EEG Signal Preparation, Storage & Analysis – Brain Computer Interface

Faculty of Information Technology, Brno University of Technology

Juraj Dedič, xdedic07

December 2023

1 Introduction

The aim of this project is the collection, storage and anylsis of data from EEG (electroencephalogram). The device provided for this project was Mindrove HeadBand[1].

The MindRove EEG headband offers precise brainwave monitoring with 4+2 channels: FP1, FP2, O1 and O2. It's a sleek wearable with semi-dry electrodes, featuring motion tracking with a 6-axis gyroscope and accelerometer. With WiFi connectivity, a rechargeable battery, and compatibility with Windows, MacOS, and Linux, it's an ideal choice for EEG research and development.

The acquired dataset contains resting state EEG from work of Torkamani-Azar et al. 2020[2]. The dataset consists of both eyes-closed and eyes-open measurements.

2 Data Storage

The storage solution consists of Python scripts:

- `db.py` – handles connection to DB server, stores ORM models,
- `upload.py` – consists of logic necessary for uploading `mat` file format of EEG data into database

The `upload.py` script first parses user input for a path to the dataset and, the name of the parameter inside the `mat` file format. It also needs the data frequency in Hertz. Usage of `upload.py` is further described in `readme.md`.

The last parameter is list of channels contained in the dataset. This must be a string of channels separated by commas, such as: `"FP1,FP2,O1,O2"`.

After parsing the CLI input, it then proceeds to load the file. It then extracts the parameters containing the EEG data. This should be an array with dimensions $(w \cdot n)$, where

[1] `https://mindrove.com/bright/`

[2] `https://ieeexplore.ieee.org/document/9034192`

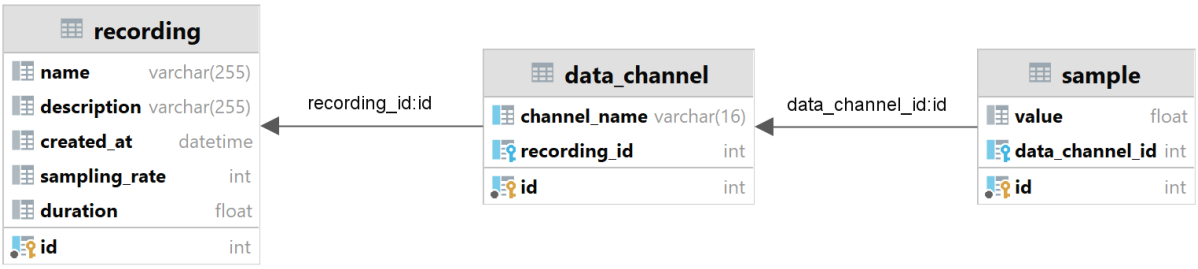

Figure 1: DB Schema of the Cloud Storage

w is the number of electrodes or additional datapoints (such as trigger/event channels) and n represents the number samples.

If the number of provided channels in the CLI argument is equal to w, the program proceeds to the next steps. If the provided count is higher than w, the script will exit with an error message. In a case where the provided channel count is lower than w. The array will get truncated to the provided size (ignoring the last channels). In such a case the user is warned.

The program then connects to the cloud storage. This storage is implemented via MySQL database hosted on Microsoft's Azure. Credential privacy is guaranteed by use of .env for their storage. To represent the EEG data, following objects were defined:

- Recording – stores general information about one recording, which is the same for all the channels.
- Data Channel – represents one channel (e.g. FP1) and holds it's name.
- Sample – contains one data value from single measurement.

The upload script first creates the Recording object with name, description and other parameters. It then takes the first channel and creates the DB object for it.

The problematic part is storing the samples, as there can be a large amount of them in each channel. To address this problem, other database solutions implement array data types. Unfortunately, MySQL does not include this type. Instead, bulk insert is used in this project to speed up the upload process.

After uploading all the channel samples, the script proceeds to the next channel and repeats the process until all the channels are in the database.

3 Loading & Analysis

Stored data can be then downloaded using *load_recoding* function in the database script. This function extracts the samples for each channel and reconstructs it to the original array. The download function then returns the list of all channels and the reconstructed array.

The analysis is performed in notebooks `project_EC.ipynb, project_EO.ipynb`, the first Jupyter notebook is used for analyzing the eyes-closed dataset and the other for the eyes-open set.

After downloading the EEG data from the database, the script creates *mne* info object and sets the appropriate montage for the EEG headset used.

3.1 Eyes-closed resting state

After plotting the initial signal (fig. 2) we can see multiple problems. First I identified the slow drift in some of the channels.

For referencing the data I used the average reference method. To resolve the drift I used a high-pass filter with cutoff frequency 1 Hz. The effect of this filter can be seen on figure 3 as the drift is removed and the signal is mostly straight. Nonetheless we can still easily identify some of the major artifacts. Most notable are the distortions at approximately 1 second in measurements from electrodes above the frontal lobe. This could potentially be caused by eye blinks.

