

Technical University of Cluj-Napoca
Faculty of Electronics, Telecommunications, and Information Technology

MATLAB Project

Computer Aided Graphics

Theme: Emotions Detector Device via
Electroencephalography

Student:
Alexa Alexandru Andrei

Coordinators:
Cîrlugea Mihaela Cristina
Farago Paul

Introduction	3
Short history of MATLAB	3
Foundation and History.....	3
Evolution over time.....	4
The Programming Language	4
Short History of Electroencephalography	5
Foundation and History.....	5
The Technology	6
Evolution over time.....	7
Uses of the technology	8
NeuroSky	9
The company.....	9
The Headset	9
The Software	11
The place of this project in the field of EEG devices.....	13
Theoretical Presentation.....	14
EEG Device Structure	14
A complex way to collect the EEG data.....	14
Machine learning algorithm for data analysis.....	15
EEG Signals	15
Theme and Implementation	17
My implementation in conjunction with the model.....	17
The signal given by the headset.....	18
Recording of the data set.....	18
Emotions and feelings induced through videos.....	18
The dataset	19
The emotion calculation algorithm.....	21
Theoretical results.....	22
The Raw EEG Analyzer.....	23
Experimental Presentation.....	23
The Raw Data Analysis Menu	23
Menu with no data.....	24

Initial values	25
Derivative	26
Integral	27
Fourier Transform.....	28
The Emotion handling menu.....	29
Practical results	29
Recording of the EEG data – Tutorial	31
Significant code parts	32
Derivative computation.....	32
Integral computation.....	32
Fourier Transform computation	33
Emotion algorithm	34
Conclusions	35
Own thoughts	35
Future improvements	35
Bibliography	36
The whole program code	37
Others.....	58

Introduction

Short history of MATLAB

MATLAB is the tool, developed by MathWorks, that engineers use nowadays to create, design, and improve the things that surround us, in our everyday lives. It provides a more user-friendly interface and a much simpler code syntax to work with, it can compute anything from signal analysis to image processing and network systems. And those are just some of the applications of the MATLAB programming language, in the following part of the project more will be presented.

Foundation and History

The modern full-featured technical computing environment MATLAB, which stands for "Matrix Laboratory," started just as a simple interactive matrix calculator, written in Fortran, built on top of a considerable set of subroutines from the LINPACK and EISPACK matrix software libraries. Three men, J. H. Wilkinson, George Forsythe, and John Todd had an important role in the origins of the program. With only 71 words and built-in functions, they achieved a functional matrix calculator, to perform more the Fortran source code had to be recompiled. The mathematical basis of the first version was a series of research papers by J. H. Wilkinson and 18 of his colleagues, published around late 1960s and later collected in the handbook for Automatic Computation, Volume II, Linear Algebra, edited by Wilkinson and C. Reinsch. These papers included algorithms that could solve eigenvalue problems and matrix linear equations.

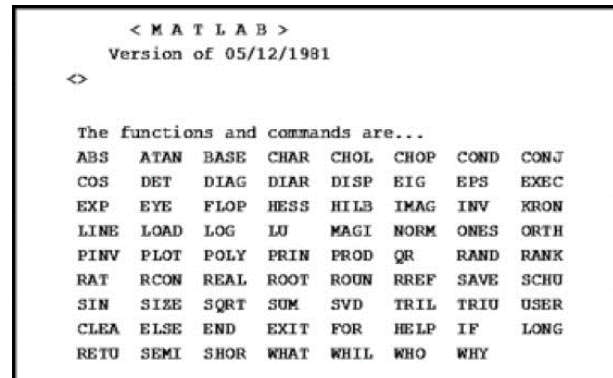


Figure 1 - First MATLAB set of commands.

In 1984, MATLAB was first released at the Automatic Control Conference in Las Vegas and became a commercial product and transformed into a full-fledged programming language. The first MATLAB sale was done in the following year, when Nick Trefethen from MIT bought ten copies of it. The calculator was reimplemented by using the C programming language, and it underwent significant enhancements such as the introduction of the user functions, toolboxes, and graphics. Initially, it was available on IBM PC and the other PCs that had the physical requirements for it, followed by versions for Unix workstations and then by the Apple Macintosh.

Evolution over time

In addition to the matrix functions present in the original calculator, in the version from the year 1984 there was introduced the Fast Fourier Transforms function (FFT). Subsequently, the Control System Toolbox was introduced in 1985, which was followed by the Signal Processing Toolbox in 1987. The ability to numerically solve ordinary differential equations was also added in 1987, by adding the ODE MATLAB solver. ODEs are also the core of Simulink®. In 1992, MATLAB introduced a significant new data structure called the sparse matrix. The Image Processing Toolbox and the Symbolic Math Toolbox were both introduced in 1993. In the late 1990s, MATLAB introduced several new data types and structures, including single precision floating point, various integer and logical types, cell arrays, structures, and objects. In the recent years there have been a lot of significant advancements in the MATLAB computing environment, with enhancements to the desktop, improvements to the object and graphics systems, support for parallel computing and GPUs, and the introduction of the "Live Editor," which allows for the integration of programs, descriptive text, output, and graphics into a single interactive and formatted document. Today, MATLAB offers over 60 Toolboxes, many of which are programmed in the MATLAB language, providing extended capabilities in various specialized technical fields, such as: electronics, telecommunications, medicine, computing and so on.

The last version of MATLAB at the current time is: R2023b.

The Programming Language

The Programming Language MATLAB is a **High-Level** Programming Language, that provides a user-friendly syntax, that can lead to the implementation of more complex programs without having the programmer to deal with all the memory allocation and other bitwise operations. The IDE (integrated development environment) is a technical and interactive environment in which it is possible to perform actions such as: Data Analysis and visualization of that data, Algorithm Development and Numerical Computations. The time spent by writing algorithms and code is much shorter with the MATLAB programming language than with other programming languages like C, C++, and Fortran, in which a time-consuming part of the code development is only managing the memory, and this is due to the high-level nature of it.

It can perform a significant number of operations just like numerical computation (linear algebra, statistics, filtering, numerical derivatives, numerical integrations, Fourier analysis), Communication Systems, Control Design, Computational Biology, and virtually any engineering domain. Also, Graphical user interfaces (GUI) are easily manageable along with plotting of 2D and 3D data. Example: `disp('Hello, world!')` It displays like so: Hello, world!

Short History of Electroencephalography

Electroencephalography (EEG) is a technique used to capture the spontaneous electrical activity of the brain by recording an electrogram. This method, known as "scalp EEG," is generally non-invasive as it involves placing electrodes along the scalp of the patient. In contrast, electrocorticography, which requires surgical placement of electrodes, is referred to as "intracranial EEG." The clinical interpretation of EEG recordings is commonly conducted through visual examination of the tracing or through quantitative EEG analysis. It offers a method to investigate the functioning of the brain and to map the interconnections between different regions of the central nervous system. Nevertheless, its utility as a research instrument is constrained due to its ability to capture only a fraction of the brain's electrical activity from its surface. Numerous intricate cognitive processes, including emotions and thoughts, cannot be directly correlated with EEG patterns, but with a big enough number of people that are analyzed some approximations can be drawn. Moreover, EEG is not a valuable tool for diagnosing psychiatric disorders, due to the nature of the "random" pattern of the EEG waves in correlation with the actual disorder.

Foundation and History

Richard Caton, an English scientist who lived from 1842 to 1926, is widely recognized for his significant contribution of the understanding of the brain's electrical properties. He was the first person to achieve this by using a sensitive galvanometer to record electrical activity from the brains of animals. Through his observations, Caton was able to identify fluctuations in brain activity during sleep and the absence of any activity when the animals passed away. Hans Berger, a German psychiatrist, in 1924 was able to perform the first recording on a human being, and the term of electroencephalograms (EEGs), began to get recognition. This was a breakthrough in the field of neuroscience, as it allowed for the direct measurement and analysis of electrical brain activity in humans. In 1934, Fisher and Lowenback were the first to demonstrate epileptiform spikes, which are abnormal electrical discharges associated with epilepsy. In the following year, Gibbs, Davis, and Lennox described interictal epileptiform discharges and 3-Hz spike-wave patterns during clinical seizures.

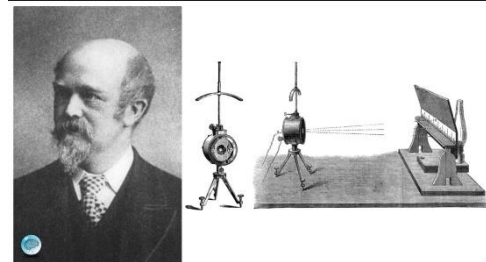


Figure 2 - Richard Caton: the first EEG

In 1936, Gibbs and Jasper made another important discovery by describing focal interictal spikes. The first clinical EEG laboratories in the United States were founded during the 1930s and 1940s, and they provided researchers with the necessary infrastructure and resources to conduct further studies on brain electrical activity. In 1947, the American EEG Society, later known as the American Clinical Neurophysiology Society, was founded. Overall, the contributions of Richard Caton, Hans Berger, and subsequent researchers have paved the way for our current understanding of the electrical properties of the brain.

The Technology

An EEG signal is a measurement of the currents that flow during synaptic excitations of the dendrites of many pyramidal neurons in the cerebral cortex. When brain cells (neurons) are activated, synaptic currents are produced within the dendrites. This current generates a magnetic field measurable by electromyogram (EMG) machines and a secondary electrical field over the scalp measurable by EEG systems. The current in the brain is mostly generated by pumping the positive ions of sodium, Na^+ , potassium, K^+ , calcium, Ca^{++} , and the negative ion of chlorine, Cl^- , through the neuron membranes in the direction governed by the membrane potential. The human head is composed of various layers, such as the scalp, skull, brain, and several other delicate layers in between. The skull can attenuate the brain signals by approximately one hundred times more than the soft tissue. Conversely, most of the noise is produced either within the brain itself (internal noise) or on the surface of the scalp (system noise or external noise).

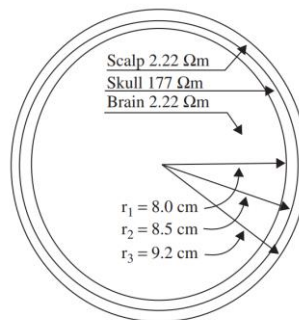


Figure 3 - The layers of the brain including their approximate resistivities and thicknesses.

More recent EEG systems consist of several delicate electrodes, a set of differential amplifiers, one for each channel, followed by filters. To obtain a higher precision and an easier method to compute the data, the need of computer software was needed, and to obtain that the signal must be converted. The process of converting the analogue EEG signals to digital is achieved using multichannel analogue-to-digital converters (ADCs). Fortunately, the effective bandwidth of EEG signals is typically restricted to around 100 Hz, and in many cases, this bandwidth can be even half of that value. As a result, a minimum sampling frequency of 200 samples/s is usually sufficient, to meet the Nyquist criterion, for sampling the EEG signals. Yet, in certain applications where a more detailed representation of brain activities in the frequency domain is required, sampling frequencies of up to 2000 samples/s may be used. Representation of each signal sample with up to 16 bits is very popular for the EEG recording systems.

The amplitudes of the raw EEG signals are in the range of μvolts , and their frequency components can have a bandwidth up to 300 Hz. To preserve the important information, it is necessary to amplify the signals and apply filtering either before or after the ADC, in order to reduce the noise and ensure the signals are suitable for processing. The filters must be designed in such a way to avoid introducing any changes or harmonic distortions to the EEG signals. To eliminate very low frequency components of the signal that are not relevant, like breathing, high

pass filters with a cut-off frequency usually smaller than 0.5 Hz are used. On the other hand, lowpass filters with a cut-off frequency of around 50-70 Hz are used to get rid of the high frequency distortions that may occur. The commonly used sampling frequencies for EEG recordings signals are 100, 250, 500, 1000, and 2000 samples/s. Noise and errors generated by the human body can include body movement and sweating. Also, noise and errors generated by the system include interference from the 50/60 Hz power supply, electrical noise from electronic components, impedance fluctuation, cable defects, and unbalanced impedances of the electrodes.

Evolution over time

The initial electrical neural signals were recorded by utilizing a precise galvanometer. To enhance the precision of the pointer tip of the device, a mirror was placed to reflect the light projected onto the galvanometer on the wall, and by that improving the ease of reading. The d'Arsonval galvanometer later incorporated a movable coil with a mirror attached, and the light directed at the mirror would reflect when an electric current passed through the coil. Lippmann and Marey introduced the capillary electrometer, while Einthoven introduced the string galvanometer in 1903, which proved to be a highly sensitive and accurate measuring device. As the years passed, the technology used for EEG changed as well. The EEG systems, in the early 2000s, were composed of multiple sensitive electrodes, individual differential amplifiers for each channel, filters, and needle-type registers. Nowadays in the era of digitalization all that has changed is the digital sampling of the data set and the computer-based computations and plots.

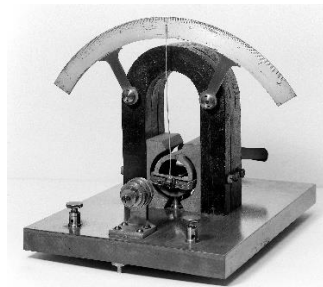


Figure 4 - Precision galvanometer

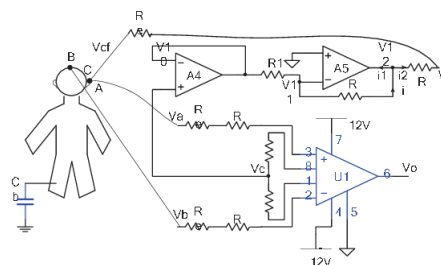


Figure 5 - The modern way to measure EEG via Electrodes and OpAmps

Uses of the technology

An electroencephalogram is a convenient tool in the diagnosis and monitoring of various brain-related conditions and diseases. It can help in identifying symptoms like seizures or memory issues. The main purpose of an EEG is to detect and investigate epilepsy, a condition characterized by recurrent seizures, followed by loss of consciousness and convulsions. By analyzing the EEG results, doctors can determine the specific type of epilepsy, identify potential triggers for seizures, and develop an optimal treatment plan. In addition to epilepsy, an EEG may also be utilized to investigate other conditions such as head injuries, dementia, brain death, encephalitis, brain tumors, and sleep disorders like sleep apnea.

There are several types of EEG recordings that serve different purposes. The routine EEG is a standard procedure that typically lasts for 20 to 40 minutes. During this test, the patient must rest quietly, open or close their eyes intermittently (for the Alpha waves measurement) and perform deep breathing exercises. For more specific information or to diagnose sleep disorders, a sleep EEG or sleep-deprived EEG diagnosis may be conducted. The sleep EEG is performed while the patient is asleep. The sleep-deprived EEG test requires the patient to stay awake the night before to ensure that they can fall asleep during the time procedure. Another type is the ambulatory EEG, which involves recording brain activity continuously over a period of one or more days. The electrodes are attached to a portable EEG recorder that can be clipped onto the patient's clothing for convenience and commodity. They can carry out their normal daily activities without getting the equipment wet. Video telemetry, also known as video EEG, is a specialized type of EEG where the patient is filmed while the EEG recording takes place. This method provides additional external information about brain activity. Lastly, invasive EEG-telemetry is a less common procedure used to determine if surgery is a viable option for individuals with complex epilepsy. It involves surgically placing electrodes directly on the brain to pinpoint the origin of seizures.



