Text Generation from Steady State Visual Evoked Potential based Brain Computer Interface

2015-FYP-022

Submitted by:

Fizza Farooq	2015-EE-004
Khadija Maryam	2015-EE-005
Haseeb Ashraf	2015-EE-021
Muhammad Ammar Malik	2015-EE-030

Supervised by: Prof. Muhammad Tahir

Department of Electrical Engineering

University of Engineering and Technology

Lahore

Text Generation from Steady State Visual Evoked Potential based Brain Computer Interface

Submitted to the faculty of the Electrical Engineering Department
of the University of Engineering and Technology Lahore
in partial fulfillment of the requirements for the Degree of

Bachelor of Science

in

Electrical Engineering.

Internal Examiner

External Examiner

Final Year Project
Coordinator

Department of Electrical Engineering

University of Engineering and Technology Lahore

Declaration

We declare that the work contained in this thesis/report is our own, except where explicitly stated otherwise. In addition this work has not been submitted to obtain another degree or professional qualification.

Signed:

_____

_____

_____

_____

Date: _____

Acknowledgments

Firstly, we are grateful to Allah Almighty for granting us this opportunity to work on such a project and explore the depths of our own brains. Secondly, we would like to thank our worthy Advisor Prof. Muhammad Tahir for his through out guidance, suggestions, reviews, constant help and also for lending us a work-place to do our work in a peaceful environment. Thanks to Bio Informatics Lab (BIL), KICS Department, for providing us the eletrode gel. Thanks to NWN lab, KICS Department, for letting us use the 3D printer. Last, but not the least we are thankful to all our friends and family for moral support all this time.

Dedicated to our Beloved Parents and Respectable Teachers who have always supported us both morally and financially and guided us in every walk of life by giving us the trust we needed.

Contents

List of Figures

Abbreviations

BCI	**B**rain **C**omputer **I**nterface
CCA	**C**anonical **C**orrelation **A**nalysis
EEG	**E**lectro **E**ncephalo **G**raphy
ERP	**E**vent **R**elated **P**otential
ITR	**I**nformation **T**ransfer **R**ate
LSL	**L**ab **S**treaming **L**ayer
SNR	**S**ignal to **N**oise **R**atio
SSVEP	**S**teady **S**tate **V**isually **E**voked **P**otential
WiFi	**Wi**reless to **Fi**delity

Abstract

The aim of this project is to use platform of Brain Computer Interface (BCI) towards application of a BCI speller. BCI as the name suggests is a mode of communication made possible by translating the brain's electrical activity into commands for the computer hence eliminating the dependency of neuromuscular activity which is needed otherwise. So, it is a possible communication method for motor-impaired patients resulting from a spinal cord injury. The Event Related Potential (ERP) technique is one of the BCI control strategies i.e. occurrence of an event helps to enhance the accuracy of brain wave acquisition by making it prominent. The Visual Evoked Potential (VEP) technique is one of the ERP techniques which can be further divided into Steady State Visual Evoked Potential (SSVEP) and P300 (P for positive, 300 for 300 ms). In SSVEP, visual stimuli modulated at certain frequencies excites retina to generate potentials at those frequencies. This visual stimulus behaves as an input to the brain and the signals from the brain will be the output i.e. there will be some reflection of the input stimuli frequency and its harmonics in the brain waves. Hardware of this project constitutes of an electrode cap with 8-electrodes, a board (Cyton board) for EEG signals acquisition which sends data to computer over WIFI using WIFI Shield. Electrodes are placed at occipital lobe at the back of head site. The project aims to implement control of a speller using SSVEP technique stimulated by speller screen. Speller screen is divided into 4 sections each flickering with different frequency. The alphabets are divided in these sections and the desired alphabet is approached by the selection of one of the sections and further using divide and conquer technique.

Chapter 1

Introduction

Many types of neurological disorders can disconnect brain's both verbal and non-verbal communication with the rest of the body by disrupting the neuromuscular channels. Patients who have some remaining motor skills through supportive techniques have a chance to regain contact but in some cases like hemorrhage, muscular dystrophies and spinal cord injury, they suffer from complete paralysis which do not allow them to communicate with the outer world. The technique like BCI is essential for such patients. In this study, we implemented the SSVEP based BCI to make a BCI Speller application.

1.1 Motivation

As Pakistan is still a developing country, we wish to introduce this technology and open up doors to new research areas here that have never been worked on before and to provide a platform for more research entities in field of BCI and its relevant techniques. Along with our spirit of patriotism and problem solving cravings, it's a basic human instinct to be curious about ones own body and much more in the most intriguing organ of body, the brain.

1.2 Problem Statement

There are over 10 million people in Pakistan alone which is about 5% of the Pakistan's population who are suffering from paralysis and can't move around or even write [5]. In order to help physically challenged people control a computer, communicate with others and to be able to send text messags, this project aims to use BCI which will help them to generate and send text by just looking at the alphabets on laptop screen.

1.3 Objectives

This project aims:

- To develop an EEG board from open source design for EEG signal acquisition.
- To implement SSVEP based BCI with the development of BCI Speller app that allows paralyzed people to send text messages to cell phones.

The goal of the project is to present a complete system that can be used both in hospitals and in homes, which can help these people in regaining some mobility so that they can gain back their individuality in this world.

1.4 Overview

Chapter 2 is comprising of the literature survey that starts by introducing all related terms one needs to comprehend and understand in this project, a brief history to BCI and then jumping to the explanation of technology we are focusing in this project. It also presents a fair history about SSVEP technique and scope of using this technique.

Chapter 3 details the description of our complete hardware setup, from making of PCB to soldering components on it, to testing the hardware and completely ensuring the connectivity of whole apparatus.

Chapter 4 explains the software part that includes how we configured our setup all together and integrated hardware and software parts. The software level filters and signal processing on EEG data. Feature extraction techniques and classifier model used to make this project work. And finally about the design of SSVEP based Speller app.

Chapter 5 is regarding experimental setup, tests performed and result compilations of this project.

Lastly, Chapter 6 will take you through the conclusion of this project and future directions.

Chapter 2

Literature Survey

2.1 Brain's Neural Network

Brain (central nervous system) is the part of body which controls all the activities going on in the body. The control is facilitated by the network of neurons scattered throughout the body. These neurons actually provide a channel for the brain, connecting parts of the body with it. This network of neurons consisting of billions of neurons that senses different activities from all parts of the body and passes on this information to the brain, after processing these signals makes a decision and sends commands to a specific part of the body through the cells of neurons [6]. This procedure is performed in nanoseconds because these signals are electric hence have some potential differences.

2.2 Brain Computer Interface

In BCI, the communication with the computer is not carried by neuromuscular activity rather the voltage potentials produced by the neural activities are captured, recorded, made sense of and used accordingly to give commands to computers or devices [7]. Hans Berger was the first one to discover the electrical movement of the human brain which led to the development of electroencephalography (EEG) technique. The first step in a BCI system is recording the electrical activity of the brain which can be done invasively (electrodes are surgically implanted in the brain) or non-invasively (electrodes are just placed on the scalp using the electrode cap).

2.2.1 Invasive Technique

Invasive technique means implant the micro-electrode array in the brain surgically to record neuron activity. This technique has been practiced by several laboratories to record signals with electrodes implanted in brains of monkeys and their studies show that monkeys were able to use BCIs to produce movements and they navigated computer cursors on screen. They were also able to control robotic arm to make simple movements [8].

2.2.2 Non-invasive Technique

In non-invasive technique, electrodes are placed on the scalp of a patient at different sites and the currents due to excited neurons in any cortex are captured through those electrode.

2.2.3 Types of Electrode

Electrodes are used to make good conductive contact with the scalp to collect better brain signals. It consists of a metal plate covered by a conducting paste or gel on the inside contact surface and is connected to the EEG board through a wire. Two types of electrodes, wet and dry are used in non-invasive technique.

2.2.3.1 Wet Electrodes

The wet electrodes need preparation of skin by applying a good amount of conductive gel to decrease the skin-to-electrode resistance. The main advantages of a gel based active electrode is that it's signal strength is really better as compared to dry electrode. The electrode-skin impedance should be less than 5k ohms for the wet electrodes to operate optimally which is often not achieved and is quite problematic. For wet electrodes one needs to prepare the skin before and after use, the residuals of the gel needs to be cleaned as well. So overall these preparations are time taking and it can also cause discomfort to many patients. Conductive gel is very thin in it's texture so it can also slip out from the EEG electrodes sites and cause a short circuit between two electrodes. So, in projects which require large number of electrodes dry electrodes are preferred.

2.2.3.2 Dry Electrodes

For projects based on EEG that need large number of electrodes in range 64-128, it's not suitable to use wet electrodes. Dry electrodes that do not need gel for better conductivity are used instead of wet electrodes. A dry electrode has higher electrode-to-skin resistance and small skin contact as compared to wet electrodes. Dry electrodes are also more costly than wet ones [**?**]. Since, this project needs

only eight channels so we are using wet electrodes.

2.2.4 EEG Montages

There are two types of EEG montages

1. Bi-polar Montage
2. Mono-polar Montage

2.2.4.1 Bi-polar Montage

In bi-polar montage, the potential difference is measured between two pair of electrodes. All channels have a single reference which may be ear electrode or cortex electrode.

2.2.4.2 Mono-polar Montage

In mono-polar montage, an electrode channel is chosen as reference electrode and each other channel shows the difference between itself and that reference electrode channel.

2.2.5 Artifacts

In any BCI project the main challenge that has to be faced is to get the right and healthy signals and that obviously is the very first hurdle as well. There are many artifacts that need to be taken care of by first recognizing and laterwards removing them. There are two types of artifacts. One is caused by the subject for example body movements, sweating of skin, eye movements, muscle clenching etc and the other are technical ones like the 50 Hz frequency of power systems in Pakistan and electrode gel impedance. All of these artifacts are dealt separately and resolved eventually. Some artifacts are simply removed by apply certain filters and at times the data is dropped when a jerk is detected. The jerky data is understood by looking at the accelerometer reading placed on the EEG board.

2.2.6 Electroencephalography (EEG)

Electroencephalography (EEG) a way of monitoring the electrical activity of the brain. It is achieved by placing electrodes on the scalp and the potential difference at that site is captured through that electrode, so it is a non-invasive method. EEG is also called recording of brain electrical activity over a period of time [8].

EEG studies are further divided into motor imagery and ERP (Event related potentials) which are explain in depth later.

2.2.7 Recording EEG Activity

Electrodes are placed over the scalp of the subject. The number of electrodes used in an application varies accordng to requirement. Some applications need only few and some use a large number such as 128 electrode.

These electrodes capture the potential voltages at their sites which are small in magnitude and then are passed through the process of signal processing such as amplification and filtration. Most filters are low pass and high pass filters. These signals are passed through analog-to-digital converter, ADC [9]. The electronic hardware EEG board is liable for amplifying, filtering and digitizing the EEG signal. EEG signal vacillations give different amplitudes at different frequencies. Studies have distributed these frequencies into set of ranges as given below. Each range displays different state of brain. So by looking at amplitudes corresponding to these frequency ranges one can understand the current state the brain is in. There are mainly four types of brain waves [10]

1. Delta Waves (0.4 to 3) Hz
2. Theta Waves (3 to 7) Hz
3. Alpha Waves (7 to 12) Hz
4. Beta Waves (12 to 38) Hz
5. Gamma Waves (38 to 42) Hz

2.2.8 Electrodes Placement (10-20 System)

In 10-20 international system [6], the brain is divided in various subsections each given a unique name to it, and that helps in the placement of electrodes at perfect sites while performing an experiment. Besides these electrodes placed on the head scalp, there are two more elctrodes, called bias and reference. These are placed on ear lobes and act as reference for the rest of the electrodes. Electrodes collect the potential difference from there sites and those potentials tell about the activities going on in the brain. In 10-20 system the '10' and '20' means the distance between the adjacent electrodes which is either 10% or 20% of the total distance of head. Skull is divided into different sections and each section has one alphabet and one number to specify it's site. P, C, T, F, and O stands for parietal, central, temporal, frontal, and occipital lobes. All even numbers belong to right hemisphere and odd numbers to the left hemisphere. See figure 2.1.

