Chair of Cognitive Systems
TUM School of Computation, Information and Technology (CIT)
Technical University of Munich

# Project report

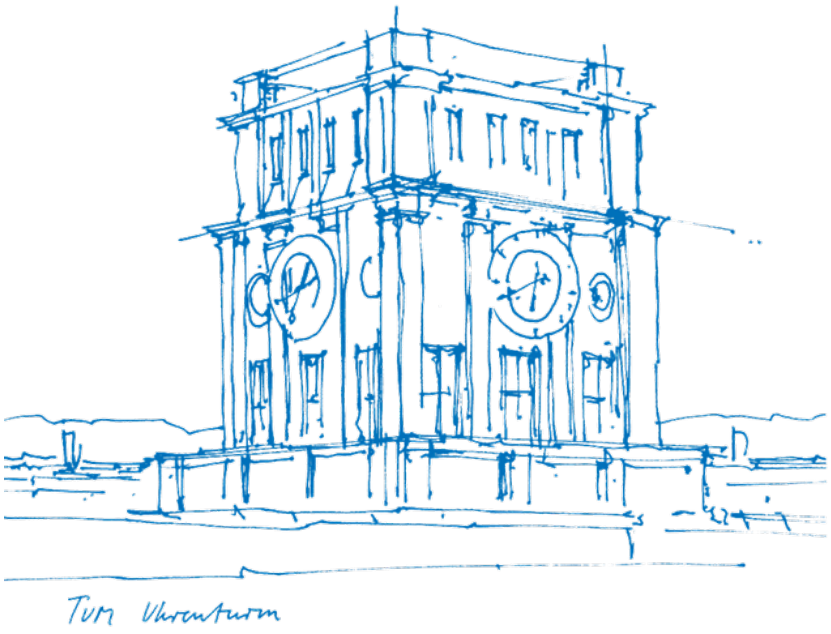
## P-300 detection

**Natascha Niessen**
**Pamela Reyna Gauna**
**Oscar Soto**
**Alena Starikova**

**Examiner**
Prof. Dr. Gordon Cheng

**Tutor**
Alireza Malekmohammadi

**Submitted on**
xx.xx.xxxx



## Contents

## Introduction

Brain Computer Interfaces (BCI) are systems that enable direct communication between the brain and an external device. These can be used to allow individuals to control or interact with technology using their neural activity and is especially useful for patients that might not be able to use their limbs due to neural or muscular diseases. Among different kind of interfaces, there is one particularly cheap, non invasive, affordable and with an acceptable temporal resolution. It involves the use of an Electroencephalogram (EEG) to read the brain's electrical activity (Minguillon et al., 2017). With this kind of imaging, one is able to obtain the electrical signals recorded from the neurons, amplify them, filter them, and process to extract the most relevant information.

The EEG can be used to measure different things such as evoked related potentials, which are brain responses to primitive stimulus, as well as event-related potentials that are responses to more cognitive or motor events (Turnip and Hong, 2012). In this project, we used an event-related potential measurement called P300. The "P" stands for positive and the "300" for the miliseconds that it takes for the brain to respond to a stimulus. In other words, the P300 is a positive deflection in the EEG that occurs approximately 300 miliseconds after a stimulus is presented.

As mentioned before, BCI can be used to help patients interact with technology and control it. With this in mind, we trained a model that is able to detect P300 signals from a subject's EEG and with it, the subject can dial the emergency number by just looking at the numbers displayed on a screen. In this article we will describe how we created the paradigm and what it consists about, the tools that we used, the pre-processing steps of the data, the classification methods, the training of the model and all of the details that helped us achieve about 96.667% accuracy.

# Methods

## Experimental design

The main goal of our project was to implement a non-invasive BCI system to imitate the situation of making an emergency call "spelling" the sequence of three digits via the P300 peaks detection with a machine learning approach. The project was divided into two parts based on the classification model: training (offline analysis) and testing (online application). We had the same one male pilot participating in both parts in all trials since the classification model changes from individual to individual.