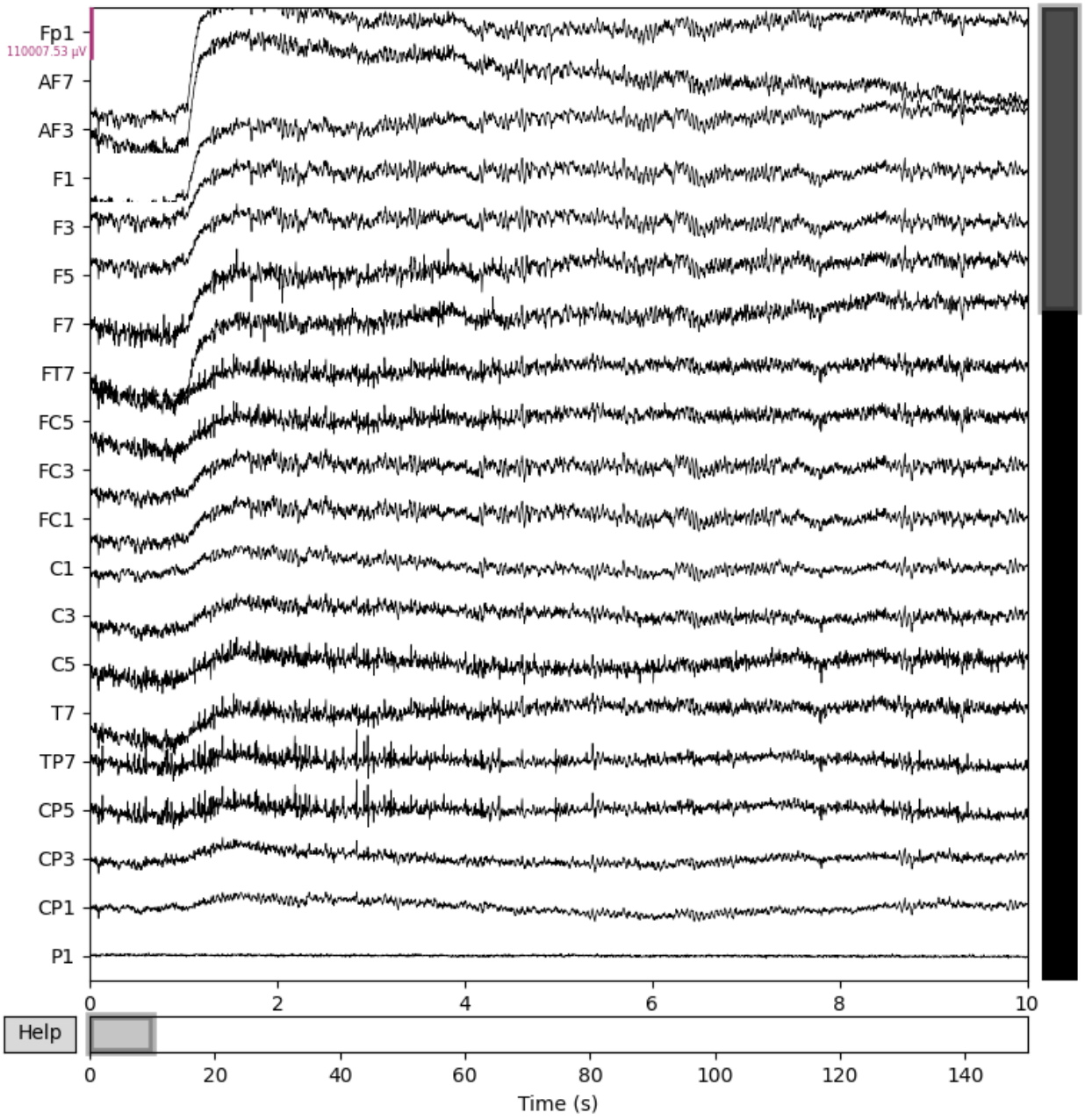

Figure 2: Initial EEG signal.

