An update on the project: ***Development of an Automatic Instrument for Schizophrenia(SZ) Diagnosis*** , for the MCIP Innovation Prize 2022.

April 3, 2023

1 Summary

The previous report presented the data acquisition and processing methods along some obtained results. Also highlighted were technical issues noticed during data acquisition and issues with ergonomics of developed software for the clinicians. The previous report also presented the next steps to be taken.

This report will highlight the challenges and next steps stated within the previous report, then present the progress made from the time of submission of the last report till the moment of submission of this report, more recent challenges and the next steps to be taken.

2 From February's Report

The last report presented results on some extracted features and stated some challenges the project is facing which include scheduling and mobility issues, subject attitude towards participation in exercise, cue communication and dellivery method, absence of clinician and subject feedback system, fuzzy entropy spatial complexity algoritm problems and non-uniform session times.

Also stated within the last report were the set of steps to be taken next. This steps included establishing the best pre-processing path in terms of features being more discriminable, resolving left over issues with audio cue delivery mechanism, development of hand held annotator for taking feedback from clinician and subject, developing montage analysis algorithm, improvement of data acquisition paradigm and more data acquisition.

In the next section, what has been achieved from the previously stated next steps is discussed.

3 Progress

In order to establish uniform data acquisition paradigm so that processing of data can be made easier, from the already acquired data, for each phase, the minumum permissible timeframe that can be met from the phase data of all subjects was adopted for data processing and set into the Generis software as the default phase data acquisition (DAQ). As stated that further data acquisition will

take place, since then data has been acquired from twenty more subjects, distributed almost equally between patients and controls.

To ensure ease of interpretation of arithmetic cues and instructions, audio recordings of the cues and instructions has ben incorporated into the Generis software for all major Nigerian languages in the areithmetic DAQ phase.

A handheld annotator is currently under design to take subject and clinician feedback on comfort, artifact activity and arithmetic task completion. More details willbe given on this as the work progresses.

In trying to establish a best data processing path, certain areas of possible improvements were noted and acted on. One of such is the processing of the auditory phase signals from which the mismatch negativity (MMN) waveform is computed. Previously,the three tone classes, one standard and two deviant were plotted directly and investigated. The MMN waveform is meant to be computed as the difference between the deviants and the standard tone. This has been corrected and the results shown for each subject under section 5. Also to further smoothen the MMN waveform, a five point moving average was used and a minimum-maximum scaler was used on each waveform to make visualization easier.

Formerly, in computing the fuzzy entropy, the library used had a minimum space complexity requirement. Electrode of cortical regions of high spatial proximity were combined to overcome this. This might lead to some information loss. This has been improved by augmenting electrode of each region with spatial dimensions(channels) of gaussian noise of zero mean and a unit standard deviation. The new results are shown under section 5. After extensive literature review a self developed fuzzy entropy algorithm has been written and tested on univariate time-series, it is to be tested on multivariate time-series next. The code is given in section 6

4 Next Steps

Over the period of four weeks, I will be doing the following

- COntinue development of hand held annotator device
- Improving self-developed fuzzy entropy library to work with multivariate time-series(2D data).
- Recomputing fuzzy entropy features.
- Comparing fuzzy entropy features from developed library to sourced library.
- Computing auditory steady state features.