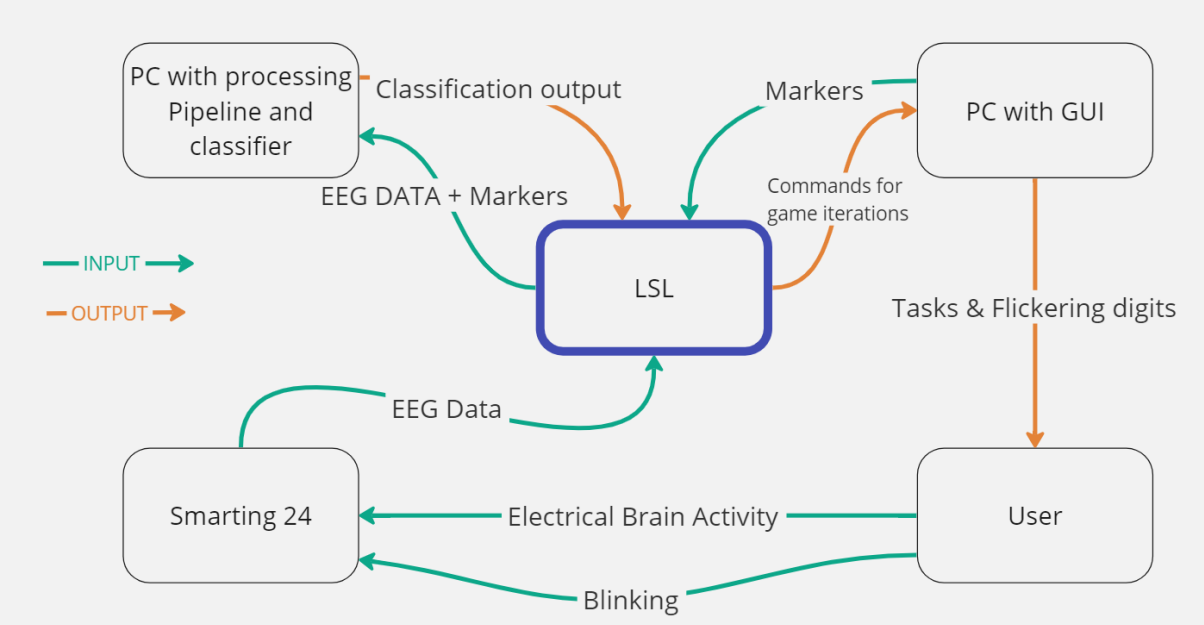
**Hardware and software setup.** The setup included a commercial wireless headset for electroencephalography (EEG) detection "Smarting 24" with an amplifier, the EasyCap (cap for the 24 electrodes placement with one reference electrode and one ground, fig. 1) and supporting software (BlueSoleil), and graphical user interface (GUI), discussed below.

**Figure 1**
*The illustration of the wireless EEG device Smarting 24.*
*https://mbraintrain.com/smarting-wireless-eeg/*

The user received information through a user-friendly graphical user interface (GUI). The detected brain activity was recorded for further processing either offline or in near-real-time. This was made possible by utilizing the lab streaming layer (LSL), a system of tools designed for real-time synchronization of data collection, viewing, and recording. The recorded brain activity was then processed and analyzed using a custom-coded data processing pipeline implemented in Python. The results of the analysis were utilized to generate commands that were sent back to the GUI, enabling the system to provide feedback to the user based on the processed data. The LSL played a critical role in connecting the GUI, data processing pipeline, and the Lab Recorder application from the LSL distribution. It facilitated seamless communication and data transfer between these components, ensuring the smooth operation of the system.

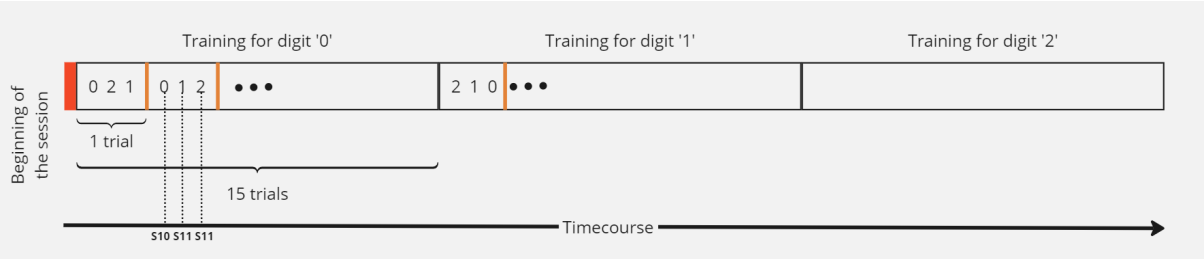
**Figure 2**

*The general schematics of the experimental setup information flows.*



**Training.** During the training session the user was instructed to focus on one of three alternatively flickering digits (0,1, and 2). The session included three tasks corresponding to the number of digits. Each tasks consisted of 15 trials. For each trial 3 different digits flickered in a random order (FIG. 3). The GUI system sent a time-point marker every time when the digit flickered. In case if the digit matched with the task the marker was 'S10' which meant that we were expecting P-300 to occur in brain activity recording otherwise it was 'S11'. Each task aimed to provide the classifier the data for training so at the end of the training session it would have 15 time-stamp markers for 'event' and 30 for 'non-event' (no P-300 expected) per task.

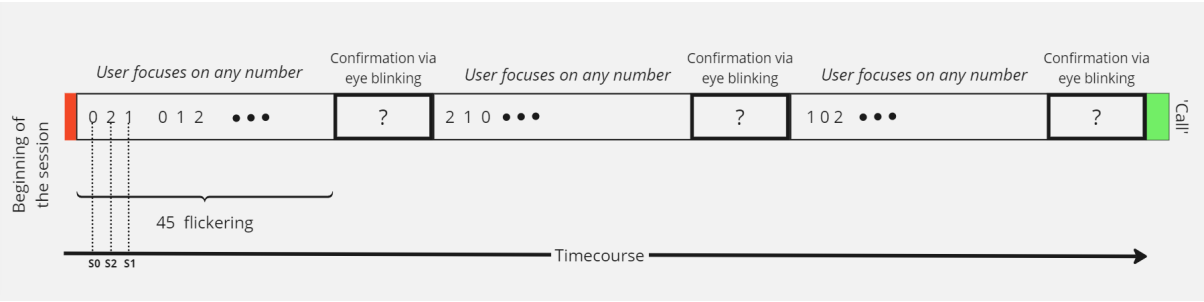
**Figure 3**
*Schematics of the paradigm for the training session.*




**Testing.** During the testing session the user was asked to focus on a digit of his choice (out of the same three as in the training session) after that the series of 15 trials started. The pre-trained classifier guessed the digit of user's choice and the user received a feedback in a form of question on the screen to confirm or deny the correctness of the classifier. In a case of the correct digit user was asked to give a reply in a form of eye blinking. The GUI sent the marker to activate the blinks detector. If the positive reply was detected by the blink detector, the user could see the sequence of already chosen digits and the game switched to the P300 detecting state with flickering digits for the

next digit selection. The game kept going until 3 digits in a row are detected and saved correctly to "execute the call".