Figure 6 - Routine EEG Test

NeuroSky

The company

Founded in 2004 in Silicon Valley, California by Stanley Yang, JongJin Lim and KooHyoung Lee; NeuroSky, Inc. is a company specializing in brain-computer interface (BCI) technologies for consumer product applications. With a variety of field that is involved in, just as in: entertainment, education, health, and automotive industries. The company has successfully adapted medical electroencephalography (EEG) and electromyography (EMG) technology to fit a wider range of consumers. By utilizing inexpensive and dry sensors and incorporating built-in electrical "noise" reduction software and hardware, NeuroSky enables low-cost EEG-linked research and product development by every passionate developer or student. Operating primarily as an original equipment manufacturer (OEM), the company collaborates with third party industry partners to develop, and research ways into making their technology compatible with a bigger number of usual consumer devices.

The Headset

NeuroSky provides a wide range of headsets and hardware solutions for the consumer market by itself on in collaboration with other third-party companies. Among the headsets, they also have on sale dry electrodes that can measure brainwaves from a distance of a few millimeters away from the scalp thus it makes it possible to wear the headsets over the hair, a breakthrough in technology. Their own headsets are: MindWave, Non-Contact Sensors, Pre MindKit, Mind Kit & Mind Set; and the ones made in junction with partners: Mindflex, MindRDR, Star Wars Force Trainer (I and II).

All the above products use the so called NeuroSky EEG chip that collects and process the data collected from the brain of the user. This electronic device performs all the computations and signal preprocessing described in an earlier chapter. It gets the raw voltages values (μV) read from the electrodes placed on the scalp of the user, then it amplifies them using precision operational amplifier, with a high common rejection mode, to reduce the noise from the environment and the headset itself. The amplified signal goes into a series of filters, of type low pass, high pass, band reject, in order to assure a better isolation of each EEG signal (alpha, beta, delta, gamma and theta), and better noise reduction. All the analog values of the data must be converted to a digital signal so the microprocessor will be able to analyze them; to obtain this each signal channel goes into an individual ADC (analog to digital converter) to sample it. After all the preprocessing of the data, the microcontroller is able to connect to a computer program with the intention of better signal analysis and powerful processing power. Loading the values into a computer ensures a bigger number of experiments that can be done.



Figure 7 - The NeuroSky EEG Chip TGAT1/TGAM1

Data Sheet:

Features:

- One EEG channel + Reference + Ground
- Extremely low-level signal detection
- Advanced filter with high noise immunity
- RAW EEG at 512Hz
- Extremely low-level signal detection
- Direct connect to dry electrode

Data Outputs:

- RAW EEG Signal
- Attention
- Meditation
- Delta, Theta, low alpha, high alpha, low beta, high beta, and gamma waves
- Additional algorithms available in SDK

Dimensions:

- TGAM1 Module: 27.9 x 15.2 x 2.5mm
- TGAT1 Chip: 9 x 9 x 1.6mm
- Weight (Max) 130mg

Specifications:

- 512Hz sampling rate at 12 bits
- 3-100Hz frequency range
- ESD Protection: 4kV Contact Discharge; 8kV Air.
- Max Power Consumption: 15mA @ 3.3V
- Operating voltage 2.97 ~3.63V
- UART (Serial): 1200, 9600, 57600 baud
- 8-bits / No parity 1 stop bit

The headset that has been used for this project is the Star Wars Force Trainer II, that was released on August 1, 2016. It is designed as a toy that can control some holograms by the attention data. It gets connected to other devices via Bluetooth 4.0 and has 3 electrodes that get in contact with the scalp, the forehead and behind each ear. It has a power button, a button for connection and an LED to ensure the user whether the device is connected or not, and the battery level.



Figure 8 - The Force Trainer II and how it should be worn.

The Software

Not only the hardware solutions are available from the NeuroSky library but also a significant number of software solutions, which include not only test code but a lot of algorithms that can perform from the raw reading of the signals to detection of blinks, attention, and meditation detection. It has a variety of programs ranging from some simple games such as Float One and Infinite Brain Runner, to educational apps as: Effective Learner with Study Trainer and Math Trainer, to wellness apps as: Art of Zen and Beauty Within. Those apps are available on all platforms, PC / MAC / iOS / Android. The development of other apps and games is sincerely possible by using the provided Development Tools, the programming languages that is needed to code the headset are C, C++, C#, Python.

ThinkGear Connector is the app and the only connection between any of the above headsets and an actual computer. Narosky's claims the ThinkGear technology has been tested at 96% as accurate as that within research grade EEGs. The ThinkGear Connector (TGC) is a

program that functions as a service to handle communication with ThinkGear devices, such as the Star Wars Force Trainer II, which is linked to the computer. The TGC operates persistently in the background, maintaining an active socket on the user's computer, enabling applications to establish a connection and obtain information from the connected ThinkGear devices.

Consequently, any application, regardless of the programming language used, that is capable of opening and reading from sockets can establish a connection and retrieve data from a headset.

The app can provide the following measurements: eSense Attention value, eSense, Meditation value, poor signal quality, EEG band powers (delta, theta, alpha, beta, gamma), raw EEG wave samples (at 512Hz), ensure the real time visualization of them. It can save the values of the waves into a .csv file for further analysis in programs such as MATLAB.

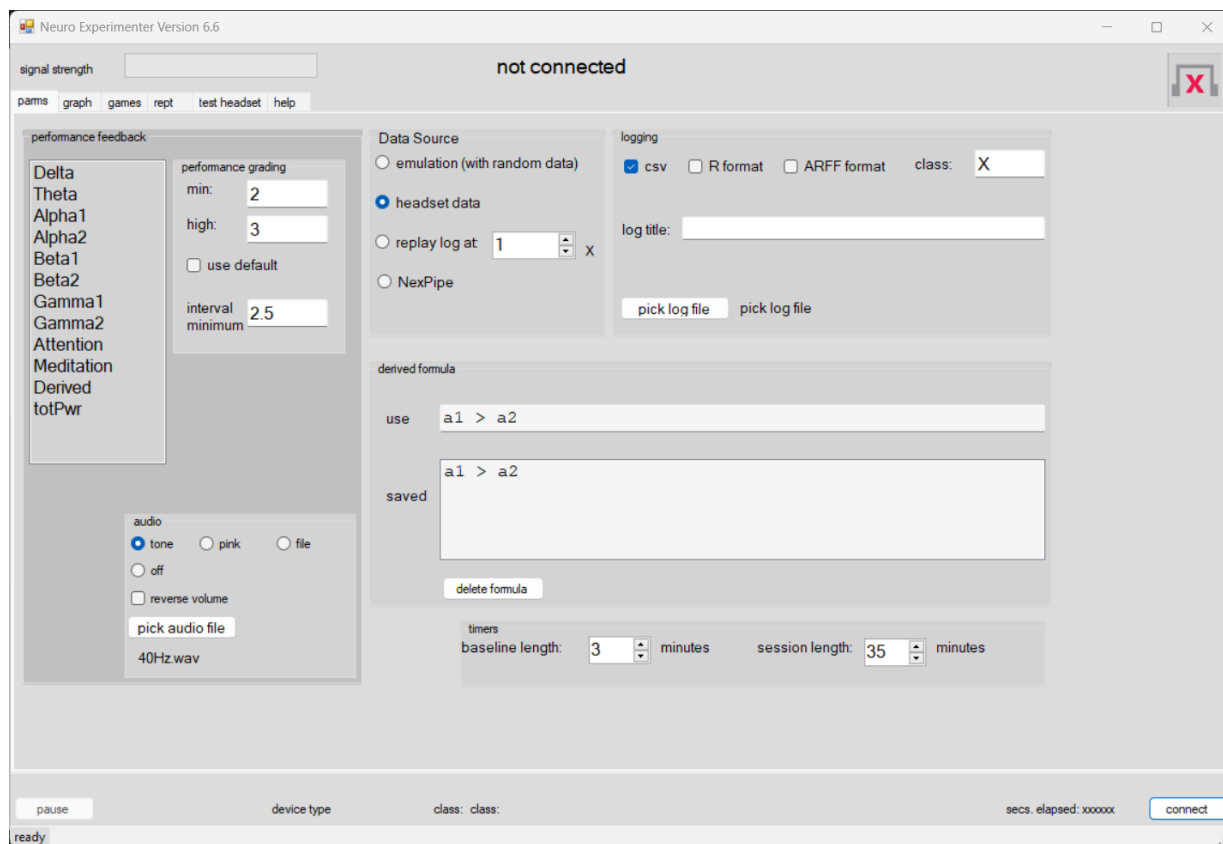


Figure 9 - The ThinkGear Connector

The place of this project in the field of EEG devices

The field of EEG devices is great populated with medical equipment and devices with the scope of helping people that have brain related diseases, as epilepsy, but beside those equipment's, there are also devices that try to find the other aspects of the human brain, as, emotions, feelings, and memories. Regardless of the application, the main point is the reading of the brain activity, and the interpretations of the readings.

The medical side of the projects in the field of Electroencephalography are designed to operate and analyze the patient in a controlled manner, more like laboratory conditions, to find the physical defects of the brain's structures. The mounting of the electric pads is done by a trained technician and the readings are interpreted by a doctor, who can prescribe the right treatment to the patient. All the steps in the procedure are done carefully, paying attention to every detail.

The feelings side of the projects in the field of Electroencephalography are designed to interpret the values of the EEG signal in ratio with the emotions and feelings they might mean. This side of the EEG field considers more of the unpredictability and the nature of humans more than the nature of the brain itself, since feelings are personal and different people might have different responses to the same stimulus, based on their past experiences. These kinds of projects are more present in universities, where they have people with a better understanding of programing and computing, and a bigger number of volunteers. And by that they can approximate the emotions values based on a large group with similar EEG values generated from the same stimulus.

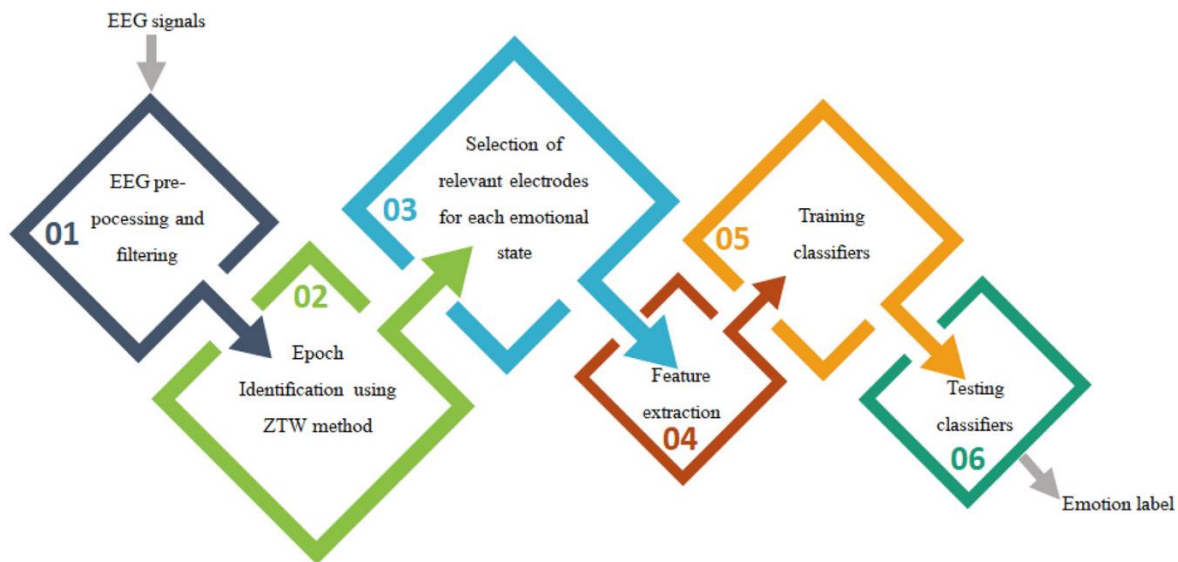
The presented project falls in the second category of the EEG projects, by trying to find the emotions of a person via their EEG signals. It's also an educational project, by being simple enough to be able to make it without machine learning algorithms but difficult enough to push me out of my comfort zone and to make learn about a new field. Its performance is at a medium percentage in order of accuracy, speed, and efficiency. It is a great way for beginners to enter the EEG field and to start learning about it and to see how their brain works.

The intention of this project is to make people aware of the way their emotions have similarities with others and the differences between them. There is no need for very precise equipment to do small college experiments and to draw scientific conclusions, but of course the more precise the better.

As a conclusion, of the role the project has in the entire field, it is a great way to enter the field of Electroencephalography by learning and playing with it.

Theoretical Presentation

EEG Device Structure



The model that I tried to follow was the one found in the “Emotion detection using electroencephalography signals and a zero-time windowing-based epoch estimation and relevant electrode identification” scientific paper. Using the hardware and the software that I had on hand.

A complex way to collect the EEG data

The method uses a set of 32 electrodes that are places around the scalp. EEG signals provide a representation of the brain's electrical activity over a certain period, offering valuable insights into the control of the entire human brain. In this study, it is proposed a new approach for extracting valuable data from EEG sources by considering the individual's brain state. This method involves identifying specific epochs within the EEG signals with a great precision and analyzing the variations in brain activity across selected electrodes, after filtering and choosing the ones with the best quality. By choosing only the EEG channels that output the most significant changes during emotional states, it is possible to get a furthermore precise output state. It is important to note that these electrodes differ from person to person, depending on their mental state. Each subject possesses a unique set of electrodes chosen from the four EEG rhythms: alpha, beta, delta, and gamma. These identified sources are then utilized to calculate the features necessary for the recognition and classification of nine distinct emotions.

Machine learning algorithm for data analysis

In this study, using QDC (Quadratic Discriminant Classifier), the problem was addressed in two parts. In the first part, the aim is to predict the emotional state to which a given trial could belong in. To obtain this, the development of an eight QDC based classifier was necessary, each one corresponding to a specific emotional state. These classifiers were trained using a set of trials representing the target emotional state, along with some outlier trials from other emotions. By analyzing changes in brain activity in these electrodes across different frequency bands, each classifier stated if it was in a proper emotion state or not. In the second part, the objective is to determine the specific emotional state that the trial most likely belonged to. To achieve this, a majority voting algorithm was implemented. If the majority of the classifiers voted for the target in part one, it indicated an error. On the other hand, if the majority of the classifiers voted for the outlier in phase one, we concluded that the trial corresponded to the target of the classifier with the highest accuracy rate during the training phase among all classifiers. The selection of EEG electrodes using an adaptive method results in varying sizes of feature vectors for different individuals. Consequently, employing an RNN (Recurrent Neural Network) in this scenario proves to be suitable, particularly for efficient and rapid processing within this type of neural network. RNN offers flexibility in terms of the number of inputs and outputs, including one-to-one, one-to-many, and many-to-many relationships.

EEG Signals

The brain is characterized by five distinct frequency ranges known as brain waves, named alpha (α), theta (θ), beta (β), delta (δ), and gamma (γ). These waves vary in frequency from low to high. The electroencephalogram spectrum of a person varies in time, in years.

