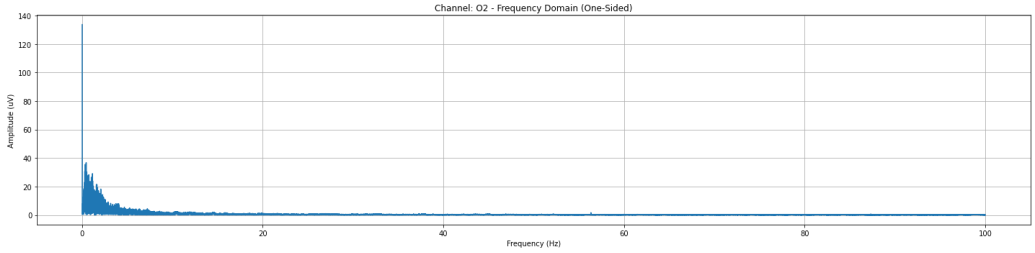

Channel: C3 - Frequency Domain (One-Sided)
Amplitude (uV)
Frequency (Hz)

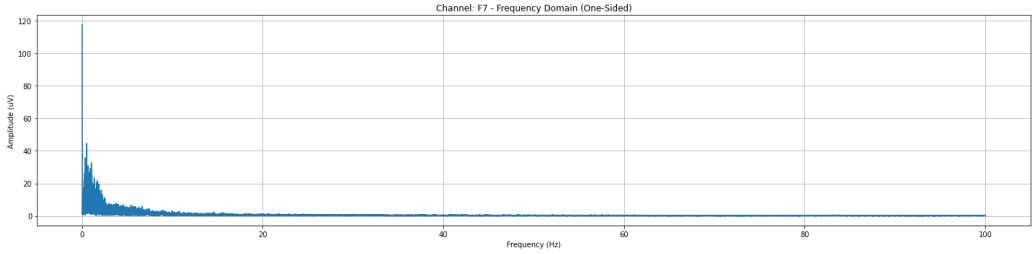

Channel: C4 - Frequency Domain (One-Sided)
Amplitude (uV)
Frequency (Hz)

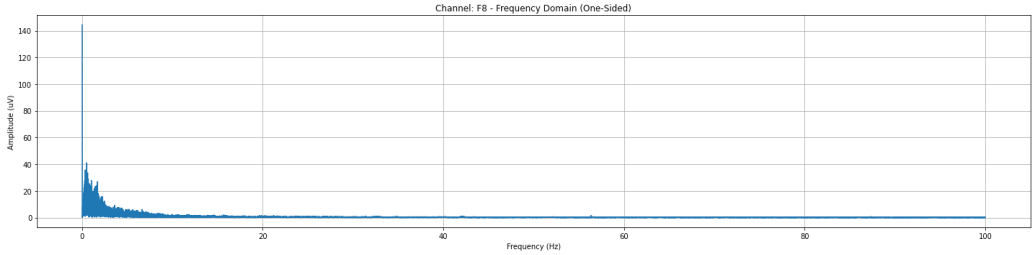

Channel: P3 - Frequency Domain (One-Sided)
Amplitude (uV)
Frequency (Hz)

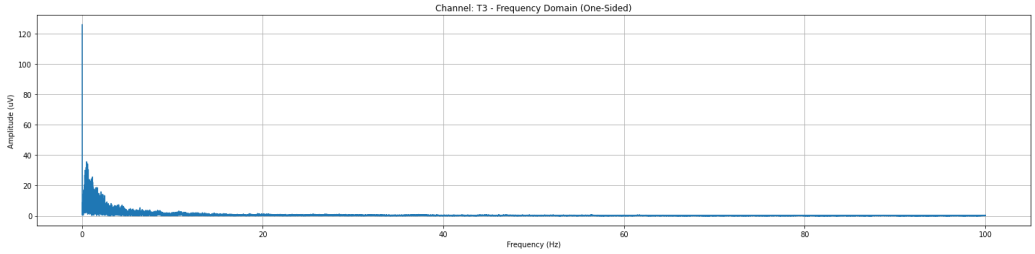

Channel: P4 - Frequency Domain (One-Sided)
Amplitude (uV)
Frequency (Hz)