**Figure 4**
*Schematics of the paradigm for the testing session. Perfect-case scenario when every digit was correct.*




### *Speller matrix*

For some time, researchers tried different methods to trigger the P300 measurement. None were as successful as Farewell and Donchin in 1988 [Farwell and Donchin, 1988].They came up with a speller matrix that had the 26 letters of the alphabet, as well as several characters and commands. Without going into much depth, the characters were originally organized in rows and columns and the P300 was elicited whenever the specific character which the user was focusing on was highlighted. We were inspired by this work and other papers that have used speller matrices [Brunner et al., 2011] and made our own version. In comparison with past speller matrices that elicit the P300 by highlighting the rows and columns, we decided to follow Jin's approach that places a human's face on top of the characters [Jin et al., 2012]. This method has proved that we as humans feel more comfortable and familiar with other people's faces, which is why our P300 deflections are stronger. In order to compare the performance of the

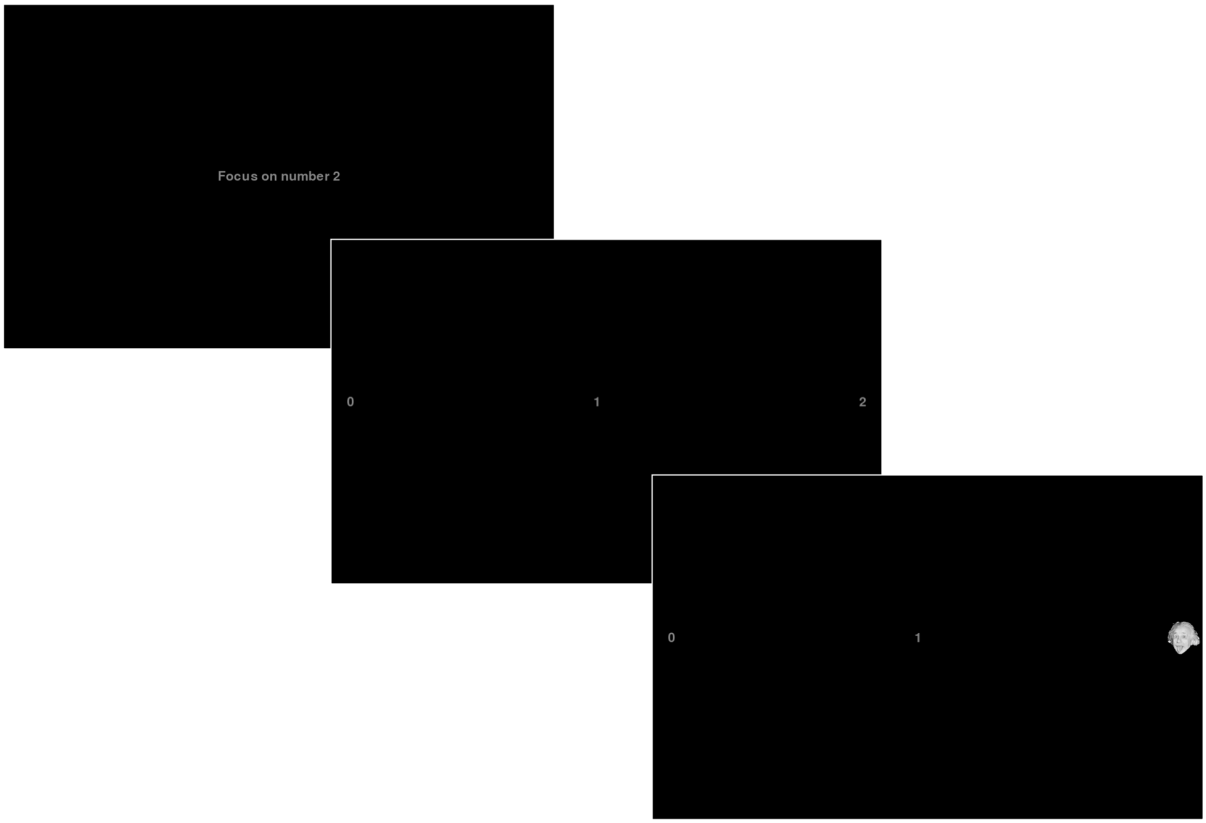
classifier we implemented three variations of the experiment: The selected numbers are highlighted with a brighter color, the face of a famous person is superimposed on the selected numbers, a familiar face (own face) is superimposed on the numbers selected. The following descriptions of the experiment will be based on the case of a famous face superimposed on the selected numbers.

We used Python's Pygame that was initially designed to create video games, to create our experiment. As mentioned before, we also had to create two different experiments. The first is to obtain the offline signals and train the model, whereas the other is used to obtain the data and interact with the machine.