1. Alpha waves are typically observed in the posterior half of the head. These waves can be detected throughout the posterior lobes of the brain and have a frequency ranging from 8-13 Hz. They often appear as a rounded or sinusoidal signal. Alpha waves have traditionally been associated with a state of relaxed awareness. They are considered the most prominent rhythm in brain activity. Opening the eyes, hearing unfamiliar sounds, experiencing anxiety, or engaging in mental concentration or attention can reduce or eliminate alpha waves. These waves exhibit a higher amplitude over the occipital areas and typically have an amplitude of less than 50 μ V.
2. The beta wave refers to the brain's electrical activity that fluctuates between 14-26 Hz. It is commonly observed during wakefulness and is associated with active cognitive processes, focused attention, problem-solving, and engagement with the external environment. This type of brain activity is present in the frontal and central regions of the brain. The amplitude of beta rhythm is typically below 30 μ V.
3. Delta waves, which have a frequency range of 0.5-4 Hz, are commonly linked to deep sleep, and can also be observed during wakefulness waking state. However,

distinguishing between genuine delta responses and artefact signals caused by the neck and jaw muscles can be challenging. The proximity of these muscles to the skin surface leads to the production of significant signals, while the desired delta signal originates deep within the brain and undergoes substantial attenuation as it passes through the skull.

4. The frequency range of approximately 30-100 Hz is known as gamma. Gamma rhythms are believed to indicate the synchronization of various groups of neurons, forming a network that facilitates specific cognitive or motor functions.
5. Theta waves are typically found in the frequency range of 4-7.5 Hz and are observed when the mind transitions into a state of drowsiness. These waves have been linked to the exploration of unconscious thoughts, fostering creativity, and facilitating deep meditation. It is worth noting that theta waves often coexist with other frequencies and are believed to be connected to the level of alertness. Researchers analyze alterations in theta wave patterns to investigate developmental and emotional aspects.

The waves that my headset can provide:

- Delta: 1-3Hz
- Theta: 4-7Hz
- Alpha1: 8-9Hz
- Alpha2: 10-12Hz
- Beta1: 13-17Hz
- Beta2: 18-30Hz
- Gamma1: 31-40Hz
- Gamma2: 41-50Hz

In addition to the raw EEG waves the headset can provide another 4 additional signals, that are the:

- Attention – that varies between 0-100% and shows the level of attention and active thinking of the person.
- Meditation – that varies between 0-100% and shows the level of relaxation and calm of the person.
- Total Power – that shows the total power consumed during the recording session.
- Derived – it's consisted of two binary values '0' and '1', for which I couldn't find a meaning.

Theme and Implementation

The theme of my project consists of the realization of the EGG Emotion Detector previously presented with the technical resources, human resources, and computing resources that I had access to. It is a simpler, not that precise, implementation that can show and extend the values of the emotions of the person that wears the headset. The program is divided in two main sections, the first being the raw signal analysis and the second part focuses on the emotion of the person.

My implementation in conjunction with the model

The project is divided into 2 major parts, those being two different screens in the interface of the program. Each screen is designed to perform a set of action over the signal read from the Force Trainer headset, and manipulate the parameters of each EEG signal that is introduced in the app.

On the main screen the user can perform the actions of, introducing the data and upload it in the program to be analyzed, later in the project documentation a tutorial will be provided. The program ensures that the data is introduced only once, thus no unwanted overwrites can occur, due to the curiosity of the user. No functionality of the program will work if no data was loaded. Without entering any submenus, the visualization / plotting of the data is possible and the visualization in time domain of the raw signals. Then the person can perform 3 main and basic mathematical operation over each signal, which are: the derivation, the integration, and the Fourier transform. At the derivation submenu it is possible to apply the first and the second order derivative and to visualize the plot of that function. Comparisons with the raw signal and other signals is possible by visually comparing the plots. At the integral submenu the user can visualize the curve of the improper integral of every raw signal and the value of the proper integral. It is allowed to modify the upper limit of the integration, the lower limit being always 0. It also shows the mean value of the signal. At the Fourier transform submenu the user can plot the frequency response of each signal and see the harmonic component of the signal that has the bigger amplitude.

On the second screen the user can visualize an animation with the sum of all EEG signals summed in time, and the corresponding emotion to each second. It shows the percentage of each emotion felt by the user in each second. The model displays AI generated images of people that feel the same emotion. The longer the recording session the longer will the animation play last.

The image generative AI that was used to create the emotion pictures is:
<https://www.krea.ai/> . By providing the right and meticulously chosen descriptions I was able to generate some decent pictures in which is a clear difference between each emotion.

The first 4 steps of the mode are done directly in the headset itself, my being left with the necessary signals at the right amplitude and with the lowest level of noise, due to the NeuroSky ThinkGear Connector technology. Steps 5 and 6 are merged in my algorithm of determining the emotion. And there were only 5 emotions that were considered: anger, disgust, fear, joy and sadness.

The signal given by the headset

Although the EEG signals can be represented as time continuous signals, this is not the way how the headset used works. Firstly, the microprocessor included on it, sends a bunch of error checking bits and quality bits to ensure that the only the best recordings get to the computer. The data itself is a sampled signal, with the values of the EEG signals and the other signals of the headset being sent with an almost equal period of 1 second. Thus, all the computations and the calculations performed in the whole program had to be done in numeric methods and approximations.

Recording of the data set

Emotions are a subjective and personal thing, that fluctuates in a man depending on his current mental state and can drastically vary among individuals. The measure of emotion is the measure of the EEG signals generated by several people when they are subjected to the same stimulus. By making the right algorithm and approximations, a proper way of detecting an accurate mental state of a person is possible.

Emotions and feelings induced through videos

Due to the lack of volunteers, the only person that I could perform the measurement was myself. The goal of the experiment was to obtain a diverse set of emotion data, to set the threshold values of the emotion calculation algorithm.

The experiment took place in 10 sessions in which I watched different emotion inducing videos, each session took around 3 minutes. There were 2 videos reserved for each emotion to create diversity in the data base and to simulate, as best as possible, the recording of another person's mental state. I tried to enhance parameters of the emotion felt by also thinking about things and situations, that I knew that will make that possible. The videos were watched one after another to not "loose" the emotions, and to not modify the overall emotion. This is since even the state of relaxation, calmness and Zen can change the value of the Alpha and Beta waves.

Videos watched for each emotion:

Sadness:

- video 1: https://www.youtube.com/watch?v=_FOm8eCyUpY
- video 2: <https://www.youtube.com/watch?v=YweYRI7IUPc>

Disgust:

- video 1: <https://www.youtube.com/watch?v=yekWI59YWTg>
- video 2: <https://www.youtube.com/watch?v=SAeiQY2itHE>

Anger:

- video 1: <https://www.youtube.com/watch?v=7wKdJOqFwYE>
- video 2: <https://www.youtube.com/watch?v=01PRo9-8RzI>

Joy:

- video 1: <https://www.youtube.com/watch?v=Nq5CR5mTgzA&t=339s>
- video 2: https://www.youtube.com/watch?v=6iltHBuxm_E

Fear:

- video 1: <https://www.youtube.com/watch?v=gbWE47w2oLE>
- video 2: <https://www.youtube.com/watch?v=ck1NO9MyQsM>

To be noted that, those are not scientifically proven videos that can ensure to induce certain emotions in people. These are just some of the many videos of their kind that I found on the online platform YouTube.

The dataset

After the measuring session of the emotion dataset, all the values measured values were saved individually in csv documents. All the parameters that the headset gave are included in the dataset but, I was interested only in the raw EEG signals that were measured each emotion measuring experiment. To manipulate and make use of the database a MATLAB program was needed, in that program the data of each csv file is read using the function: `readmatrix()`. Working with matrices makes the process of computing equations with a big number of variables much easier and makes the code more understandable and easier to code. For safety reasons all the documents have a Backup document, in case the user is not choosing the right document to save his data in.

For the algorithm implementation, we needed a set of threshold values for the comparison of the new data introduced. First, the dataset is divided into 5 emotions: anger, disgust, fear, joy, and sadness. The MATLAB program is firstly reading and saving into matrices variables the whole data from the documents; then the mean values are taken for each EEG signal from each emotion data set. The mean values function ignores the other data from the documents, beside the EEG waves. There being two documents for each emotion, another average value of each EEG waves is taken between the 2 previously averaged documents. The newly computed values are saved into a new csv document which will contain the EEG waves values for each emotion as a 5 x 10 matrix (5 emotions, 10 EEG signals).

By using this method any possible spiked or artifacts can be reduced, and a great mediation is made. This might not be the most accurate way into interpretation of the EEG signals, yet for the small data set and the lack of machine learning skills, this method gives a decent resolution into each emotion.












 EEG_ANGER_1	1/4/2024 5:44 PM	Microsoft Excel Co...	15 KB
 EEG_ANGER_2	1/4/2024 5:54 PM	Microsoft Excel Co...	15 KB
 EEG_DISCUST_1	1/4/2024 5:16 PM	Microsoft Excel Co...	16 KB
 EEG_DISCUST_2	1/4/2024 5:39 PM	Microsoft Excel Co...	16 KB
 EEG_FEAR_1	1/4/2024 6:08 PM	Microsoft Excel Co...	21 KB
 EEG_FEAR_2	1/4/2024 6:13 PM	Microsoft Excel Co...	16 KB
 EEG_JOI_1	1/4/2024 5:59 PM	Microsoft Excel Co...	15 KB
 EEG_JOI_2	1/4/2024 6:03 PM	Microsoft Excel Co...	16 KB
 EEG_SADNESS_1	1/4/2024 5:03 PM	Microsoft Excel Co...	17 KB
 EEG_SADNESS_2	1/4/2024 7:18 PM	Microsoft Excel Co...	16 KB
 EEG_THRESHOLD	1/5/2024 7:17 PM	Microsoft Excel Co...	1 KB

Figure 10 - The Backup folder of the Database

The emotion calculation algorithm

The algorithm used does not include any machine learning or AI to process the data, it only compares the current introduces data with the thresholds read before, and it was implemented in a MATLAB function file that is called each time the user wants to replot the data. The function at the beginning of the algorithm reads the data from the Threshold values document and the document that contains the data introduces by the user. The dates from the headset are 1 second apart of each other, the algorithm takes in each second the values of the EEG signals and then subtracts the EEG signals for each emotion from themselves. That makes a new matrix with 1 x 5 (5 emotions) for each second. The EEG signals are added together for each emotion and a percentage of each emotion on each is made. In the following explanations I'm going to refer at matrices as tables.

The initial table of data, from the user:

- The number of columns is equal to the number of seconds.

Delta	Theta	Alpha1	Alpha2	Beta1	Beta2	Gamma1	Gamma2
22635	38001	18912	24922	10265	7227	4705	2435
707185	37067	14192	17098	3271	5317	3975	2890
7588	7677	2671	25954	6717	5325	2938	2536
52551	26819	6779	11148	14223	7180	4937	3005
192491	71554	16236	6936	9903	7415	2328	2204
47976	49625	33956	37182	7813	7663	5016	5179
81415	28417	5197	23352	8489	9772	6105	6800
10502	3189	10427	5537	6674	10930	7215	6780
216044	127506	5285	18522	7590	5913	6728	2850
44520	38075	5501	2623	7505	5004	7720	3007
39789	40543	39888	10354	5639	7585	5249	2436
20297	15165	5773	21206	5917	3166	3606	3423
22628	15827	30434	7162	5233	2870	3560	4207

The table of threshold values:

- The values are computed only once and then they are used to compute the emotion.

Anger	125643.9	39289.9	19168.8	14731.8	11489.6	10701.4	8024.5	6250.1
Disgust	97057.5	41178.6	15484.4	12055.3	9743.6	8962.7	6294.9	4655.4
Fear	71967.0	23974.9	15205.4	13929.8	10626.1	11850.0	9178.9	6380.8
Joy	90226.4	33736.5	17886.2	11845.8	9861.1	10435.3	8165.3	5907.6
Sadness	71583.8	31152.6	20653.4	14796.0	9403.6	8559.1	5443.0	4633.5

The columns of the first data table express the EEG parameters in as well as in the threshold table, as it was stated before, each row of the first data table will subtract each row of the thresholds that will give a table of this kind.

Anger level sec 1	Disgust level sec 1	Fear level sec 1	Joy level sec 1	Sadness level sec 1
Anger level sec 2	Disgust level sec 2	Fear level sec 2	Joy level sec 2	Sadness level sec 2
Anger level sec 3	Disgust level sec 3	Fear level sec 3	Joy level sec 3	Sadness level sec 3

Because numbers by themselves don't possess a significant meaning for humans, using percentages is a better way of showing the ratio of each emotion per second. The percentages were determined by finding the sum of the signal in each row then, dividing each emotion value to the sum. To be noted that that there were used the absolute values of each division, to prevent more errors in the precision of the computation to occur. Thus, creating a final output table that the user will see.

Time [sec]	Anger [%]	Disgust [%]	Fear [%]	Joy [%]	Sadness [%]
1	8.24	13.53	31.18	17.55	29.48
2	19.32	19.91	20.39	20.02	20.34
3	32.26	21.55	12.88	19.58	13.71
4	4.37	15.37	31.38	19.02	29.84
5	17.44	19.67	21.48	20.08	21.31
6	13.89	19.22	23.54	20.20	23.12
7	12.45	19.04	24.38	20.25	23.86

Theoretical results

The results obtained after the algorithm computations, look promising at a first glance. There are no visible spikes in values or rather strange patterns. The values in each part of the algorithm didn't cause any unexpected errors or obviously wrong values. The patterns of the data for each emotion follows the expected path. As an example, the waves of the fear row show high levels of Beta waves which are activated in a stressful situation and indicated the level of active thinking. Alpha waves are affected also by blinking, the values of the Alpha waves are small in comparison to the others because the brain were more alert to catch any possible danger, thus by not blinking so often it could achieve that.

The Raw EEG Analyzer

The program is divided into 2 parts that were previously presented. To improve the signals interpretations, a menu of signal analysis was needed. This menu considers the raw values of the data read from the brain and performs a series of transformations over the data for a better understanding of the nature of it. There is no much theoretical description needed for those submenus because they consist in some blocks that perform operation over the data. A broadly description of each will be provided at the experimental part, where they play a major role.

Experimental Presentation

In this part of the documentation the actual interface and its functionality will be presented. There are 2 main figures or screens that are available for the user to work around the signals that he just got. Each of the consists of a set of actions that are performed easily due to the simple and intuitive graphical user interface. To embed both worlds that use Electroencephalography, the interface theme wants to mimic a piece of medical equipment and a piece of university laboratory equipment. The learning curve of the program is a stipe one due to the simplicity and the logic of the program.

The Raw Data Analysis Menu

The menu that focuses on the mathematics behind the signals, and the raw interpretation of them. By using the **Save** command from the top the figure every plot or computation can be saved for future references. The computations are performed by sampled time signals, thus there is no “right” value computer, only a better approximation using numerical methods.

Menu with no data

If the user does not press the **Introduce the eeg data** an error will be thrown, and no function or screen of the Graphical User Interface will work. This section plays a major role in the security of the program but especially of the functions that perform direct data manipulation. By pressing the **Emotions handling** button, it is not possible to bypass the security, because those menus also cannot function if no data is loaded. In the case of a user that wants to reupload a set of dates the program must be closed from the **Close** button. By pressing the **Close** button, a clear Workspace and a clear Command Window are ensured, for best performance of the new measurements.