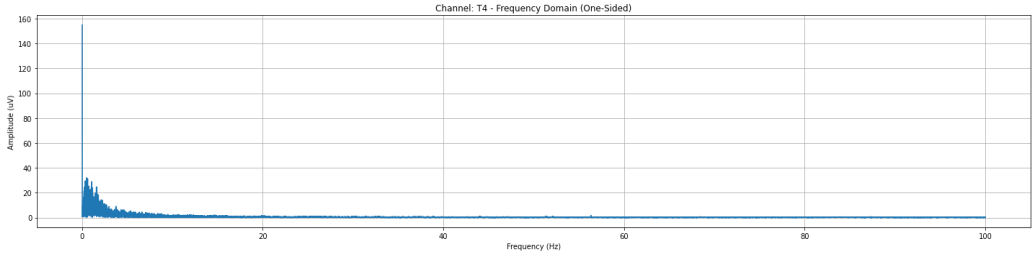

Channel: O1 - Frequency Domain (One-Sided)
Amplitude (uV)
Frequency (Hz)

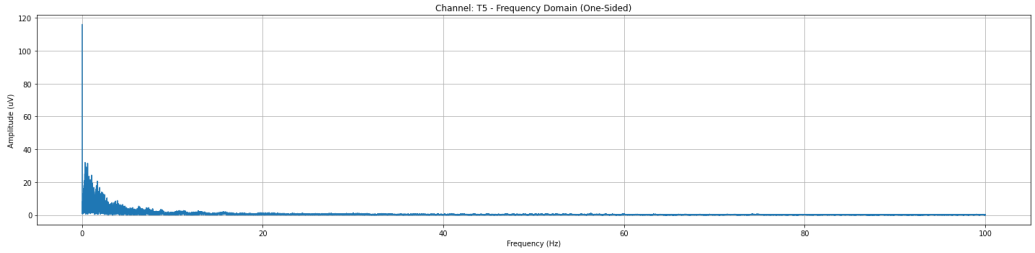

Channel: O2 - Frequency Domain (One-Sided)
Amplitude (uV)
Frequency (Hz)

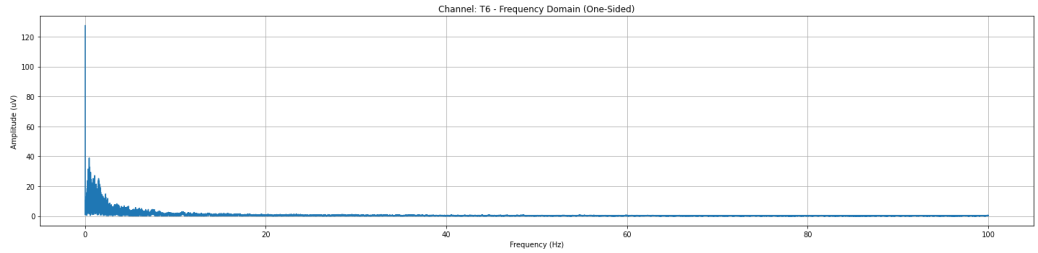

Channel: F7 - Frequency Domain (One-Sided)
Amplitude (uV)
Frequency (Hz)

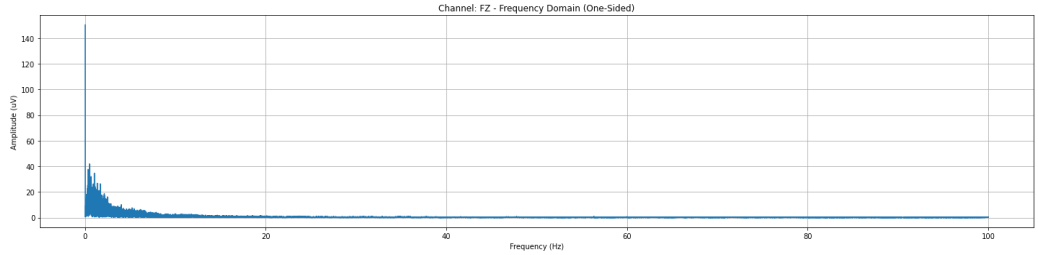

Channel: F8 - Frequency Domain (One-Sided)
Amplitude (uV)
Frequency (Hz)

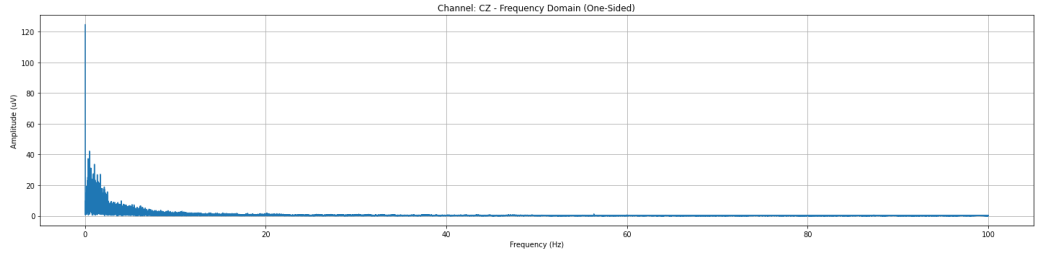

Channel: T3 - Frequency Domain (One-Sided)
Amplitude (uV)
Frequency (Hz)