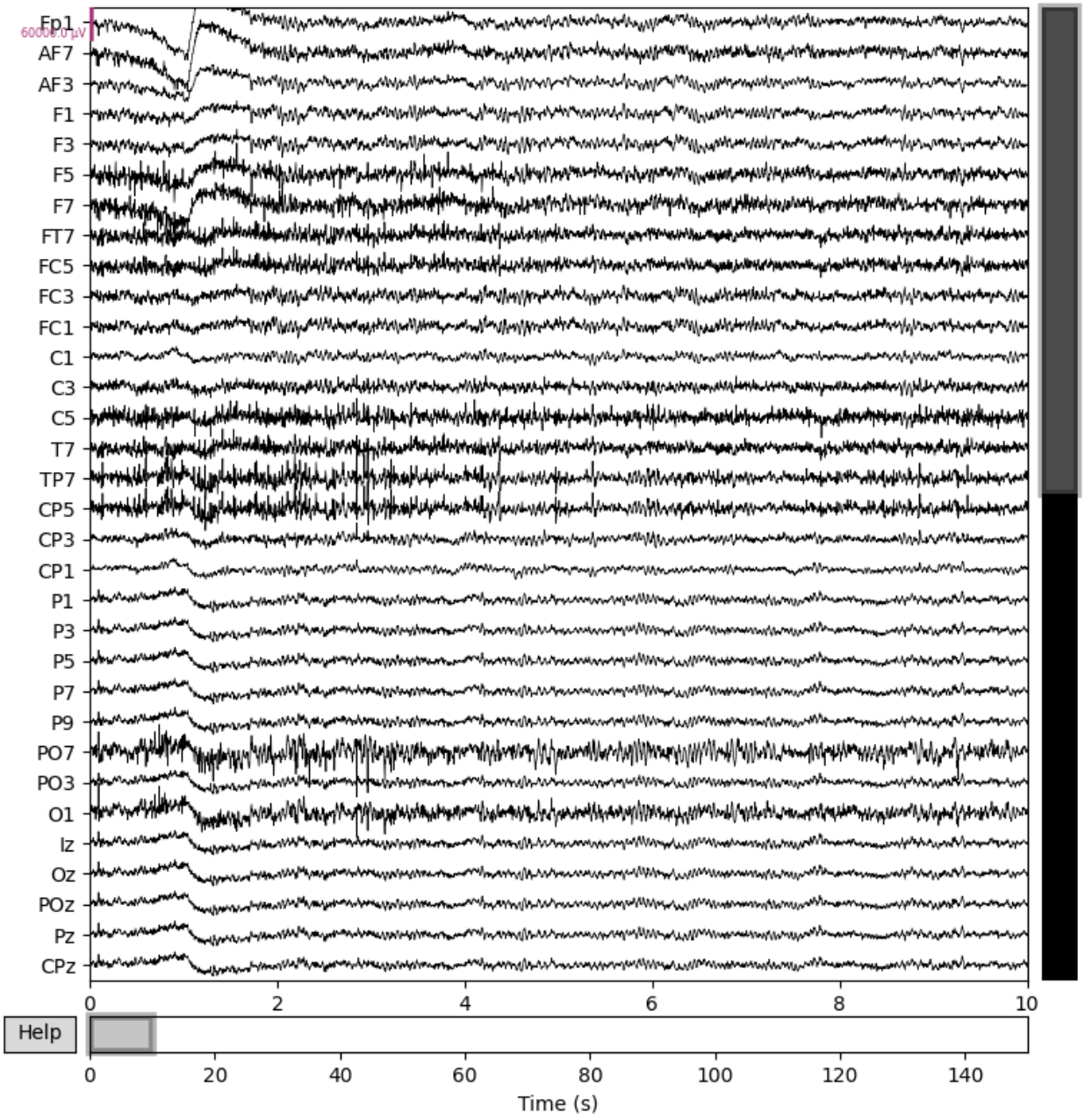

Figure 3: Straightened EEG signal.

To better filter out the airtifacts I used ICA (independent component analysis). I first used the piccard algorithm to identify the independent components visible on figure 4.