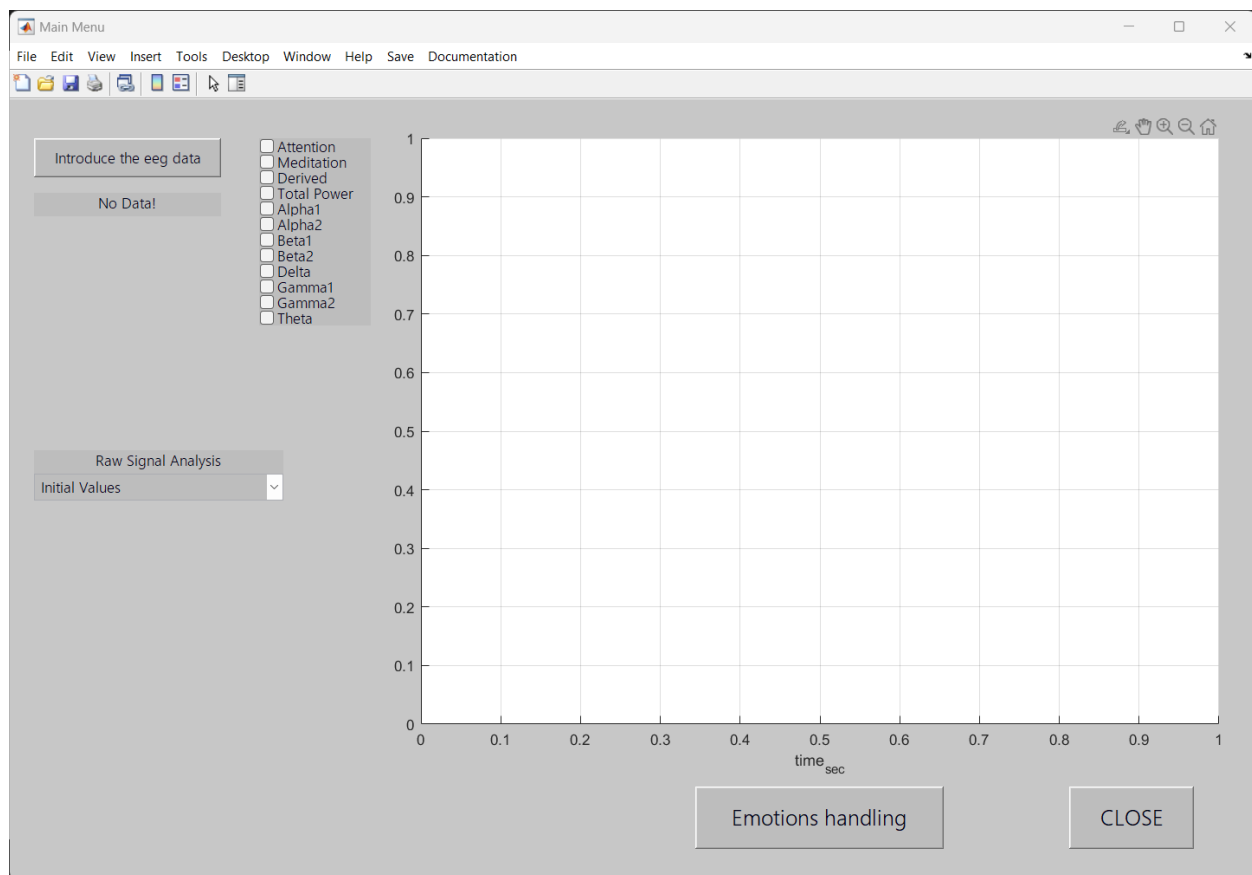


Figure 11 - The Raw Signal Menu with no data

Right after hitting the **Introduce the eeg data**, the text label below will be changed from **No Data!** to **The data was introduced!** and all the applications listed below will be possible to be run.

Initial values

The **Initial values** submenu is designed to perform quick plotting of all the 12 channels of the Input data. By choosing the **Initial values** menu the only thing that is possible is plotting and visual analysis of the plot of the waves. The first 4 channels (**Attention**, **Meditation**, **Derived**, **Total Power**) can be plotted only one at the time, because a couple of factors associated with their units of measure that are different for each other, the magnitudes of the signals are a lot different, and their frequency differs a lot. But one of the most important things that lead to the separation of them was the fact that, they mean completely different things and analyzing signals that not correlate one with each other is confusing.

The last 8 channels (**Alpha1**, **Alpha2**, **Beta1**, **Beta2**, **Delta**, **Gamma1**, **Gamma2**, **Theta**) can be plotted on the same plot because they have the same nature and possess similar parameters. Their amplitudes fluctuate in the range of μV and their 1 – 50 Hz frequencies are comparable. To make the reading and what every line from the plot means the color of the checkbox takes the color of the signal that is linked to.

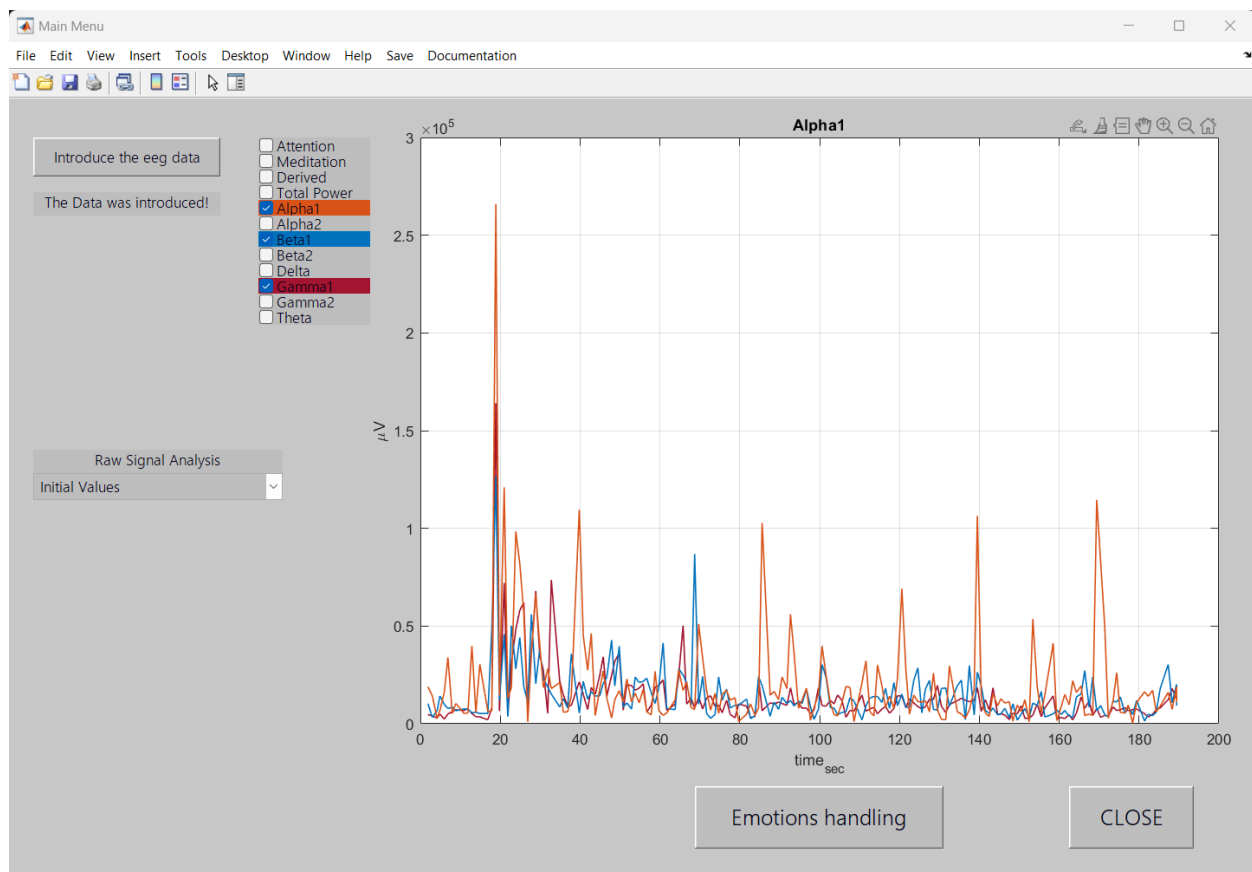


Figure 12 - Initial values

Derivative

The **Derivative** submenu is designed to provide a new way of looking at the data by analyzing what can happen to the signal when is differentiated. A lot of deeper understanding of the waves can be drawn from their derivatives like their positivity and negativity on the given domain and their concavity or convexity. This operation can be performed on all 12 channels, regardless of their origin. The analysis of the signal is also done only visually by studying the plot of it.

As soon as the submenu is accessed for the first time, a second order derivative cannot be performed, due to the way of how the software works. The function that calculates the second order derivative needs to be firstly called by the first order derivative function. Plotting the derivative along with the initial function can be made but is not allowed to plot the first order derivate and the second order derivative at the same time.

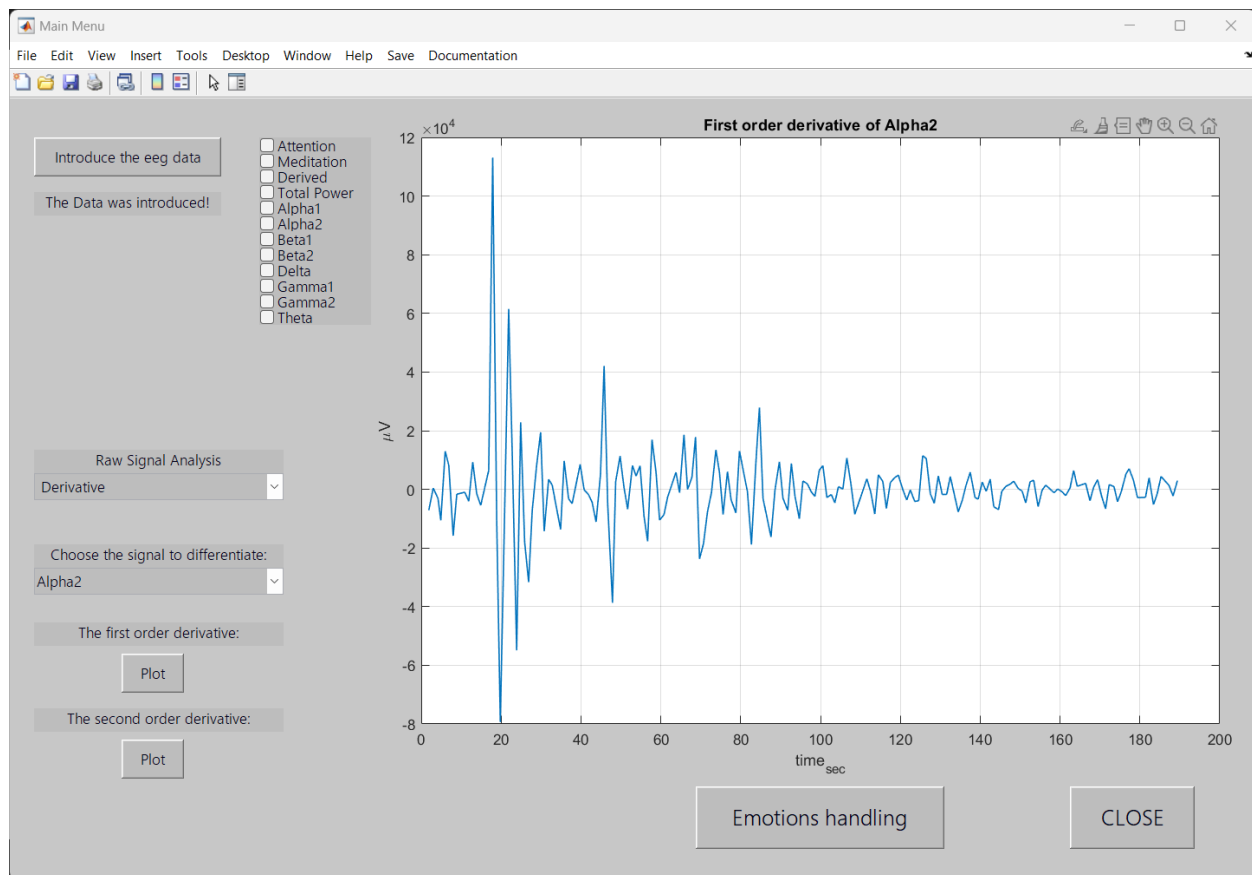


Figure 13 – Derivative

Integral

For the **Integral** submenu the user can change the parameters of the operation in the interface not only with its brain. By taking the integral over the functions period, which is the session length, parameters like the average values of the voltages and the values of the integral can indicate factors like, the gain of the headset, and the power consumed over the session. It is possible to find those parameters not only over the entire function but also on discrete intervals.

The user can use the slider to change the value of the upper limit of the integral but the lower one is always 0, regarding the signal that is applied on, or the unit of measure. On the figures plot, it is shown the indefinite integral of that function, but the limits of the plotted line are between the chosen values. Is also shown the value of the definite closed integral, which calculates the area under the graph. The average value of the signal is shown below the value of the integral. It is possible to plot the raw data at the same time with the plot of the integral but is not recommended, due to the large-scale differences and plot shapes.

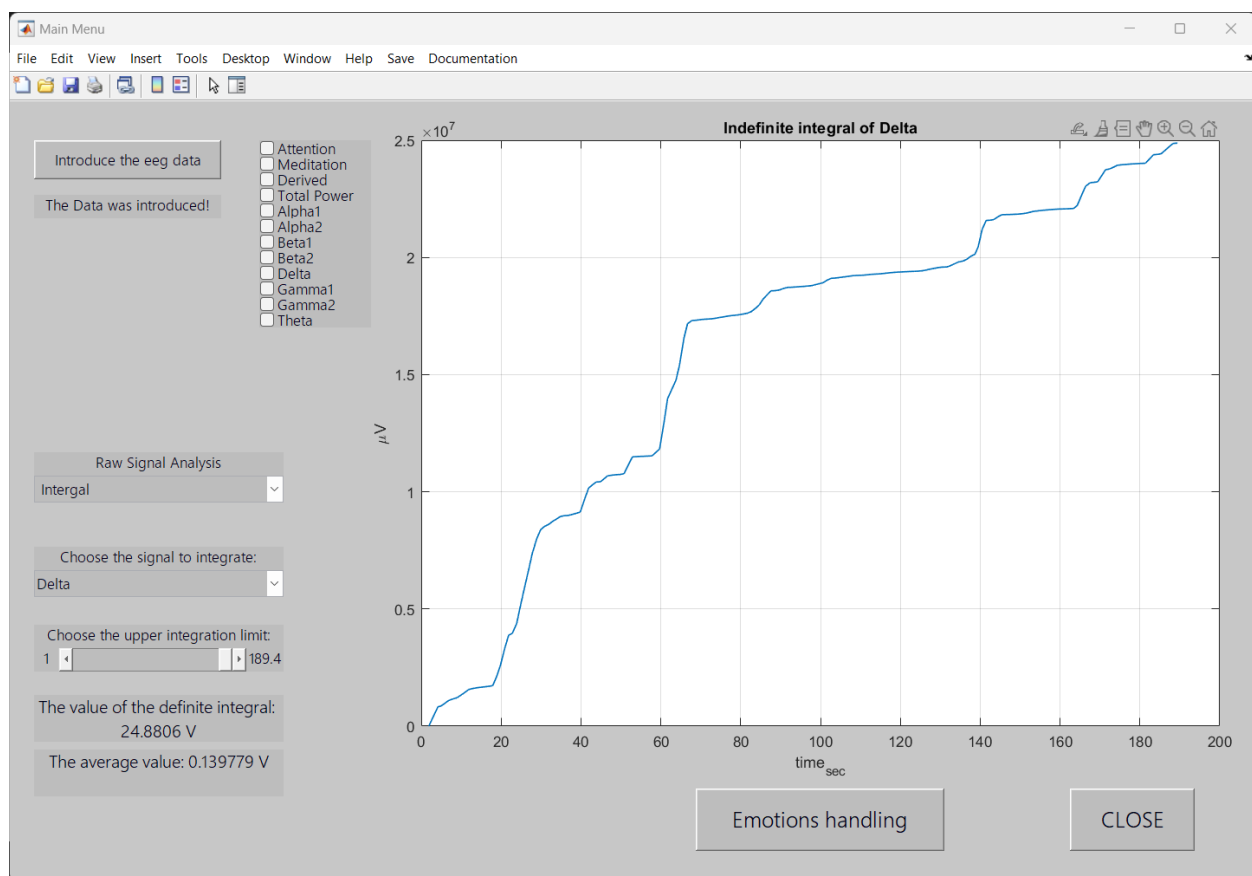


Figure 14 – Integral

Fourier Transform

The **Fourier Transform** submenu is meant to indicate the frequency response of every signal that is selected. The need of seeing the signals in their frequency domain came from the complex non-sinusoidal shape of the EEG signals that contains a significant number of harmonics with higher frequencies than the desired one. Such harmonics can be induced by sudden movements of the body, or by the electrical noise found in the headset and around it.