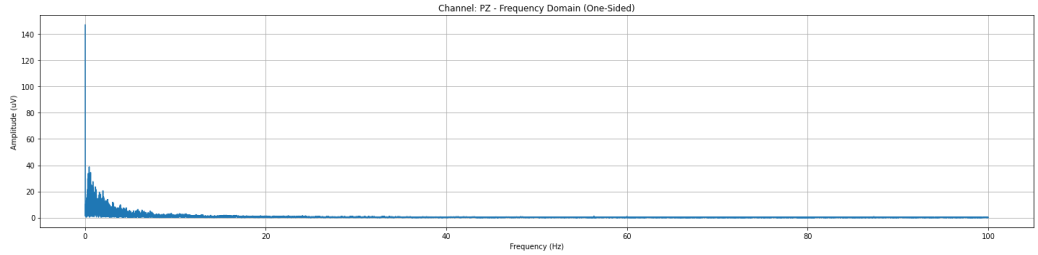

Channel: T4 - Frequency Domain (One-Sided)
Amplitude (uV)
Frequency (Hz)


Channel: T5 - Frequency Domain (One-Sided)
Amplitude (uV)
Frequency (Hz)









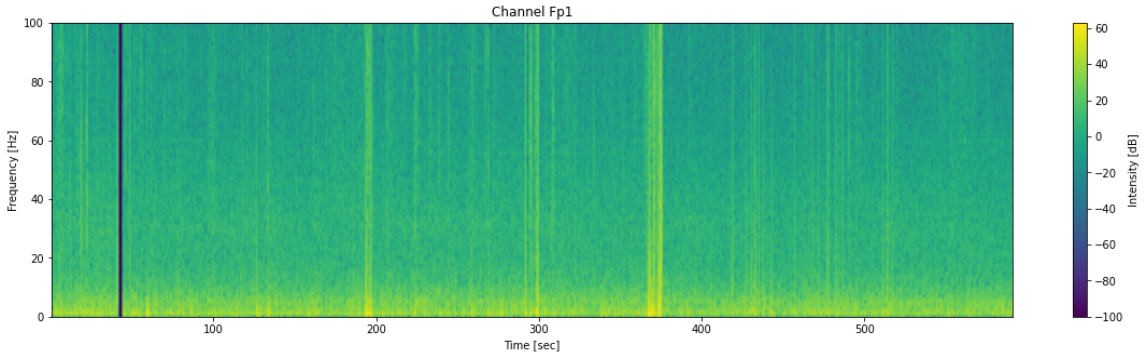
// in analyzing progress

▼ Time-Frequency Domain Analysis

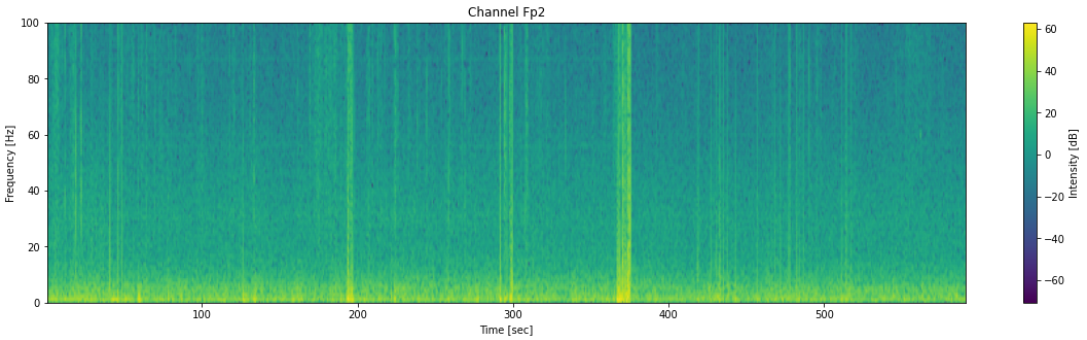
▼ Scripts

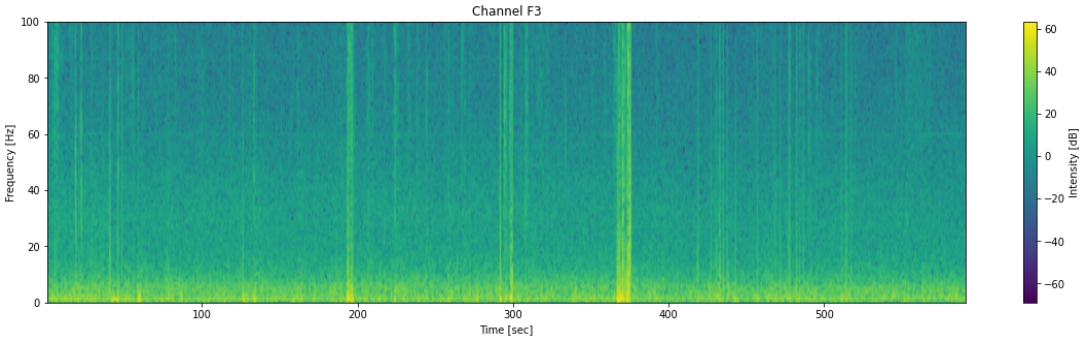
```
%matplotlib inline
import matplotlib.pyplot as plt
from scipy.signal import spectrogram
import numpy as np

s_freq = raw.info['sfreq']  # sampling frequency in Hz
```