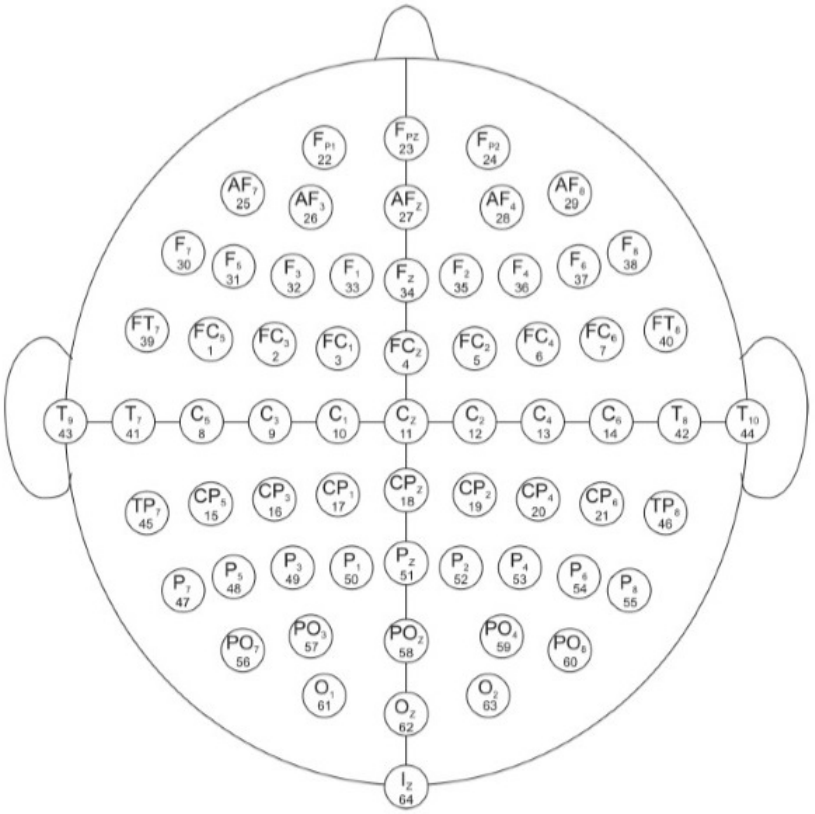


FIGURE 2.1: Electrode placement in International 10-20 System [3].

2.2.9 Motor Imagery

Motor imagery is a mental exercise in which a person practices or simulates a particular activity. It has been known to find use in sports training as a mental exercise of sport actions and neurological rehabilitation. It has been used as a research model in cognitive neuroscience and cognitive psychology to study the content and structure of hidden processes preceding the execution of an act [11, 12]. It can also be defined as a moving condition where the user is projecting an image in his mind means that the user feel himself following that movement [13].

Mental practice refers to the use of visual-motor images to improve motor behaviors in patients having troubles or for sports-men to sharpen their skills. This technique is one of well known BCI paradigms.

2.2.10 Event Related Potentials (ERP)

ERP (event related potentials) are obtained by taking derivatives of EEG over time for any specific event. These are of kind visual, somatosensory, or auditory [14]. ERP are stimulated by unexpected events, be it appearance of an image on screen or hearing any unique tone or by sudden pressing of a button and the list

goes on. All these stimuli will result in a peak in EEG pattern after some time [15]. The technique of ERP is non-surgical method to study a brain's response to certain events. ERP waveform contains many positive and negative voltage peaks. See figure 2.2.

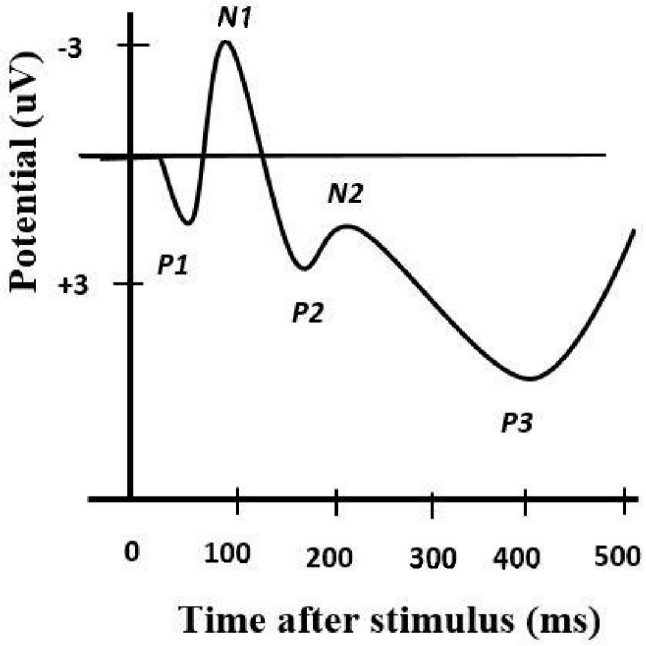


FIGURE 2.2: Some event causing a peak in EEG signal [4].

2.2.11 Visual P300

P300, here P stands for positive and 300 for 300ms delay, is another type of VEP based BCI brought by infrequent, task-relevant stimuli associated with the event 300 milliseconds after each stimulus arrival. The P300 evoked potential is very stable cerebral response belonging to the appropriate event group (ERP). For now the origin of P300 is still unknown, but it is believed that it occurs during a decision making and at times when a rare target is perceived in mind [15].

The best site for P300 is parietal zone on head scalp and the intensity of P300 gets weaken as we move further from here. There are three types of paradigms known when it comes to P300. One is oddball, second is single stimulus and the last one is three stimulus [16]. In all these paradigms, the user has to either wait for a certain stimuli to appear and during that wait user can do mental counting.

2.2.12 Steady State Visual Evoked Potential

The SSVEP is a type of Visual Evoked Potential based BCI. This technique is based on frequency coding of excitation source of the visual stimuli [17]. In these

type of systems, every target is made to flicker with different frequency, so to select any particular target. It's frequency with which it was flickering is detected in VEP as the most prominent frequency, It is possible to recognize which target the user is looking at. In figure 2.3, there are six targets each flickering with a different frequency. When user focuses at one of these, the corresponding frequency components dominates in EEG waves.

SSVEP-based BCI system is more preferred over other techniques because of its higher Information Transfer Rate (ITR) and best accuracy. Also, for application based on SSVEP short or no training time is required and it also requires fewer EEG channels.

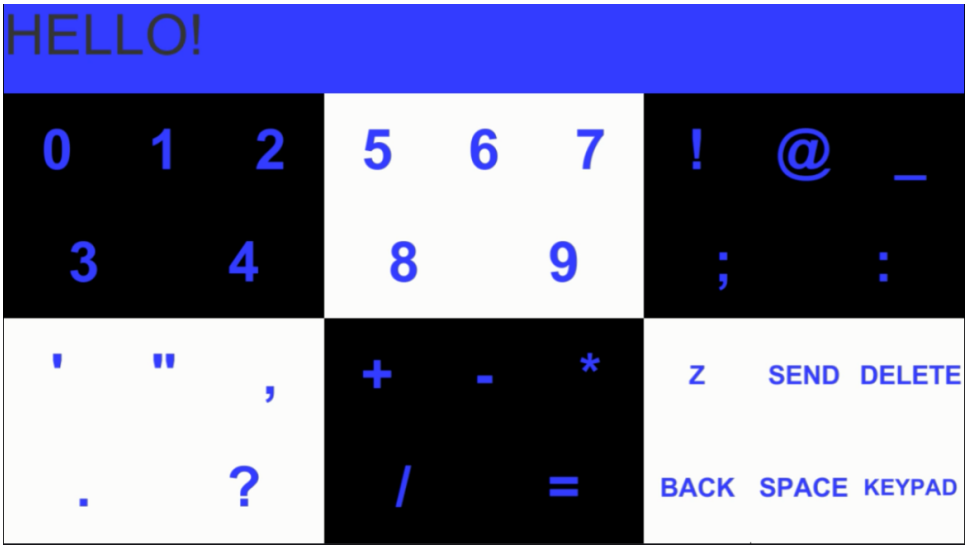


FIGURE 2.3: Text generation Speller App based on SSVEP Technique

2.3 Brief History to SSVEP based Speller

In neurology and neuroscience research, SSVEPs results in signals which project any frequency that was shown in the visual stimuli. SSVEP-based BCIs [18] are widely used, as majority of users can work with them using as few as one recording channel, minimal hardware setup, tuning, and training. Technique of SSVEP has been used extensively to study a subject's vision and in studies related to subject's attention and focus. One of the reason for good amount of research on SSVEP is due to it's high signal-to-noise ratio [19] and also because unlike other EEG techniques SSVEP is very less affected by the artifacts [20]. SSVEP-based BCIs have been proven very effective for selection tasks [21].

Researchers have worked on the development of applications that can be useful in providing a mode of communication for amyotrophic lateral sclerosis (ALS)

patients and BCI-speller is one of the most commonly studied application for that reason. The previous studied speller application focused on objective of facilitating unfortunate patients suffering from motor neuron disease (MND) who can not even communicate with their surrounding with any means, those kind of patients also became the aspiration of this project. But now-a-days the recent developments are exploring other areas by integrating technology such as Virtual Reality (VR) with BCI.

This briefing would like to describe details about some successful recently published BCI-speller systems similar to this project. The spelling system is a screen display having our typical letters, numbers and symbols [22]. And it is controlled and selected by blocks each flickering with certain frequency containing set of those letters, numbers and symbols. Spelling application are developed in order to make possible communication for paralyzed patients. Brain signals are recorded and analyzed, the subject selects and type the desired letter by just looking at it on the screen. Farwell and Donchin were the first ones to work on a spelling application in 1988 [23]. After that the research of SSVEP based speller focused on the construction of last layer of system to work on the techniques responsible for signal recognition and understanding for better accuracy.

2.3.1 Performance Constraints

There are two constraints upon which the BCI Spellers is analyzed. One is the speller accuracy and other is it's Information Transfer Rate (ITR). Accuracy is defined as number of correctly classified commands divided by the total number of commands. The ITR formula was given by Wolpaw in [24].

2.3.2 Categorizing SSVEP

There are two ways in which the BCI speller can be classified.

One is dependent or independent. The gaze dependent BCI speller are the ones where eye movement is involved. On the other hand gaze independent BCI spellers are not eye movement dependent rather they depend more on mental attention.

The other way the BCI spellers are classified is asynchronous and synchronous. In asynchronous the subject and stimuli interact more naturally, the subject does not have to wait for some sort of cue like in synchronous ones. SSVEP based BCI speller fall under this category.

2.3.3 SSVEP Spellers

Bremen was the first one to work on SSVEP-based BCI Spellers. The speller application had a diamond shaped keyboard on screen with five boxes containing total of 32 letters. A cursor was used which can move and select any box, so that the desired character can be selected. Each box segment had a unique frequency with which it was flickering. That frequency presented as a visual stimuli to the user. This paper stated it's accuracy as 93.27% and it's average ITR was 25.6 bits/min [25].

In Multi-Phase SSVEP Spellers, one research paper experimented with two groups, one with young aged subjects and other with old. In multi-phase the character is not selected in first selection rather it takes more than one selection. This paper reported for young aged group the accuracy 98.5% and for old aged group 91.13%, both groups having ITR of 27.36 bits/min and 16.10 bis/min respectively [26].

Some BCI spellers are based on row/column (RC) paradigm, it was introduced into SSVEP speller world by [27]. In this a GUI is created with 6 rows and 6 columns, so in total 6 different frequencies are used. A similar GUI application also used to combine the P300 with SSVEP and enhance the overall system's performance [28].

The above discussed researches are recent ones and SSVEP based BCI Speller is a good research area which has a lot of room for improvement.

2.4 Applications of BCI

This project will pave the path to endless list of applications of using BCI considering the world's attention towards and obsession with technology.

1. Applications to facilitate disable or paralyzed people.
   (a) Text generation
   (b) Automated wheelchair
   (c) Home automation
2. Applications in entertainment.
   (a) Advancement in game technology
   (b) Advancement in entertainment industry
3. Applications in education and training.

4. Applications in health.

    (a) Developing a system that can monitor mental state or diagnose mental illness.

Chapter 3

Hardware Setup

The complete setup has following components

1. Electrode Cap
2. Cyton Board
3. WIFI Shield

The hardware setup of this project includes putting on the electrode cap on head and the electro-gel placement for conductivity from the electrodes as shown in figure 3.1. The figure 3.2 shows the block diagram which explains the assembly

FIGURE 3.1: Complete Setup

of the hardware items. The block diagram is divided in three subsystems. The subsystem-1 named User, shows that the user is wearing the electrode cap and is looking onto the screen for visual input. The signals from subsystem-1 flow towards the subsystem-2 which is EEG board. EEG board then sends the data to subsystem 3 over WIFI. In subsystem-3, the signal processing, feature extraction and classification has been performed and a control command is sent to the app displayed. This chapter explains the setup of the above mentioned items included in the hardware. The subsystem-3 is explained in next chapter.