The plot X axis changes from the time domain to the frequency domain, where the frequency bandwidth is only 50 Hz, because this is the range of frequencies that the headset can output. The text box with the value of the **predominant frequency** shows at which frequency the harmonic has the biggest amplitude. In this way anomalies and spikes can be detected. It allowed only on plot at the time of each signal. The plot of the initial values of the signals at the same time as the Fourier plot is possible but not recommended, due to the difference in axes scales, magnitudes of the signals and the meaning of them.

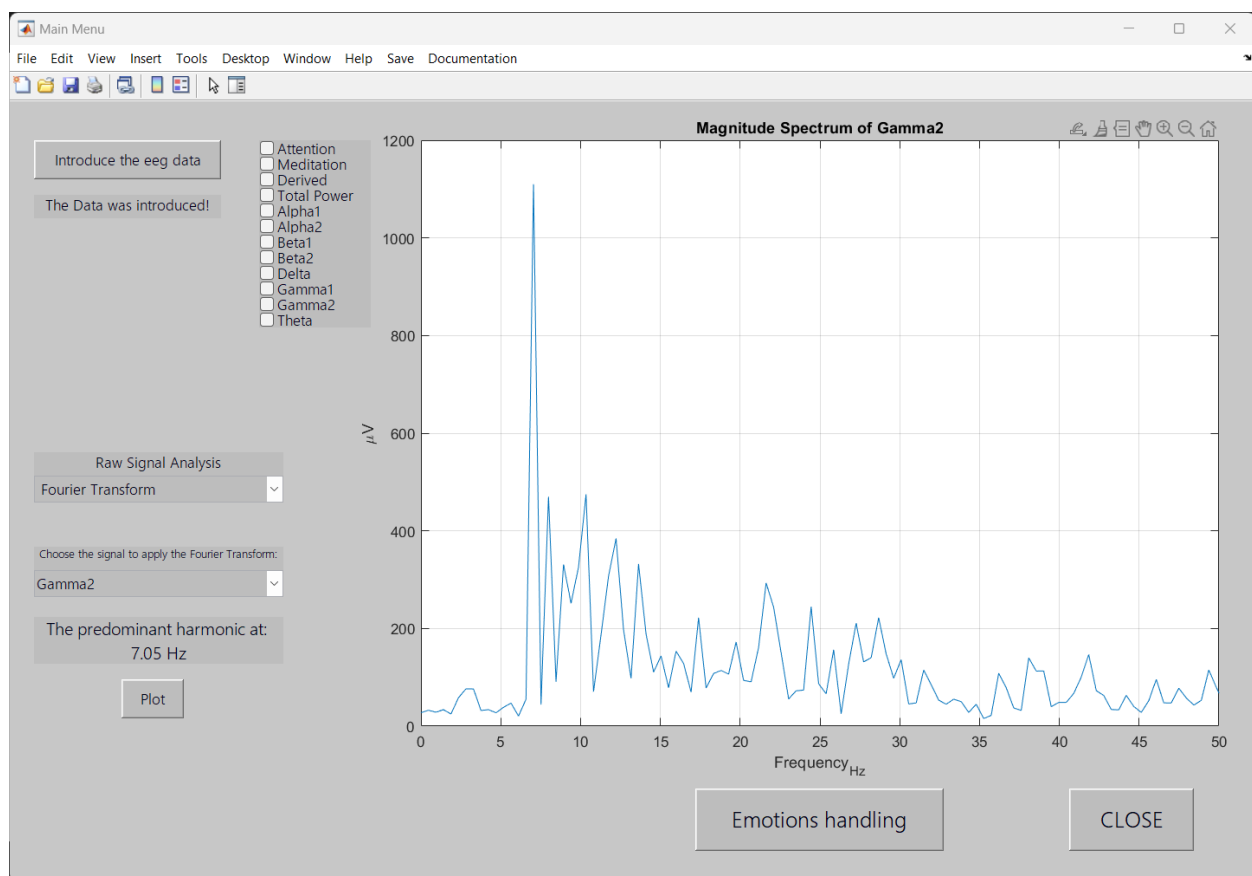


Figure 15 - Fourier Transform

The Emotion handling menu

The second screen is meant only for the abstract emotion presentation, and no computations are performed directly on the Graphical User Interface. Here the user can see his variation in emotions in each second. The computation is done in the backend of the program so no mathematics or programming skills are needed to use this section at its designed maximum. The visualization of the emotion would blend well with the visualization of the data, that's why a plot is provided in this part.

The plot axes have the same dimensions as the ones from the previous part, but they are used in kind of a different way. To generalize the 8 well defined and separated signals into one, they were added into a single signal. The only one summed signal is displayed on the plot. To make it more enjoyable to watch and to simulate the real time visualization, the plot is animated. As the seconds pass by the graph appears on the plot of the figure and they stay on for about 90 milliseconds.

The textbox placed at the left of the screen shows the values of the percentages of emotion in each second. Dates which are directly read from the algorithm that computed them.

To avoid a too technical and non-user-friendly menu, I added pictures for each emotion that is displayed the in same way as the graph and the emotion percentages. A function calculated the emotion with the biggest percentage value and assume that this is the predominant emotion, and shows the picture related to that emotion.

Practical results

The practical results showed some perturbances and minor errors that were quite noticeable. The data used for testing was a 3-minute-long data set taken into an uncontrolled environment and doing various activities and thinking about random things. The testing showed that this kind set had spikes of fear and anger, those being the only ones that showed in the visualization. A couple of factors might indicate that behavior, and those are: that fact that the whole data set was taken from a single person and its missies a factor of diversity, the unequal recording time of the data and all the approximations that were done in the whole process.

To conclude a proper solution a bigger number of tests must be performed on a significant number of people.

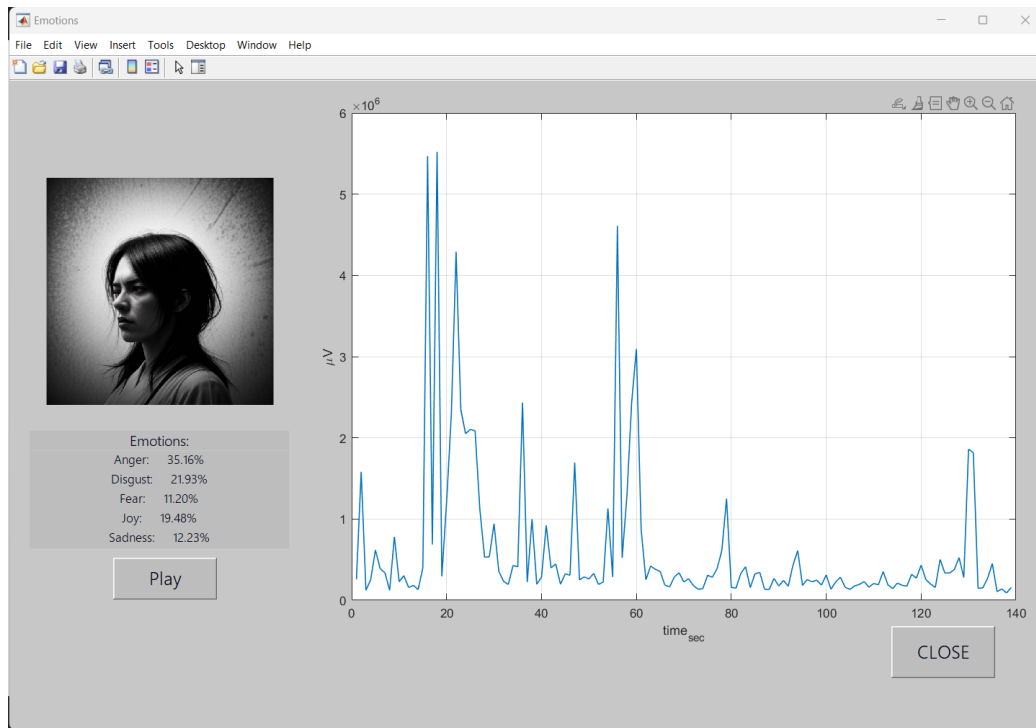


Figure 16 - Emotion handling I

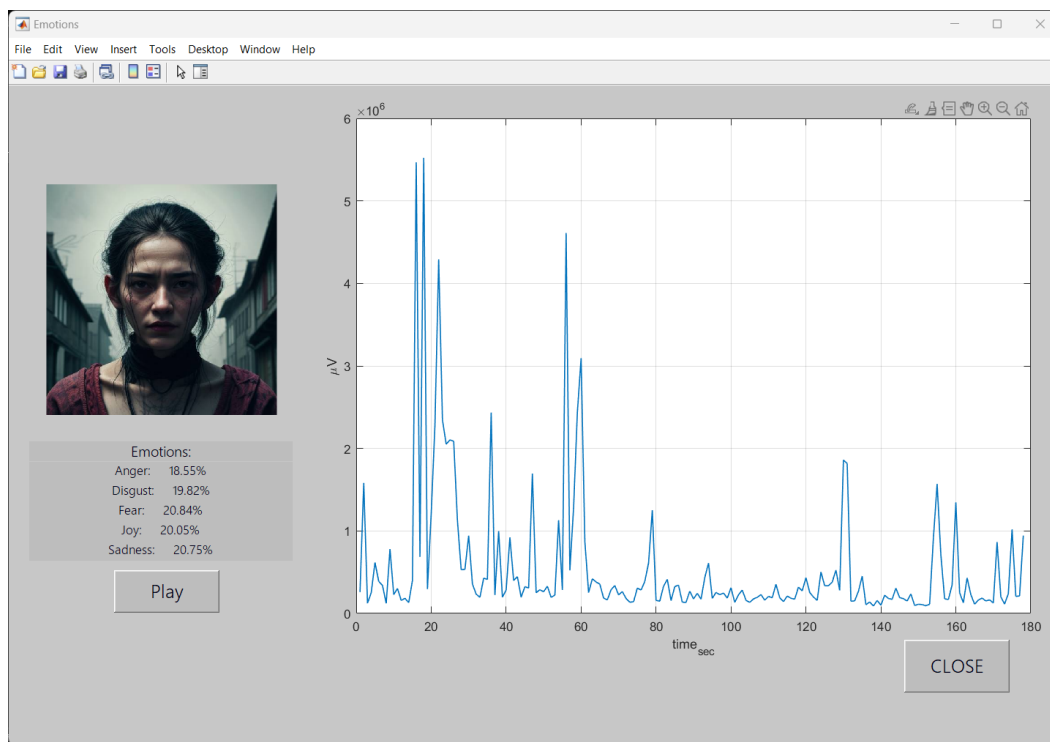


Figure 17 - Emotion handling II

Recording of the EEG data – Tutorial

Step 1: Open the ThinkGear Connector and at the **logging** section check the **csv** box and check **headset data** if is not checked already.

Step 2: In the section of **pick log file** browse for the folder where the program functions and source code are and set the name as **EEG_Device_Data**. To be noted, if the name of the csv file is not this the data will not be read by the program. The name of the **log tittle** is not important.

Step 3: Connect the headset to the app, if it does not want to connect on the first try, do one or two times more connections until it connects.

Step 4: The recording starts right after the headset is connected. Record your session as long as you want. To stop the recording either disconnect the headset or pause it.

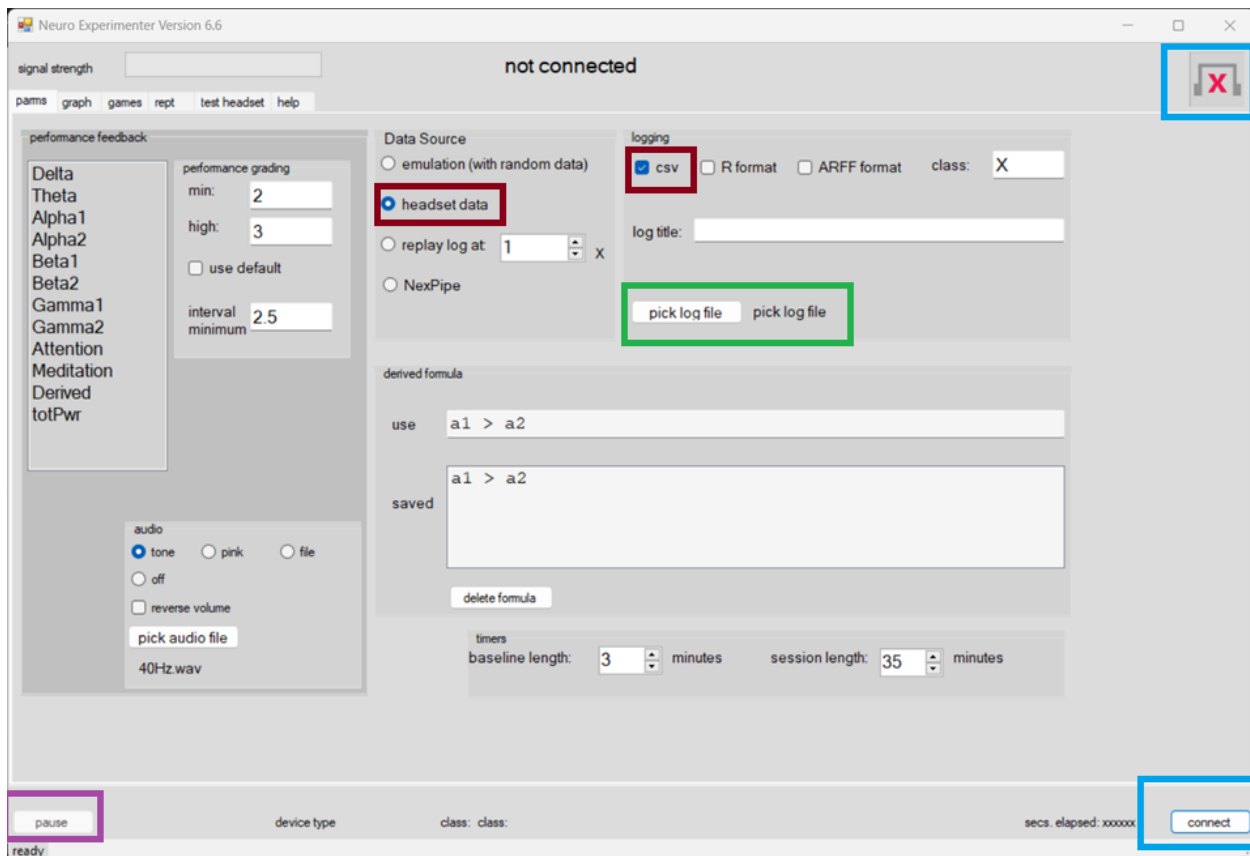


Figure 18 - Introduce data - Tutorial.