5 Figures

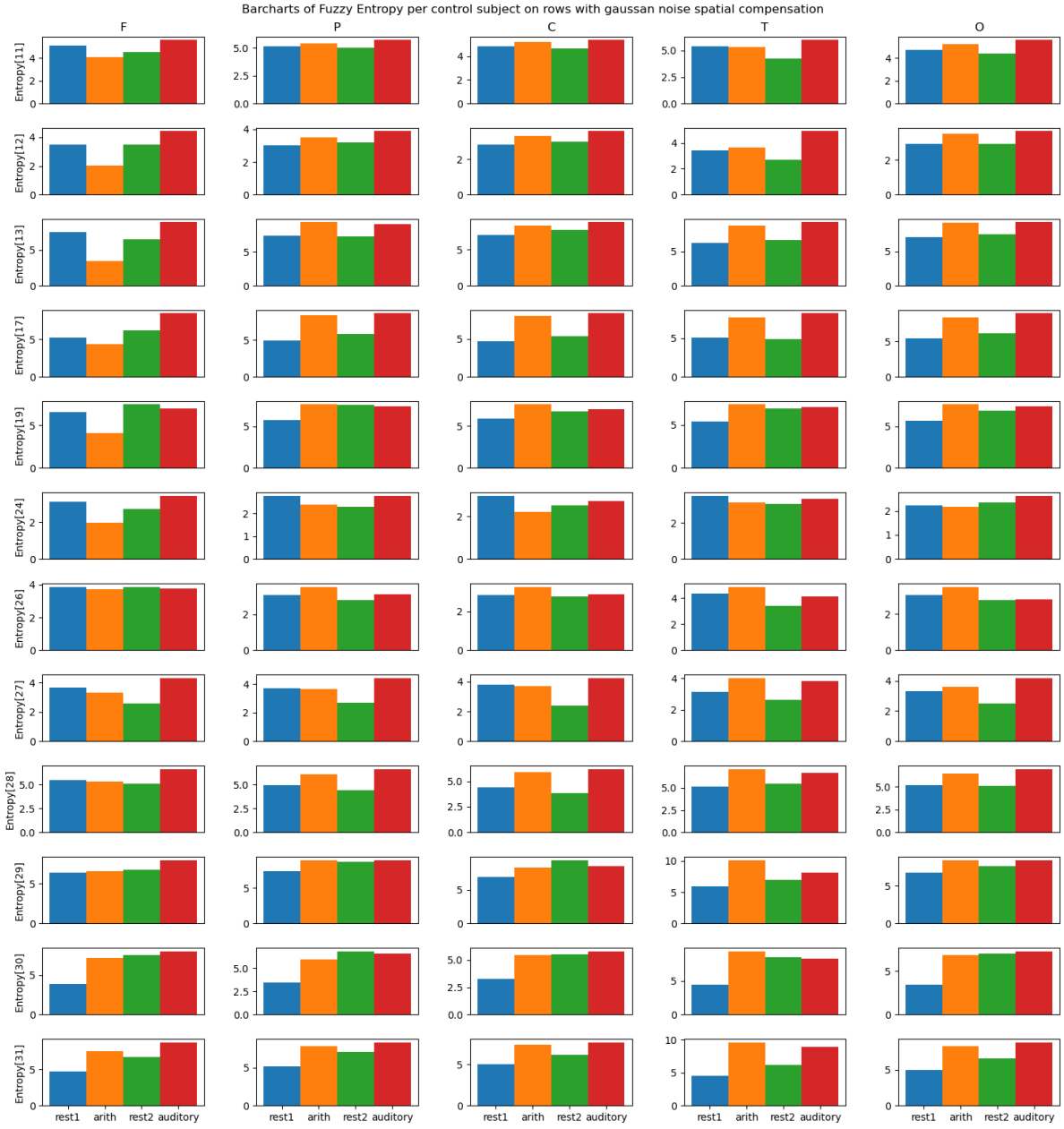

Figure 1: Fuzzy Entropy from controls

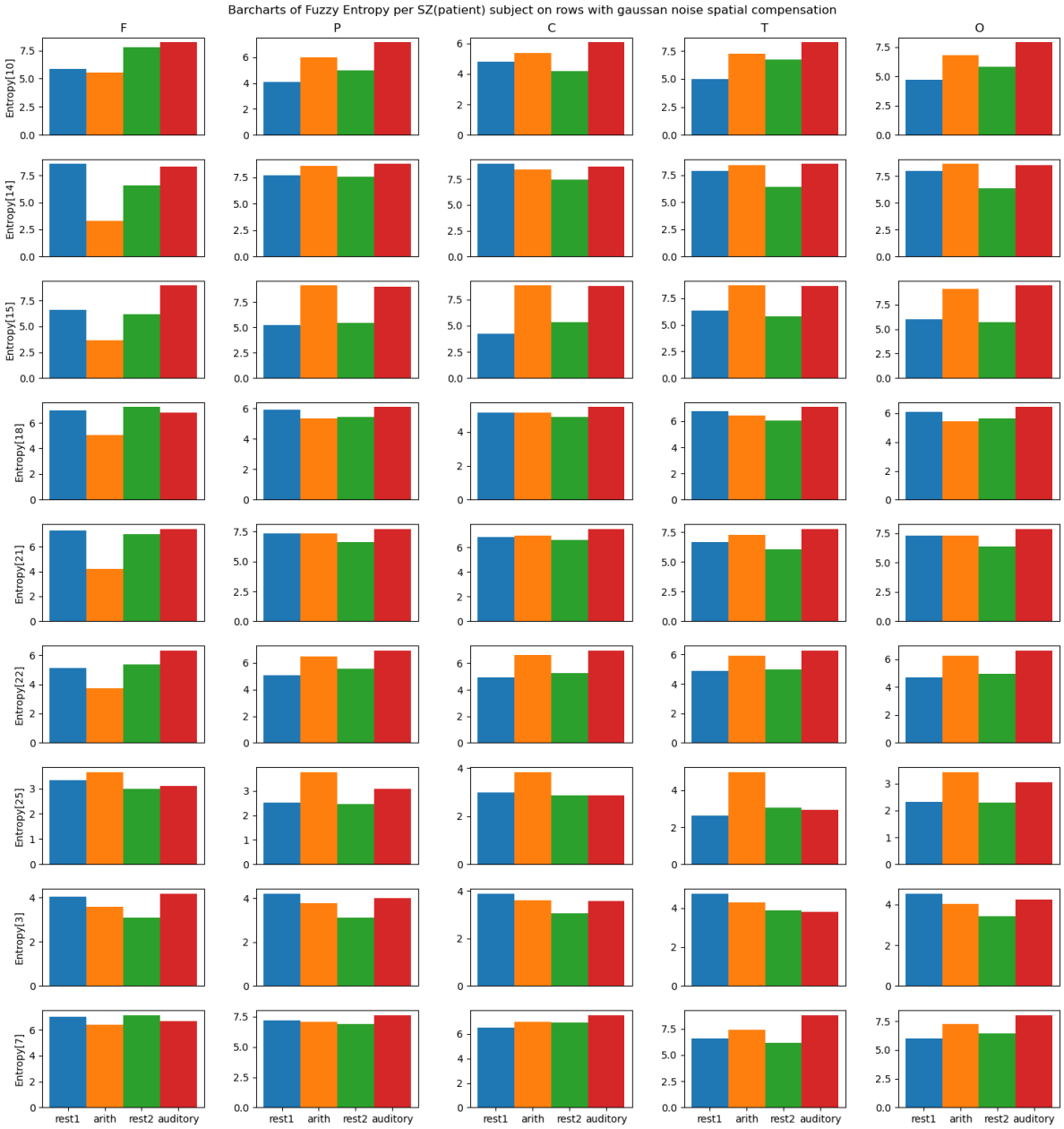

Figure 2: Fuzzy Entropy from patients

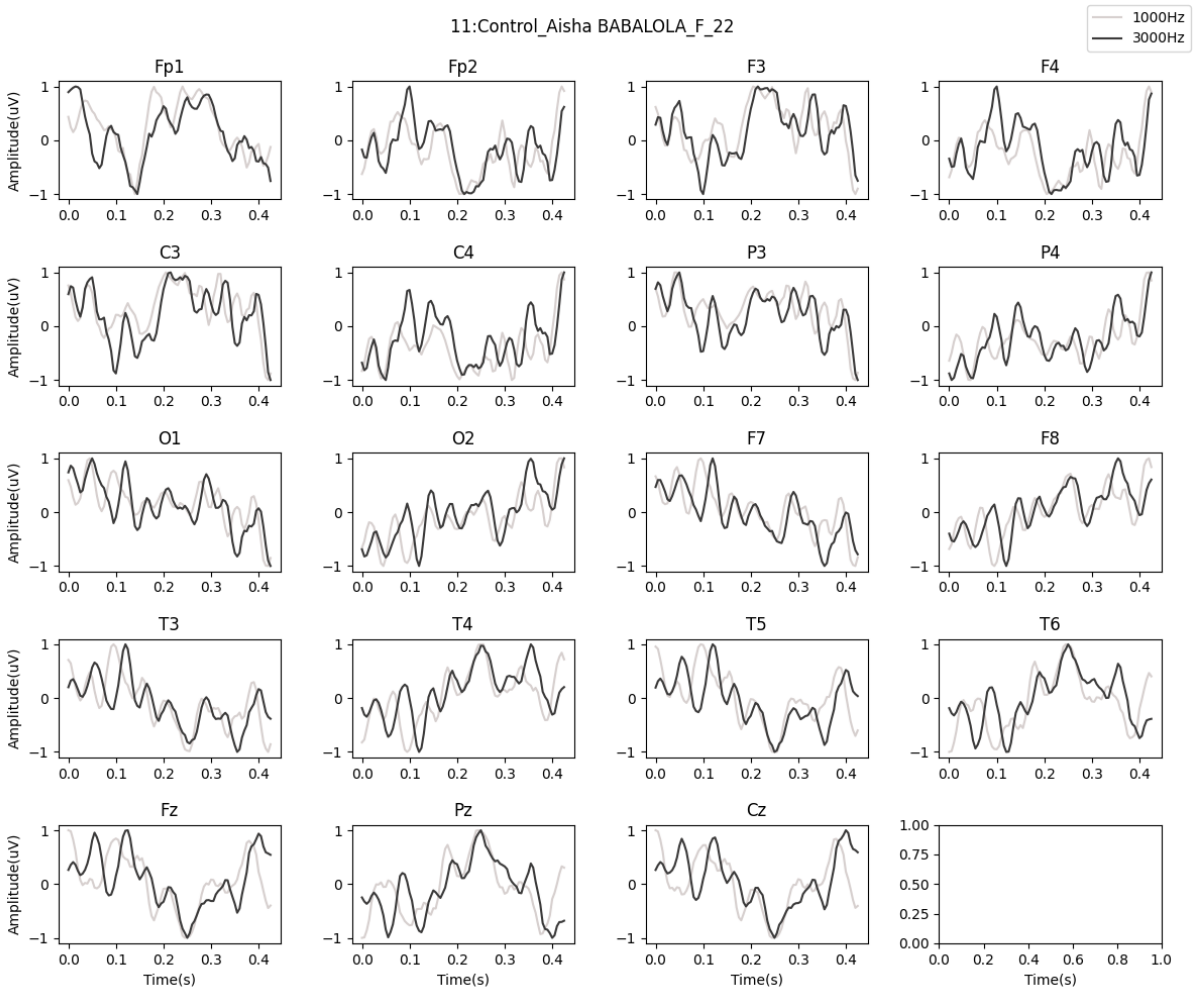

Figure 3: A control subject MMN plots

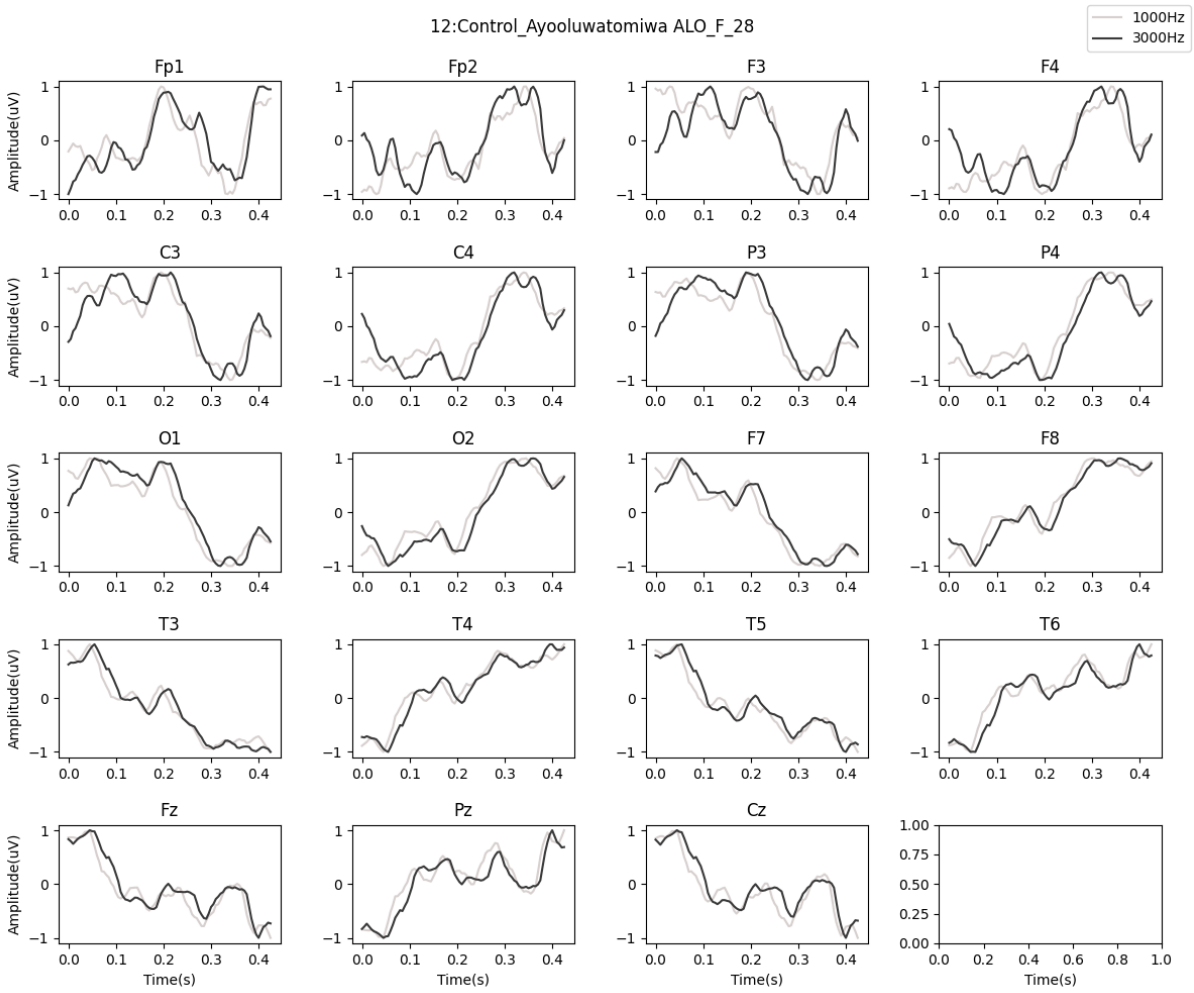

Figure 4: A control subject MMN plots

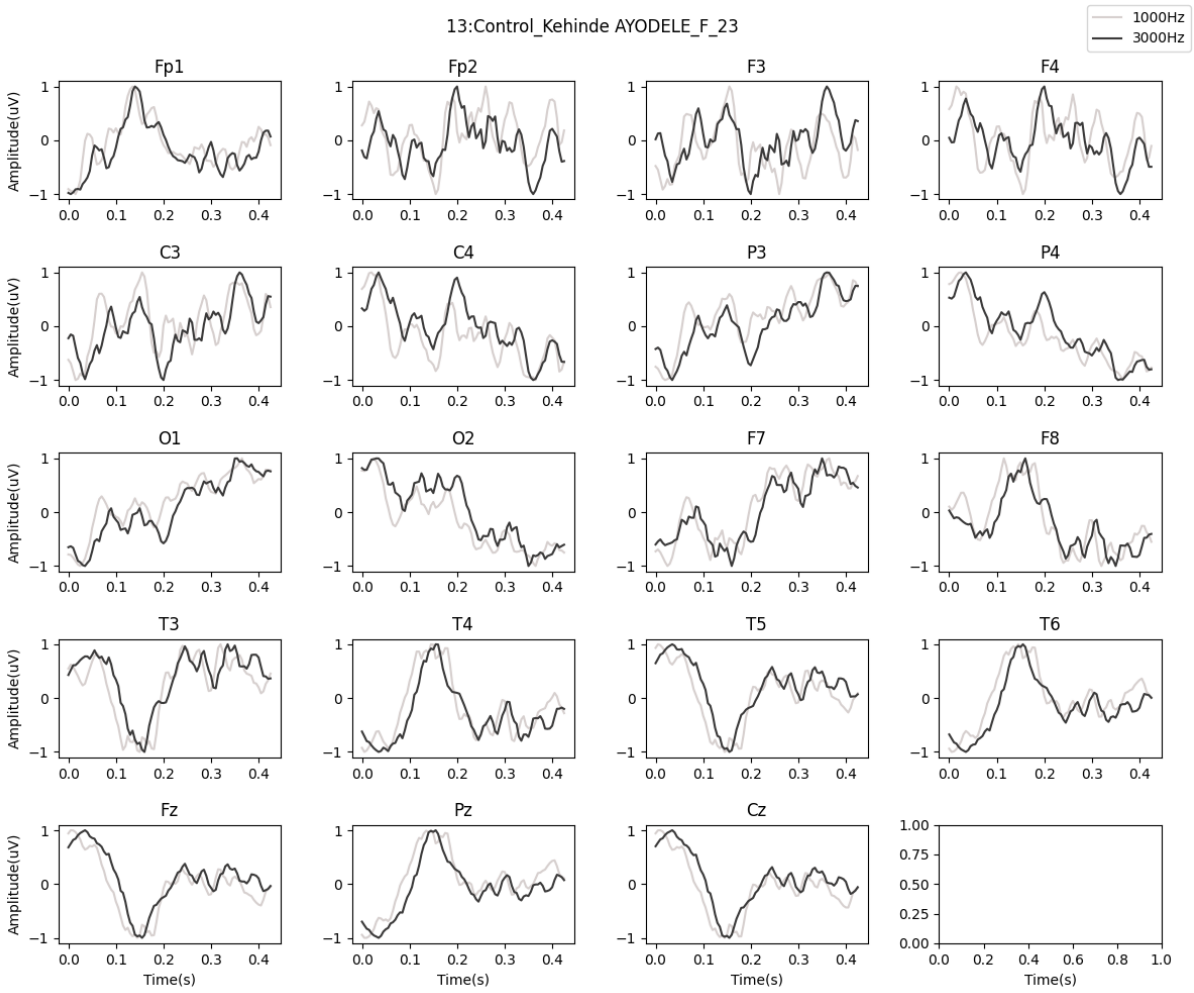

Figure 5: A control subject MMN plots

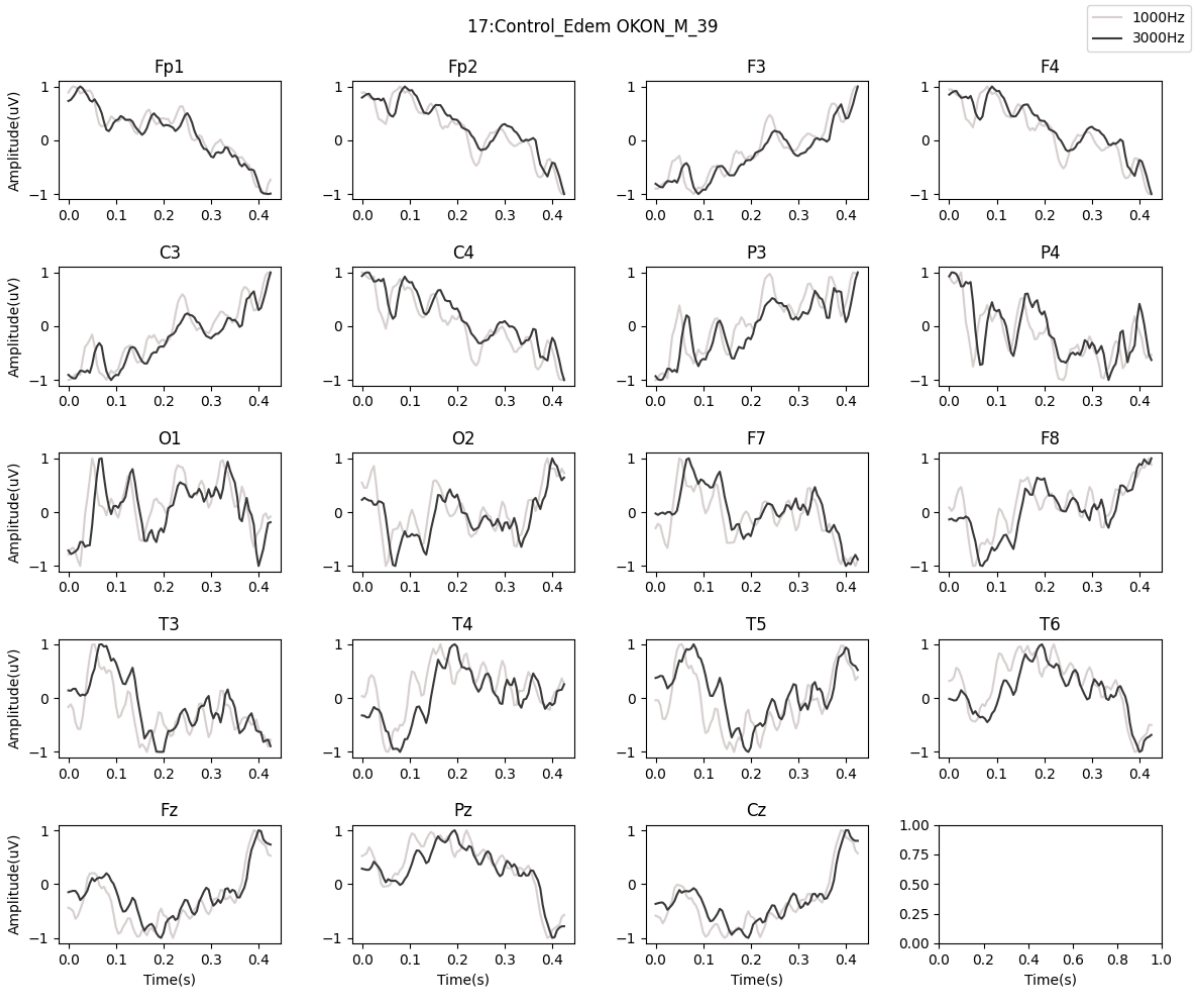

Figure 6: A control subject MMN plots

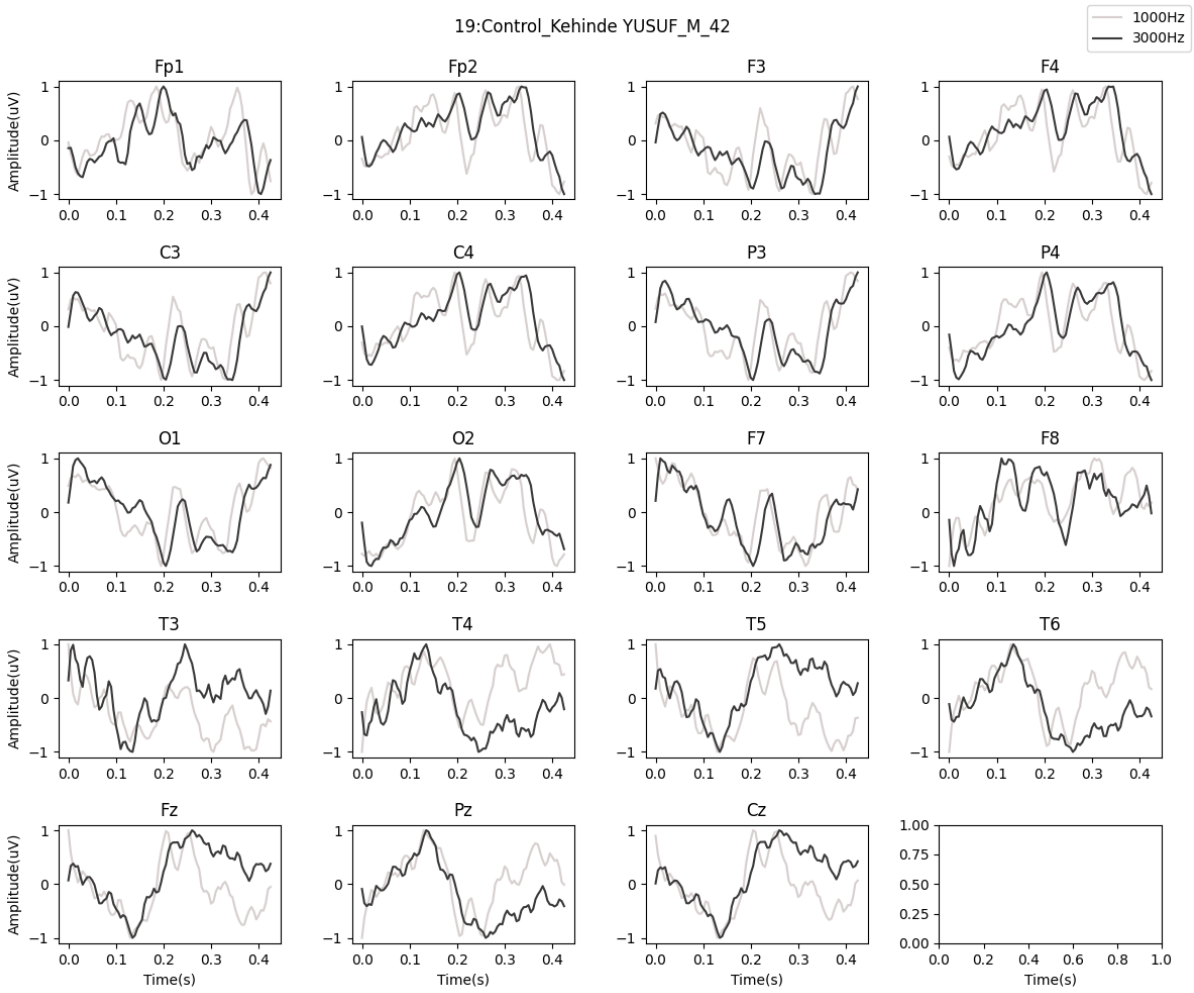

Figure 7: A control subject MMN plots

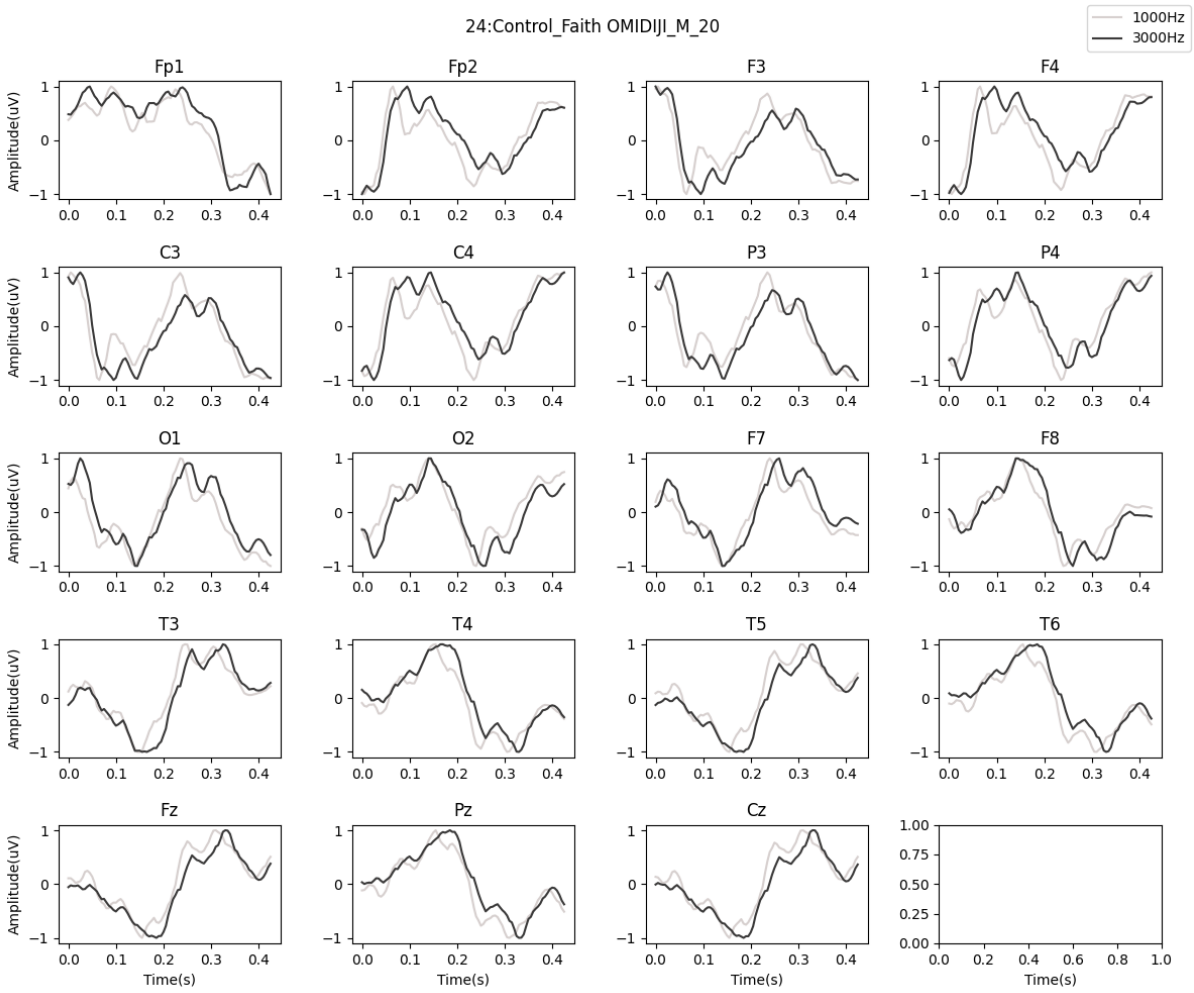