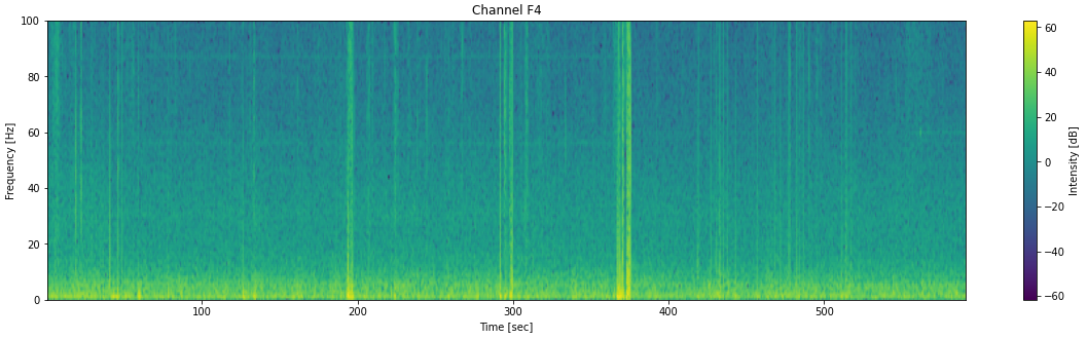
```
signals, _ = raw[:] # return data, time
signals = signals * 10e6 # V to uV

ch_names = raw.ch_names
for ch_name, signal in zip(ch_names, signals):
    frequencies, times, Sxx = spectrogram(signal, s_fr
eq, nperseg=int(s_freq))
    Sxx = np.maximum(Sxx, 1e-10) # avoid log(1)

    plt.figure(figsize=(20, 5))
    plt.pcolormesh(times, frequencies, 10 * np.log10(S
xx), shading='gouraud')
    plt.ylabel('Frequency [Hz]')
    plt.xlabel('Time [sec]')
    plt.title(f'Channel {ch_name}')
    plt.colorbar(label='Intensity [dB]')
    plt.show()
```