Significant code parts

Derivative computation

```
% The first order derivative calculation

fprintf('Case %d selected\n', selected);
derivative_1{selected} = gradient(eeg_data{selected}) ./ gradient(eeg_data{13});
assignin('base', 'derivative_1', derivative_1);

% The second order derivative calculation
derivative_2{selected} = gradient(derivative_1{selected}) ./
gradient(eeg_data{13});
assignin('base', 'derivative_2', derivative_2);
```

The code that performs the first and second order derivative. It uses the `fprintf()` function to display on the Command Window the selected case. The use of the `assignin()` function is to write the values into the Workspace, for easier use of the data across the program. For the data type a numerical way of calculating the derivative is needed and the function `gradient()` calculates the central difference for interior data points, and it gives indeed a good resolution and precision. Only the beginning and the end of the signal get calculated as the difference between two consecutive points.

Integral computation

```
% Modifies the sizes of vectors in order to be the same
max_value = get(source, 'Value');
eeg_max = zeros(size(eeg_data{13}));
eeg_var_max = zeros(size(eeg_data{selected}));

for i = 1:length(eeg_data{13})
    if eeg_data{13}(i) <= max_value
        eeg_max(i) = eeg_data{13}(i);
        eeg_var_max(i) = eeg_data{selected}(i);
    end
end

last_nonzero_index = find(eeg_max, 1, 'last');
last_nonzero = find(eeg_var_max, 1, 'last');
eeg_max = eeg_max(1:last_nonzero_index); % Time x axis
eeg_var_max = eeg_var_max(1:last_nonzero); % Variable y axis

% The area under the curve using the trapezoidal method
def_int{selected} = trapz(eeg_max, eeg_var_max) / 1000000; % unit measure in V/W
not uV/uW
```

```
%The average value
avr{selected} = def_int{selected} / length(eeg_max);

% The function that describes the indefinite integral
indef_int{selected} = cumtrapz(eeg_max, eeg_var_max);
cla;
```

For the integral, the user can modify the upper limit of the integration but that will lead into two vectors with unequal values, and that is not possible for the computation of the integral. Firstly, two new matrices are needed to store the values of the needed signal until a certain point is reached, that point is chosen by the user. This happens in the code provided above the first integration. The function `trapz()` computes the approximate integral of the function via the trapezoidal method with unit spacing of 1, for the definite integral. For the indefinite integral, `cumtrapz()` computes the approximate cumulative integral of the function via the trapezoidal method with unit spacing of 1. The average value is determined by the division of the definite integral and the period.

Fourier Transform computation

```
% Function that calculates the fft
fs = 1024; %The Nyquist frequency
N = length(eeg_data{selected});
eeg_pad{selected} = [eeg_data{selected}; zeros(2000,1)]; % Initialize for Zero-
padding

% Paramters after Zero-padding
N2 = length(eeg_pad{selected});
one_sided = eeg_pad{selected}(1:floor(N2/2)); % Use floor to ensure integer
division
f_eeg = fs * (0:floor(N2/2)-1) / N2; % Adjust the limits accordingly
eeg_ampl{selected} = abs(one_sided) / (N / 2);

% Find the index of the maximum magnitude
[~, maxIdx] = max(eeg_ampl{selected});
% Calculate the corresponding frequency
freq_max = f_eeg(maxIdx);

xlim([0, 50]); % EEG signals have low frequencies
```

To obtain the Fourier transform is not such a straightforward process due to the complexity of the procedure. Firstly, the Nyquist frequency had to be chosen double of the working frequency of the headset which has the value of 512 Hz. The zero-padding method was

used to clear the noise and the unwanted frequencies, also making the overall plot much cleaner. In this case it is presented and plotted the one-sided spectrum, just a personal preference. The other `max()` function finds the maximum number in the column vector and finds its frequency by running it again though the computation. The original frequency (x axis) goes up to 512 Hz, but the EEG signals have much lower frequencies around 50 Hz, so there is no need of an empty plot, and that's why the x limit is set to 50.

Emotion algorithm

```
% Convert the cell data to matrix data
eeg_data_mat = cell2mat(eeg_data);
out_emotion = zeros(length(eeg_data{2}), 5); % The initialization of out_emotion
outside the loop

for i = 1 : length(eeg_data{2}) % loop for the rows of eeg_data_mat
    sum_mat = zeros(5, 1); % sum_mat resets inside the loop for each iteration
    for j = 1 : 5 % loop for the emotion threshold
        for k = 3 : 10 % loop for the eeg signals
            % Calculates the difference between the data and the threshold
            sum_mat(j) = sum_mat(j) + (eeg_data_mat(i, k + 1) - eeg_threshold(j,
k));
        end
    end
    out_emotion(i, :) = sum_mat; % Store the results for each iteration in the
corresponding row of the emotion
end

emotion_values_percentage = zeros(length(eeg_data{2}), 5);

for i = 1:length(eeg_data{2})
    emotion_values = out_emotion(i, :);

    % Function that transforms all values into percentages
    row_values_percentage = (abs(emotion_values) / sum(abs(emotion_values))) *
100;

    % Store the percentage values for the current iteration in the matrix
    emotion_values_percentage(i, :) = row_values_percentage;
end

% Writes the data in the Workspace
assignin('base', 'emotion_values_percentage', emotion_values_percentage);
```

The code consists in a 3-dimensional array that performs the computations. The first loop goes over the length of the entire data set, which is the effective time of the recording. For each iteration of `i` a 5 x 1 matrix `sum_mat` is created; then another loop goes along the rows of the

threshold values, which are only 5; inside the j loop there is another loop that goes along the columns of the threshold values. The function inside the all the 3 loops subtracts each row of the threshold values from the i row of the eeg_data; then adds all the columns in the sum_mat variable that created a matrix with a row for each emotion. These matrices are saved in each iteration in the variable out_emption. Then for easy reading the next code structure converts the EEG values into percentages, and then final output is a matrix consisting of 5 columns, one for each emotion, and n number of rows, depending on the length of the session.

Conclusions

In conclusion, the field of Electrography is a broad one, that includes a lot of applications ranging from medical use to university research. My project tries to combine them but is also more focused on the research part. For sure the place of it is in a university or a research lab, this being far from a commercial product.

Own thoughts

As my opinion, I am satisfied with the overall results that I got. It was a long journey with a lot of ups and downs, now as I'm thinking of it, there were more downs than ups, but was worth every minute, hour spend making the project. It was a great way into discovering a rather interesting domain, that extends far beyond my field of study, but is also rooted in it. Making this project I also learn things like: "start as soon as possible", and things like time management.

Future improvements

- Implementation of a real time reading and plotting function.
- Use of specialized drivers and software to introduce the headset data directly into MATLAB thus bypassing the NeuroSky software.
- A bigger data set and a bigger number of volunteers, to perform a more objective and diverse study.
- Some kind of machine learning algorithm to get more accurate results from the dataset.
- The submenus to not interfere so easily one with each other.
- Overall, a bigger efficiency in the code.

Bibliography

- <https://dl.acm.org/doi/10.1145/3386331>
- <https://www.mathworks.com/company/newsletters/articles/the-origins-of-matlab.html>
- <https://www.slideshare.net/ideas2ignite/matlab-introduction-7045222>
- <https://www.mathworks.com/company/newsletters/articles/a-brief-history-of-matlab.html>
- <https://en.wikipedia.org/wiki/MATLAB>
- https://journals.lww.com/clinicalneurophys/abstract/2013/02000/early_history_of_electroencephalography_and.2.aspx
- <https://en.wikipedia.org/wiki/Electroencephalography>
- <https://www.nhs.uk/conditions/electroencephalogram/>
- <https://www.youtube.com/watch?v=-K78gJGduNg>
- <https://www.britannica.com/science/electroencephalography>
- <https://en.wikipedia.org/wiki/NeuroSky>
- <https://neurosky.com/>
- <https://www.nature.com/articles/s41598-021-86345-5>
- <https://faculty.washington.edu/seattle/brain-physics/textbooks/sanei.pdf>

The whole program code

To be noted, the csv files with the database and the photos used in the program are not included.

```
clc;
clear all;
close all;

% Main interface
Menu = figure('Name','Main Menu', ...
    'Units','normalized', ...
    'Position',[0.04 0.058 0.75 0.835], ...
    'NumberTitle','off', ...
    'Color', [0.78 0.78 0.78]);

% Saves the workspace in the working directory
h = uimenu('Label','Save','Callback','savefig');
% Opens the documentation
d = uimenu('Label','Documentation','Callback', @Documentation_CallBack);

% Menu for introducing the data
In_Data = uicontrol('Style','pushbutton', ...
    'Units','normalized', ...
    'String','Introduce the eeg data', ...
    'Position', [0.02 0.90 0.15 0.05], ...
    'BackgroundColor', [0.74 0.74 0.74], ...
    'ForegroundColor', [0 0 0.1], ...
    'FontName', 'Yu Gothic UI Semilight',...
    'FontSize', 10, ...
    'Callback', @In_Data_CallBack);

% Menu to check if there is any data loaded
Data_Check = uicontrol('Style','Text', ...
    'Units','normalized', ...
    'String','No Data!', ...
    'Position', [0.02 0.85 0.15 0.03], ...
    'BackgroundColor', [0.74 0.74 0.74], ...
    'ForegroundColor', [0 0 0.1], ...
    'FontName', 'Yu Gothic UI Semilight',...
    'FontSize', 10, ...
    'Callback', '');
assignin('base', 'Data_Check', Data_Check);

% Button GUI element
Button_Check = uicontrol('Style','Text', ...
    'Units','normalized', ...
```

```

        'String', '', ...
        'Position', [0.02 0.75 0.15 0.08], ...
        'BackgroundColor', [0.74 0.74 0.74], ...
        'ForegroundColor', [0 0 0.1], ...
        'FontName', 'Yu Gothic UI Semilight',...
        'FontSize', 15, ...
        'Visible', 'off', ...
        'Callback', '');
assignin('base', 'Button_Check', Button_Check);

% Menu to show the raw data from the headset
max_boxes = 12; % Total number of checkboxes
data_boxes = cell(max_boxes, 1);
assignin('base', 'data_boxes', data_boxes);
assignin('base', 'max_boxes', max_boxes);

data_Names = {'Attention', 'Meditation', 'Derived', 'Total Power', ...
    'Alpha1', 'Alpha2', 'Beta1', 'Beta2', 'Delta', 'Gamma1', ...
    'Gamma2', 'Theta'};

for i = 1 : max_boxes
    % Use a function handle to pass the correct arguments to the callback
    data_boxes {i} = uicontrol('Style','checkbox', ...
        'Units', 'normalized', ...
        'String', data_Names{i}, ...
        'Position', [0.2 0.95 - i * 0.02 0.09 0.02], ...
        'BackgroundColor', [0.74 0.74 0.74], ...
        'ForegroundColor', [0 0 0.1], ...
        'FontName', 'Yu Gothic UI Semilight',...
        'FontSize', 10, ...
        'Callback', {@display_initial_value, i});
end

%Empty plot
plotPosition = [0.33 0.2 0.64 0.75];
axes_plot = axes('Position', plotPosition);
getframe;
grid on;
xlabel('time_{sec}');
assignin('base', 'axes_plot', axes_plot);

% Menu to do some analysis operations on the collected data
Signal_Analysis_Label = uicontrol('Style', 'Text', ...
    'Units', 'normalized', ...
    'String', 'Raw Signal Analysis', ...
    'Position', [0.02 0.52 0.2 0.03], ...
    'BackgroundColor', [0.74 0.74 0.74], ...
    'ForegroundColor', [0 0 0.1], ...
    'FontName', 'Yu Gothic UI Semilight',...
    'FontSize', 10, ...
    'Callback', '');

Signal_Analysis = uicontrol('Style', 'popupmenu', ...

```

```

'Units', 'normalized', ...
'String', ' Initial Values | Derivative | Intergal | Fourier Transform ', ...
'Position', [0.02 0.5 0.2 0.02], ...
'BackgroundColor', [0.74 0.74 0.74], ...
'ForegroundColor', [0 0 0.1], ...
'FontName', 'Yu Gothic UI Semilight',...
'FontSize', 10, ...
'Callback', @Signal_Analysis_Callback);

%Choose the signal gui
Choose_Label = uicontrol('Style', 'Text', ...
    'Units', 'normalized', ...
    'String', 'Choose the signal to differentiate:', ...
    'Position', [0.02 0.4 0.2 0.03], ...
    'BackgroundColor', [0.74 0.74 0.74], ...
    'ForegroundColor', [0 0 0.1], ...
    'FontName', 'Yu Gothic UI Semilight',...
    'FontSize', 10, ...
    'Visible', 'off', ...
    'Callback', '');
assignin('base', 'Choose_Label', Choose_Label);

Choose = uicontrol('Style', 'popupmenu', ...
    'Units', 'normalized', ...
    'String', data_Names, ...
    'Position', [0.02 0.38 0.2 0.02], ...
    'BackgroundColor', [0.74 0.74 0.74], ...
    'ForegroundColor', [0 0 0.1], ...
    'FontName', 'Yu Gothic UI Semilight',...
    'FontSize', 10, ...
    'Visible', 'off', ...
    'Callback', @Choose_Callback);
assignin('base', 'Choose', Choose);

%Derivative gui

% Plot the first order derivative
Dev1_Label = uicontrol('Style', 'Text', ...
    'Units', 'normalized', ...
    'String', 'The first order derivative:', ...
    'Position', [0.02 0.3 0.2 0.03], ...
    'BackgroundColor', [0.74 0.74 0.74], ...
    'ForegroundColor', [0 0 0.1], ...
    'FontName', 'Yu Gothic UI Semilight',...
    'FontSize', 10, ...
    'Visible', 'off', ...
    'Callback', '');
assignin('base', 'Dev1_Label', Dev1_Label);

Plot1 = uicontrol( 'Style','pushbutton', ...
    'Units','normalized', ...
    'String', 'Plot', ...

```



```

'Position', [0.09 0.24 0.05 0.05], ...
'BackgroundColor', [0.74 0.74 0.74], ...
'ForegroundColor', [0 0 0.1], ...
'FontName', 'Yu Gothic UI Semilight',...
'FontSize', 10, ...
'Visible', 'off', ...
'Callback', @Plot_dev1_CallBack);
assignin('base', 'Plot1', Plot1);

% Plot the second order derivative
Dev2_Label = uicontrol('Style', 'Text', ...
    'Units', 'normalized', ...
    'String', 'The second order derivative:', ...
    'Position', [0.02 0.19 0.2 0.03], ...
    'BackgroundColor', [0.74 0.74 0.74], ...
    'ForegroundColor', [0 0 0.1], ...
    'FontName', 'Yu Gothic UI Semilight',...
    'FontSize', 10, ...
    'Visible', 'off', ...
    'Callback', '');
assignin('base', 'Dev2_Label', Dev2_Label);

Plot2 = uicontrol( 'Style','pushbutton', ...
    'Units','normalized', ...
    'String', 'Plot', ...
    'Position', [0.09 0.13 0.05 0.05], ...
    'BackgroundColor', [0.74 0.74 0.74], ...
    'ForegroundColor', [0 0 0.1], ...
    'FontName', 'Yu Gothic UI Semilight',...
    'FontSize', 10, ...
    'Visible', 'off', ...
    'Callback', @Plot_dev2_CallBack);
assignin('base', 'Plot2', Plot2);

% Integral gui

% The upper limit of the integral
Int_Label = uicontrol('Style', 'Text', ...
    'Units', 'normalized', ...
    'String', 'Choose the upper integration limit:', ...
    'Position', [0.02 0.3 0.2 0.03], ...
    'BackgroundColor', [0.74 0.74 0.74], ...
    'ForegroundColor', [0 0 0.1], ...
    'FontName', 'Yu Gothic UI Semilight',...
    'FontSize', 10, ...
    'Visible', 'off', ...
    'Callback', '');
assignin('base', 'Int_Label', Int_Label);

Integral_up_limit = uicontrol('Style','slide',...
    'Units','normalized',...
    'Position',[0.04 0.27 0.15 .03],...
    'Min',1 , 'Max',100, 'Value',1 ,...