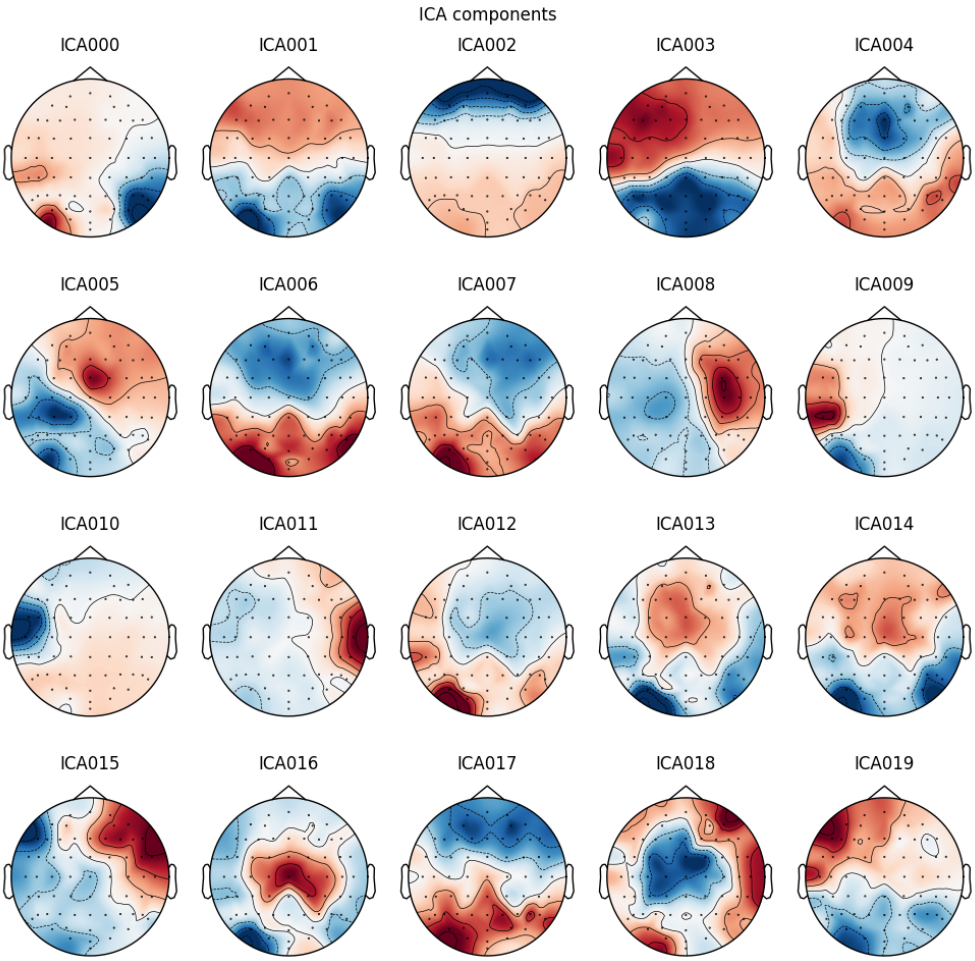

Figure 4: Independent components

We can identify the *ICA002* as being responsible for the eye-blink artifact. For better clarity of the components I plotted the time series of the individual components (fig. 5). The signal contains multiple other artifacts which sources were identified with the help of ICA. Therefore I removed the following ICA components: 0, 1, 2, 9, 10, 17.

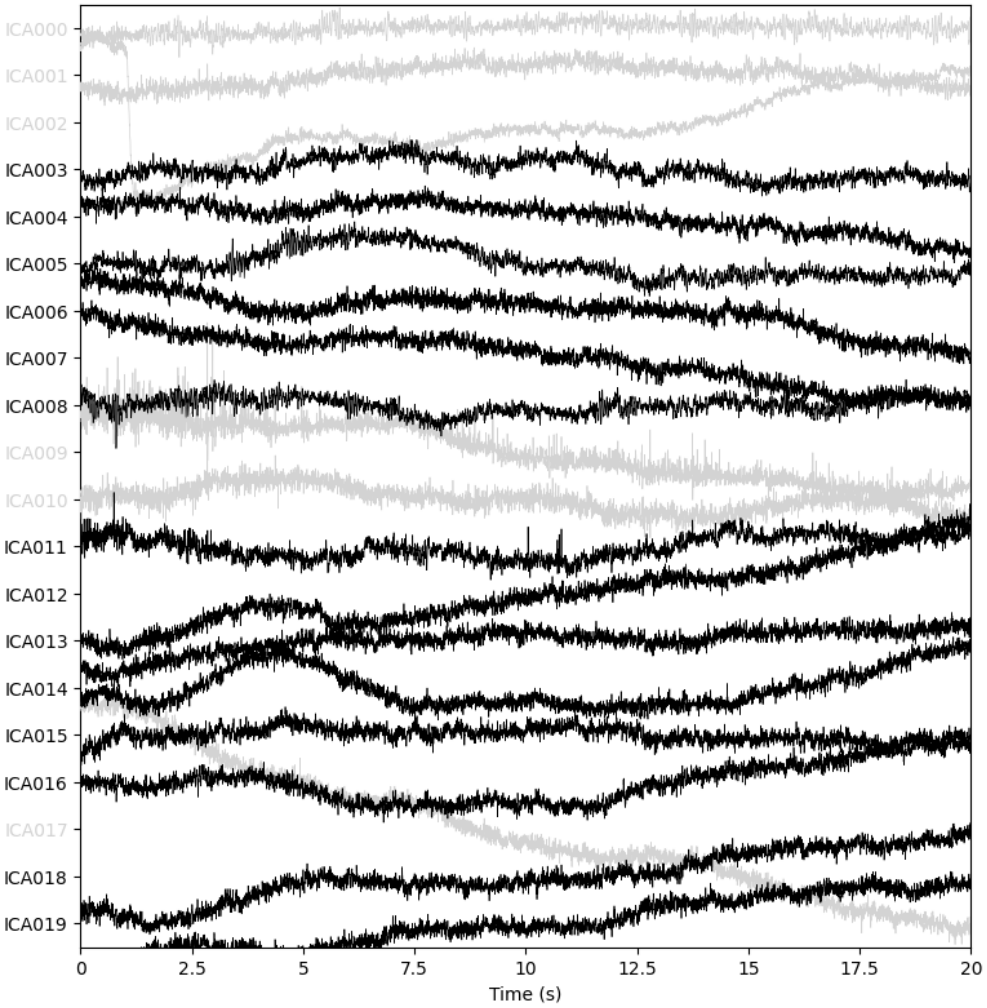

Figure 5: Time series of the independent components of the EEG signal. The greyed out components were removed from the signal.

After clearing out the EEG data using ICA, the plot can be seen on figure 6.