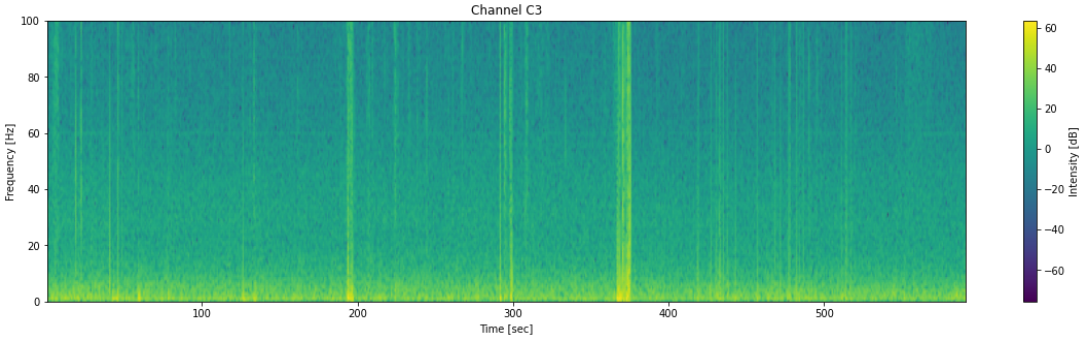


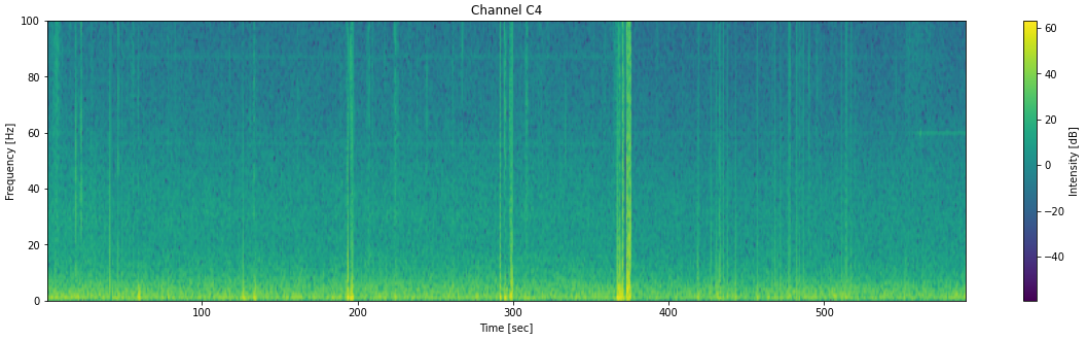
▼ Others channel

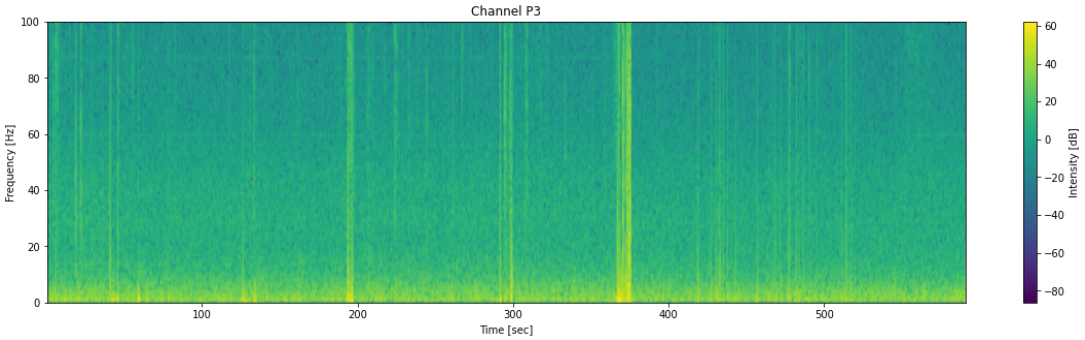


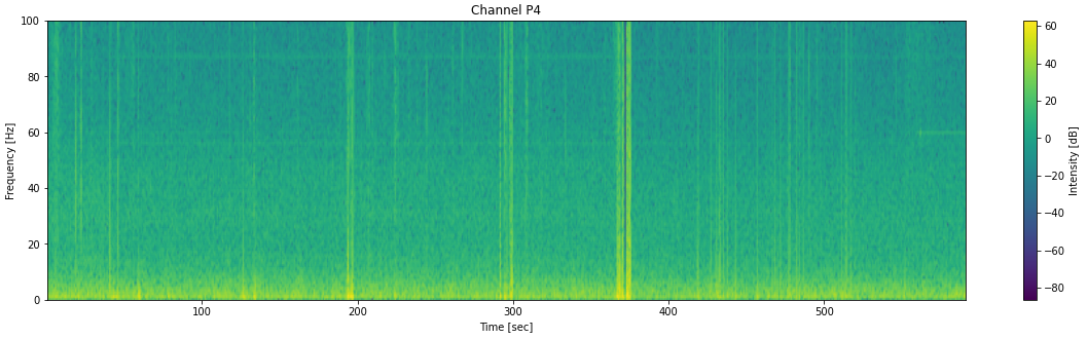

Channel F3
Frequency [Hz]
Time [sec]
Intensity [dB]

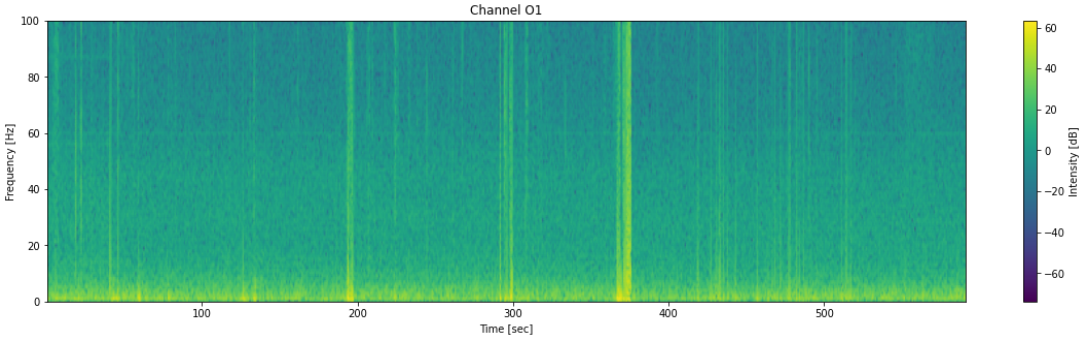

Channel F4
Frequency [Hz]
Time [sec]
Intensity [dB]