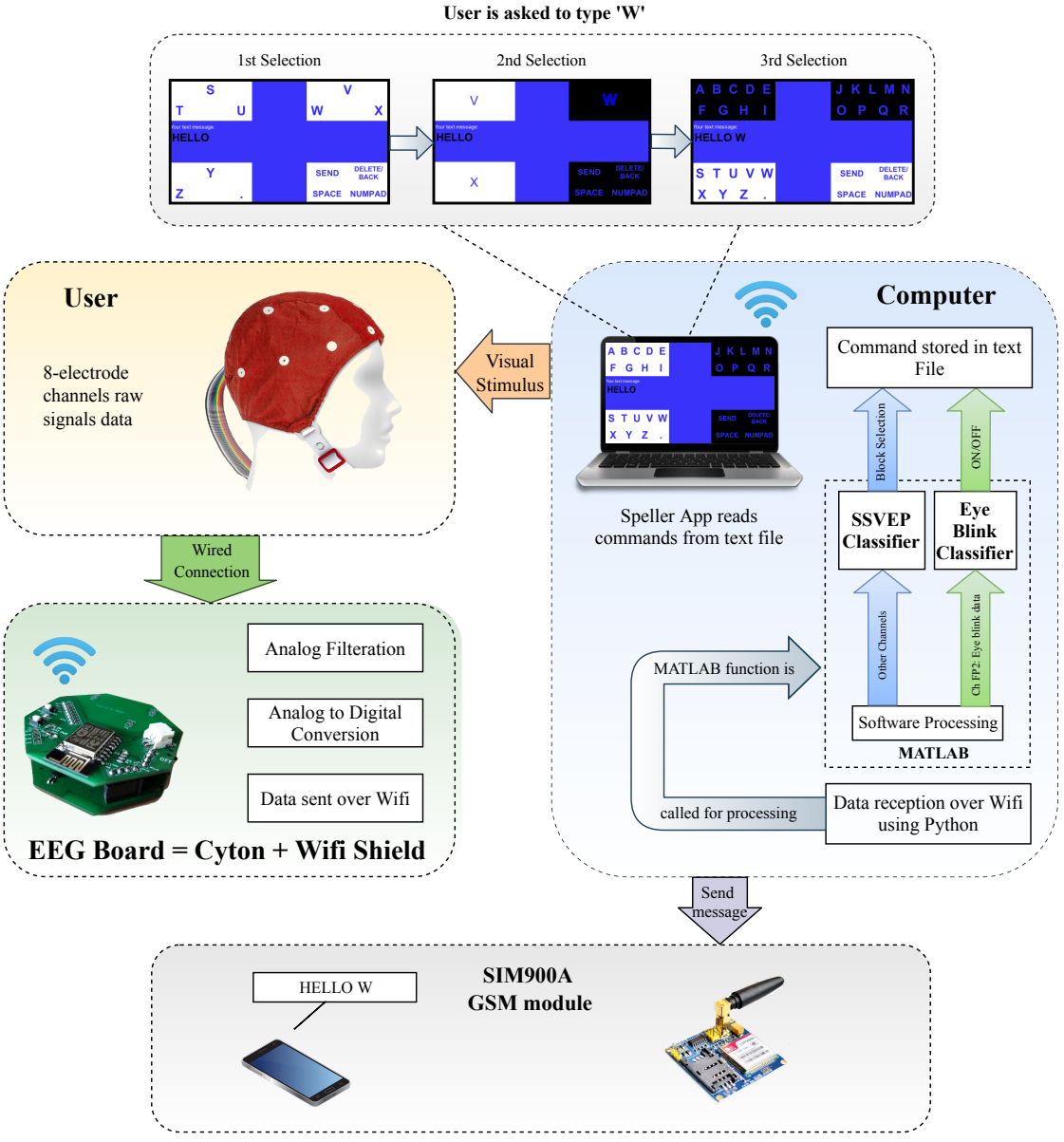


FIGURE 3.2: Block Diagram

3.1 Electrode Cap

Figure 3.3 shows the electrode cap and conducting gel. This electrode cap consists of 16 electrodes. We are using 8 electrodes out of 16. The positions of electrodes that we used are FP2, P3, P4, PO3, PO4, O1, O2, and Oz. These are tin electrodes and are wet electrodes i.e. electrodes will contact with the head through conducting gel. In addition to these electrodes two reference electrodes are used which are placed on each ear lobe. We added locally made electrodes to this electrode cap at positions PO3, PO4 and Oz. The reference electrodes are also locally made as shown in figure 3.4. To use this electrode cap, we applied the gel and checked the impedance which was around 10k.

(a) Electrode Cap (b) EEG Conducting Gel

FIGURE 3.3: Electrode Cap and Conducting Gel

FIGURE 3.4: Flat and Cup Shaped Electrodes

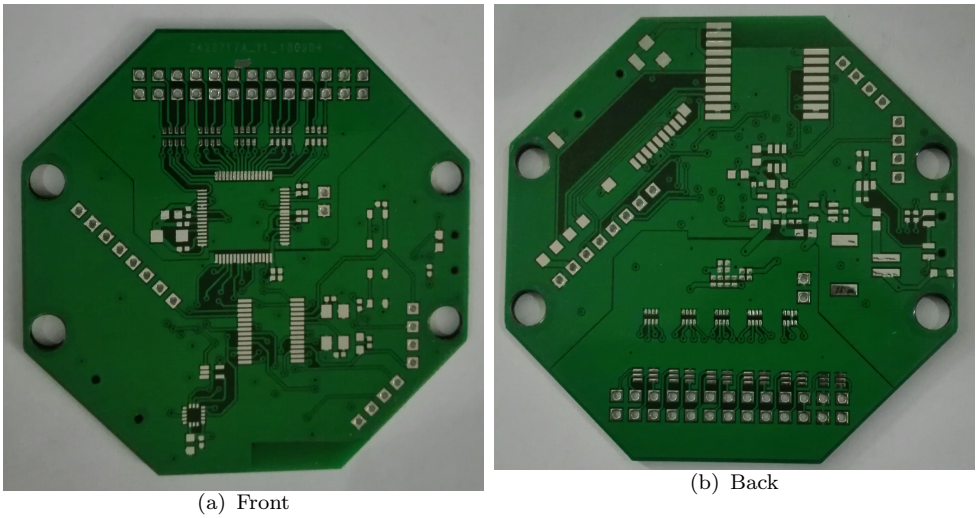
3.2 Cyton Board [1]

The design for the EEG board named as Cyton board has been taken from https://www.openbci.com which is an open source website for BCI. It is an

8-channel biosensing board. It costs for $499 so we decided to clone it from the open source design. It works with PIC32 micro-controller which communicates with 8-channel 24-bit ADC (ADS1299IPAG), SD card and WIFI shield (ESP-12E-8266) over SPI interface. The 8 signals coming from the EEG cap first passes through transient voltage suppressor IC (TPD4E1B06) then from analog RC filters to ADC. The micro-controller then reads digital signal from ADC, sends it to the WIFI module on WIFI shield and also writes it in the SD card. It can also communicate over bluetooth connection but we preferred to use WIFI over bluetooth. The SD card data can be used for offline training of machine learning model. For online working, the data is sent over WIFI to computer.

3.2.1 PCB Fabrication of Cyton Board

This PCB was fabricated from China (https://www.jlcpcb.com). The PCB is shown in figure 3.5.



(a) Front

(b) Back

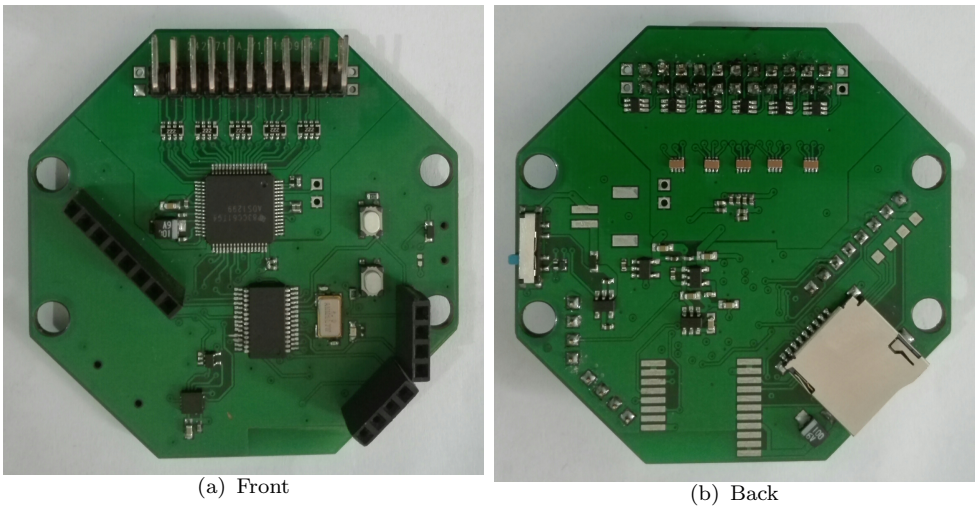
FIGURE 3.5: Cyton Board PCB Fabrication

3.2.2 PCB Soldering of Cyton Board

The PCB components were imported from Mouser Electronics and we finally soldered the components on PCB and made the PCB work. Figure 3.6 shows the PCB after soldering.

3.2.3 Uploading Bootloader and Firmware

We used the pickit 3 to upload the bootloader provided by the OpenBCI website. The bootloader allows serial communication with the micro-controller and the firmware (also provided by OpenBCI website) then was uploaded using the Rx and Tx pins of the micro-controller.



(a) Front (b) Back

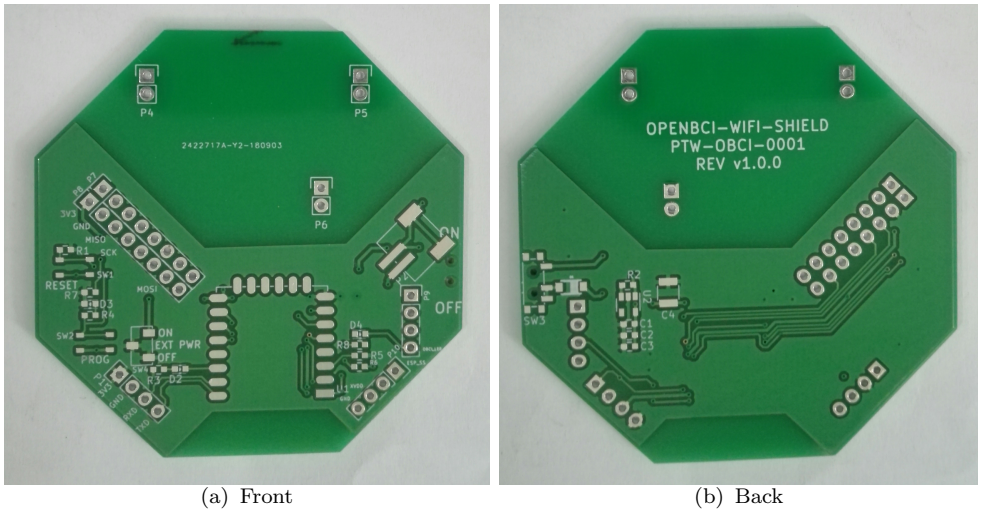
FIGURE 3.6: Cyton Board PCB after soldering

3.3 WIFI Shield [2]

WIFI shield is an extension of the Cyton board which adds the functionality of data streaming over WIFI to computer. It costs for $149 so we decided to clone it from the open source design which has been taken from the same website https://www.openbci.com. It is attachable to the Cyton board through header pins. It receives data over SPI interface from the micro-controller and then transmits over wireless connection to computer.

3.3.1 PCB Fabrication of WIFI Shield

This PCB was fabricated from China (https://www.jlcpcb.com). The PCB is shown in figure 3.7.

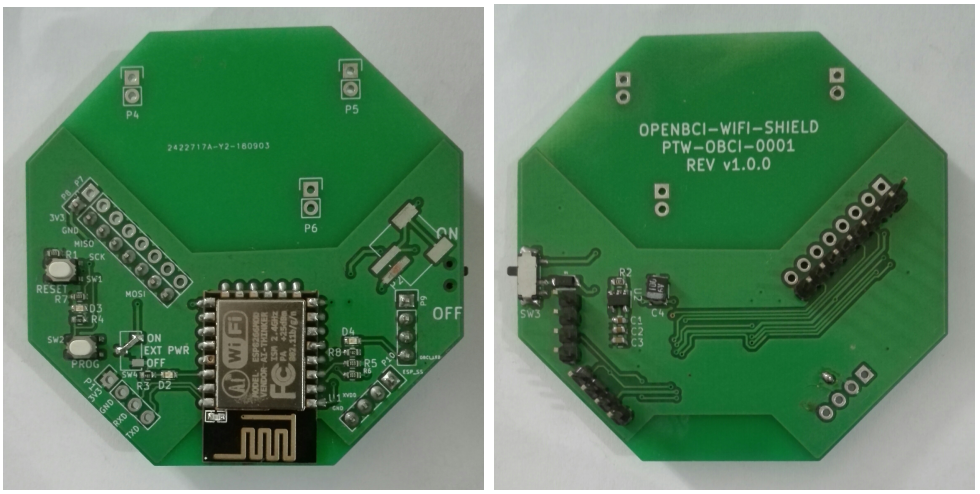


(a) Front (b) Back

FIGURE 3.7: WIFI Shield PCB Fabrication

3.3.2 PCB Soldering of WIFI Shield

The PCB components for WIFI shield were also imported from Mouser Electronics and we soldered the components on PCB and made it work with the Cyton board. 3.8 figure shows the PCB after soldering.



