**Methods**

In this project we developed a system that will be used in conducting experiments based on real-time eye and EEG data. We calibrated and set up the eye tracker, recorded eye data, researched its data structures, and analyzed eye data. We created a virtual environment containing Python recording packages for eye tracker and EEG.

* **EEG System** *ant-neuro eego-mylab.*
* **Python packages:**
  + *lab streaming layer LSL.* Records eye tracker, triggers, and EEG data and provides xdf files containing all the channels recorded.
  + *pandas*
  + *numpy*
  + *matplotlib*
  + *sklearn*
* **Eye tracker** *Tobii Spectrum Pro.*

**Preliminary Experiments**

First Mission – System Calibration

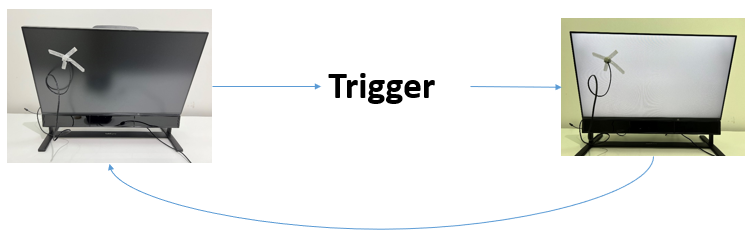
The system included a computer sending triggers via EGG system and another one recording them from it. The first goals were calibrating the system, getting familiarized with the recording equipment, and getting a better understanding of Lab streaming layer (LSL) python package and interfaces of data collection. To complete this goal, after collecting some data and understanding how to analyze it, we conducted a photodiode experiment.

**Photodiode Experiment**

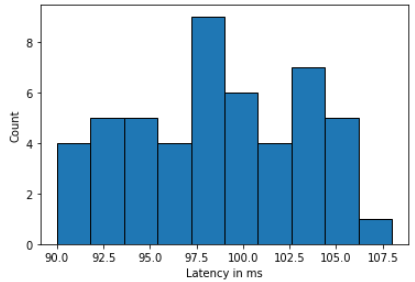
Main goal: calculate the average latency of the system, between sending a trigger and recording it as input.

We taped a photodiode on a computer screen and ran a Python script that sends a trigger through the EEG and turns the screen white for 200ms, multiple times. Then, using the second computer, we recorded the EEG data. We collected, preprocessed, and labeled the time series data from EEG.

We measured the latency between the time the trigger was sent and the time the photodiode recorded a significant change in brightness. We conducted this experiment twice: while recording 2 channels – photodiode and triggers, and while recording multiple channels – photodiode, triggers and 64 EEG electrodes.



Results: the average in both cases is about 10ms, thus, the system is robust to the number of recorded channels, and it has no cost in terms of latency. This finding must be taken as consideration when running real-time experiences; in case the time difference between components of the experiment is shorter than 10ms, the results obtained will be wrong. The results below are presented in seconds.

2 channels

* Mean – 0.098
* Standard deviation – 0.005
* Minimum – 0.089
* Maximum – 0.107

Chart, histogram

Description automatically generated66 channels

* Mean – 0.097
* Standard deviation – 0.005
* Minimum – 0.087
* Maximum – 0.110

Chart, bar chart, histogram

Description automatically generated

2 channels

Chart, histogram

Description automatically generated

64 channels

Second Mission – Eye data classifier

The goals are collecting, preprocessing, and labeling time-series eye data, using it to train and validate a classification ML algorithm. Later, use this classifier to decode eye data and predict what the subject is looking at.

**Eye Cues Experiment**

We created a dataset of 240 black photos with a white dot in it. The dots’ locations were divided to 4 labels: up, down, left, and right. In each category we sampled the locations from a uniform distribution in range that match the category.

The experiment held 240 trials. We shuffled randomly the photos and presented in each trial a fixation cue followed by a photo from the dataset. A trigger was sent with each fixation (and dot-photo???) presented. The subjects were told to look at the dot when it appears, back on the fixation cue, and so on. We recorded the eye tracker data and analyzed it. Need to understand the times to talk about it in the results.

**Eye tracker data**

We recorded from *Tobii spectrum pro* eye tracker using LSL-record. We used 4 channels in this experiment: x and y axes of left and right gaze point on display area. The data is relative to the screen the subject is looking at: both x and y coordinated range from 0 to 1, when (0, 0) is the top left corner of the screen and (1, 1) is the bottom right corner.

Figure: Example of eye data plot with triggers (preprocess exp 1)

Analyzing: while shuffling the photos randomly for each subject, we saved the premutation received and used it to label each trial with the corresponding direction of the point presented in it. We divided the time-series data from each trial (fixation to fixation \ fixation to dot?) into 10 timeframes to find out when the eye data is more informative. In each timeframe we calculated the average of x and y coordinates from the left eye, as we found out this data from each eye is almost identical. Then we trained two classifiers: maximum correlation and SVM, tested and compared their results.

Results: we found out the first half of the time right after the fixation cue is non-informative, whereas each one of the timeframes in the second half is very informative. There is no significant difference between the models.

Scatters of the eye data:

1 Scatter chart

Description automatically generated

2 Chart, scatter chart

Description automatically generated

3 Chart, scatter chart

Description automatically generated

4 Chart, scatter chart

Description automatically generated

5 Chart, scatter chart

Description automatically generated

6 Chart, scatter chart

Description automatically generated

7 Chart, scatter chart

Description automatically generated

8 Scatter chart

Description automatically generated

9 Chart, scatter chart

Description automatically generated

10 Scatter chart

Description automatically generated