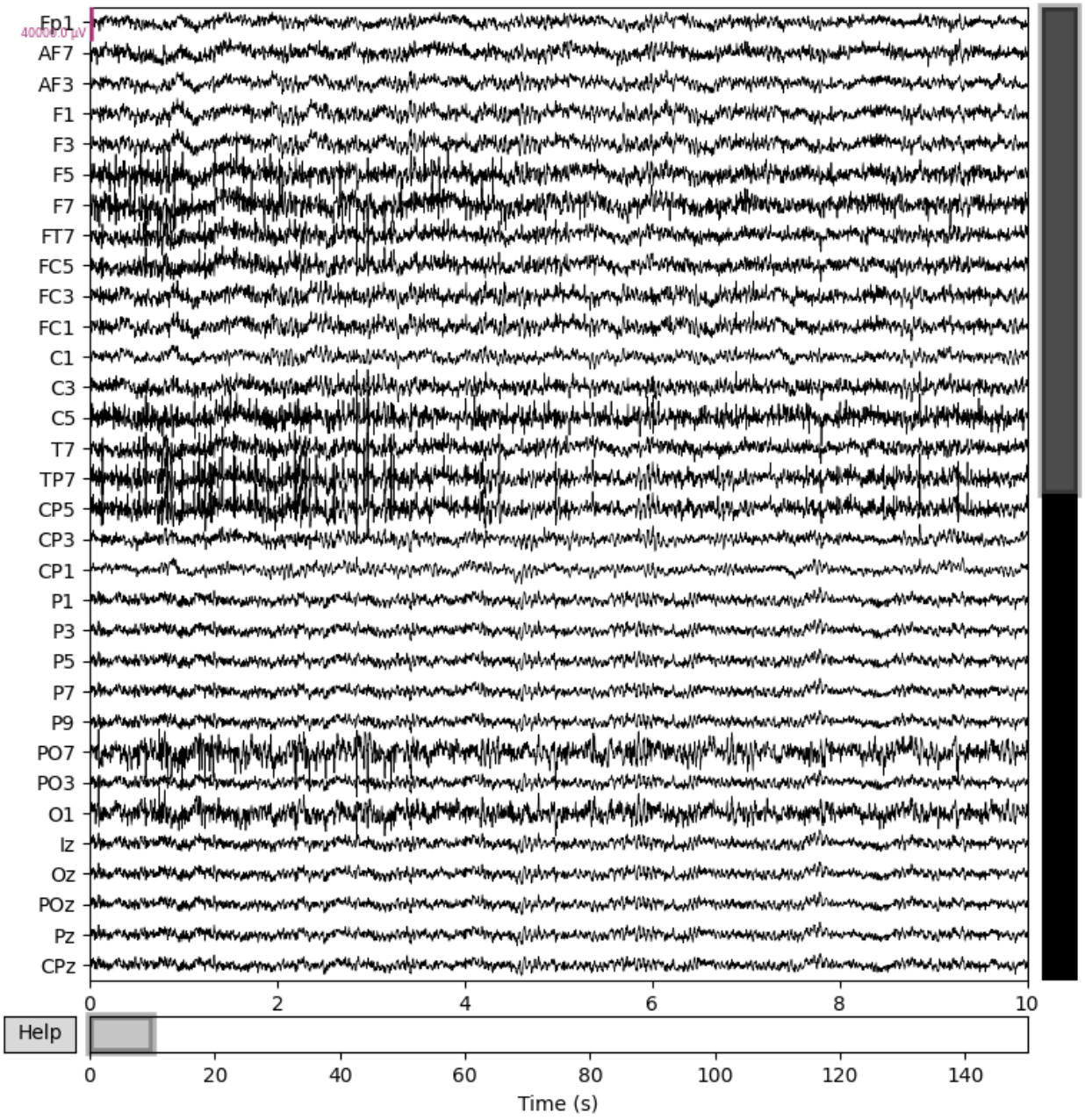

Figure 6: Corrected EEG signal.

To calculate the spectral connectivity in channel space, I split the data into epochs of duration 0.5 s with 0.25 s overlap between epochs. Next I used *mne_connectivity* library to calculate the spectral connectivity on frequencies between 4.0 and 9.0 Hz.