For the former, we initialize our experiment by using several commands and setting the environment with a black full screen. We designed the experiment to have four different states called "game states". Each of these contain the individual sections and were created to have a specific order throughout the whole assessment. The experiment starts with a black screen which remains for five seconds and then displays a text that contains the instructions for the user, asking them to focus on a specific number (FIG. 5). Consequently, it proceeds to the next state of the experiment, where the characters are displayed in a horizontal matter in the middle of the screen for a second and then a familiar face pops-up on top of one of the characters. The iterations starts here, making the face move from one character to the other in a random matter, without it repeating on any of the numbers. The face is shown for 128ms, removed for 128ms and then shown on the next character for the same amount of time as done by Medina [Medina-Juliá et al., 2020]. This loop goes on for fifteen rounds - making the shifting movements reach a total of 45. The order in which the face was shifted from character to character is also saved on an array to be able to confirm the accuracy of our model. Furthermore, we created markers that state which and when a character was truly highlighted

in order to send this through the LSL. Once we are done with this, we jump to the next state that is simply a black screen that lasts for two seconds. Finally, the last state feeds a counter that will keep the experiment running until it reaches the amount of trials needed and takes the assessment back to the initial stage. The order in which the characters are attended to and the order in which the face moves is completely random, the only constant is the order of the displayed numbers, i.e. 0, 1, 2 as seen in Fig (FIG. 5).

**Figure 5**
*Experiment with the speller matrix on the training phase.*




The second experiment is the most interactive and end product. It is very similar to the first assessment, same black full screen and same digits. The image of the face will also pop on character to character but there is some variation. Instead of asking the user to focus on a specific character, this time the instruction will ask the user to focus on

any number. After having the same state in which the face shifts, the experiment goes to a new stage in which it asks the user to confirm if the number coincides with the one that the model classified. The program enters a loop and then waits for confirmation of the user, which in this case is eye blinks. If it receives the marker assigned to the eye blink, then the number is saved on an array. Otherwise, the experiment is repeated. This experiment will run until the array reaches three digits and then makes the call.
Both experiments can be terminated whenever the user wishes it by pressing the escape key.

## Data Processing

### *Preprocessing*

The EEG data used for P-300 detection is filtered with a one-pass, zero-phase, non-causal bandpass filter from two to 16 Hz. All the methods were implemenred with an application of MNE package for Python (https://mne.tools/stable/index.html).
The FIR filter parameters used are the following:

- Windowed time-domain design (firwin) method
- Hamming window with 0.0194 passband ripple and 53 dB stopband attenuation
- Lower passband edge: 2.00
- Lower transition bandwidth: 2.00 Hz (-6 dB cutoff frequency: 1.00 Hz)
- Upper passband edge: 16.00 Hz
- Upper transition bandwidth: 4.00 Hz (-6 dB cutoff frequency: 18.00 Hz)
- Filter length: 825 samples (1.650 s)

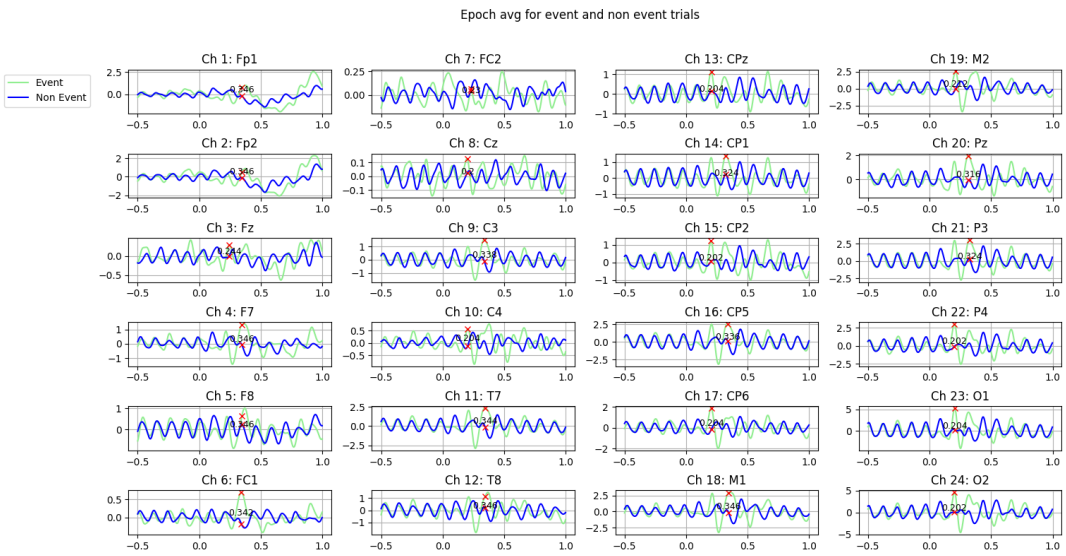
The filtered data was epoched using markers sent by the GUI with a window ms before the time stamp[0.5s <marker<1s] related to the time stamp. This would allow to focus on the interval where P-300 peak is supposed to be detected.

### *Feature extraction and evaluation*

To extract features we first calculated the averaged trials over each task for events and non-events. So we got 30 'event' samples and 30 'non-event' samples. We visualized the results for epochs to analyze which channel are representing the activity including the P-300 peak (Fig. 6) and we focused on the window between 250ms and 350ms after the marker because it is the timing of P-300 appearance. Out of samples we created a 60 pieces feature matrix. Since we had markers from the speller matrix for 'event' (the moment when the highlighted digit was the one that the person was focusing on) and 'non-event' time-stamps we also constructed a vector of labels for two classes.