Figure 8: A control subject MMN plots

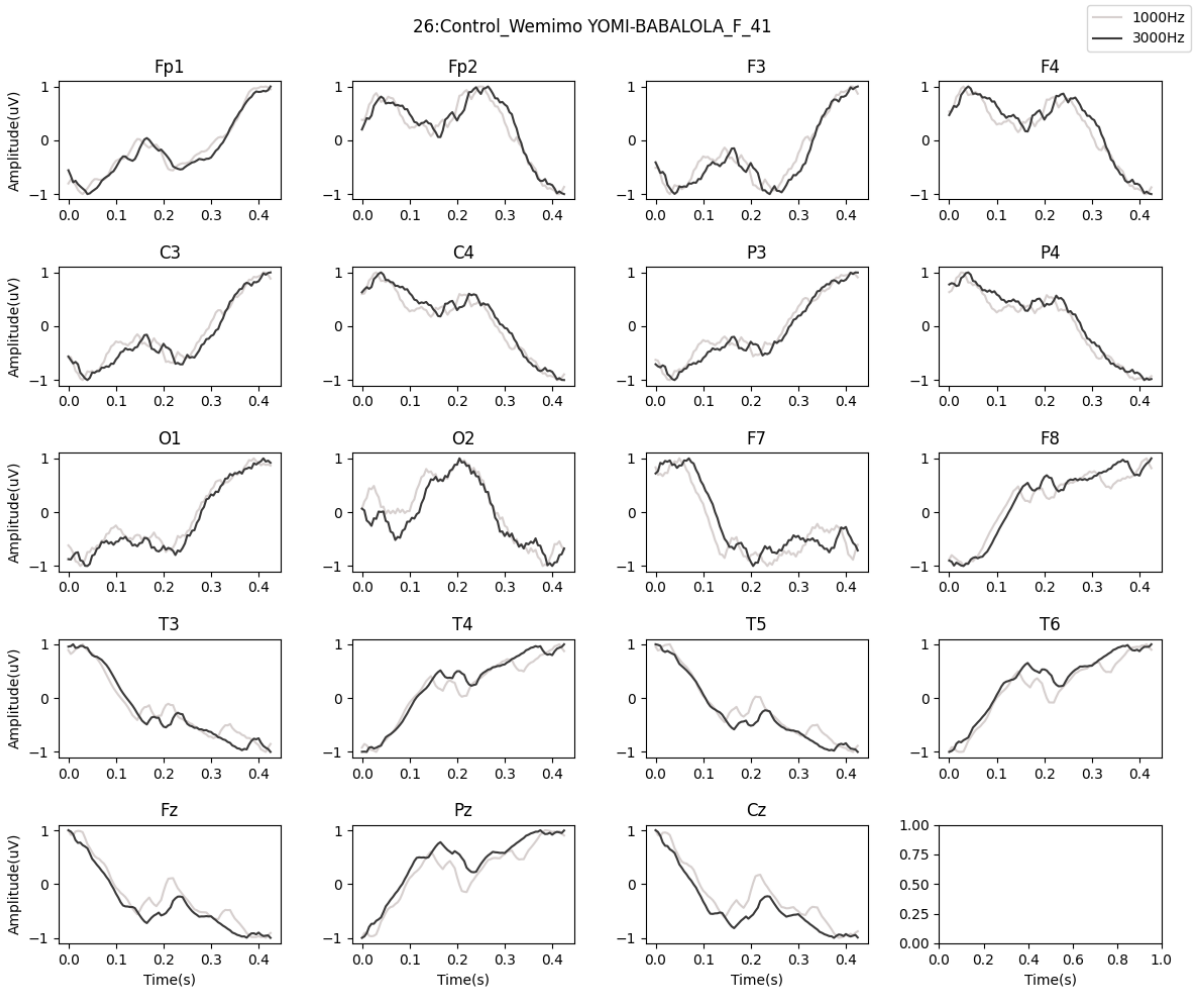

Figure 9: A control subject MMN plots

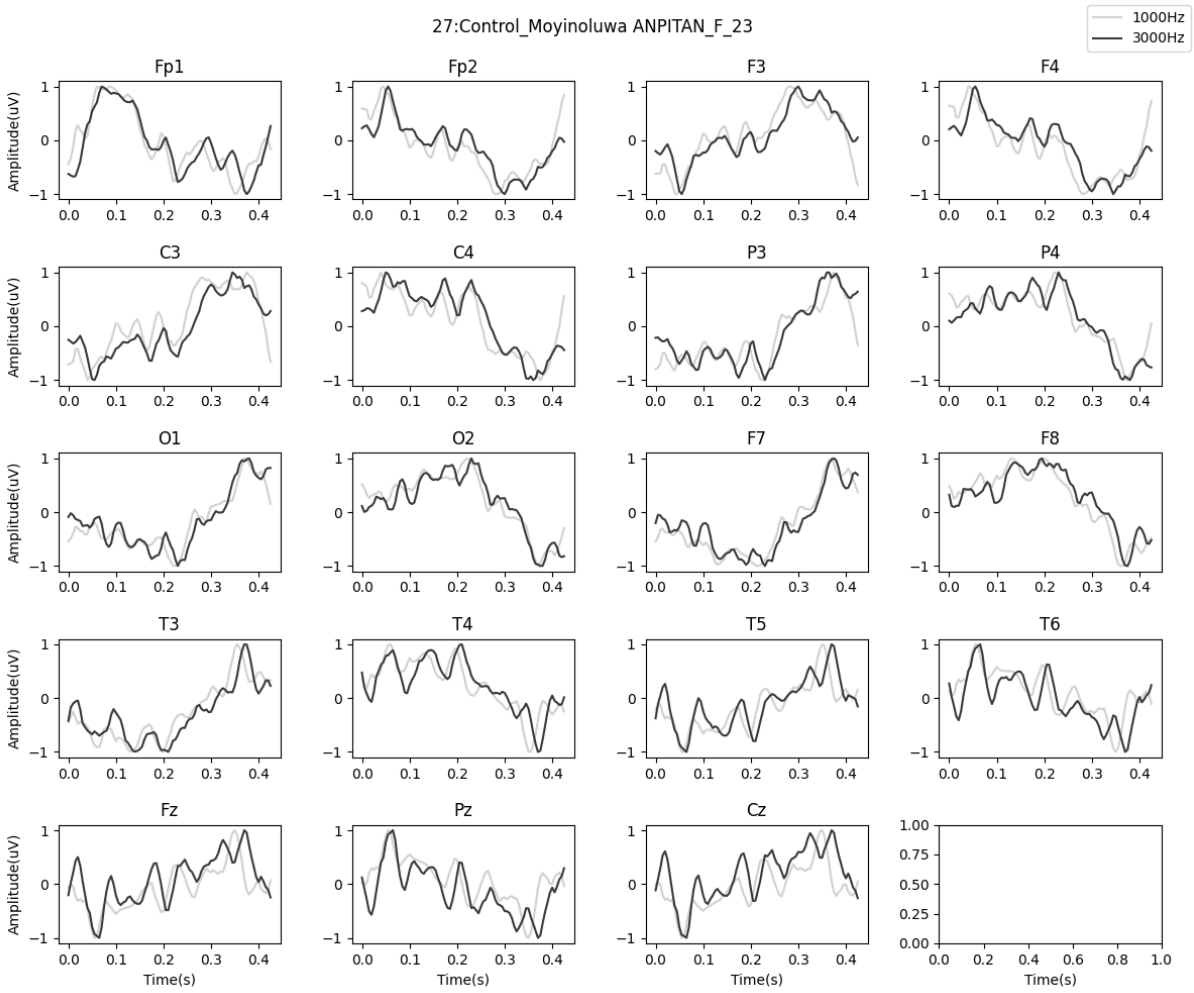

Figure 10: A control subject MMN plots

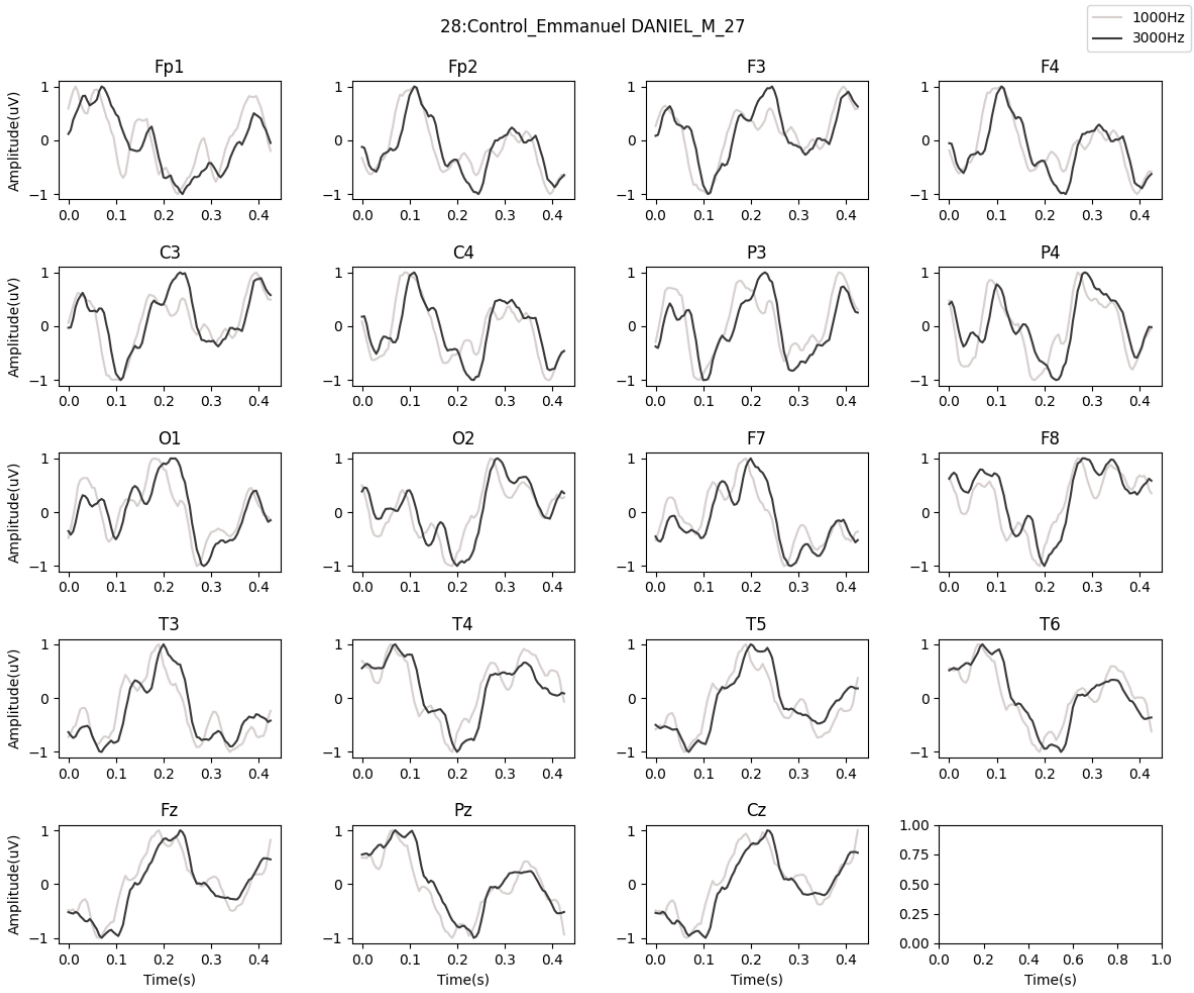

Figure 11: A control subject MMN plots

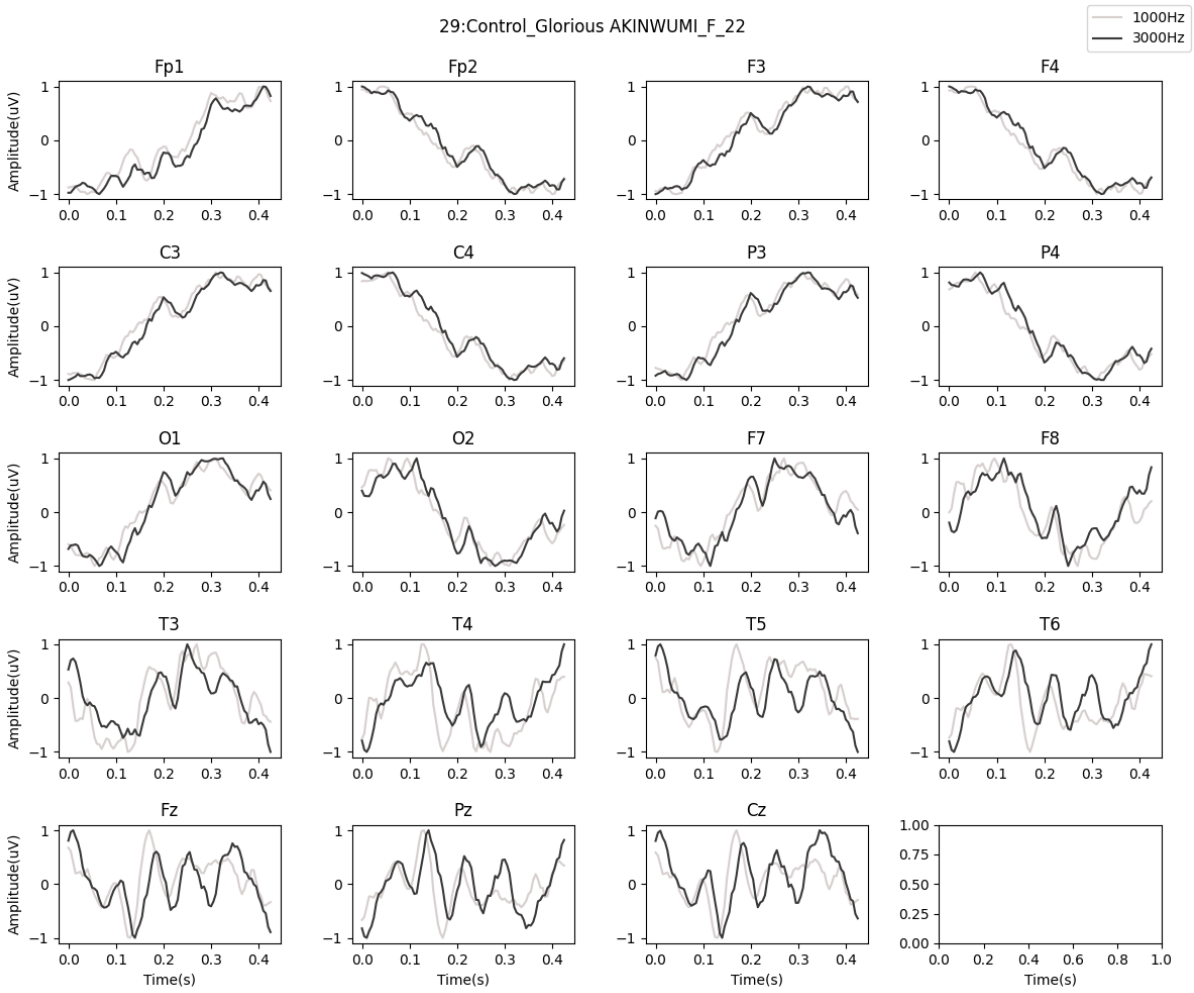

Figure 12: A control subject MMN plots

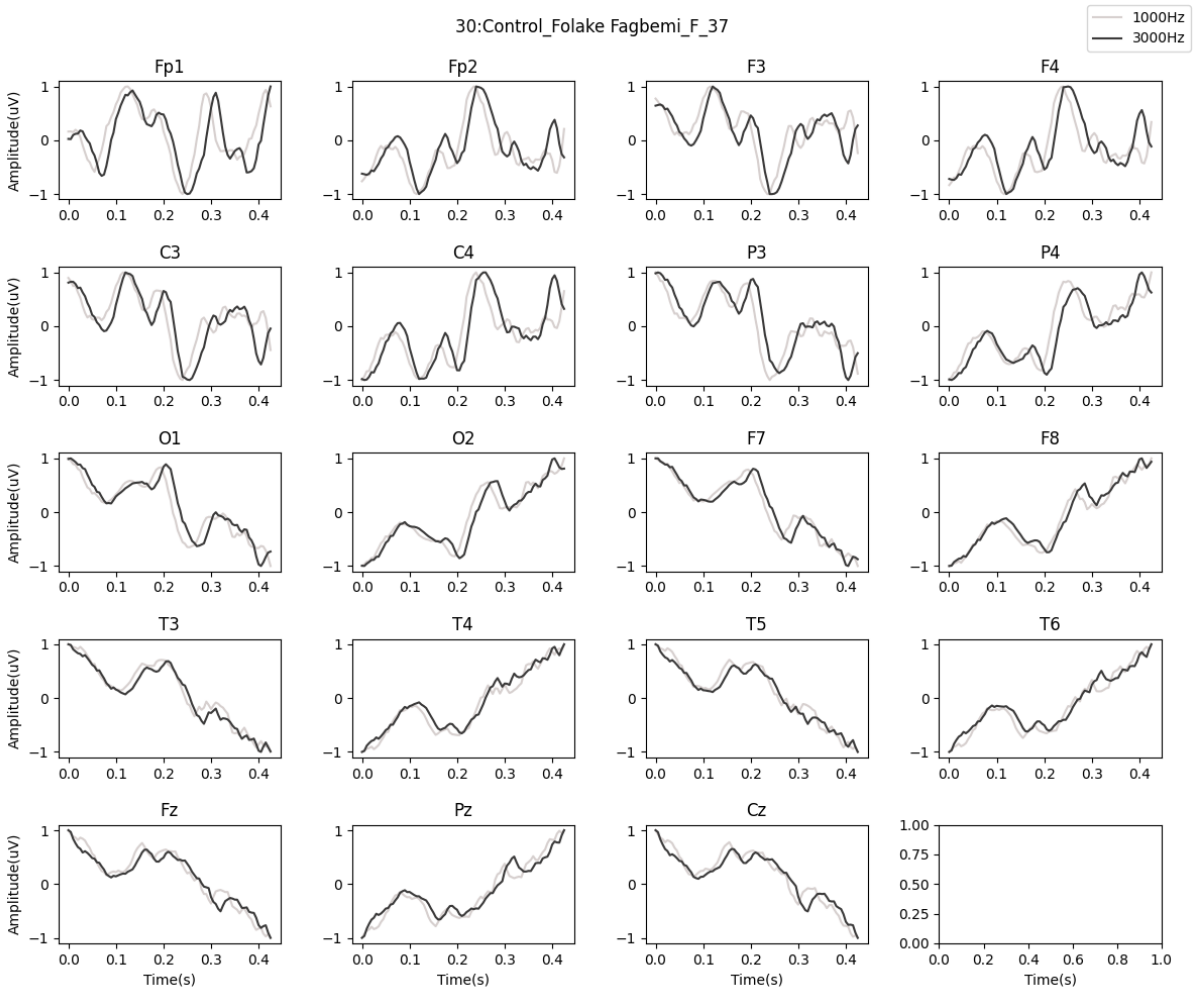

Figure 13: A control subject MMN plots

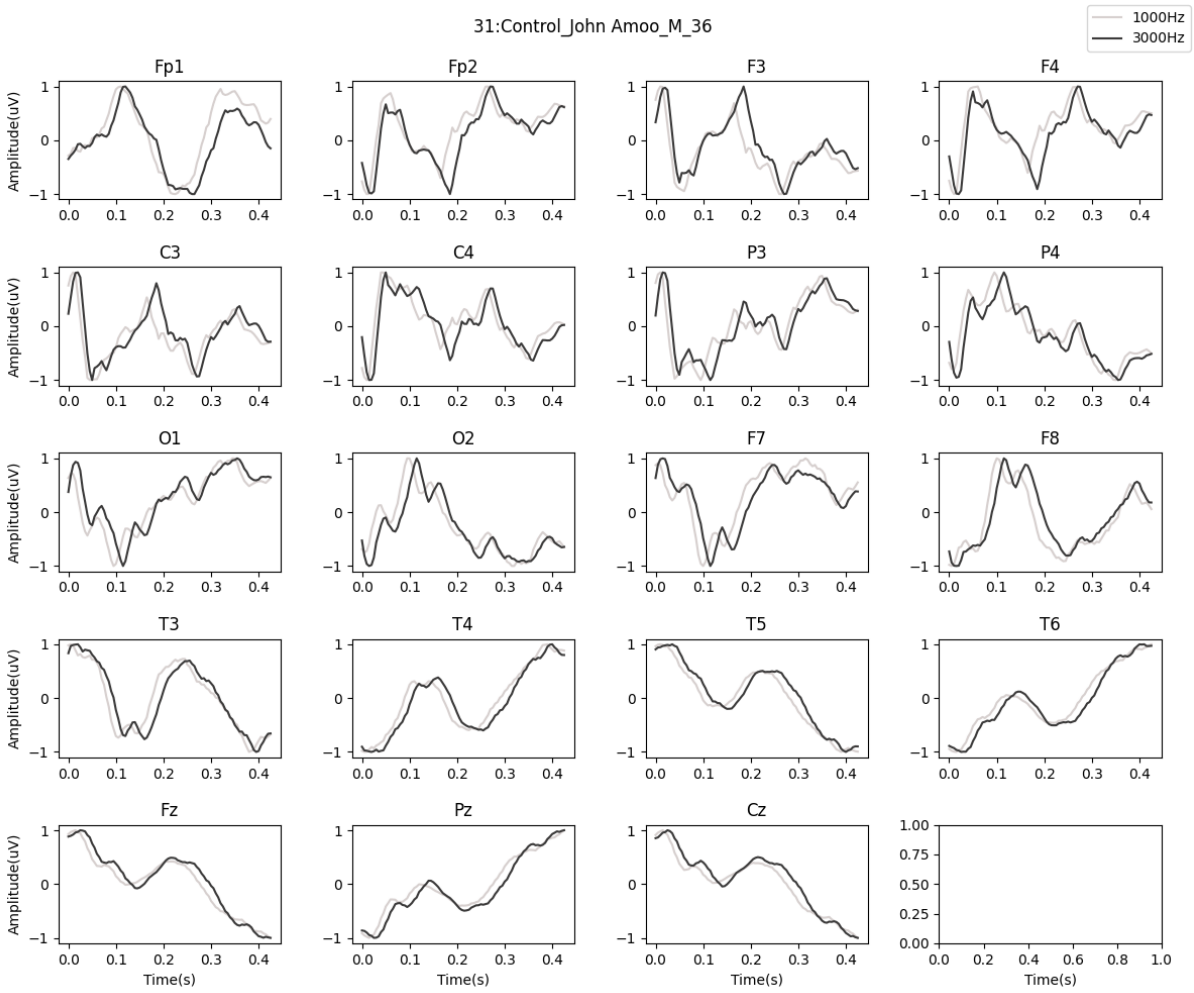

Figure 14: A control subject MMN plots

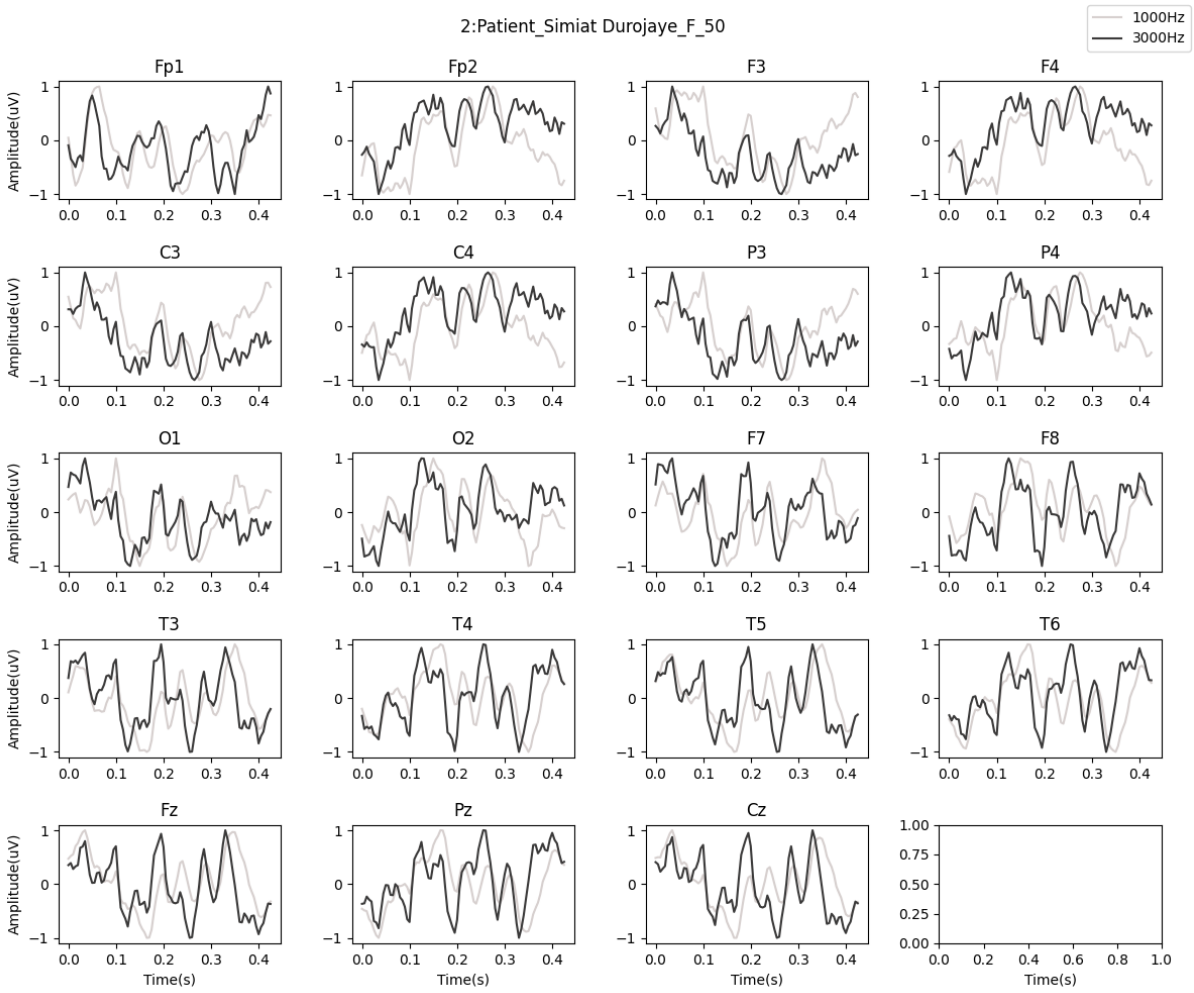

Figure 15: A schizophrenia (SZ) subject MMN plots