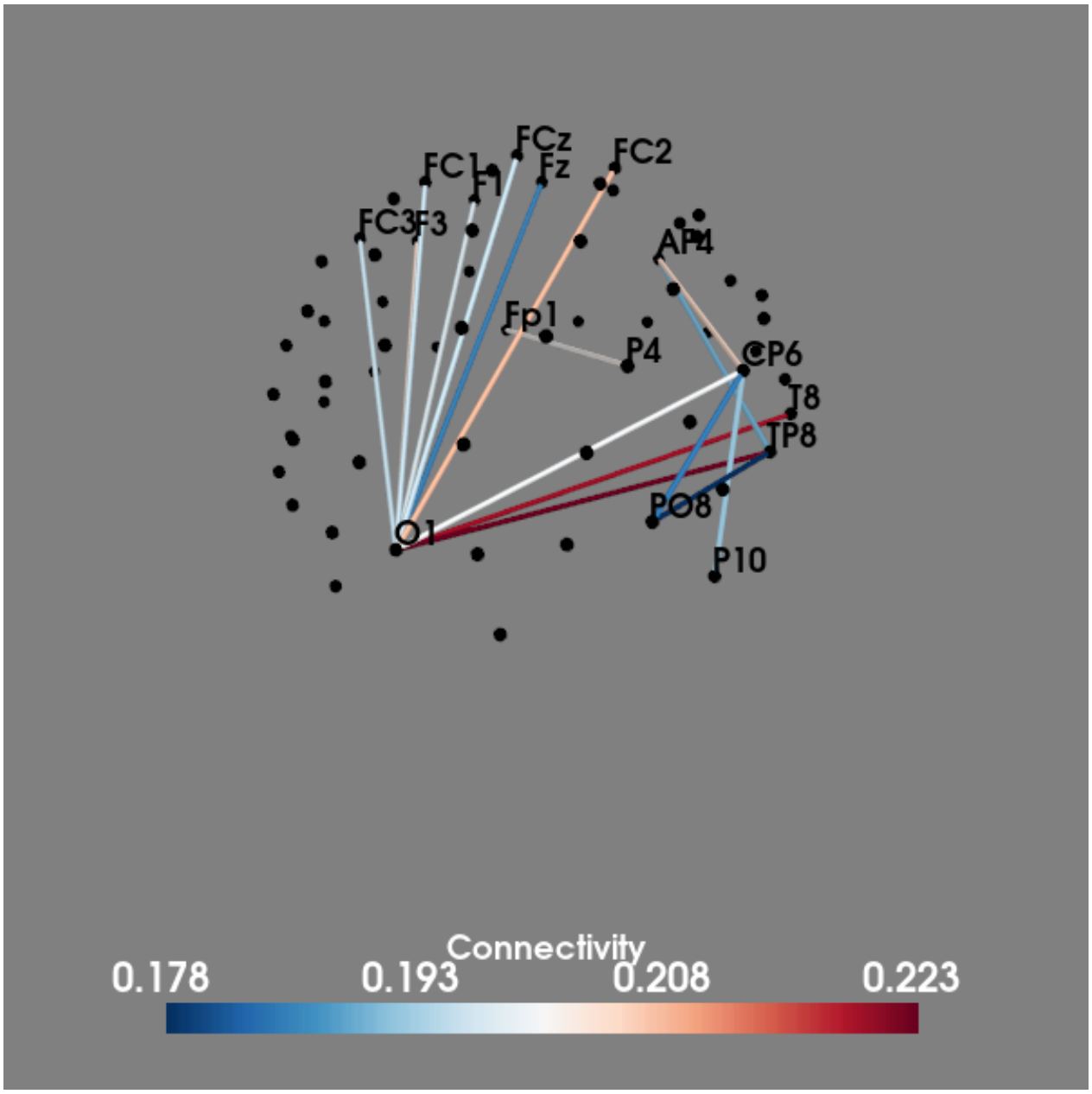

Figure 7: Spectral connectivity 3D visualisation.

We see a significant degree of connectivity between the occipital region and the left hemisphere of the frontal lobe and the parietal lobe of the right hemisphere.

3.2 Eyes-open resting state

The eyes-open EEG dataset can be seen on figure. Again as in the previous dataset in the eyes-open set, we see both slow downward signal drifts and some major artifacts. To mitigate these artifacts I used the high-pass filter and ICA, removing components based on this dataset.