**Figure 6**

*Average signal over trials with extracted maximum (marked) for two classes of features.*




To evaluate features we used Fisher Score and picked 6 best features based on the rule of thumb of having at least 10 times more observations than features. For visualization we plot the feature score as well as the two best features (FIG.7, FIG.8, and FIG.9).

**Figure 7**

*Highlighted numbers experiment: The resulting distribution of the features contribution correspondingly to the Fisher score (left) and representation of two best features (right).*

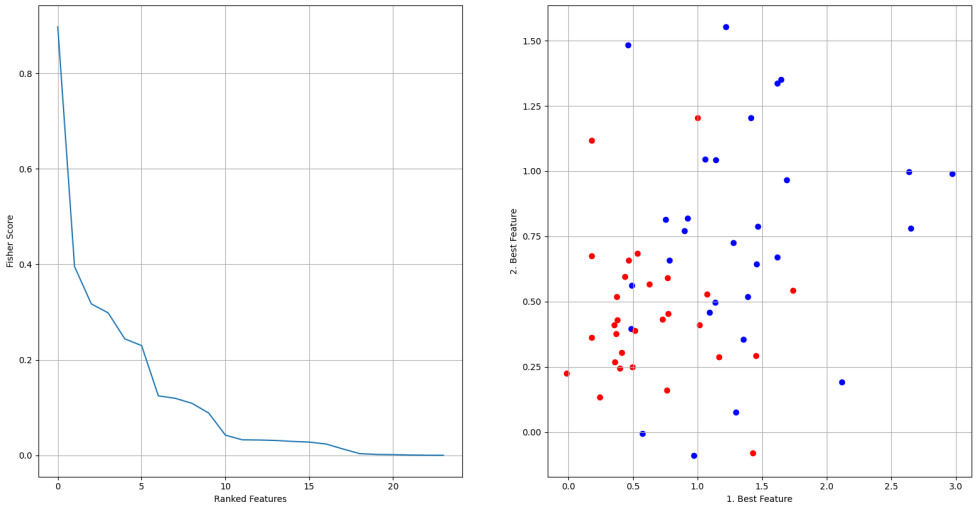



**Figure 8**

*Famous face experiment: The resulting distribution of the features contribution correspondingly to the Fisher score (left) and representation of two best features (right).*

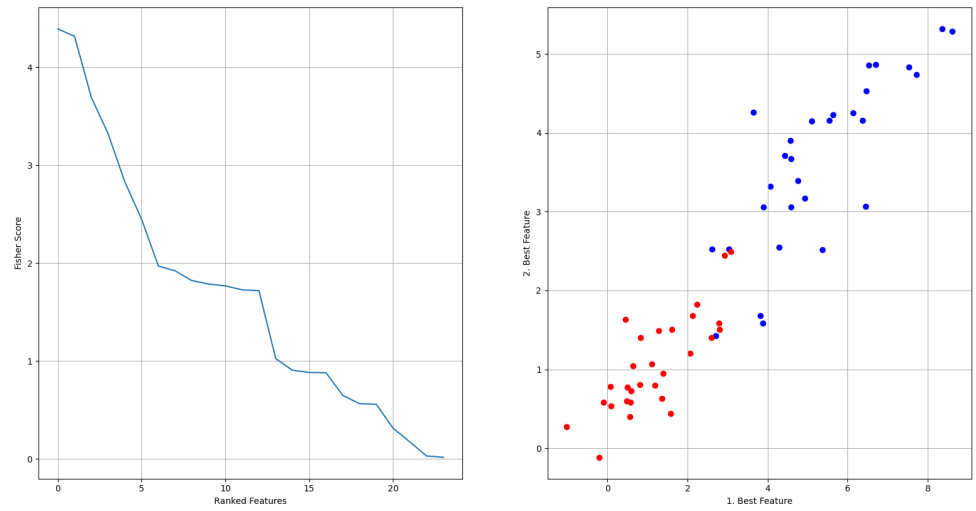
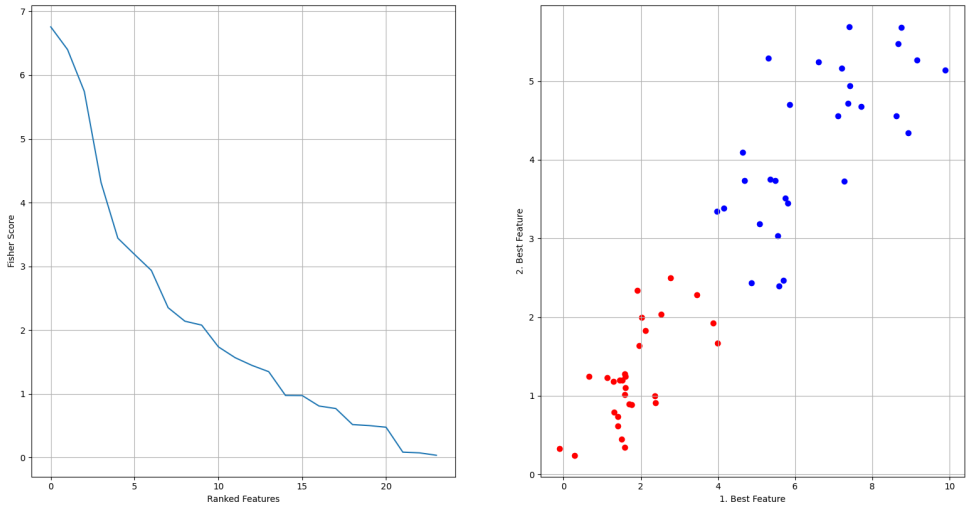