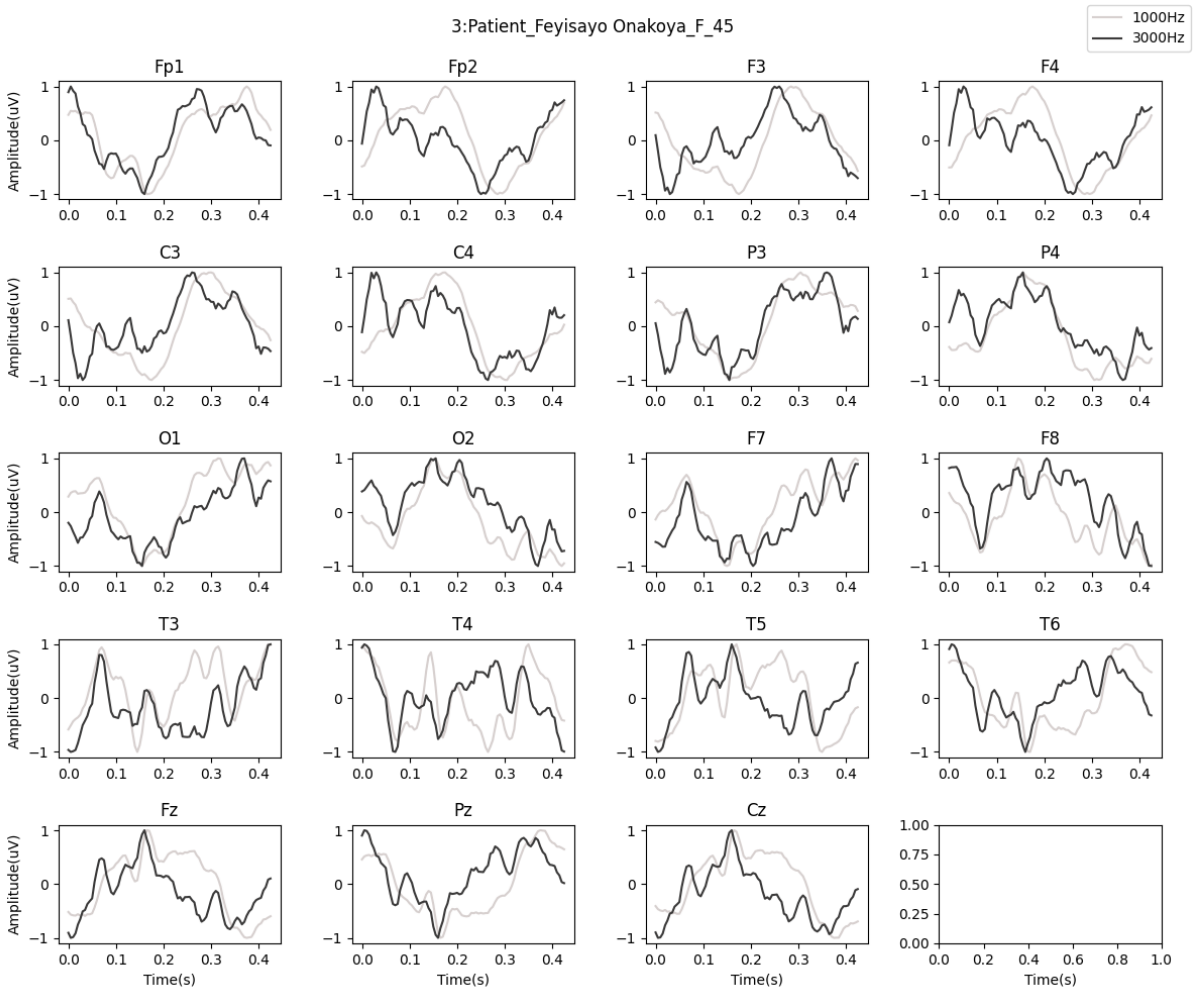

Figure 16: A SZ subject MMN plots

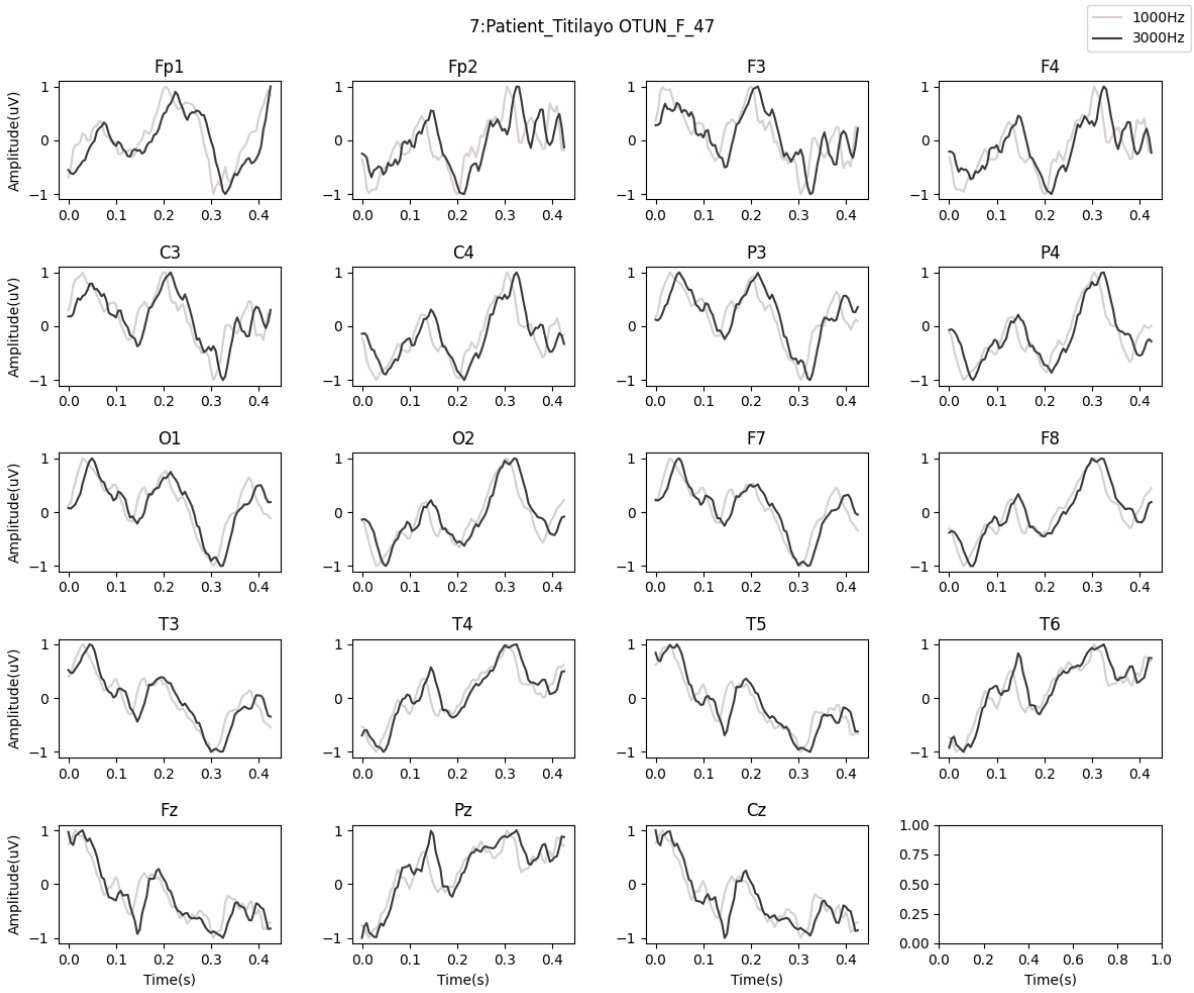

Figure 17: A SZ subject MMN plots

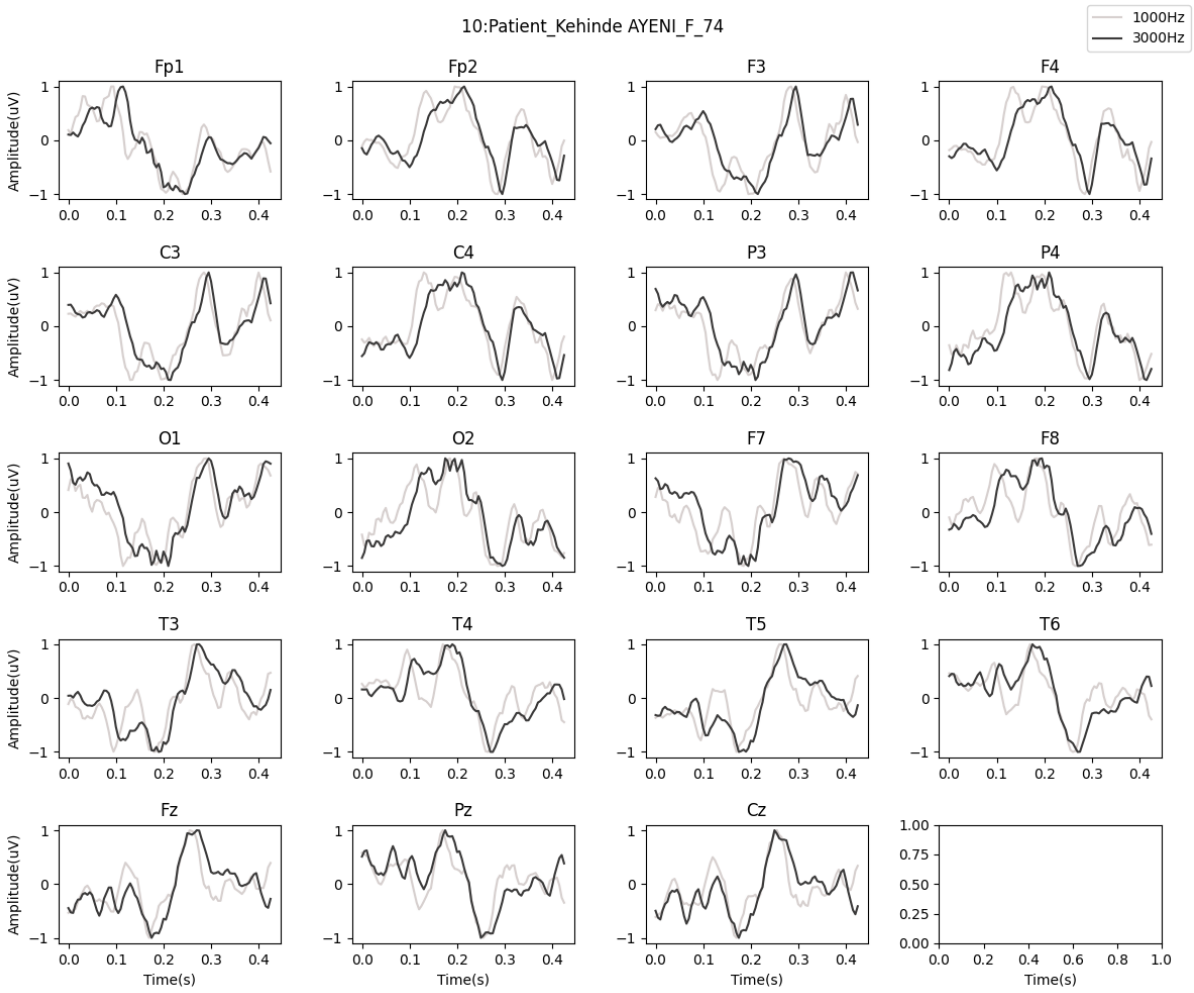

Figure 18: A SZ subject MMN plots

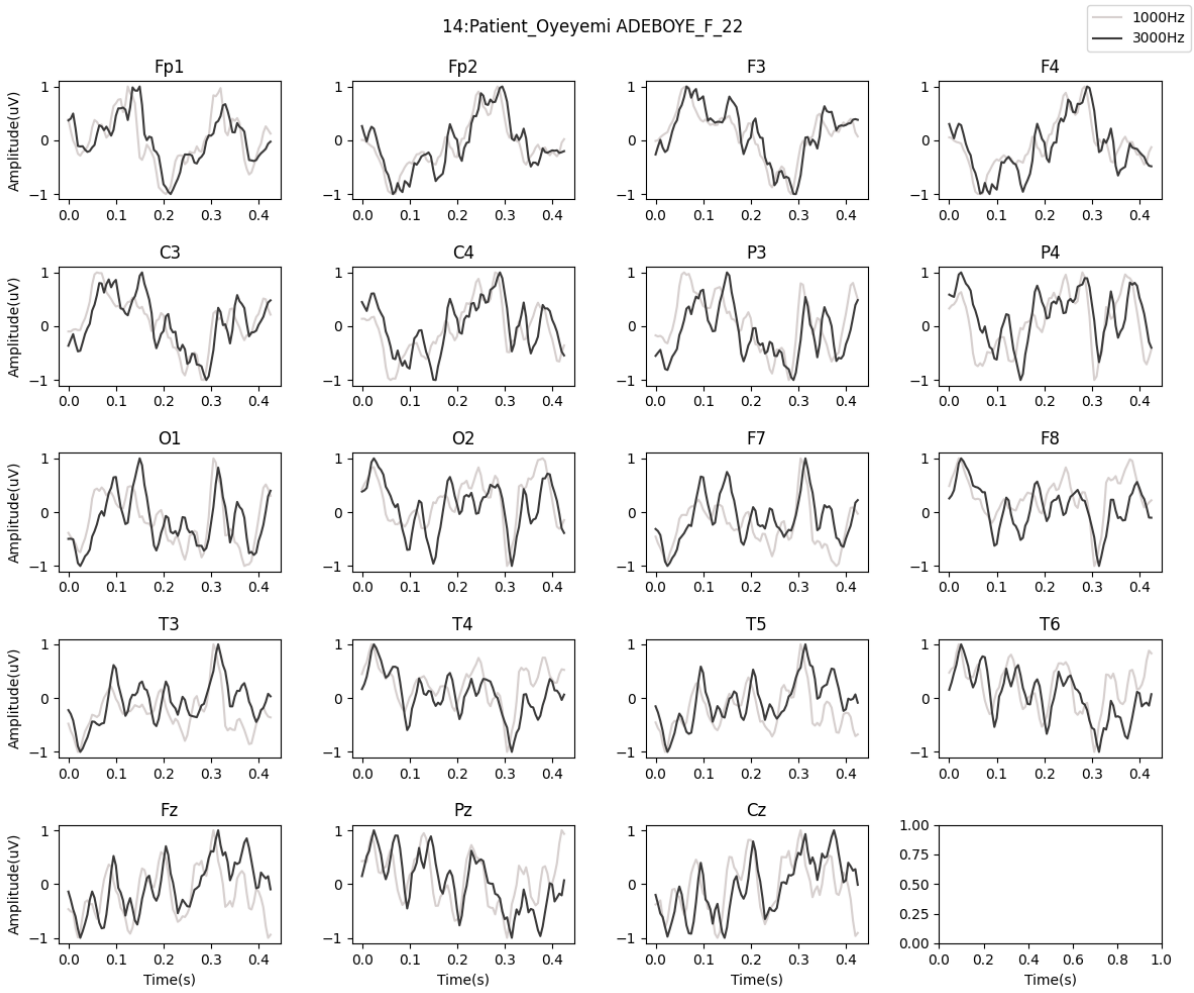

Figure 19: A SZ subject MMN plots

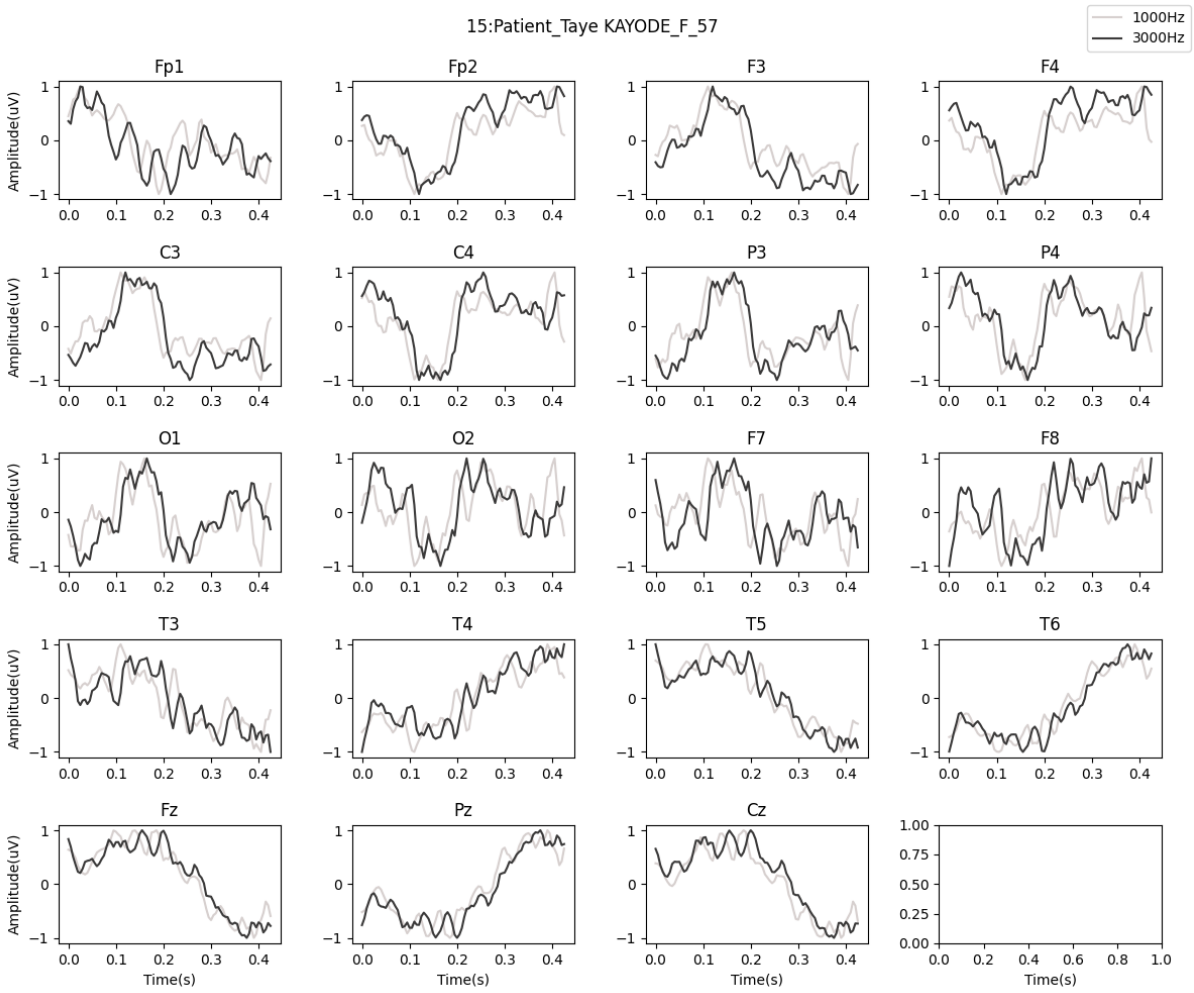

Figure 20: A SZ subject MMN plots

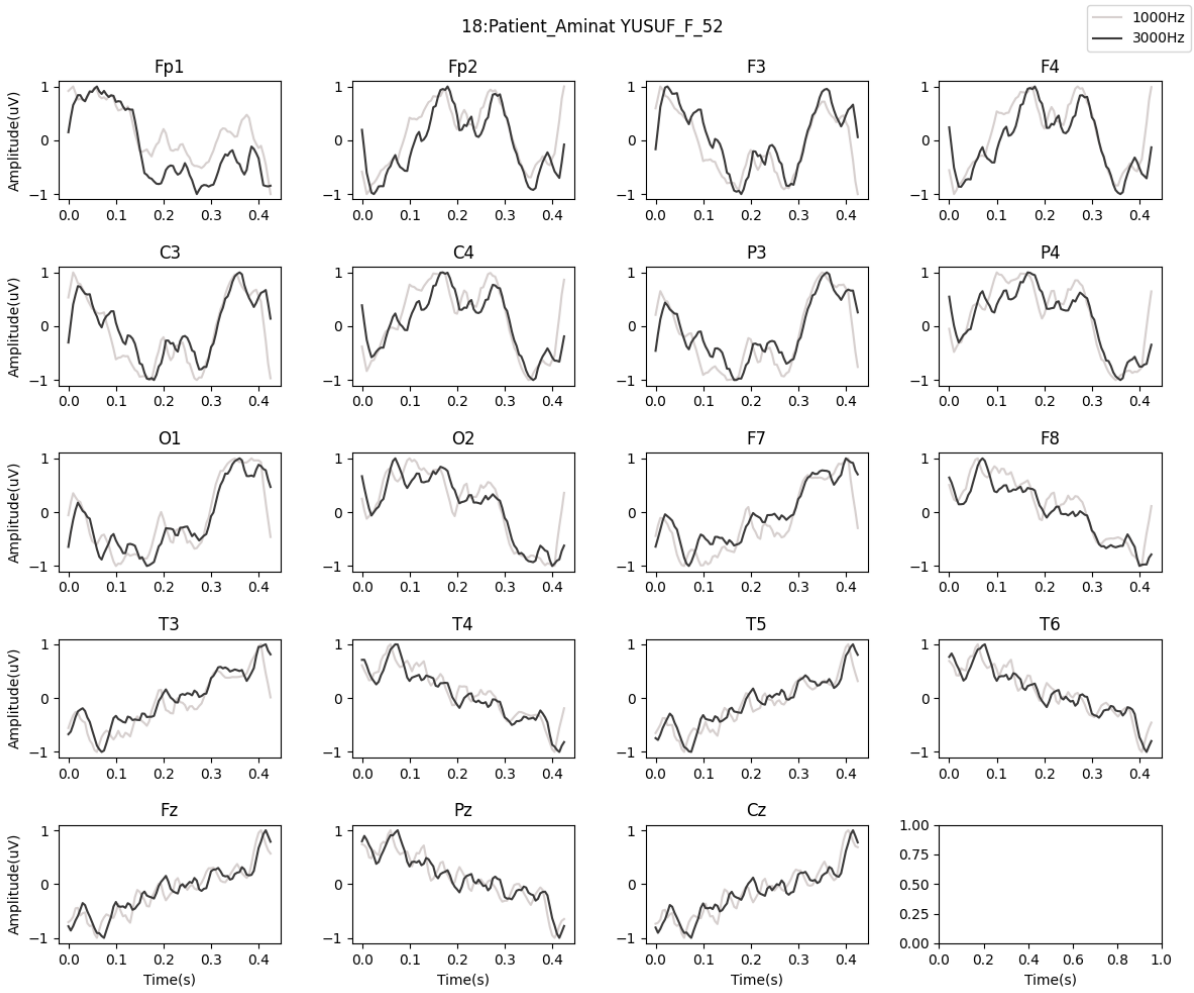

Figure 21: A SZ subject MMN plots

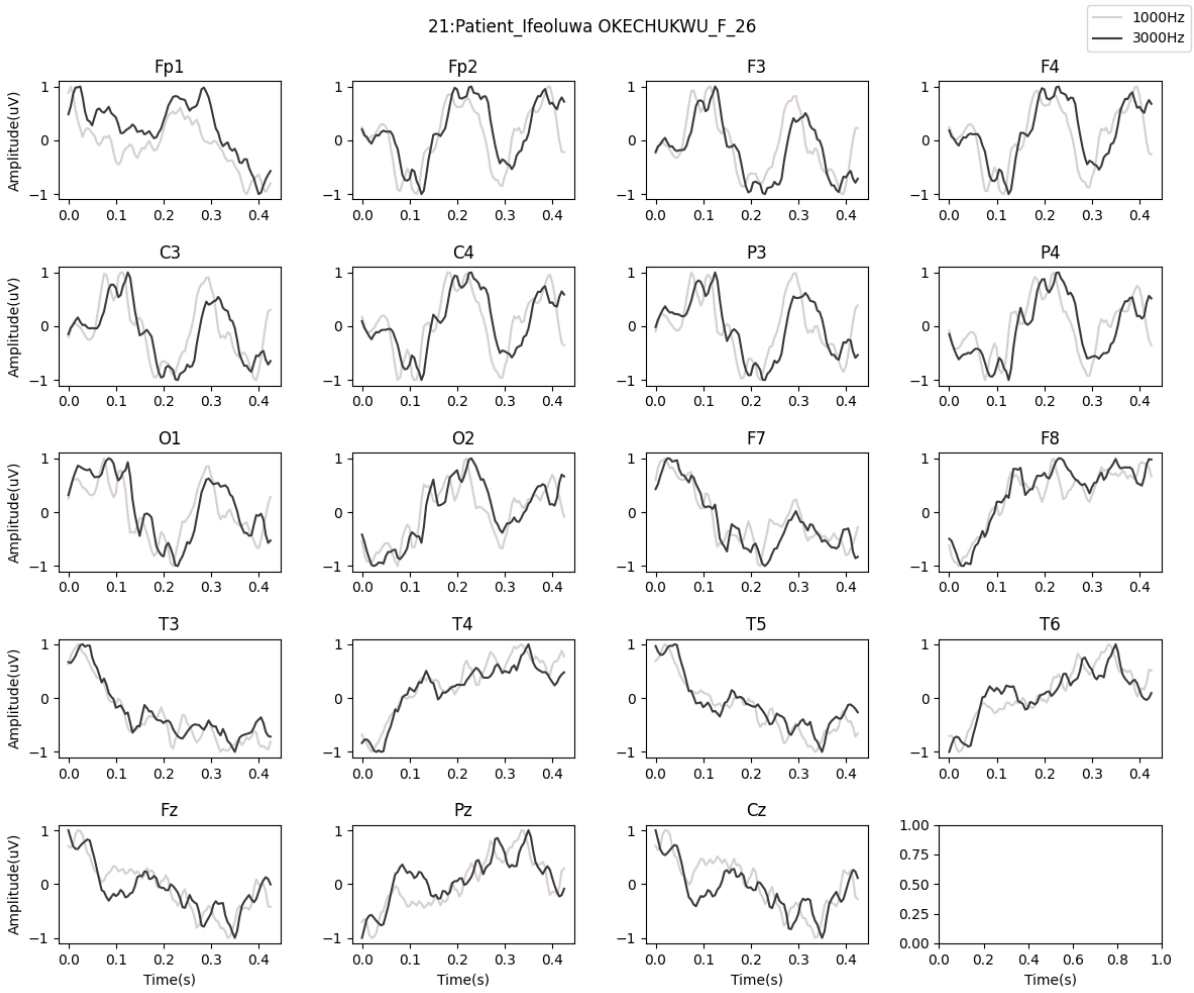

Figure 22: A SZ subject MMN plots

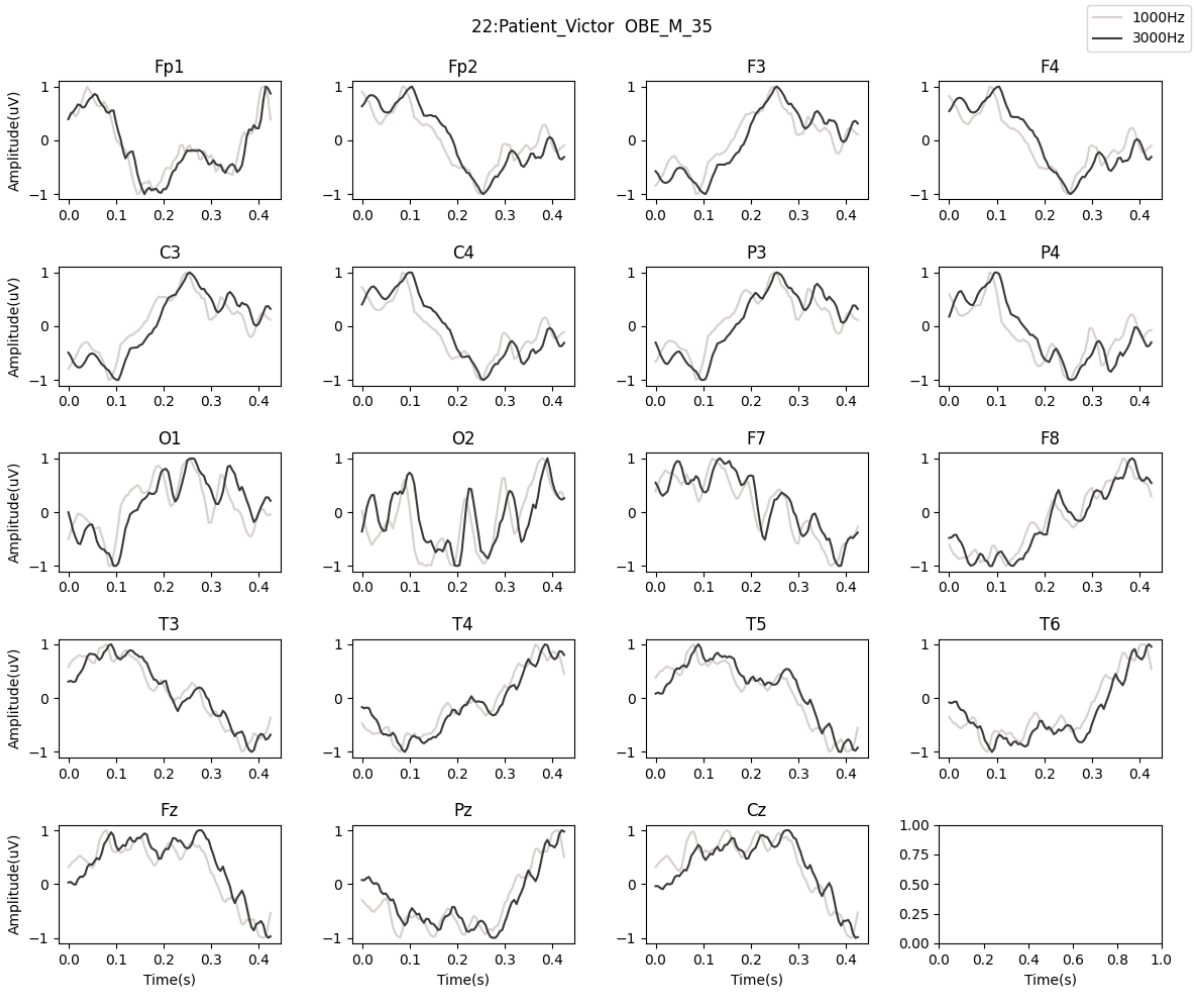

Figure 23: A SZ subject MMN plots