(a) Front (b) Back

FIGURE 3.8: WIFI Shield PCB after soldering

3.3.3 Uploading Firmware

Uploaded the firmware provided by the OpenBCI website using the serial pins on the WIFI shield.

Chapter 4

Software Design

The project aims to develop a BCI Speller app which will provide SSVEP stimulus to the user. The figure 4.1 shows the flow chart of this project. The EEG signals are captured by electrodes at different sites. The EEG signals are sampled at 1000 Hz and after processing through the EEG board are then sent to the PC where these are processed and control command is generated and sent to the app. The details of the steps to receive data from the Cyton board, its processing and the design of apps is explained in this chapter.

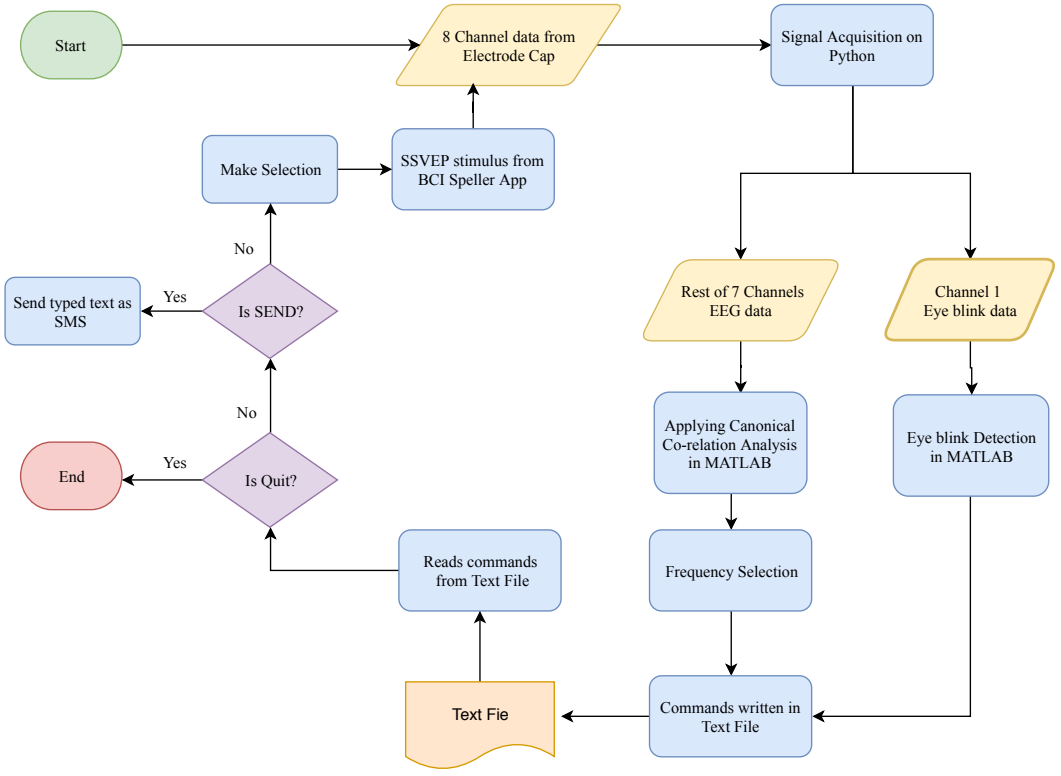


FIGURE 4.1: Flow Chart

4.1 Data Reception in PC

We used two different ways to receive data in PC for off-line and on-line working on data.

4.1.1 Data Reception for Off-line Working

We used the OpenBCI_GUI software for data reception from the EEG board. OpenBCI_GUI is also an open source software from OpenBCI website. This software can

1. Send commands over WiFi to Cyton board
2. Enable channels, set gains, sampling frequency and other settings of ADC
3. Enable data write to SD card
4. Apply notch filter of 50 or 60 Hz
5. Apply band pass filter
6. Plot 8-channel signals of EEG along with FFT
7. Plot x, y and z axes data of accelerometer
8. Send data to Lab Streaming Layer (LSL)

It has a lot of other features. We used only the above mentioned functions of this software. We set the gain of each of the 8-channels to x24, sampling frequency to 1000Hz. Enabled the accelerometer. Applied notch filter of 50Hz to cancel the humming noise. Then sent the data to LSL. LSL can be used to send data from one software to the other running on same PC.

4.1.2 Data Reception for On-line Working

The OpenBCI_GUI software uses LSL to send data to other softwares for further processing but the LSL buffer introduces lag of 15-20 seconds. We also tried by reducing the size of the LSL buffer but in vain. So, we used python for data reception in PC for on-line working. The initial driver was already available to communicate over WIFI with EEG board. We modified it to receive and process data. Python code receives data and then for further processing, it invokes the MATLAB functions.

4.2 Data Processing and Classification in MATLAB

Data processing and classification parts are same for on-line and off-line working. Two classifiers are used in this project.

4.2.1 Eye Blink Detection Classifier

The starting and closing of the app is controlled by three consecutive eye blinks. So a classifier is needed to detect the presence of three or more consecutive eye blinks in the signal. Front channel (FP2) signal is used for this purpose. The signal is first filtered through a band stop filter of range 5 to 75 Hz. Only low frequency part is needed since eye blinks lies in the low frequency part of the spectrum. The signal is first inverted about the time axis. Since the eye blink causes dip in the signal, inverting the signal makes it a peak in the signal which is then detected using MATLAB peak detector function with tuning of minimum peak prominence attribute of the function. Before peak detection, signal is down-sampled such that there are only 50 samples per second. The signal processed is 5 seconds long and data is collected in portions of 2.5 seconds. The features used are the heights of peaks, number of peaks and the distance between the peaks. We have classified the signals without using the machine learning model.

4.2.2 SSVEP Frequency Detection Classifier

The second classifier classifies the four classes corresponding to each SSVEP frequency. The signal is first passed through 4th order band stop filter of 25 to 75 Hz range then through a 4th order notch filter at 13 Hz and finally through the band pass filter of 10th order ranging from 4 to 25 Hz. These filters been selected on basis of experimental results to suppress noise part at 13 Hz, 50 Hz and near 50 Hz humming noise. The canonical correlation analysis (CCA) has been used for feature extraction [29]. CCA gives the correlation coefficients using the cross correlation between two vectors of random variables. Here the first vector is the EEG data of seven channels i.e. first vector has seven variables each of length equal to 2500 samples (we are using 1000 Hz sampling rate and data segment of 2.5 seconds for processing). The other vector contains the sine and cosine signals of same samples length and frequency equal to SSVEP frequency. The SSVEP frequencies that we found in this project are 4.3, 6, 7.6 and 10 Hz. For the first frequency, second harmonic has been used i.e. 8.6 Hz because the signal at low frequency was noisy, it was not useful even after the filtration and the second harmonic was more dominant for this case only. For other frequencies, fundamental

harmonics have been used. The CCA of first vector is calculated with second vector of sine and cosine of one frequency at a time. So, we have four sets of correlation coefficients and the sum of each set is used as a feature corresponding to each frequency. Further, a threshold is set to build the confidence about the selection of frequency. The threshold is set in a way such that the feature value for the selected frequency must be large enough as compared to other features.

4.3 SSVEP Based Speller App

SSVEP has been discussed in earlier chapter. In this technique, a visual input is flickering of the objects and the frequency of flickering is expected in EEG plot. We decided to choose four different frequencies which are not multiples of each other as mentioned in previous section. We divided the screen in four boxes and placed the alphabets, numeric and other symbols in these boxes each flickering with different frequency from other ones. The app is shown in figure 4.2. There are four targets each flickering with a different frequency. When user focuses at one of these, then the corresponding frequency components dominates in EEG waves. This app writes the alphabet in three selections. It uses the COM port to communicate with the GSM900A module. This module then sends message to the mobile number.

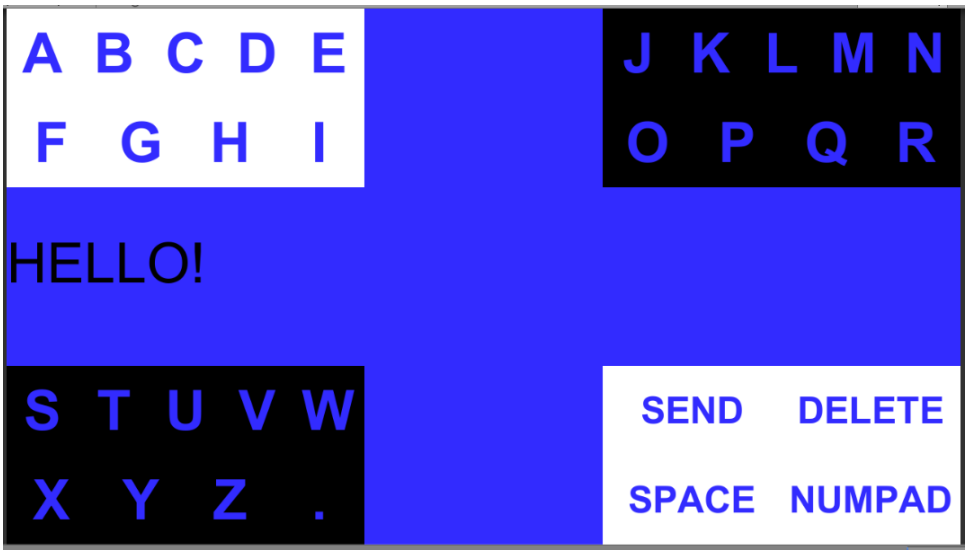


FIGURE 4.2: SSVEP based Speller

4.3.1 Control Command to Speller App

This app is designed to run on PC. The classification is done in MATLAB and the command is sent using the text file. It contains three comma separated values. First value to indicate the detection of three consecutive eye blinks, second for

the detection of SSVEP frequency and third to indicate whether the file has been modified or not. Since file is read continuously by app in each update frame that's why the third value is required. MATLAB whenever writes the file, it toggles the third value in file from one to zero and vice versa.

Chapter 5

Experiments and Results

In this chapter, we'll list all the tests and experiments that we performed to complete this project. Firstly, we made the EEG board as mentioned earlier in chapter 3. To verify the functionality of the board, we performed a test to plot the ECG signal because its shape is known already.

5.1 Board Testing by ECG Test

We used three electrodes, one for positive, other for negative and third bias terminal which behaves the same way so as to common the ground of body and the board, reducing the noise. The figure 5.1 shows the setup and the resulting signal. Then, we verified the plot for each channel one by one. Though, in the beginning two channels (channels 2 and 3) were not working. The problem for one channel (channel 3) was in soldering, which we could fix successfully. The problem with the channel 2 was not fixed completely, the negative of the channel 2 was working which was fine with us since we need only the negatives of each of the eight channels with one common reference (SRB2 pin).

5.2 Initial Tests for Each Setup

There are two basic connectivity checks performed every time before starting use of the setup to ensure that it's working properly.