**Figure 9**
*Familiar face experiment: The resulting distribution of the features contribution correspondingly to the Fisher score (left) and representation of two best features (right).*




### *Offline classification and model training*

We recruited two models to train offline on the recorded data. First one was using Linear Discriminant Analysis (LDA) and the second - Support Vector Machine. For the validation we used K-folds method with 10 folds. The accuracy of the classification was 96.667% for LDA so we decided to implement this model for the online classification. In addition, we tested both algorithms with the unseen data, using another recorded dataset.

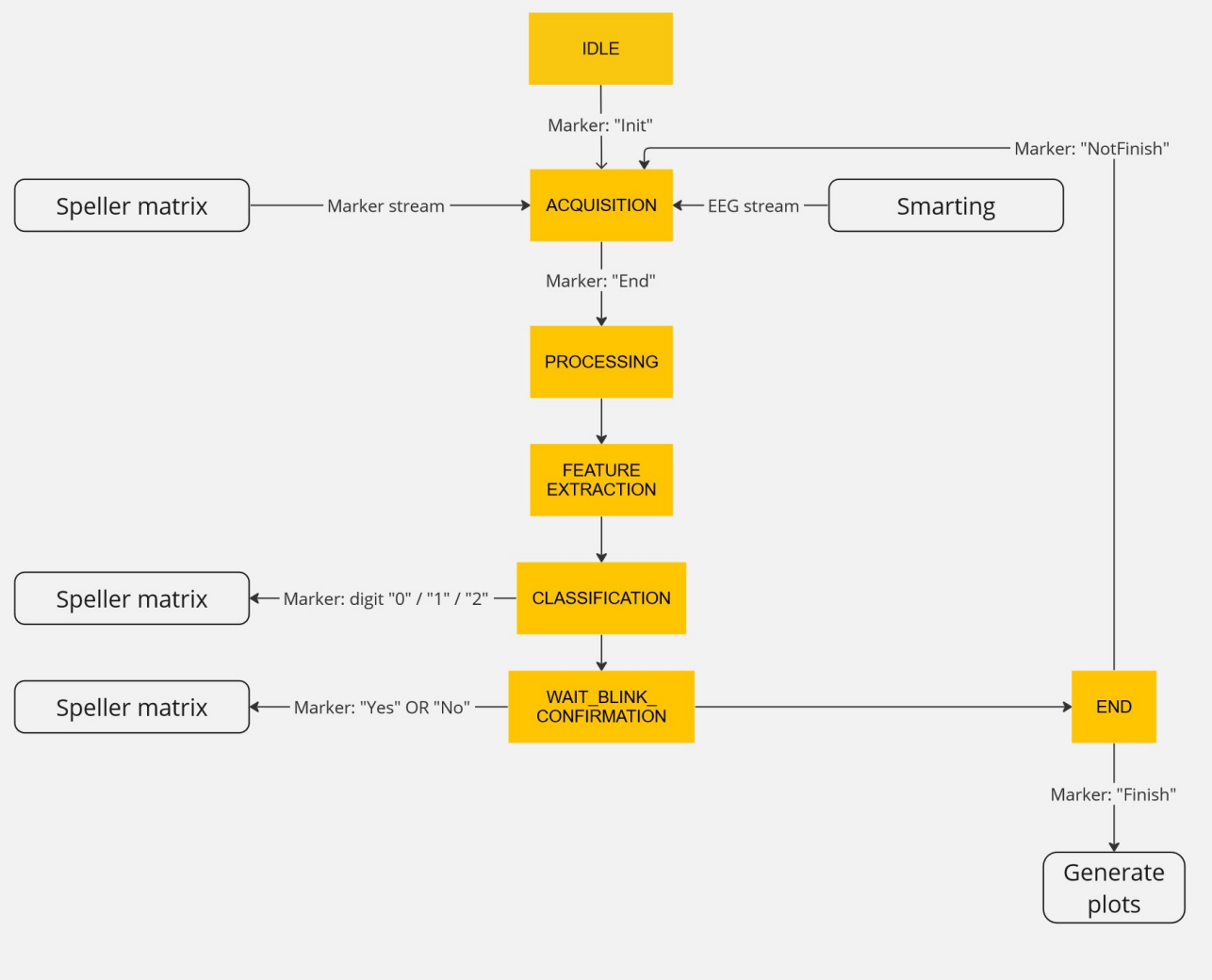
### *Online classification*

As a next step, the models obtained during offline classification and training can be used to perform online classification and thus perform the final task as described in the experimental design section.