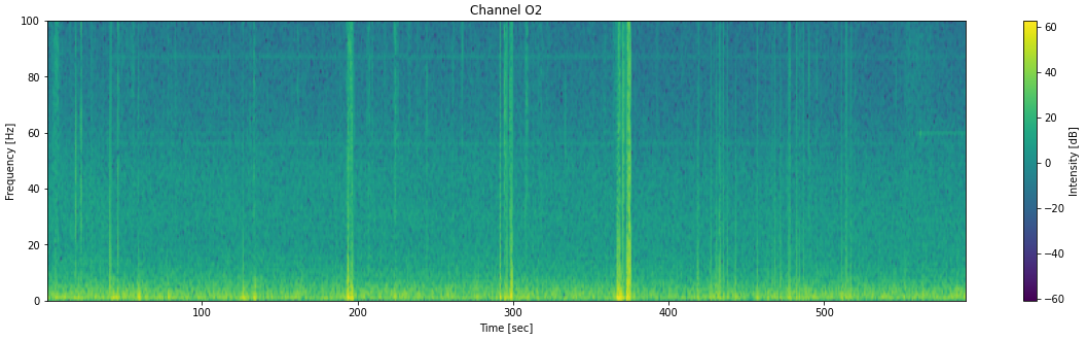

Channel C3
Frequency [Hz]
Time [sec]
Intensity [dB]

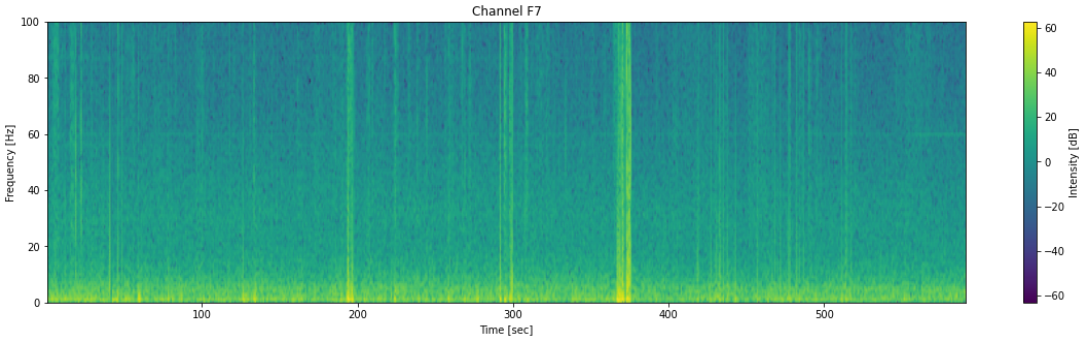

Channel C4
Frequency [Hz]
Time [sec]
Intensity [dB]

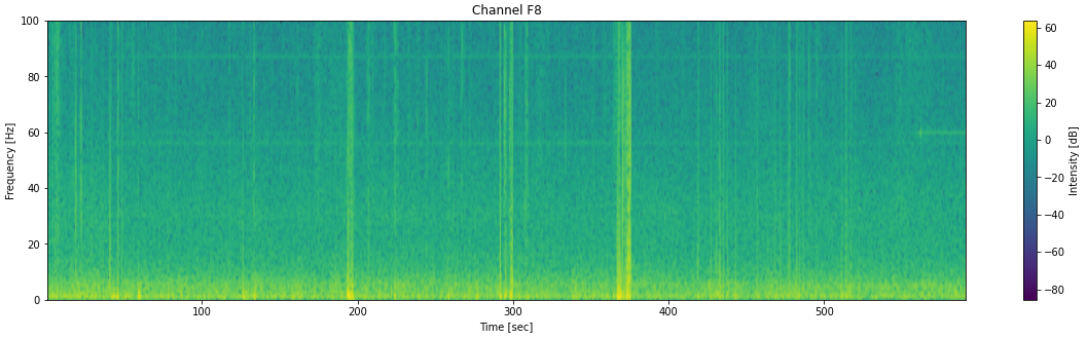

Channel P3
Frequency [Hz]
Time [sec]
Intensity [dB]