1. Electrode Resistance Test
2. Electrode Impedance Test
3. Eye Blink Test
4. Jaw Clench Test

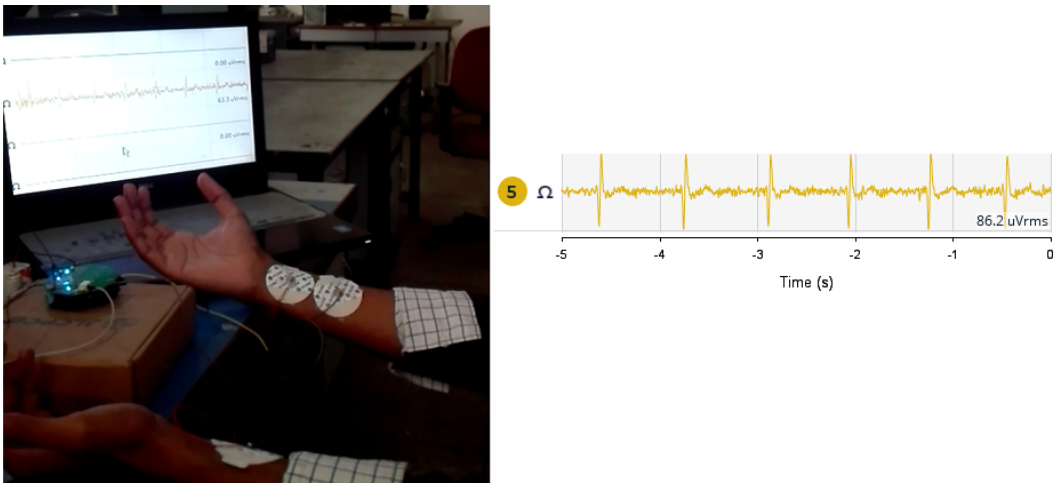


FIGURE 5.1: ECG Plotting

5.2.1 Electrode Resistance Test

The pole resistance or the DC resistance is measured when the electrical continuity is interrupted from the electrodes, the conductive wires and the connector. This measurement is made when the electrode is not attached to the scalp. The uninsulated ends of the electrodes are connected to a micrometer which passes through a small DC current through the electrode. The resistance of an intact electrode should measure no more than a few ohms.

5.2.2 Electrode Impedance Test

Electrode impedance or opposition to alternating current ow is measured application of the electrode at the recording site to evaluate the contact between electrode and scalp. Each electrode impedance should be nearly 5000 ohms.

5.2.3 Eye Blink Test

After setting up the EEG cap on the subject's head. This test is performed called the eye blink test that is just another specific pattern of a voltage dip when the subject blinks the eyes. This effect is most prominently viewed on the electrode site FP2. This test also leads us to conclude that our reference electrodes are working just fine. One more thing we take notice of here is the variation of voltage amplitude. The eye blinks pattern is shown in figure 5.2.

5.2.4 Jaw Clench Test

In this test, the subject is instructed to clench his pair of jaw against one another. A specific pattern is a result in brain signals. But this time the effect is prominent in almost all the electrode sites, but most prominent on the electrode sites at the

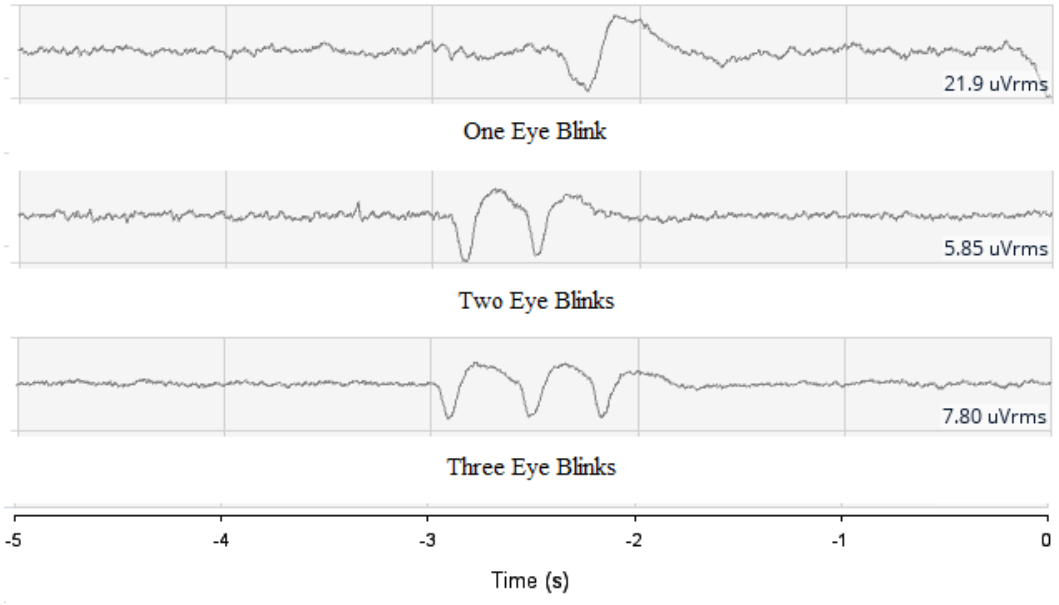


FIGURE 5.2: Eye Blinks

back of the subject head. The jaw clench pattern is shown in figure 5.3. This test insures the perfect working of all the electrode sites. Meanwhile the voltage amplitude is also noted and it shots up if one was to press their finger against any electrode site.

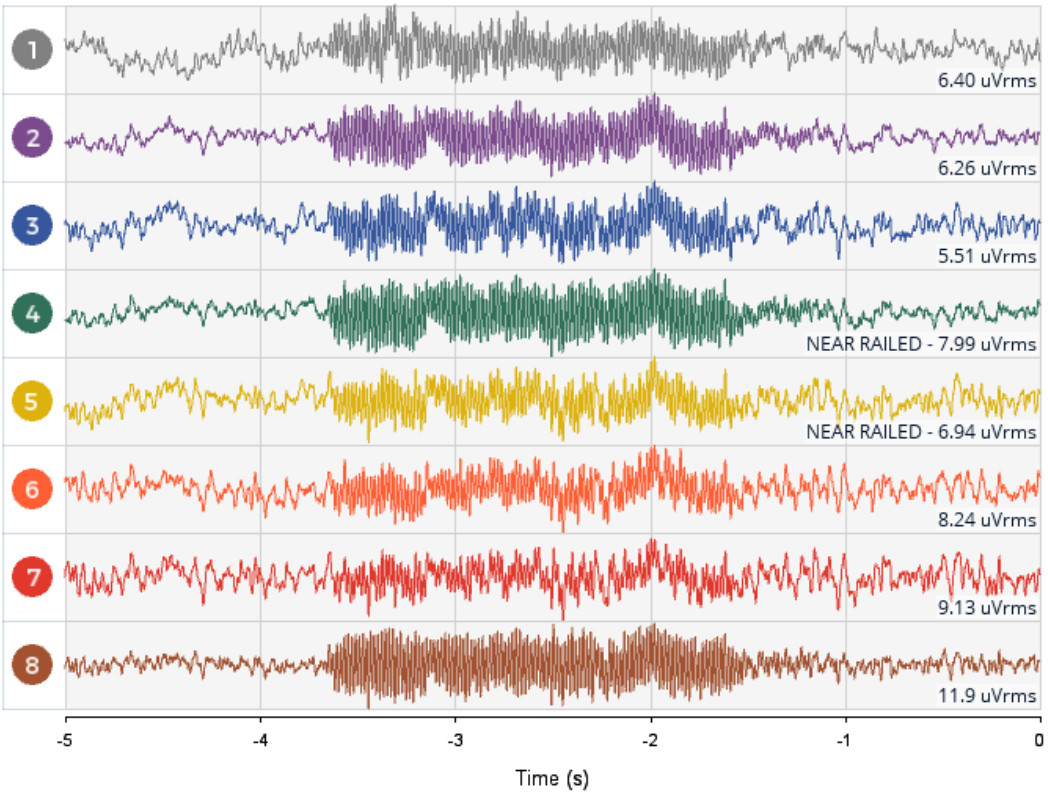


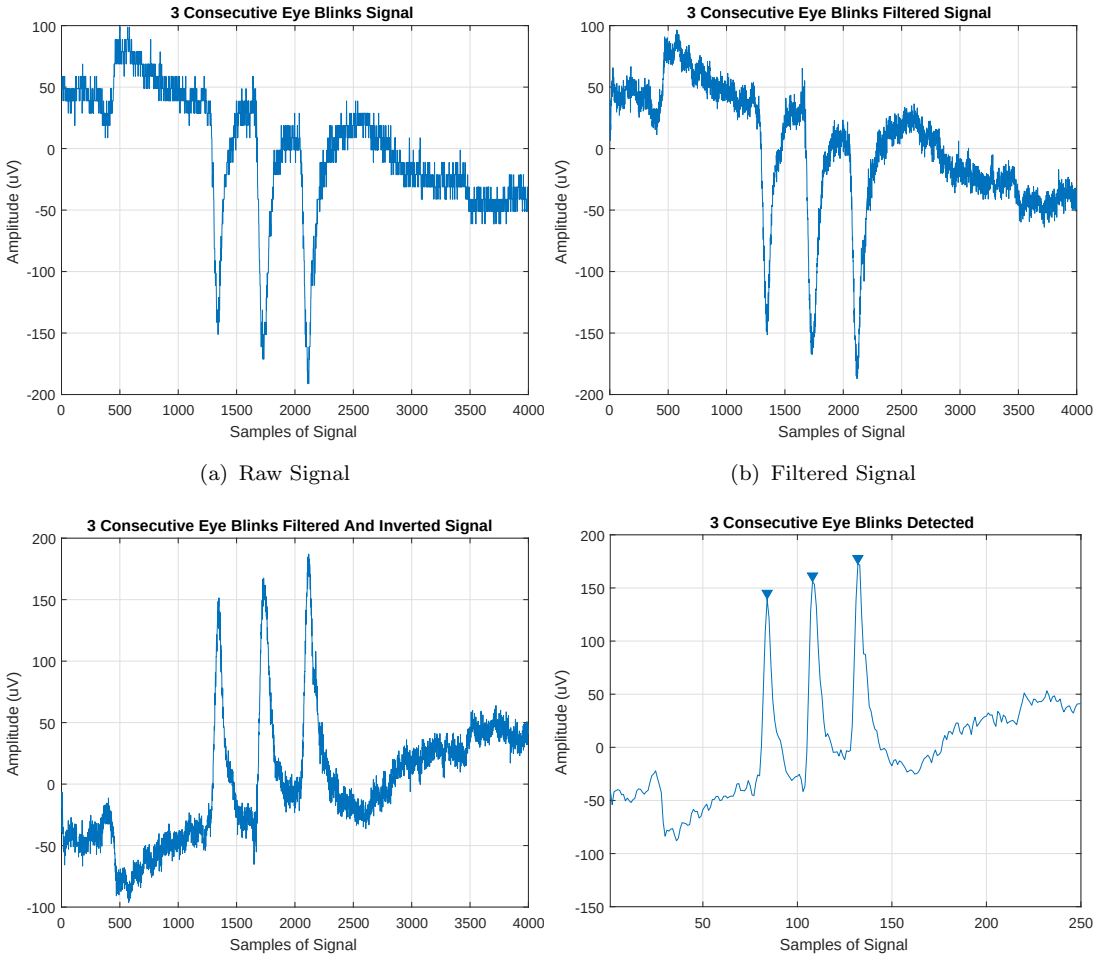
FIGURE 5.3: Jaw Clench in all 8 channels.

5.3 Signals Preprocessing

Signal Preprocessing part is explained in the chapter 4. Here the graphical results are shown for each classification and the preprocessing of signals.

5.3.1 Eye Blink Detection

As mentioned earlier only the front channel is used for eye blink detection. The plot of raw signal, filtered signal and the decimated signal with eye blink detection is shown in the figure 5.4.



(a) Raw Signal

(b) Filtered Signal

(c) Inverted Signal

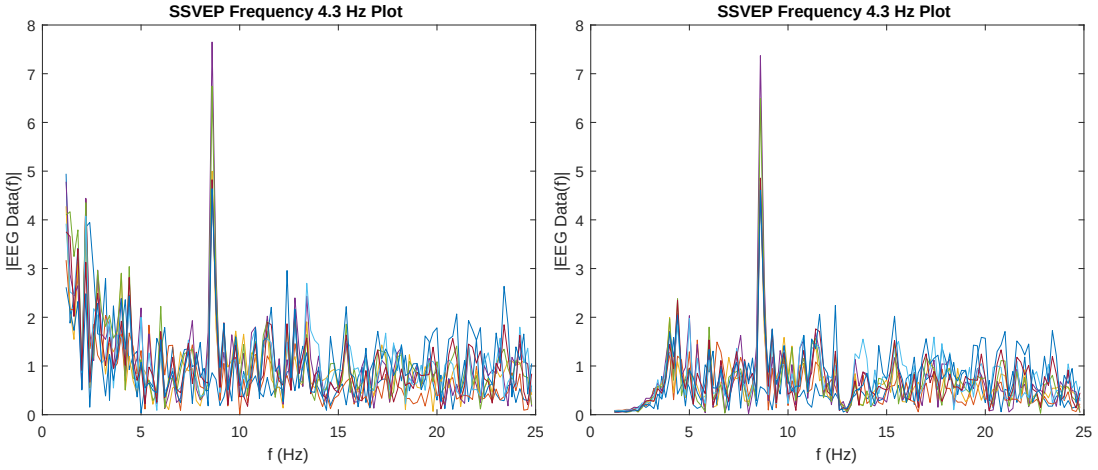
(d) Consecutive Eye Blinks Detected

FIGURE 5.4: Plot of 3 Consecutive Eye Blinks

5.3.2 SSVEP Frequency Detection

The results are shown for each SSVEP frequency before and after the filtration of signals. Each figure has 8-channel data though channel-1 is not used for SSVEP frequency classification.