**Finite State Machine.** The online classification is divided into different states. The Finite State Machine (FSM) describing these different states is visualized in Figure 8. The yellow boxes correspond to the different states the online classification FSM can be in. The markers on the arrows pointing from one state to another correspond to

**Figure 10**

*Flow chart of the Finite State Machine (FSM) used for online classification. The markers on the arrows pointing from one state to another correspond to markers sent by the speller matrix to switch states in the online classification.*



markers sent by the speller matrix to switch states in the online classification. The steps will be described in more detail in the following:

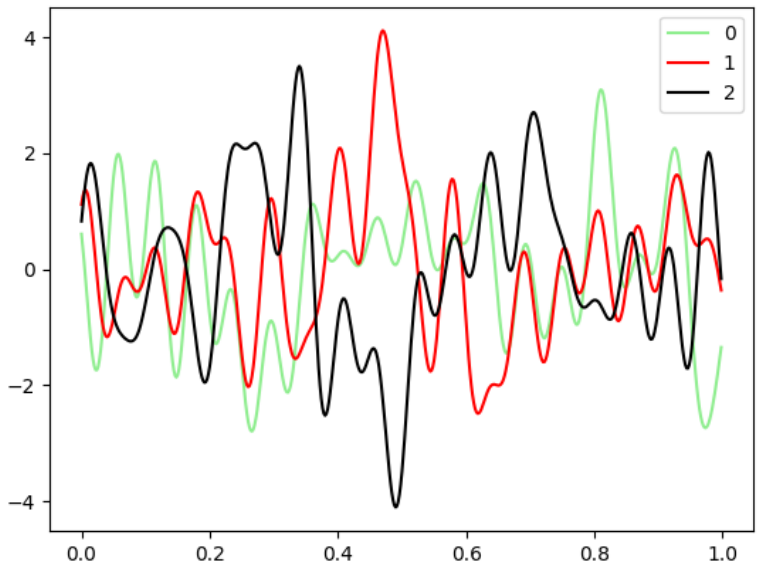
In the IDLE state, the online classification scripts listens to the stream that has been set up with the speller matrix and waits for the "Init" marker that announces that the pygame instance has been initiated and the user now will see the different screens depending on the different game states. As soon as the "Init" has been received, the current EEG stream and marker stream buffers are being emptied to avoid working with old data corresponding to a previous session.

In the ACQUISITION state, the stream to receive the EEG data from the Smarting device is used to save the EEG samples while the speller matrix is active. Also, the marker stream from the speller matrix is used to take into account the timing of the different digits that are flickering.

As soon as the "End" marker that indicates the end of the number selection in the main window has been received from the speller matrix, the program enters the PROCESSING state. First, the EEG data is bandpass filtered between two and 16 Hz. Then, epoching is conducted around the markers "S0", "S1" and "S2" that have been received whenever the respective digit has been highlighted in the speller matrix. E.g. when the digit "0" has been highlighted in the speller matrix, the latter sends a marker "S0" that indicates the timing when the digit "0" was highlighted. During epoching, a time window from 0 seconds (when the event occured) to 1 second after the event is extracted from the EEG data. The marker indices are used to find the EEG time stamp that is closed to when the markers has been received. All the epochs corresponding to "S0", "S1" and "S2" events respectively are assembled to allow treating them alltogether in the next state.

**P-300 detection.** In the FEATURE EXTRACTION state, the epoched data regrouped by digit for "S0", "S1" and "S2" events is averaged across all epochs corresponding to the same digit. An example for the curves obtained during this averaging can be seen in Figure 9. As a next step, the time window that is used to detect the P-300 peak is defined. It reaches from 250ms to 350ms after the time point of the "S0", "S1" or "S2" event marker. The maximum amplitudes of the average of the epochs within this time window are then stored in a feature vector.

**Figure 11**
*Average of epoched data regrouped by digit. The prominent peak in the black curve corresponding to the digit "2" around 300 ms will help to select the digit during classification.*




The FSM then switches to the CLASSIFICATION state in which the best features as determined during offline classification are being kept for further processing.The predictions ("-1" NonEvent and "1" Event) are determined with the help of the model obtained during training. The corresponding prediction probabilities are also determined and stored as ["probability of NonEvent/-1", "probability of Event/1"]. If the classifier returns "1" for more than one digit, the digit where the label "1" has the highest prediction

probability is selected. The selected digit is then pushed to the marker stream to the speller matrix.

**Eye blinking detection.** When the *WAIT_BLINK_CONFIRMATION* state is entered, the program waits to receive the "Calibration" marker from the speller matrix that indicates that the screen has switched from the speller matrix to the screens used to ask the user if the detected digit is indeed the one the user wanted to select. As soon as a window showing "Waiting for calibration" is shown, the *eye_blink_detection* function is called. The reason for this calibration time is that the *eye_blink_detection* function works with real time EEG data and needs a certain amount of samples in its buffers to be fully working. During the calibration time these buffers are thus filled with EEG samples.

The eye blink detection works as described in Tutorial 2 for the EEG devices. The incoming EEG samples in the buffer are first bandpass filtered between 1 and 40 Hz and then smoothened with a moving average filter. With the help of a baseline window, the threshold for eye blinking is updated constantly so as to avoid any influence of random signal fluctuations or a change in the overall signal amplitude over time. As eye blinks appear as prominent peaks in the EEG signal, this threshold is then used to indicate an eye blink by comparison with the EEG signal values in the activity window.

Once an eye blink has been detected, the function enters a refractory period during which the eye blink peak moves out of the activity window. Thus, a single eye blink is not detected several times but only once.