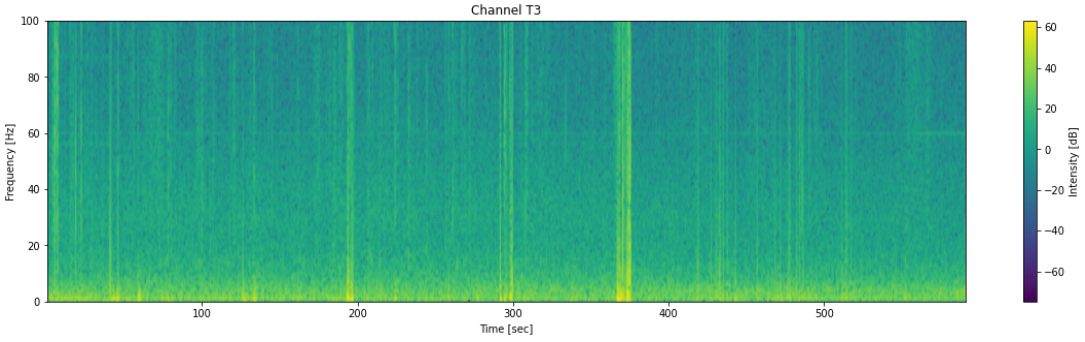

Channel P4
Frequency [Hz]
Time [sec]
Intensity [dB]

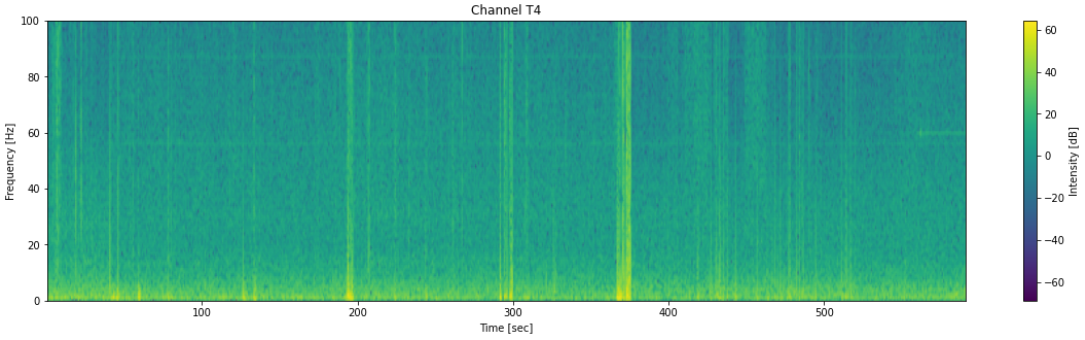

Channel O1
Frequency [Hz]
Time [sec]
Intensity [dB]

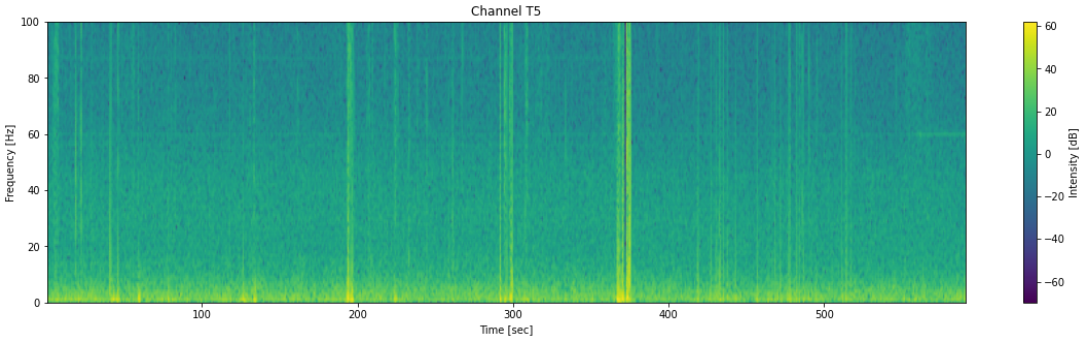

Channel O2
Frequency [Hz]
Time [sec]
Intensity [dB]

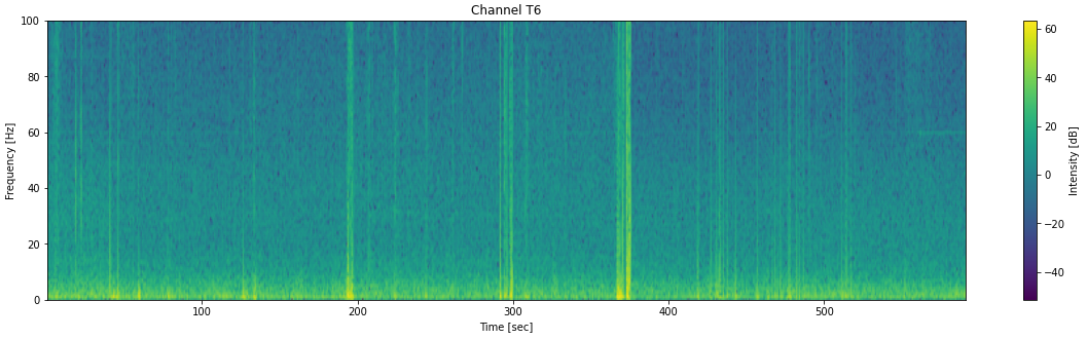

Channel F7
Frequency [Hz]
Time [sec]
Intensity [dB]