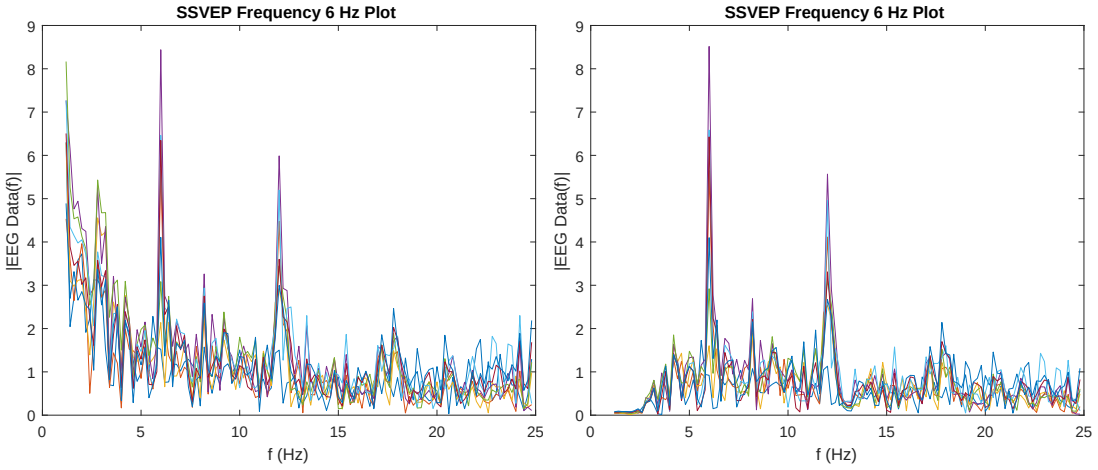
The frequency response for SSVEP frequency 4.3 Hz before and after filtration is shown in figure 5.5. You can see that the second harmonic at 8.6 Hz is more dominant.



(a) Raw Signals (b) Filtered Signals

FIGURE 5.5: SSVEP Frequency 4.3 Hz

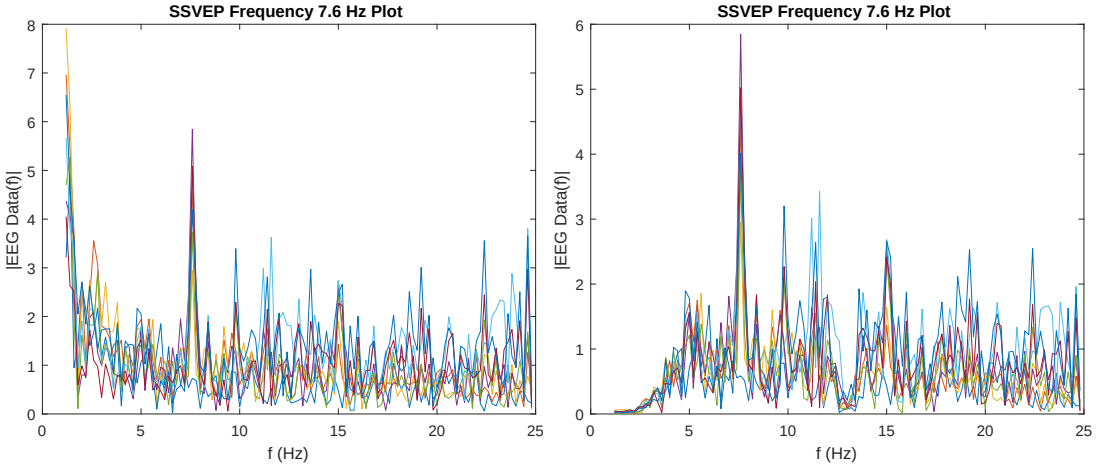
The frequency response for SSVEP frequency 6 Hz before and after filtration is shown in figure 5.6. You can see that the fundamental harmonic at 6 Hz is dominant.



(a) Raw Signals (b) Filtered Signals

FIGURE 5.6: SSVEP Frequency 6 Hz

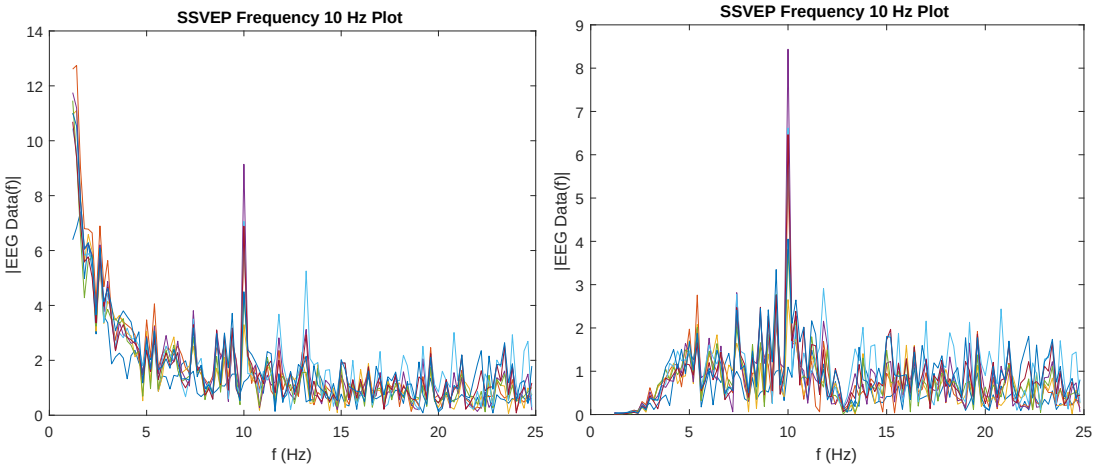
The frequency response for SSVEP frequency 7.6 Hz before and after filtration is shown in figure 5.7. You can see that the fundamental harmonic at 7.6 Hz is dominant.



(a) Raw Signals (b) Filtered Signals

FIGURE 5.7: SSVEP Frequency 7.6 Hz

The frequency response for SSVEP frequency 10 Hz before and after filtration is shown in figure 5.8. You can see that the fundamental harmonic at 10 Hz is dominant.



(a) Raw Signals (b) Filtered Signals

FIGURE 5.8: SSVEP Frequency 10 Hz

5.4 Classification Results

The eye blink detection classifier has 100% accuracy. For SSVEP frequency detection classifier, the data set contains 480 files (120 of each SSVEP frequency). Out of these files 215 were classified as class 5 which actually represents that no selection should be made (this happens because of the less attention of the user towards the screen). These files have not been counted for calculating the accuracy since this class do not make decisions it only says collect the data again. Out of

265 remaining files only 17 were miss-classified giving an accuracy of 93.58%. The accuracy during live testing was also above 90%.

5.5 Information Transfer Rate

In BCI research, ITR is the performance measuring method which measures the amount of subject information transferred per unit time in a BCI system [30]. The ITR provides a way to measure the performance of a BCI system when N, s, and p are known. The ITR can be calculated as

$$ITR = \frac{60}{s} * [log_2 N + p * log_2 p + (1-p) * log_2(\frac{1-p}{N-1})] \tag{5.1}$$

where

$$ITR = Information\ Transfer\ Rate\ per\ Minute$$
$$N = Number\ of\ Options$$
$$s = Time\ per\ Selection\ in\ Seconds$$
$$p = Probability\ of\ Correct\ Selection$$

The time for each selection on average is 4 second. With probability of 0.9358 for 4 options the ITR came out to be 23.315 bits per minute.

Chapter 6

Conclusion and Future Directions

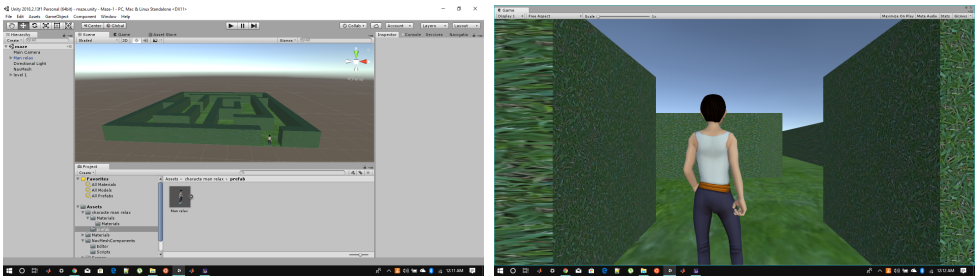
6.1 Conclusion

After experimenting, we found that the SSVEP frequency response is prominent at some frequencies which one has to figure out and the response for users was similar for these frequencies. We were able to find four frequencies 4.3, 6, 7.6 and 10 Hz. We used the common PC screen for flickering which has limitation of refresh rate. This screen does not support flickering at frequencies more than 15 Hz. When the screen is flickered at a frequency of more than 15 Hz, the frequency does not remain fixed, we observed more than one peaks in near region of 15 Hz rather than only a single sharp peak in the frequency plot when the screen was flickered at 15 Hz. This project has been tested on five persons. It has shown quite good results as discussed in the previous chapter the accuracy of more than 90 percent and the ITR of 23.315 bits per minute.

6.2 Future Directions

The project response can be improved by using a dedicated screen which is being controlled by micro-controller. This will increase the range of frequencies and hence more boxes can be made in the app that can reduce the number of selections to approach one alphabet.

The initial aim of this project was an extended form where Virtual Reality was to be integrated with BCI, from 2D to 3D application. This could lead to vast number of applications. Our aim was to implement this idea as well for that a 3D application has had been designed too on unity 3D, see figure 6.1 in which right, left, forward and backward movements can be made by gazing at SSVEP blocks. Because of the shortage of time we couldn't see that through. This project could



(a) Top View (b) Player View

FIGURE 6.1: 3D VR Game

lead to various research topics in the field of BCI and further more interesting techniques likes P300. A hardware prototype has been fully constructed in this project which could benefit anyone who wishes to further this interesting study. It is highly recommended to keep working on this area to keep pace with the advanced and emerging technology in the world.

Appendix A

BCI Speller Application Development

A.1 Unity Scripts for GUI

The BCI Speller application was developed in Unity. The BCI Speller scripts are given below. The flickering.cs script was associated with each flickering box. The changeText.cs script is the main script that is responsible for changing the text on each selection. For running these scripts, a GUI is first created in Unity and the tags are assigned to each object that is to be referenced in the script. Once the application is complete, the executable file can be generated from Unity.

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.UI;
5  using System.IO;
6  using System.IO.Ports;
7  using UnityEngine.SceneManagement;
8
9  public class changeText : MonoBehaviour {
10
11   private GameObject testObject;
12   private Image testImage;
13
14   /* To hold the references of four big boxes of type Text */
15   private Text[] testText = new Text[4];
16
17   /* Messsage to be displayed */
18   private Text message;
19
```

```
20  /* Confirmation Box text holder*/
21  private Text confirmationBox;
22
23  /* A 2D string array to hold all the strings to be displayed.
      */
24  private string[][] characters = new string[4][];
25
26  /* A flag variable to switch between NUMPAD and KEYPAD */
27  private bool showNumpad = false;
28
29  /* Following variables can have values from 0 to 4
30  * 0 -> no box selected
31  * 1 -> first box selected
32  * 2 -> second box selected
33  * 3 -> third box selected
34  * 4 -> fourth box selected
35  * The alphanumeric is printed in two selections.
36  */
37  private int firstSelectionNo;
38  private int secondSelectionNo;
39  private int thirdSelectionNo;
40
41  /* Count occurrence of the event 3 consecutive eye blinks*/
42  private int noOf3EyeBlinks;
43
44  /* Time for which confirmation box waits */
45  private double timeLeft;
46
47  /* Serial Port to send data to GSM900a module */
48  SerialPort mySerialPort;
49
50  /* Contact number */
51  private String contactNo = "03106200106";
52
53  // Use this for initialization
54  void Start () {
55
56    mySerialPort = new SerialPort("COM6");
57
58    mySerialPort.BaudRate = 9600;
59    mySerialPort.Parity = Parity.None;
60    mySerialPort.StopBits = StopBits.One;
61    mySerialPort.DataBits = 8;
62    mySerialPort.Handshake = Handshake.None;
63    mySerialPort.RtsEnable = false;
64
```