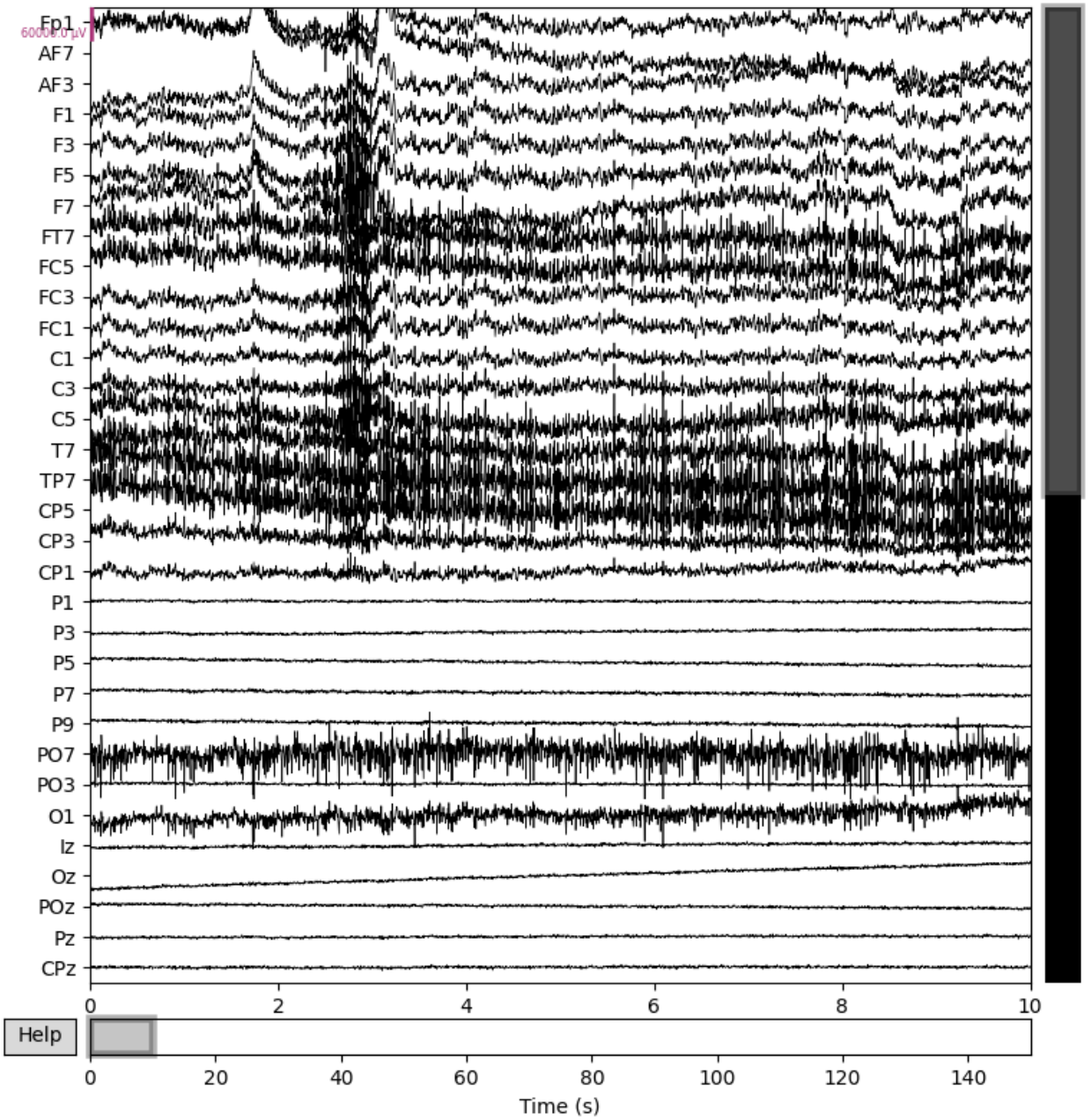

Figure 8: Raw eyes-open EEG signal.

Using the described techniques I corrected the signal to this state (fig. 9).
I again split the signal to epochs as described in the previous dataset. To identify the