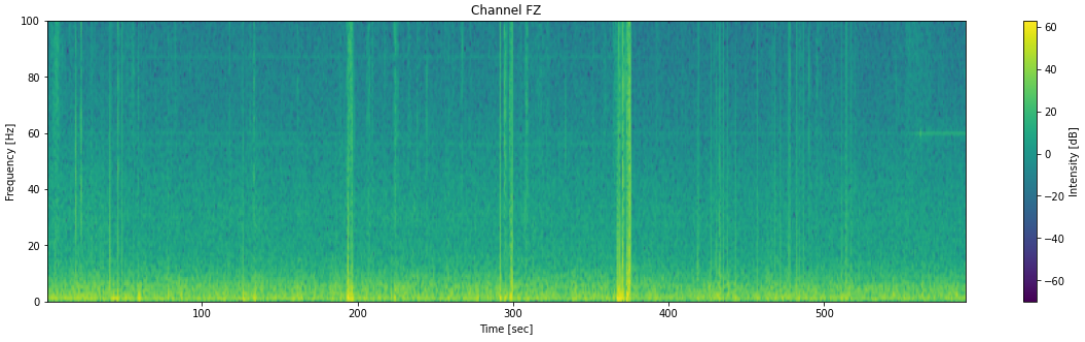

Channel F8
Frequency [Hz]
Time [sec]
Intensity [dB]

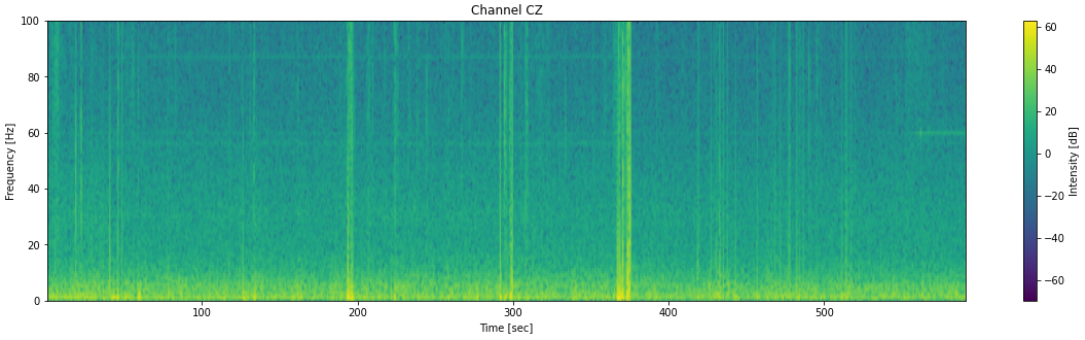

Channel T3
Frequency [Hz]
Time [sec]
Intensity [dB]

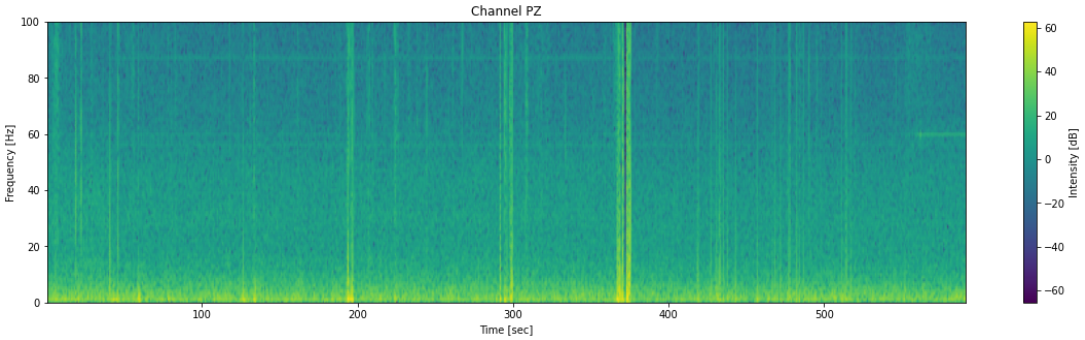

Channel T4
Frequency [Hz]
Time [sec]
Intensity [dB]


Channel T5
Frequency [Hz]
Time [sec]
Intensity [dB]


Channel T6
Frequency [Hz]
Time [sec]
Intensity [dB]


Channel FZ
Frequency [Hz]
Time [sec]
Intensity [dB]


Channel CZ
Frequency [Hz]
Time [sec]
Intensity [dB]



// in analyzing progress

Next work

- [x] ~~Spectrogram~~
- [ ] Analyze and define noise
- [ ] Define filter with each corresponding noise
- [ ] ML method: ICA, PCA