```
65
66      /*
        ************************************************************
67      Tags are used here to find the reference of the objects. In
        the heirarchy, Text object is a children of the tagged
        object so GetComponentInChildren<Text> method is called here
        . testText array holds all the references of the parent text
        box. Parent text box: A bigger box contains 9 or 4 (in last
        box) child Text objects.
68      ************************************************************
        */
69      testObject = GameObject.FindGameObjectWithTag ("topLeftTag"
        );
70      testText [0] = testObject.GetComponentInChildren<Text> ();
71
72      testObject = GameObject.FindGameObjectWithTag ("topRightTag
        ");
73      testText [1] = testObject.GetComponentInChildren<Text> ();
74
75      testObject = GameObject.FindGameObjectWithTag ("
        bottomLeftTag");
76      testText [2] = testObject.GetComponentInChildren<Text> ();
77
78      testObject = GameObject.FindGameObjectWithTag ("
        bottomRightTag");
79      testText [3] = testObject.GetComponentInChildren<Text> ();
80
81      /*
        ************************************************************
82      Message will hold the reference of the Text object that
        displays message on screen
83      ************************************************************
        */
84      testObject = GameObject.FindGameObjectWithTag ("messageTag"
        );
85      message = testObject.GetComponent<Text> ();
86      message.text = "";
87
88      /*
        ************************************************************
89      confirmationBox will hold the reference of the Text object
        that is displayed on screen and testImage holds reference of
        the image containing confirmationBox
90      ************************************************************
        */
```

```
91      testObject = GameObject.FindGameObjectWithTag ("
       confirmationBoxTag");
92      confirmationBox = testObject.GetComponentInChildren<Text>
       ();
93      confirmationBox.text = "";
94      testImage = testObject.GetComponent<Image> ();
95
96      /* Make the confirmationBox image transparent */
97      var tempColor = testImage.color;
98      tempColor.a = 0f;
99      testImage.color = tempColor;
100
101     /* Count number of 3 eye blinks if they are greater more
       than one within 10 secs then close the app */
102     noOf3EyeBlinks = 0;
103     timeLeft = 10;
104
105     Reset ();
106   }
107
108   /*
       ****************************************************************

109   This function displays all NUMPAD/KEYPAD strings on the
       screen.
110
       ****************************************************************
       */
111   void Reset () {
112
113     firstSelectionNo = 0;
114     secondSelectionNo = 0;
115     thirdSelectionNo = 0;
116
117     /* In 2D string characters, the rows contain parent strings
       (string to be displayed in
118     * big boxes). The matrix representation is:
119     * [ A       B       C       D       E     F    G    H    I]
120     * [ J       K       L       M       N     O    P    Q    R]
121     * [ S       T       U       V       W        X    Y    Z    .]
122     * [ SEND    DELETE/BACK   SPACE    NUMPAD ]
123     * Its not a perfect matrix. First five rows contain 9
       elements and last row contains
124     * 4 elements.
125     */
126     if (showNumpad) {
```

```
127         characters [0] = new string[]{ "0", "1", "2", "3", "4", "
    5", "6", "7", "8" };
128         characters [1] = new string[]{ "9", "!", "@", "#", "$", "
    &", "_", ";", ":" };
129         characters [2] = new string[]{ "?", "+", "-", "*", "/", "
    =", "'", "\"", "," };
130         characters [3] = new string[]{ "SEND", "DELETE", "SPACE",
     "KEYPAD" };
131       } else {
132         characters [0] = new string[]{ "A", "B", "C", "D", "E", "
    F", "G", "H", "I" };
133         characters [1] = new string[]{ "J", "K", "L", "M", "N", "
    O", "P", "Q", "R" };
134         characters [2] = new string[]{ "S", "T", "U", "V", "W", "
    X", "Y", "Z", "." };
135         characters [3] = new string[]{ "SEND", "DELETE/\nBACK", "
    SPACE", "NUMPAD" };
136       }
137       /*
    ************************************************************
138       testText array holds all references of four parent Text
    boxes. testText[0] contains reference of first parent box
    and so on. Using GetComponentsInChildren<Text> returns
    parent address at zeroth index and all child Text object
    references on indices 1 and onward i.e. childText[0]
    contains the same address as testText[i] while childText[j]
    where j >=1 contains the references of child Text objects.
    Outer loop iterates from 0 to 3 since there are four boxes.
    Inner loop iterates for 9 times for first three boxes and
    for 4 times for the last box. The parent text box should
    contain null string. The strings should only be displayed in
     smaller boxes.
139       ************************************************************
    */
140       Text[] childText;
141
142       for(int i = 0; i < 4; i++) {
143         childText = testText[i].GetComponentsInChildren<Text> ();
144
145         childText [0].text = "";
146         for (int j = 0; j < ((i == 3) ? 4 : 9); j++) {
147           childText [j + 1].text = characters [i] [j];
148         }
149       }
150     }
151
```

```
152  /*
     ************************************************************

153  This function displays the strings of the box selected in
     first selection.
154
     ************************************************************
     */
155  void DisplayText(int boxNo, int selectionStage){
156    /*
     ************************************************************
157    * testText array holds all references of four parent Text
     boxes. testText[0] contains reference of first parent box
     and so on. Using GetComponentsInChildren<Text> returns
     parent address at zeroth index and all child Text object
     references on indices 1 and onward i.e. childText[0]
     contains the same address as testText[i] while childText[j]
     where j >=1 contains the references of child Text objects.
     Outer loop iterates from 0 to 3 since there are four boxes.
     Inner loop iterates for 9 times for first three boxes and
     for 4 times for the last box. The parent box should display
     a string and the smaller boxes should have null string.
158    ************************************************************
     */
159    Text[] childText;
160    if (boxNo == 4) {
161      /* Box 4 has only one selection stage so mapping it onto
     second stage */
162      selectionStage = 2;
163    }
164    if (selectionStage == 1) {
165      int index = 0;
166      /* Only first three boxes are modified here*/
167      for (int i = 0; i < 3; i++) {
168        childText = testText [i].GetComponentsInChildren<Text>
     ();
169
170        for (int j = 0; j < 9; j++) {
171          if ((j + 1) % 3 == 0) {
172            childText [j + 1].text = characters [boxNo - 1] [
     index];
173            index++;
174          } else {
175            childText [j + 1].text = "";
176          }
177        }
```

```
178       }
179     } else if (selectionStage == 2) {
180       if (boxNo == 4) {
181         for (int i = 0; i < 4; i++) {
182           childText = testText [i].GetComponentsInChildren<Text
      > ();
183 
184           childText [0].text = characters [boxNo - 1] [i];
185           for (int j = 0; j < ((i == 3) ? 4 : 9); j++) {
186             childText [j + 1].text = "";
187           }
188         }
189       } else {
190         Text[] tempText;
191         tempText = testText [boxNo - 1].GetComponentsInChildren
      <Text> ();
192         for (int i = 0; i < 3; i++) {
193           childText = testText [i].GetComponentsInChildren<Text
      > ();
194 
195           childText [0].text = tempText [3 * (i + 1)].text;
196         }
197         for (int i = 0; i < 3; i++) {
198           childText = testText [i].GetComponentsInChildren<Text
      > ();
199 
200           for (int j = 2; j < 9; j = j + 3) {
201             childText [j + 1].text = "";
202           }
203         }
204       }
205     }
206   }
207 
208   /***********************************************
209    This function uses gsm900a module to send message
210    ***********************************************/
211   void SendMessage(string phoneNo, string msg){
212 
213     int ctrl_Z_ascii = 26;
214     mySerialPort.Open();
215     mySerialPort.Write ("AT+CMGF=1\r");
216     Delay(100);
217     mySerialPort.Write ("AT+CMGS=\"" + phoneNo + "\"\r\r");
218     Delay(100);
```

```
219     mySerialPort.Write (msg + "\nMessage sent from gsm" + "\r")
      ;
220     Delay(100);
221     mySerialPort.Write ((char)ctrl_Z_ascii + "\r");
222     mySerialPort.Close();
223   }
224
225   /* an approximate delay function */
226   void Delay(double timeInMilliSeconds){
227     int i = (int) 1000000 * timeInMilliSeconds;
228     while (i-- != 0);
229   }
230
231   void Update(){
232
233     /* Read the command from file written by MATLAB */
234     StreamReader reader = new StreamReader (@"B://
      matlab_to_unity.txt");
235     string line = reader.ReadLine();
236     reader.Close();
237
238     if (line != null) {
239       string[] command = line.Split(',');
240
241       /* The string written in the file is in the format A,B,C
      where
242       * A is the number indicating the detection of 3
      consecutive eye
243       * blinks. A == 1 indicates detection of the event. B
      indicates
244       * the box selection number. C can have two values 1 or 0
      its
245       * value is toggled every time the file is written to tell
       unity
246       * if the file is modified or not since unity reads file
      in every
247       * update frame. A can have value 0 or 1 and B can have
      values from
248       * 0 to 4 */
249       changeScene.LastVal = changeScene.Val;
250       changeScene.Val = command [2];
251       if (noOf3EyeBlinks == 1) {
252         timeLeft -= Time.deltaTime;
253         confirmationBox.text = "REPEAT THREE OR MORE EYE BLINKS
       TO QUIT\n" + timeLeft.ToString("#0");
254         if (timeLeft < 0) {
```

```
255           noOf3EyeBlinks = 0;
256           timeLeft = 10;
257           confirmationBox.text = "";
258           var tempColor = testImage.color;
259           tempColor.a = 0f;
260           testImage.color = tempColor;
261         }
262       } else if (noOf3EyeBlinks == 2) {
263         noOf3EyeBlinks = 0;
264         timeLeft = 10;
265         confirmationBox.text = "";
266         var tempColor = testImage.color;
267         tempColor.a = 0f;
268         testImage.color = tempColor;
269         SceneManager.LoadScene (0);
270       }
271       if (string.Compare (changeScene.Val, changeScene.LastVal)
      != 0
272       || Input.GetKeyDown(KeyCode.Alpha1) || Input.GetKeyDown(
      KeyCode.Alpha2)
273       || Input.GetKeyDown(KeyCode.Alpha3) || Input.GetKeyDown(
      KeyCode.Alpha4)
274       || Input.GetKeyDown(KeyCode.Alpha5)) {
275
276         if (string.Compare (command [0], "1") == 0 || Input.
      GetKeyDown(KeyCode.Alpha5)) {
277           noOf3EyeBlinks++;
278           confirmationBox.text = "REPEAT THREE OR MORE EYE
      BLINKS TO QUIT\n" + timeLeft.ToString("#0");
279           var tempColor = testImage.color;
280           tempColor.a = 1f;
281           testImage.color = tempColor;
282
283         }
284         if (noOf3EyeBlinks == 0) {
285           if (string.Compare (command [1], "1") == 0) { // If
      command 1 occured
286
287             /*
      ********************************************************
288             * case 0 -> No selection was made earlier
289             * case 1, 2, 3, 4 -> Previously the selection was
      made correspondingly 1, 2, 3 and 4
290             * One level nested switch cases mean the selection
      at level 2
```

```
291             * The inner most nested switch cases mean the
    selection at level 3
292
    **********************************************************/
293             switch (firstSelectionNo) {
294             case 0:
295               firstSelectionNo = 1;
296               DisplayText (1, 1);
297               break;
298             case 1:
299             case 2:
300             case 3:
301               switch (secondSelectionNo) {
302               case 0:
303                 secondSelectionNo = 1;
304                 DisplayText (1, 2);
305                 break;
306               case 1:
307               case 2:
308               case 3:
309                 switch (thirdSelectionNo) {
310                 case 0:
311                   thirdSelectionNo = 1;
312                   message.text = message.text + characters [
    firstSelectionNo - 1] [3 * (secondSelectionNo - 1) +
    thirdSelectionNo - 1];
313                   Reset ();
314                   break;
315                 case 4:
316                   SendMessage (contactNo, message.text);
317                   message.text = "";
318                   Reset ();
319                   break;
320                 }
321                 break;
322               case 4:
323                 SendMessage (contactNo, message.text);
324                 message.text = "";
325                 Reset ();
326                 break;
327               }
328               break;
329             case 4:
330               SendMessage (contactNo, message.text);
331               message.text = "";
332               Reset ();
```

```
333                 break;
334               }
335             } else if (string.Compare (command [1], "2") == 0) {
336               switch (firstSelectionNo) {
337               case 0:
338                 firstSelectionNo = 2;
339                 DisplayText (2, 1);
340                 break;
341               case 1:
342               case 2:
343               case 3:
344                 switch (secondSelectionNo) {
345                 case 0:
346                   secondSelectionNo = 2;
347                   DisplayText (2, 2);
348                   break;
349                 case 1:
350                 case 2:
351                 case 3:
352                   switch (thirdSelectionNo) {
353                   case 0:
354                     thirdSelectionNo = 2;
355                     message.text = message.text + characters [
    firstSelectionNo - 1] [3 * (secondSelectionNo - 1) +
    thirdSelectionNo - 1];
356                     Reset ();
357                     break;
358                   case 4:
359                     int tmpSelection = firstSelectionNo;
360                     Reset ();
361                     firstSelectionNo = tmpSelection;
362                     DisplayText (firstSelectionNo, 1);
363                     break;
364                   }
365                   break;
366                 case 4:
367                   Reset ();
368                   break;
369                 }
370                 break;
371               case 4:
372                 if (message.text.Length != 0) {
373                   message.text = message.text.Substring (0,
    message.text.Length - 1);
374                 }
375                 Reset ();
```