```

```

        'BackgroundColor', [0.74 0.74 0.74], ...
        'ForegroundColor', [0 0 0.1], ...
        'FontName', 'Yu Gothic UI Semilight',...
        'FontSize', 10, ...
        'Visible', 'off', ...
        'Callback', @int_of_data) ;
assignin('base', 'Integral_up_limit', Integral_up_limit);

min_lim = uicontrol('Style','text',...
    'Units','normalized',...
    'String',num2str(get(Integral_up_limit,'Min')), ...
    'Position',[0.02 0.27 0.02 .03],...
    'BackgroundColor', [0.74 0.74 0.74], ...
    'ForegroundColor', [0 0 0.1], ...
    'FontName', 'Yu Gothic UI Semilight',...
    'FontSize', 10, ...
    'Visible', 'off', ...
    'Callback','');
assignin('base', 'min_lim', min_lim);

max_lim = uicontrol('Style','text',...
    'Units','normalized',...
    'String',num2str(get(Integral_up_limit,'Max')), ...
    'Position',[0.19 0.27 0.03 0.03],...
    'BackgroundColor', [0.74 0.74 0.74], ...
    'ForegroundColor', [0 0 0.1], ...
    'FontName', 'Yu Gothic UI Semilight',...
    'FontSize', 10, ...
    'Visible', 'off', ...
    'Callback','');
assignin('base', 'max_lim', max_lim);

% Definite integral value
Int_val = uicontrol('Style', 'Text', ...
    'Units', 'normalized', ...
    'String', 'The value of the definite integral: 0 ', ...
    'Position', [0.02 0.18 0.2 0.06], ...
    'BackgroundColor', [0.74 0.74 0.74], ...
    'ForegroundColor', [0 0 0.1], ...
    'FontName', 'Yu Gothic UI Semilight',...
    'FontSize', 12, ...
    'Visible', 'off', ...
    'Callback', '');
assignin('base', 'Int_val', Int_val);

% Average value
Avr_val = uicontrol('Style', 'Text', ...
    'Units', 'normalized', ...
    'String', 'The average value: 0 ', ...
    'Position', [0.02 0.11 0.2 0.06], ...
    'BackgroundColor', [0.74 0.74 0.74], ...
    'ForegroundColor', [0 0 0.1], ...
    'FontName', 'Yu Gothic UI Semilight',...
    'FontSize', 12, ...
    'Visible', 'off', ...

```

```
'Callback', '');
assignin('base', 'Avr_val', Avr_val);

% Fourier gui

% The fundamental freq.
Fun_Freq = uicontrol('Style','Text', ...
    'Units','normalized', ...
    'String','The predominant harmonic at:   Hz', ...
    'Position',[0.02 0.28 0.2 0.06], ...
    'BackgroundColor',[0.74 0.74 0.74], ...
    'ForegroundColor',[0 0 0.1], ...
    'FontName','Yu Gothic UI Semilight',...
    'FontSize', 12, ...
    'Visible','off', ...
    'Callback','');
assignin('base', 'Fun_Freq', Fun_Freq);

% The plot in frequency domain
Plot_freq = uicontrol( 'Style','pushbutton', ...
    'Units','normalized', ...
    'String','Plot', ...
    'Position',[0.09 0.21 0.05 0.05], ...
    'BackgroundColor',[0.74 0.74 0.74], ...
    'ForegroundColor',[0 0 0.1], ...
    'FontName','Yu Gothic UI Semilight',...
    'FontSize', 10, ...
    'Visible','off', ...
    'Callback', @four_of_data);
assignin('base', 'Plot_freq', Plot_freq);

% Menu to enter the other part of the app
Emotion_Enter = uicontrol( 'Style','pushbutton', ...
    'Units','normalized', ...
    'String','Emotions handling', ...
    'Position',[0.55 0.04 0.2 0.08], ...
    'BackgroundColor',[0.74 0.74 0.74], ...
    'ForegroundColor',[0 0 0.1], ...
    'FontName','Yu Gothic UI Semilight',...
    'FontSize', 15, ...
    'Callback', @emotion);

% Close pushbutton
uicontrol('Style','pushbutton', ...
    'Units','normalized', ...
    'String','CLOSE', ...
    'Position',[0.85 0.04 0.1 0.08], ...
    'BackgroundColor',[0.74 0.74 0.74], ...
    'ForegroundColor',[0 0 0.1], ...
    'FontName','Yu Gothic UI Semilight',...
    'FontSize', 15, ...
    'Callback','close; clc; clear all; close all;');
```

%%%

```
function Documentation_Callback(~, ~)
    % Function that opens the documentation of the project
    open('Project Documentaion.pdf');
end
```

%%%

```
function eeg_data = get_eeg_data(~, ~)
    data = readtable("EEG_Device_Data.csv");

    eeg_data_time = data.time;
    eeg_data_Att = data.Attention;
    eeg_data_Med = data.Meditation;
    eeg_data_Der = data.Derived;
    eeg_data_Pwr = data.totPwr;
    eeg_data_Al1 = data.Alpha1;
    eeg_data_Al2 = data.Alpha2;
    eeg_data_Bt1 = data.Beta1;
    eeg_data_Bt2 = data.Beta2;
    eeg_data_Del = data.Delta;
    eeg_data_Gm1 = data.Gamma1;
    eeg_data_Gm2 = data.Gamma2;
    eeg_data_Tht = data.Theta;

    eeg_data = {eeg_data_Att, eeg_data_Med, eeg_data_Der, eeg_data_Pwr, ...
        eeg_data_Al1, eeg_data_Al2, eeg_data_Bt1, eeg_data_Bt2, eeg_data_Del, ...
        eeg_data_Gm1, eeg_data_Gm2, eeg_data_Tht, eeg_data_time};

    % To save the data in the workspace
    assignin('base', 'eeg_data', eeg_data);
end
```

%%%

```
function In_Data_Callback(~, ~)

    Data_Check = evalin('base', 'Data_Check');
    Button_Check = evalin('base', 'Button_Check');

    % Increment a counter to keep track of button presses
    persistent buttonPressCount;
    if isempty(buttonPressCount)
        buttonPressCount = 0;
    end

    buttonPressCount = buttonPressCount + 1;

    if buttonPressCount == 1
        get_eeg_data;
        % Ensures the user about the data
        set(Data_Check, 'String', 'The Data was introduced!');
        disp('The Data was introduced!');
    end
```

```

end

% Check the number of button presses and shows a message to the user
if buttonPressCount >= 10 && buttonPressCount < 20
    set(Button_Check, 'String', 'STOP PRESSING THE BUTTON!', ...
        'Visible', 'on');
elseif buttonPressCount >= 20 && buttonPressCount < 50
    set(Button_Check, 'String', 'SERIOUSLY STOP!', ...
        'Visible', 'on');
end

% Save buttonPressCount in the MATLAB workspace
assignin('base', 'buttonPressCount', buttonPressCount);

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function display_initial_value(src, ~, checkboxIndex)
    eeg_data = evalin('base', 'eeg_data');
    axes_plot = evalin('base', 'axes_plot');
    data_Names = evalin('base', 'data_Names');
    data_boxes = evalin('base', 'data_boxes');

    % Initialize cell array to store line handles
    persistent data_boxes_handles
    if isempty(data_boxes_handles)
        data_boxes_handles = cell(1, numel(data_boxes));
    end

    persistent data_title_handles
    if isempty(data_title_handles)
        data_title_handles = cell(1, numel(data_boxes));
    end

    persistent data_yaxis_handles
    if isempty(data_yaxis_handles)
        data_yaxis_handles = cell(1, numel(data_boxes));
    end

    if get(src, 'Value') == 1

        % When Checkbox is checked, plot the graph
        data_boxes_handles{checkboxIndex} = plot(axes_plot, eeg_data{13}, ...
            eeg_data{checkboxIndex}, 'LineWidth', 1);
        xlabel('time_{sec}');
        grid on;
        data_title_handles{checkboxIndex} = title([data_Names{checkboxIndex}]);
        data_yaxis_handles{checkboxIndex} = ylabel('');

        color{checkboxIndex} = get(data_boxes_handles{checkboxIndex}, ...
            'Color');

        if checkboxIndex <= 2
            ylabel('%');
        end
    end
end

```

```

        hold off;
    elseif checkboxIndex == 3
        ylabel('binary');
        hold off;
    elseif checkboxIndex == 4
        ylabel('\muW');
        hold off;
    else
        ylabel('\muV');
        hold on;
        set(data_boxes{checkboxIndex}, 'BackgroundColor', ...
            color{checkboxIndex});
    end
else
    % Clear the Y-axis and the Title
    delete(data_boxes_handles{checkboxIndex});
    delete(data_title_handles{checkboxIndex});
    delete(data_yaxis_handles{checkboxIndex});
    set(data_boxes{checkboxIndex}, 'BackgroundColor', [0.74 0.74 0.74]);
    xlabel('time_{sec}');
    grid on;
end
end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function Signal_Analysis_Callback(source, ~)
    % Signal anlysis menu function
    buttonPressCount = evalin('base', 'buttonPressCount');

    selected = 1;
    assignin('base', 'selected', selected);

    if buttonPressCount >= 1

        %Derivative gui
        Choose = evalin('base', 'Choose');
        Choose_Label = evalin('base', 'Choose_Label');
        Dev1_Label = evalin('base', 'Dev1_Label');
        Plot1 = evalin('base', 'Plot1');
        Dev2_Label = evalin('base', 'Dev2_Label');
        Plot2 = evalin('base', 'Plot2');

        set(Choose, 'Visible', 'off');
        set(Choose_Label, 'Visible', 'off', 'FontSize', 10, ...
            'String', 'Choose the signal to differentiate:');
        set(Dev1_Label, 'Visible', 'off');
        set(Plot1, 'Visible', 'off');
        set(Dev2_Label, 'Visible', 'off');
        set(Plot2, 'Visible', 'off');

        %Integral gui
        Int_Label = evalin('base', 'Int_Label');
        Integral_up_limit = evalin('base', 'Integral_up_limit');
        max_lim = evalin('base', 'max_lim');
    end
end

```

```

min_lim = evalin('base', 'min_lim');
Int_val = evalin('base', 'Int_val');
Avr_val = evalin('base', 'Avr_val');

set(Int_Label, 'Visible', 'off');
set(Integral_up_limit, 'Visible', 'off', 'Max', 100);
set(max_lim, 'Visible', 'off', ...
    'String', num2str(get(Integral_up_limit, 'Max')));
set(min_lim, 'Visible', 'off');
set(Int_val, 'Visible', 'off');
set(Avr_val, 'Visible', 'off');

eeg_data = evalin('base', 'eeg_data');
lastValue = eeg_data{13}(end);

%Fourier gui
Fun_Freq = evalin('base', 'Fun_Freq');
Plot_freq = evalin('base', 'Plot_freq');

set(Fun_Freq, 'Visible', 'off');
set(Plot_freq, 'Visible', 'off');

% The cases handling
selected_opt = get(source, 'Value');

if selected_opt == 1
    % Initial Values
    disp('Selected Initial Values');
    cla;
    delete(title(''));
    ylabel('');
    xlabel('time_{sec}');

elseif selected_opt == 2
    % Derivative
    disp('Selected Derivative');

    %Make the gui components visible
    set(Choose, 'Visible', 'on');
    set(Choose_Label, 'Visible', 'on');
    set(Dev1_Label, 'Visible', 'on');
    set(Plot1, 'Visible', 'on');
    set(Dev2_Label, 'Visible', 'on');
    set(Plot2, 'Visible', 'on');

    % Call the differentiation functions
    Plot_dev1_CallBack;
    Plot_dev2_CallBack;
    delete(title(''));
    ylabel('');
    cla;

elseif selected_opt == 3
    % Integral

```

```

disp('Selected Integral');

%Make the gui components visible

set(Choose, 'Visible', 'on');
set(Choose_Label, 'Visible', 'on', ...
    'String', 'Choose the signal to integrate:');
set(Int_Label, 'Visible', 'on');
set(Integral_up_limit, 'Visible', 'on', 'Max', lastValue);
set(max_lim, 'Visible', 'on', ...
    'String', num2str(get(Integral_up_limit, 'Max')));
set(min_lim, 'Visible', 'on');
set(Int_val, 'Visible', 'on');
set(Avr_val, 'Visible', 'on');

% Call the integration function
int_of_data([]);
cla;
delete(title(''));
ylabel('');

elseif selected_opt == 4
    % Fourier Transform
    disp('Selected Fourier Transform');

    %Make the gui components visible
    set(Choose, 'Visible', 'on');
    set(Choose_Label, 'Visible', 'on', ...
        'String', ...
        'Choose the signal to apply the Fourier Transform:', ...
        'FontSize', 8);
    set(Fun_Freq, 'Visible', 'on');
    set(Plot_freq, 'Visible', 'on');

    % Call the integration function
    four_of_data([]);
    delete(title(''));
    ylabel('');
    cla;
end
end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function Choose_Callback(source, ~)
    % Makes the counter of the data variables
    cla;
    selected = get(source, 'Value');
    assignin('base', 'selected', selected);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```



```
function Plot_dev1_Callback(~, ~)
% Function that calculates and plots the first order derivative
data_Names = evalin('base', 'data_Names');
eeg_data = evalin('base', 'eeg_data');
axes_plot = evalin('base', 'axes_plot');
selected = evalin('base', 'selected');

% The first order derivative calculation
fprintf('Case %d selected\n', selected);
derivative_1{selected} = gradient(eeg_data{selected}) ./ gradient(eeg_data{13});
assignin('base', 'derivative_1', derivative_1);

%The plot
cla;
plot(axes_plot, eeg_data{13}, derivative_1{selected}, 'LineWidth', 1);
xlabel('time_{sec}');
grid on;
title(['First order derivative of ' data_Names{selected}]);
if selected <= 2
    ylabel('%');
elseif selected == 3
    ylabel('binary');
elseif selected == 4
    ylabel('\muW');
else
    ylabel('\muV');
end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function Plot_dev2_Callback(~, ~)
% Function that calculates and plots the second order derivative
data_Names = evalin('base', 'data_Names');
axes_plot = evalin('base', 'axes_plot');
eeg_data = evalin('base', 'eeg_data');
selected = evalin('base', 'selected');
derivative_1 = evalin('base', 'derivative_1');

% The second order derivative calculation
derivative_2{selected} = gradient(derivative_1{selected}) ./
gradient(eeg_data{13});
assignin('base', 'derivative_2', derivative_2);

%The plot
cla;
plot(axes_plot, eeg_data{13}, derivative_2{1,selected}, 'LineWidth', 1);
xlabel('time_{sec}');
grid on;
title(['Second order derivative of ' data_Names{selected}]);
if selected <= 2
    ylabel('%');
elseif selected == 3
    ylabel('binary');
elseif selected == 4
```