If the user does not blink for 10 seconds, the FSM sends a "No" marker to the speller matrix which will then ask the user to reselect a digit as apparently the detection was not successful. However, as soon as the user blinks, before the 10 seconds during which blinks are being detected are over, a "Yes" marker is sent to the speller matrix.

The successfully selected digit is then stored in the currently selected digit sequence. If the classifier does not detect at least one P300 then marker 4 is sent to tell the speller matrix that no digit has been detected and the user is asked to try again, in this case it is not required to go to the blink detection state

If three digits have already been selected, the speller matrix sends a "Finish" marker to the FSM which is then exited and optionally displays plots relevant for the last data acquisition. Otherwise, the speller matrix sends a "NotFinish" marker which makes the FSM switch again to the ACQUISITION state.

## Results and Discussion

The LDA classification worked smoothly and accuracy was 96.667% for test and 98.433% for training using the familiar face experiment variant. In comparison, the highlighted numbers experiment variant had an accuracy of 66,667% for test and 77.17% for training and the famous face experiment variant had an accuracy of 95% for test and 93.667% for training. As the final outcome we performed a full testing session with a timing from the first question to the "call" 1 min and 49 seconds. Even though the numbers is quite far from some of the results of our colleges (40-60 characters per minute Chen et al., 2015), we believe it is reasonable timing for our goal.

This shows that the building of the model with the offline classification as well as the online classification work well and are built in a consistent way. Ambiguities in the selection of digits (e.g. "0" and "1" were both predicted to have been selected) were successfully solved using the prediction probabilities. Furthermore, we demonstrate that the use of a familiar face elicits stronger P300 signals that eventually improve classifier performance.

A major challenge in setting up the offline classification was that in order to merge events and EEG data into raw data that can be used by the mne library, first an mne object had to be created. However, it was observed that the mne library does not align the event markers with the EEG time stamps. Another limitation of the mne library is that the mne object time member is just readable and not writable which is needed for epoching. Therefore, we implemented our own epoching function.

In the process of choosing the classifier we got the better result for the SVM in comparison to LDA model. However, to make the comparison between the three types of experiments (highlighted numbers, famous face and familiar face) we chose the LDA model since it is the one that presented the best performance for the highlighted numbers experiment.

One of multiple minor technical inconveniences was the issue with the software for Bluetooth connection between the PC and Smarting24. The connection between devices was time consuming to set. Although, during the experiment the system worked stable.

## Conclusion

This report outlines the successful development of a Brain-Computer Interface (BCI) system for user interaction with a speller matrix. Through the utilization of Electroencephalography (EEG) and the detection of P300 events with an original speller matrix, we achieved an innovative approach to facilitate communication and control. By employing a probability prediction machine learning method, we were able to accurately detect and interpret the user's intent from the recorded EEG signals. This enabled the seamless interaction between the user and the speller matrix, empowering individuals with limited motor control or communication abilities. The BCI system demonstrated remarkable accuracy and reliability in detecting P300 events, enabling effective and

efficient communication. The implementation of the online classification algorithm provided real-time predictions with high confidence levels, enhancing the user experience and reducing communication latency. Furthermore, our comprehensive analysis and evaluation of the system's performance yielded promising results. The accuracy of the probability prediction method surpassed previous approaches, indicating the efficacy of our chosen methodology. We also conducted user testing and obtained positive feedback, highlighting the system's usability and potential for practical application. Overall, this project showcases the successful integration of EEG, P300 event detection, and machine learning techniques in developing a robust BCI system. The system's accuracy, real-time performance, and user-friendliness open up new possibilities for individuals with communication impairments, offering them greater independence and improved quality of life.

## Bibliography

Brunner, P., Ritaccio, A. L., Emrich, J. F., Bischof, H., & Schalk, G. (2011). Rapid communication with a "p300" matrix speller using electrocorticographic signals (ecog). *Frontiers in neuroscience*, *5*, 5.

Chen, X., Wang, Y., Nakanishi, M., Gao, X., Jung, T.-P., & Gao, S. (2015). High-speed spelling with a noninvasive brain–computer interface. *Proceedings of the National Academy of Sciences*, *112*(44), E6058–E6067. https://doi.org/10.1073/pnas.1508080112

Farwell, L. A., & Donchin, E. (1988). Talking off the top of your head: Toward a mental prosthesis utilizing event-related brain potentials. *Electroencephalography and clinical Neurophysiology*, *70*(6), 510–523.

Jin, J., Allison, B. Z., Kaufmann, T., Kübler, A., Zhang, Y., Wang, X., & Cichocki, A. (2012). The changing face of p300 bcis: A comparison of stimulus changes in a p300 bci involving faces, emotion, and movement. *PloS one*, *7*(11), e49688.

Medina-Juliá, M. T., Fernández-Rodrıguez, Á., Velasco-Álvarez, F., & Ron-Angevin, R. (2020). P300-based brain-computer interface speller: Usability evaluation of three speller sizes by severely motor-disabled patients. *Frontiers in Human Neuroscience*, *14*, 583358.

Minguillon, J., Lopez-Gordo, M. A., & Pelayo, F. (2017). Trends in eeg-bci for daily-life: Requirements for artifact removal. *Biomedical Signal Processing and Control*, *31*, 407–418.

Turnip, A., & Hong, K.-S. (2012). Classifying mental activities from eeg-p300 signals using adaptive neural network. *Int. J. Innov. Comp. Inf. Control*, *8*(7), 5839–5850.