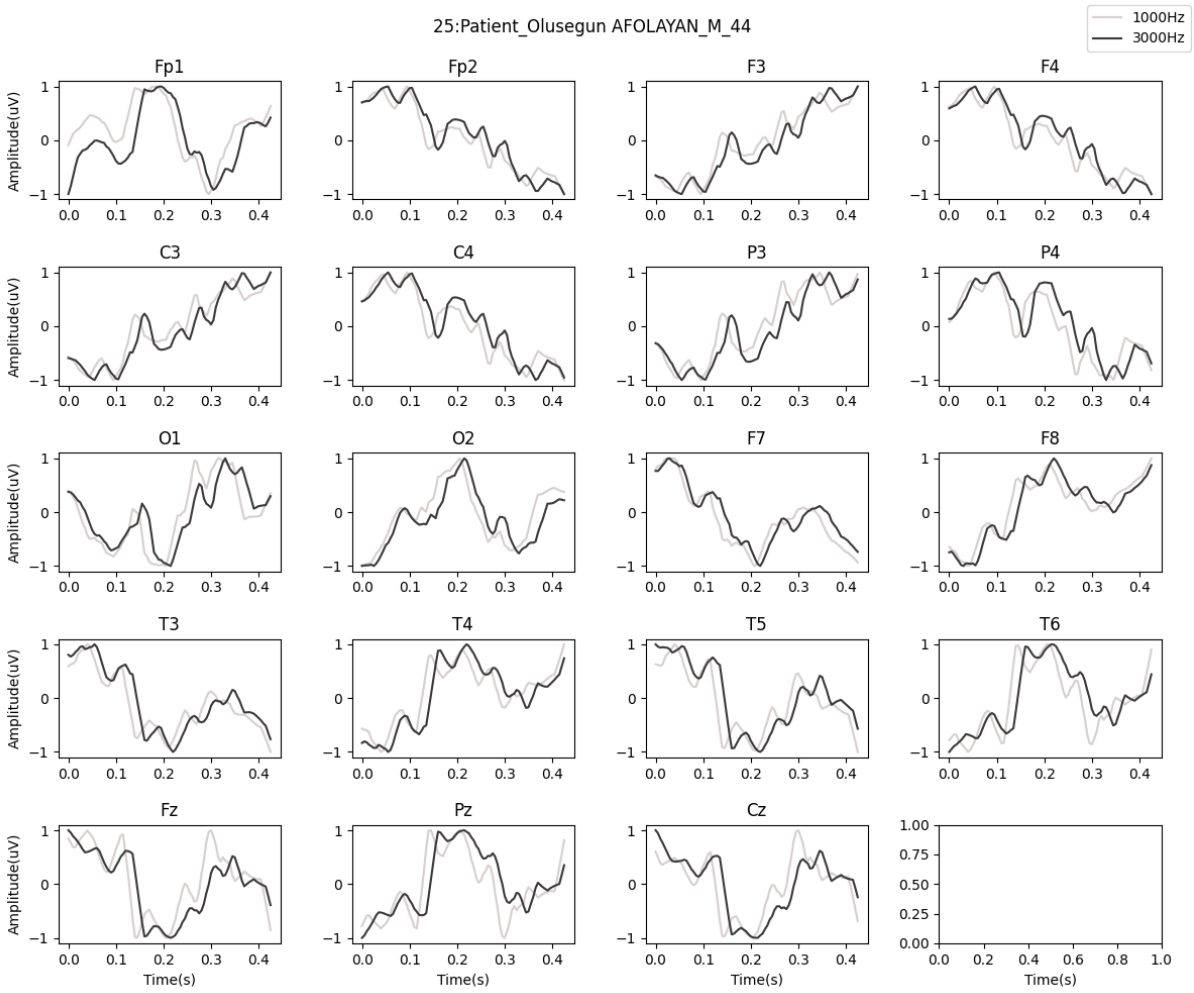

Figure 24: A SZ subject MMN plots

6 Appendix

6.1 Fuzzy Entropy Code

```
import numpy as np
from scipy.spatial.distance import cdist
import time

def sigmoid(x,r):
    assert isinstance(r,tuple), 'When Fx = "Sigmoid", r must be a two-element tuple.'
    y = 1/(1 + np.exp((x-r[1])/r[0]))
    return y
def default(x,r):
    assert isinstance(r,tuple), 'When Fx = "Default", r must be a two-element tuple.'
    y = np.exp(-(x**r[1])/r[0])
    return y
def modsampen(x,r):
    assert isinstance(r,tuple), 'When Fx = "Modsampen", r must be a two-element tuple.
    y = 1/(1 + np.exp((x-r[1])/r[0]))