```
        ylabel('\muW');
    else
        ylabel('\muV');
    end
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
function int_of_data(source, ~)
    % Function that calculates and plots the definite and indefinite integrals
    eeg_data = evalin('base', 'eeg_data');
    axes_plot = evalin('base', 'axes_plot');
    selected = evalin('base', 'selected');
    data_Names = evalin('base', 'data_Names');
    Int_val = evalin('base', 'Int_val');
    Avr_val = evalin('base', 'Avr_val');
    fprintf('Case %d selected\n', selected);

    cla;
    % Modifies the sizes of vectors in order to be the same
    max_value = get(source, 'Value');
    eeg_max = zeros(size(eeg_data{13}));
    eeg_var_max = zeros(size(eeg_data{selected}));

    for i = 1:length(eeg_data{13})
        if eeg_data{13}(i) <= max_value
            eeg_max(i) = eeg_data{13}(i);
            eeg_var_max(i) = eeg_data{selected}(i);
        end
    end

    last_nonzero_index = find(eeg_max, 1, 'last');
    last_nonzero = find(eeg_var_max, 1, 'last');
    eeg_max = eeg_max(1:last_nonzero_index); % Time x axis
    eeg_var_max = eeg_var_max(1:last_nonzero); % Variable y axis

    % The area under the curve using the trapezoidal method
    def_int{selected} = trapz(eeg_max, eeg_var_max) / 1000000; % unit measure in V/W
not uV/uW

    %The average value
    avr{selected} = def_int{selected} / length(eeg_max);

    % Display to the user the value of the integral and the mean value
    if selected <= 2
        unit_measure = '%';
    elseif selected == 3
        unit_measure = 'logical';
    elseif selected == 4
        unit_measure = 'W';
    else
        unit_measure = 'V';
    end
end
```

```

set(Int_val, 'String', ...
    sprintf('The value of the definite integral: %.4f %s', ...
        def_int{selected}, unit_measure));
set(Avr_val, 'String', ...
    sprintf('The average value: %f %s', ...
        avr{selected}, unit_measure));

% The function that describes the indefinite integral
indef_int{selected} = cumtrapz(eeg_max, eeg_var_max);
cla;

% The plot of the indefinite integral
plot(axes_plot, eeg_max, indef_int{selected}, 'LineWidth', 1);
xlabel('time_{sec}');
grid on;
title(['Indefinite integral of ' data_Names{selected}]);
if selected <= 2
    ylabel('%');
elseif selected == 3
    ylabel('binary');
elseif selected == 4
    ylabel('\muW');
else
    ylabel('\muV');
end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function four_of_data(~,~)
    % Function that calculates and plots the Fourier Transform
    axes_plot = evalin('base', 'axes_plot');
    eeg_data = evalin('base', 'eeg_data');
    selected = evalin('base', 'selected');
    data_Names = evalin('base', 'data_Names');
    Fun_Freq = evalin('base', 'Fun_Freq');
    fprintf('Case %d selected\n', selected);

    % Function that calculates the fft
    fs = 1024; %The Nyquist frequency
    N = length(eeg_data{selected});
    eeg_pad{selected} = [eeg_data{selected}; zeros(2000,1)]; % Initialize for Zero-
padding

    % Paramters after Zero-padding
    N2 = length(eeg_pad{selected});
    one_sided = eeg_pad{selected}(1:floor(N2/2)); % Use floor to ensure integer
division
    f_eeg = fs * (0:floor(N2/2)-1) / N2; % Adjust the limits accordingly
    eeg_ampl{selected} = abs(one_sided) / (N / 2);

    % Find the index of the maximum magnitude

```

```
[~, maxIdx] = max(eeg_ampl{selected});
% Calculate the corresponding frequency
freq_max = f_eeg(maxIdx);

% Display to the user the frequency
set(Fun_Freq, 'String', ...
    sprintf('The predominant harmonic at: %.2f Hz', freq_max));

% The plot of the Fourier Transform
plot(axes_plot, f_eeg, eeg_ampl{selected});
xlim([0, 50]); % EEG signals have low frequencies
title(['Magnitude Spectrum of ' data_Names{selected}]);
xlabel('Frequency_{Hz}');
grid on;
if selected <= 2
    ylabel('%');
elseif selected == 3
    ylabel('binary');
elseif selected == 4
    ylabel('\muW');
else
    ylabel('\muV');
end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function emotion(~, ~)
    % Funtion that opens and runs the other menu
    run Emotions_Interface.m;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Reads the emotion data from the dataset

%Anger 1
anger1_data = readmatrix("EEG_ANGER_1.csv");
%Anger 2
anger2_data = readmatrix("EEG_ANGER_2.csv");

%Disgust 1
discust1_data = readmatrix("EEG_DISCUST_1.csv");
%Disgust 2
discust2_data = readmatrix("EEG_DISCUST_2.csv");

%Fear 1
fear1_data = readmatrix("EEG_FEAR_1.csv");
%Fear 2
fear2_data = readmatrix("EEG_FEAR_2.csv");

%Joy 1
joi1_data = readmatrix("EEG_JOI_1.csv");
%Joy 2
```

```
joi2_data = readmatrix("EEG_JOI_2.csv");

%Sadness 1
sadness1_data = readmatrix("EEG_SADNESS_1.csv");
%Sadness 2
sadness2_data = readmatrix("EEG_SADNESS_2.csv");

% Gets the average value of each emotion

%Average values for Anger

%Avera values for Anger 1
anger1_mean = zeros(1,8);
for i = 3 : 10
    anger1_mean(i) = mean(anger1_data(:,i));
end

%Avera values for Anger 2
anger2_mean = zeros(1,8);
for i = 3 : 10
    anger2_mean(i) = mean(anger2_data(:,i));
end

%Avera values for Anger
anger_mean = zeros(1,8);
for i = 3 : 10
    anger_mean(i) = (anger1_mean(i) + anger2_mean(i))./2;
end

%Average values for Disgust

%Avera values for Disgust 1
discust1_mean = zeros(1,8);
for i = 3 : 10
    discust1_mean(i) = mean(discust1_data(:,i));
end

%Avera values for Disgust 2
discust2_mean = zeros(1,8);
for i = 3 : 10
    discust2_mean(i) = mean(discust2_data(:,i));
end

%Avera values for Disgust
discust_mean = zeros(1,8);
for i = 3 : 10
    discust_mean(i) = (discust1_mean(i) + discust2_mean(i))./2;
end

%Average values for Fear

%Avera values for Fear 1
```

```

fear1_mean = zeros(1,8);
for i = 3 : 10
    fear1_mean(i) = mean(fear1_data(: ,i));
end

%Avera values for Fear 2
fear2_mean = zeros(1,8);
for i = 3 : 10
    fear2_mean(i) = mean(fear2_data(: ,i));
end

%Avera values for Fear
fear_mean = zeros(1,8);
for i = 3 : 10
    fear_mean(i) = (fear1_mean(i) + fear2_mean(i))./2;
end

%Average values for Joy

%Avera values for Joy 1
joi1_mean = zeros(1,8);
for i = 3 : 10
    joi1_mean(i) = mean(joi1_data(: ,i));
end

%Avera values for Joy 2
joi2_mean = zeros(1,8);
for i = 3 : 10
    joi2_mean(i) = mean(joi2_data(: ,i));
end

%Avera values for Joy
joi_mean = zeros(1,8);
for i = 3 : 10
    joi_mean(i) = (joi1_mean(i) + joi2_mean(i))./2;
end

%Average values for Sadness

%Avera values for Sadness 1
sadness1_mean = zeros(1,8);
for i = 3 : 10
    sadness1_mean(i) = mean(sadness1_data(: ,i));
end

%Avera values for Sadness 2
sadness2_mean = zeros(1,8);
for i = 3 : 10
    sadness2_mean(i) = mean(sadness2_data(: ,i));
end

%Avera values for Sadness

```

```
sadness_mean = zeros(1,8);
for i = 3 : 10
    sadness_mean(i) = (sadness1_mean(i) + sadness2_mean(i))./2;
end

% Writes the treshhold values of emotions into a cvs file
writematrix(anger_mean,'EEG_TRESHOLD.csv');
writematrix(discust_mean,'EEG_TRESHOLD.csv','WriteMode','append');
writematrix(fear_mean,'EEG_TRESHOLD.csv','WriteMode','append');
writematrix(joi_mean,'EEG_TRESHOLD.csv','WriteMode','append');
writematrix(sadness_mean,'EEG_TRESHOLD.csv','WriteMode','append');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Emotions interface

Sub_Menu = figure('Name','Emotions', ...
    'Units','normalized', ...
    'Position',[0.04 0.058 0.75 0.835], ...
    'NumberTitle','off', ...
    'Color',[0.78 0.78 0.78]);

%Empty plot
plotPosition = [0.33 0.2 0.64 0.75];
plotaxes = axes('Position', plotPosition);
getframe;
grid on;
xlabel('time_{sec}');
ylabel(plotaxes, '\muV');
assignin('base', 'plotaxes', plotaxes);

%Empty image
imgPosition = [0.02 0.5 0.25 0.35];
imgaxes = axes('Position', imgPosition);
Initial_Image = imread("No_Image_Start.png");
imshow(Initial_Image);

% Emotion Label + Emotion text box
Initial_Emotion_Label = uicontrol('Style','text', ...
    'Units','normalized', ...
    'String','Emotions:', ...
    'Position',[0.02 0.43 0.25 0.03], ...
    'BackgroundColor',[0.74 0.74 0.74], ...
    'ForegroundColor',[0 0 0.1], ...
    'FontName','Yu Gothic UI Semilight',...
    'FontSize',12, ...
    'Callback','');

max_emot = 5; % Total number of emotions
emot_boxes = zeros(max_emot, 1);
assignin('base', 'emot_boxes', max_emot);
assignin('base', 'max_emot', max_emot);
```

```
emotion_Names = {'Anger', 'Disgust', 'Fear', 'Joy', 'Sadness'};
assignin('base', 'emotion_Names', emotion_Names);
```

```
for i = 1 : max_emot
    % Use a function create the emotion interface
    emot_boxes(i) = uicontrol('Style','text', ...
        'Units', 'normalized', ...
        'String', emotion_Names{i}, ...
        'Position', [0.02 0.43 - i * 0.03 0.25 0.03], ...
        'BackgroundColor', [0.74 0.74 0.74], ...
        'ForegroundColor', [0 0 0.1], ...
        'FontName', 'Yu Gothic UI Semilight',...
        'FontSize', 10, ...
        'Callback', '');
end
assignin('base', 'emot_boxes', emot_boxes);
```

```
% Play button for the animation
Play_Button = uicontrol( 'Style','pushbutton', ...
    'Units','normalized', ...
    'String', 'Play', ...
    'Position', [0.1 0.2 0.1 0.065], ...
    'BackgroundColor', [0.74 0.74 0.74], ...
    'ForegroundColor', [0 0 0.1], ...
    'FontName', 'Yu Gothic UI Semilight',...
    'FontSize', 15, ...
    'Callback', @Play_CallBack);
```

```
% Close pushbutton
uicontrol('Style','pushbutton', ...
    'Units','normalized', ...
    'String','CLOSE', ...
    'Position', [0.85 0.08 0.1 0.08], ...
    'BackgroundColor', [0.74 0.74 0.74], ...
    'ForegroundColor', [0 0 0.1], ...
    'FontName', 'Yu Gothic UI Semilight',...
    'FontSize', 15, ...
    'Callback','close');
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
function Play_CallBack(~, ~)
    % Function that display the emotions
    compare_to_data_set;
    emotion_values_percentage = evalin('base', 'emotion_values_percentage');
    emot_boxes = evalin('base', 'emot_boxes');
    eeg_data = evalin('base', 'eeg_data');
    emotion_Names = evalin('base', 'emotion_Names');
    max_emot = evalin('base', 'max_emot');
    plotaxes = evalin('base', 'plotaxes');
```



```
% Calculates the sum of all eeg signals
eeg = zeros(1,8);
for i = 4 : 12
    eeg = eeg + (eeg_data{i});
end

% Variable that stores the max value emotion in each iteration
[~, emt_img_1] = max(transpose(emotion_values_percentage));

% Function that plots and shows the results
for i = 1 : length(eeg_data{2}) % loop for the session length
    for j = 1 : max_emot % loop for fi
        set(emot_boxes(j), 'String', ...
            strcat(emotion_Names{j}, sprintf(':', %.2f%%',
emotion_values_percentage(i, j))));
    end
    plot(plotaxes, 1 : i, eeg(1 : i), 'LineWidth', 1); % Plots the sum of the
signals
    grid(plotaxes, 'on');
    xlabel(plotaxes, 'time_{sec}');
    ylabel(plotaxes, '\muV');
    drawnow; % Makes an animation
    pause(0.09);

    switch emt_img_1(i)
        case 1
            Anger_Image = imread("Angry_emj.png");
            imshow(Anger_Image);
        case 2
            Discust_Image = imread("Discust_emj.png");
            imshow(Discust_Image);
        case 3
            Fear_Image = imread("Fear_emj.png");
            imshow(Fear_Image);
        case 4
            Joi_Image = imread("Joi_emj.png");
            imshow(Joi_Image);
        case 5
            Sadness_Image = imread("Sadness_emj.png");
            imshow(Sadness_Image);
        otherwise
            break
    end
end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function compare_to_data_set(~, ~)
    % Function that assigns the emotions

    % Read the data
    eeg_threshold = readmatrix("EEG_THRESHOLD.csv");
    eeg_data = evalin('base', 'eeg_data');
```

```
% Convert the cell data to matrix data
eeg_data_mat = cell2mat(eeg_data);
out_emotion = zeros(length(eeg_data{2}), 5); % The initialization of out_emotion
outside the loop

for i = 1 : length(eeg_data{2}) % loop for the rows of eeg_data_mat
    sum_mat = zeros(5, 1); % sum_mat resets inside the loop for each iteration
    for j = 1 : 5 % loop for the emotion threshold
        for k = 3 : 10 % loop for the eeg signals
            % Calculates the difference between the data and the threshold
            sum_mat(j) = sum_mat(j) + (eeg_data_mat(i, k + 1) - eeg_threshold(j,
k));
        end
    end
    out_emotion(i, :) = sum_mat; % Store the results for each iteration in the
corresponding row of the emotion
end

emotion_values_percentage = zeros(length(eeg_data{2}), 5);

for i = 1:length(eeg_data{2})
    emotion_values = out_emotion(i, :);

    % Function that transforms all values into percentages
    row_values_percentage = (abs(emotion_values) / sum(abs(emotion_values))) *
100;

    % Store the percentage values for the current iteration in the matrix
    emotion_values_percentage(i, :) = row_values_percentage;
end

% Writes the data in the Workspace
assignin('base', 'emotion_values_percentage', emotion_values_percentage);

end
```

Others

It took me 2 days to learn how to pronounce Electrography, and sometimes I still struggle with it, so this is another reason why medicine is not for me.

There is an easter egg in the program, if the user starts to press the “Introduce the eeg data!” button at least 10 times the message “STOP PRESSING THE BUTTON!” will appear on the screen, and if he insists and goes to 20 times or above the message “SERIOUSLY STOP!” will appear.

I got the idea to buy the headset of eBay and make a project out of it from a guy who connected the acceleration pedal of his car to his mind. This is the video:

<https://www.youtube.com/watch?v=mPbtR4vorgY&t=586s>.