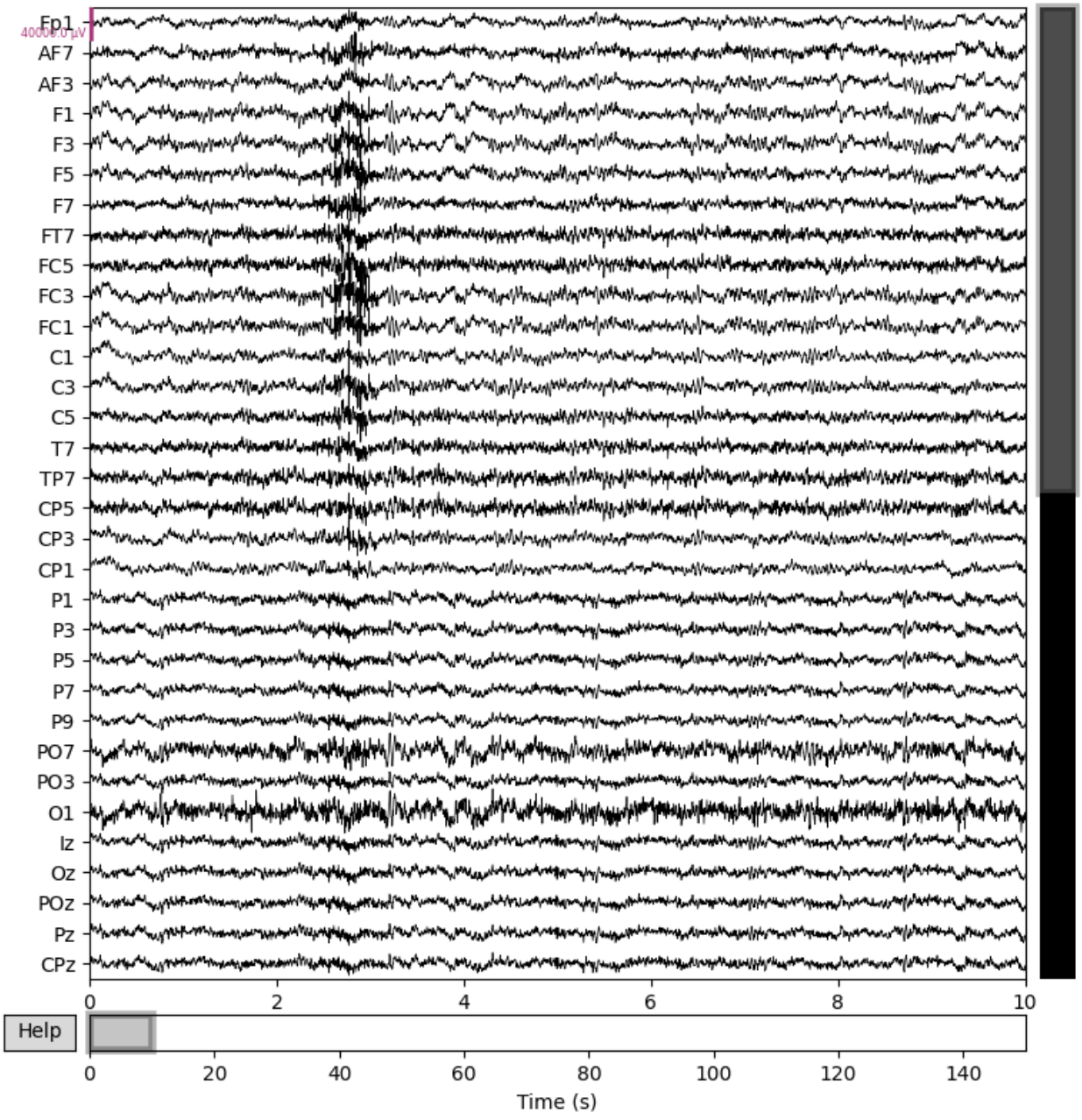

Figure 9: Corrected eyes-open EEG signal.

connectivity between the brain regions we can use this plot.

We can see significant degree of connectivity mostly between the left frontal and frontotemporal electrodes.

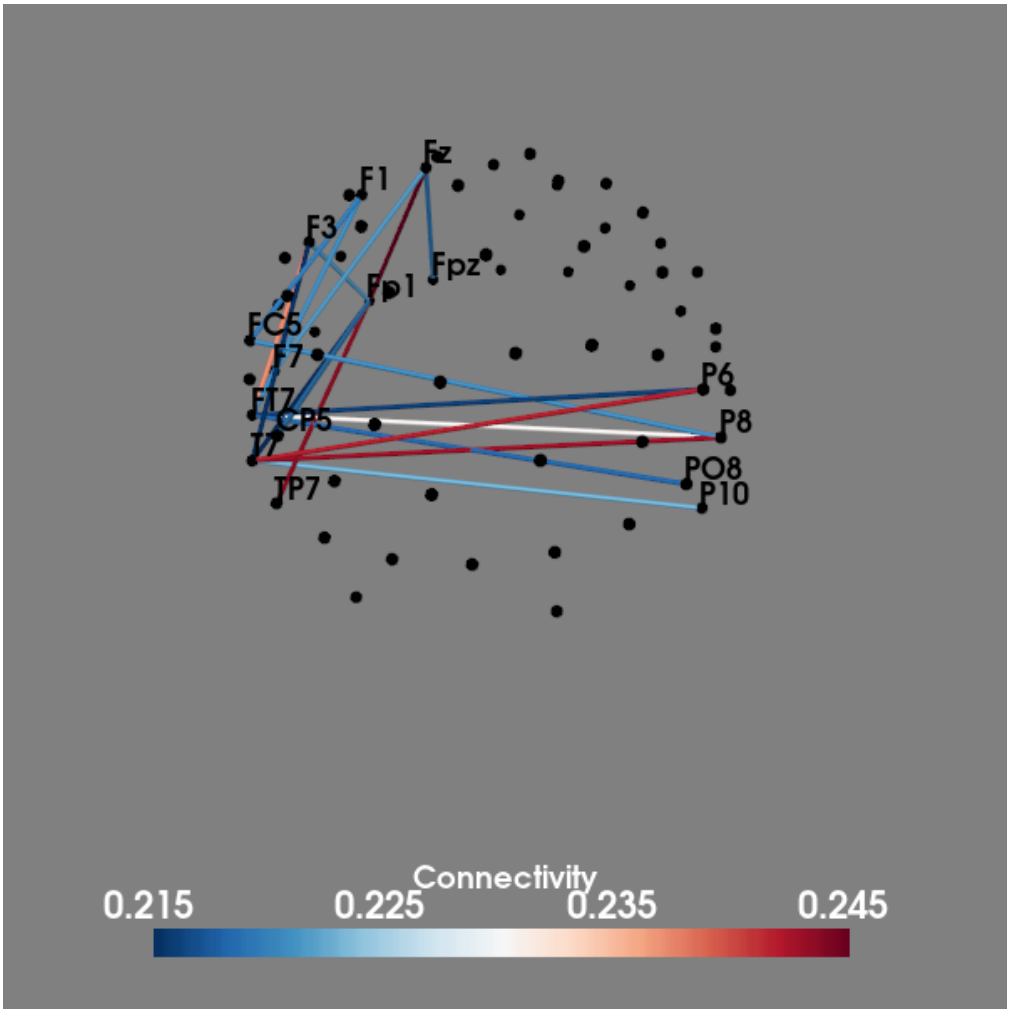

Figure 10: Spectral connectivity in sensor space of eyes-open EEG data.