    return y
def gudermannian(x,r):
    if r <= 0:
        raise Exception('When Fx = "Gudermannian", r must be a scalar > 0.')
    y = np.arctan(np.tanh(r/x))
    y = y/np.max(y)
    return y
def linear(x,r):
    if r == 0 and x.shape[0]>1:
        y = np.exp(-(x - np.min(x))/np.ptp(x))
    elif r == 1:
        y = np.exp(-(x - np.min(x)))
    elif r == 0 and x.shape[0]==1:
        y = 0
    else:
        print(r)
        raise Exception('When Fx = "Linear", r must be 0 or 1')
    return y

class fuzzEntropy:
    def __init__(self,window_size,dissimilarity_index,membership_function='linear'):
        self.m = self.window_size = window_size
        self.r = self.dissimilarity_index = dissimilarity_index
        self.mu = self.membership_function = globals()[membership_function.lower()]

    def __computeFuzzyMatrix(self,x,m):
        if x.ndim == 1:
            N = x.shape[0]
            Xm = np.array([x[i:i+m-1].tolist() for i in range(0,N-m)])
```

```
            dm = cdist(Xm,Xm,'euclidean')
            dm = self.mu(dm, self.r)
            phim = np.sum(dm, axis=1)/(N-m+1)
        return phim

    def _fuzzyEntropyCompute(self,x):
        N = x.shape[0]
        phim = self.__computeFuzzyMatrix(x, self.m)
        phim1 = self.__computeFuzzyMatrix(x, self.m+1)

        psim = (1/(N-self.m+1)) * np.sum(phim, axis=0)
        psim1 = (1/(N-self.m+2)) * np.sum(phim1, axis=0)

        with np.errstate(divide='ignore', invalid='ignore'):
            fuzz = np.log(psim)-np.log(psim1)
        return fuzz

def fuzzEntropy2D(x, window_size, dissimilarity_index, membership_function=gaussianMembe
        fuzzyent = fuzzEntropy(window_size, dissimilarity_index, membership_function)

        res = np.empty(x.shape[0])
        for i in range(x.shape[0]):
            res[i] = fuzzyent._fuzzyEntropyCompute(x[i,:])
        return res.mean()
```