```
376                 break;
377               }
378             } else if (string.Compare (command [1], "3") == 0) {
379               switch (firstSelectionNo) {
380               case 0:
381                 firstSelectionNo = 3;
382                 DisplayText (3, 1);
383                 break;
384               case 1:
385               case 2:
386               case 3:
387                 switch (secondSelectionNo) {
388                 case 0:
389                   secondSelectionNo = 3;
390                   DisplayText (3, 2);
391                   break;
392                 case 1:
393                 case 2:
394                 case 3:
395                   switch (thirdSelectionNo) {
396                   case 0:
397                     thirdSelectionNo = 3;
398                     message.text = message.text + characters [
    firstSelectionNo - 1] [3 * (secondSelectionNo - 1) +
    thirdSelectionNo - 1];
399                     Reset ();
400                     break;
401                   case 4:
402                     message.text = message.text + " ";
403                     Reset ();
404                     break;
405                   }
406                   break;
407                 case 4:
408                   message.text = message.text + " ";
409                   Reset ();
410                   break;
411                 }
412                 break;
413               case 4:
414                 message.text = message.text + " ";
415                 Reset ();
416                 break;
417               }
418             } else if (string.Compare (command [1], "4") == 0) {
419               switch (firstSelectionNo) {
```

```
420             case 0:
421               firstSelectionNo = 4;
422               DisplayText (4, 1);
423               break;
424             case 1:
425             case 2:
426             case 3:
427               switch (secondSelectionNo) {
428               case 0:
429                 secondSelectionNo = 4;
430                 DisplayText (4, 1);
431                 break;
432               case 1:
433               case 2:
434               case 3:
435                 switch (thirdSelectionNo) {
436                 case 0:
437                   thirdSelectionNo = 4;
438                   DisplayText (4, 1);
439                   break;
440                 case 4:
441                   if (showNumpad) {
442                     showNumpad = false;
443                   } else {
444                     showNumpad = true;
445                   }
446                   Reset ();
447                   break;
448                 }
449                 break;
450               case 4:
451                 if (showNumpad) {
452                   showNumpad = false;
453                 } else {
454                   showNumpad = true;
455                 }
456                 Reset ();
457                 break;
458               }
459               break;
460             case 4:
461               if (showNumpad) {
462                 showNumpad = false;
463               } else {
464                 showNumpad = true;
465               }
```

```
466                 Reset ();
467                 break;
468               }
469             }
470           }
471         }
472       }
473   }
474 }
```

Listing A.1: C# changeText.cs

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using UnityEngine.UI;
5 using UnityEngine.SceneManagement;
6
7 public class Flickering : MonoBehaviour {
8
9   Image testImage;
10  Color testColor;
11  public float frequency;
12  bool isBlack = false;
13
14  // Use this for initialization
15  void Start () {
16
17    // Hold the reference of the image with which the script is
18    // attached
19    testImage = gameObject.GetComponent<Image> ();
20
21    // Start coroutine
22    StartCoroutine (Flashing ());
23  }
24
25  IEnumerator Flashing () {
26    while (true) {
27
28      // Introduce a delay
29      yield return new WaitForSecondsRealtime (1/(2*frequency))
    ;
30
31      /*****************************************************
32      * Change color to black if white and vice versa
33      *****************************************************/
```

```
34        if (isBlack) {
35          testImage.color = Color.white;
36          isBlack = false;
37        } else {
38          testImage.color = Color.black;
39          isBlack = true;
40        }
41      }
42    }
43  }
```

Listing A.2: C# Flickering.cs

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.UI;
5  using System.IO;
6  using UnityEngine.SceneManagement;
7
8  public class changeScene : MonoBehaviour {
9
10   private static string lastVal;
11
12   public static string LastVal{
13     set{ lastVal = value; }
14     get{ return lastVal; }
15   }
16
17   private static string val = "0";
18   public static string Val{
19     set{ val = value; }
20     get{ return val; }
21   }
22
23   void Update(){
24     StreamReader reader = new StreamReader ("B://
       matlab_to_unity.txt");
25     string line = reader.ReadLine();
26     reader.Close();
27
28     if (line != null) {
29       string[] command = line.Split(',');
30
31       lastVal = val;
32       val = command [2];
```

```
33        if (string.Compare (val, lastVal) != 0) {
34          if (string.Compare(command[0], "1") == 0) {
35            SceneManager.LoadScene(1);
36          }
37          else if (string.Compare(command[0], "2") == 0) {
38            Application.Quit ();
39          }
40        }
41      }
42    }
43 }
```

Listing A.3: C# changeScene.cs

A.2 MATLAB Scripts for Signal Processing and Classification

The following MATLAB script first applies filters and then extracts features using CCA and returns the result of classification on the input EEG data.

```
1 function result = ssvepClassification(fs, secs,
    eegRecord)
2
3   % SSVEP frequencies %
4   frequency = [2*4.3 6 7.6 10];
5
6   L = secs*fs;
7   t = linspace(0, L/fs, L)';
8
9   Y = zeros(L, 2*length(frequency));
10  for i = 1:4
11    % Sine and cose signals of same length to find CCA
      of the EEG data
12    % with these signals
13    sineSignal = sin(2*pi*frequency(i)*t);
14    cosineSignal = cos(2*pi*frequency(i)*t);
15    Y(:, (2*i-1):(2*i)) = [sineSignal cosineSignal];
16  end
17
18  % Apply band stop filter %
19  [b,a] = butter(2, [25 75]/(fs/2), 'stop');
```

```
20    eegRecordBStopFiltered = filter(b, a, eegRecord);
21
22    % Apply notch filter of 13 Hz %
23    [b,a] = butter(2, [12.5 13.5]/(fs/2), 'stop');
24    eegRecordNotchFiltered = filter(b, a,
       eegRecordBStopFiltered);
25
26    % Apply band pass filter %
27    [b,a] = butter(5, [5 25]/(fs/2));
28    eegRecordFiltered = filter(b, a,
       eegRecordNotchFiltered);
29
30    % Compute Sample Canonical Correlation
31    X = eegRecordFiltered;
32
33    [~, ~, r1, ~, ~] = canoncorr(X, Y(:, 1:2));
34
35    [~, ~, r2, ~, ~] = canoncorr(X, Y(:, 3:4));
36
37    [~, ~, r3, ~, ~] = canoncorr(X, Y(:, 5:6));
38
39    [~, ~, r4, ~, ~] = canoncorr(X, Y(:, 7:8));
40
41    data = [sum(r1) sum(r2)-0.3 sum(r3)-0.065 sum(r4)];
42
43    [maxValue, result] = max(data);
44
45    difference = data - maxValue;
46
47    % Check threshold constraint %
48    if min(abs(difference(difference ~= 0))) < 0.1
49      result = 5;
50    end
51  end
```

Listing A.4: MATLAB ssvepClassification.m

This MATLAB script applies filter for eye blink classification and gives result of classification on the input EEG data of one channel (FP2).

```
function result = eyeBlinkClassification(fs, secs, eegDataCh1)
  eegRecord = eegDataCh1 - mean(eegDataCh1); % Making the data zero mean

  % Apply band stop filter %
  [b,a] = butter(2, [5 75]/(fs/2), 'stop');
  eegRecordFiltered = filter(b, a, eegRecord);

  % Invert the signal about time axis %
  eegRecordFiltered = -1*eegRecordFiltered;

  % Decimate such that there are 50 samples per second after decimation %
  eegRecordFiltered = decimate(eegRecordFiltered, length(eegRecordFiltered)/(secs*50));

  [peaks, locations] = findpeaks(eegRecordFiltered, 'MinPeakProminence', 55);

  % If peak value is greater than 500 that is signal is noisy %
  if sum(peaks>500) >= 1
    result = 0;
  else
    % if there are more than three peaks %
    if length(peaks) >= 3
      peakSeparation = diff(locations);

      % If the separations between the peaks are right one %
      rightPeakSeparation = peakSeparation > 10 & peakSeparation <= 25;
      if(sum(rightPeakSeparation) >= 2)
        result = 1;
      else
        result = 0;
      end
```

```
31      else
32        result = 0;
33      end
34    end
35  end
```

Listing A.5: MATLAB eyeBlinkClassification.m

This script integrates the output of classification with the Unity application by sending command through a text file.

```
1  function update = sendCommand(eyeBlinkDetected,
       classNo, update)
2    % Toggle the value of update variable %
3    if update == 0
4      update = 1;
5    else
6      update = 0;
7    end
8    command = [eyeBlinkDetected classNo update];
9
10   % Write the text file %
11   csvwrite("B://matlab_to_unity.txt", command);
12 end
```

Listing A.6: MATLAB sendCommand.m

This script plots the frequency spectrum of EEG data.

```
1  function plot_fft_of_eeg_data(eeg_data, sampling_rate,
       channel_range, freq_range)
2
3    L = length(eeg_data);
4    f = (sampling_rate*(0:(L/2))/L)';
5    if length(channel_range) == 1
6      channel_range = [channel_range channel_range];
7    end
8
9    if length(freq_range) == 1
10     freq_range = [freq_range f(end)];
11   end
```

```
12
13    f_range = f > freq_range(1) & f < freq_range(2);
14    Y = fft(eeg_data);
15
16    figure
17    for channel_no = channel_range(1):channel_range(2)
18      P2 = abs(Y(:, channel_no)/L);
19      P1 = P2(1:L/2+1);
20      P1(2:end-1) = 2*P1(2:end-1);
21      plot(f(f_range),P1(f_range));
22      hold on;
23    end
24    title('Single-Sided Amplitude Spectrum')
25    xlabel('f (Hz)')
26    ylabel('|EEG Data(f)|')
27  end
```

Listing A.7: MATLAB plot_fft_of_eeg_data.m

A.3 Data Reception from EEG Board in Python

This python code receives data from EEG board and calls MATLAB scripts. To run this python code, you need to first install these python libraries

1. numpy
2. pylsl
3. pyserial
4. requests
5. six
6. socketIO-client
7. websocket-client
8. wheel
9. Yapsy
10. xmltodict

The python code is available at github which gives the initial driver for reciving data over WIFI. The github link is https://github.com/OpenBCI/OpenBCI_Python. The data is received in python and MATLAB scipts are called such that the data is passed to the MATLAB function and after performing the classification by MATLAB script the result is then sent back to python script which again calls MATLAB function to send command to the Unity application.

```
import sys; sys.path.append('..') # help python find cyton.py
    relative to scripts folder
from lib import openbci_wifi as bci
import logging
import numpy as np
import matlab
import matlab.engine

current_index = 0

# Sampling rate
fs = 1000.0
secs = 2.5
array_len = fs*secs

# Instantiate the matlab to call matlab functions
eng = matlab.engine.start_matlab()

update = 1
update = eng.sendCommand(0, 0, update)

# Initialize the myDataStream with zeroes
myDataStream = np.zeros((int(array_len), 8), dtype=np.float)
def printData(sample):
  global current_index
  global array_len
  global update

  # Only consider complete sample and increment the index
  if len(sample.channel_data) == 8:
    myDataStream[current_index, :] = np.array(sample.
    channel_data)
    current_index += 1

  # When one chunk of fs*secs data is complete process it and
    classify it
  if current_index == array_len:
    current_index = 0
```

```
36
37      # Call matlab classification functions
38      result1 = eng.eyeBlinkClassification(fs, secs, matlab.
      double(myDataStream[:, 0].tolist()))
39      result2 = eng.ssvepClassification(fs, secs, matlab.double(
      myDataStream[:,1:].tolist()))
40
41      # if result2 is 5 write 0 to file
42      if result2 > 4:
43        result2 = 0
44      update = eng.sendCommand(result1, result2, update)
45
46 if __name__ == '__main__':
47   shield_name = 'OpenBCI-B405'
48
49   logging.basicConfig(filename="test.log",format='%(asctime)s -
       %(levelname)s : %(message)s',level=logging.DEBUG)
50
51   logging.info('---------LOG START-------------')
52
53   # Instantiate shield object of OpenBCIWiFi
54   shield = bci.OpenBCIWiFi(ip_address='192.168.4.1',
       shield_name=shield_name, log=False, high_speed=True)
55
56   # Set sampling rate to 1000
57   shield.set_sample_rate(sample_rate=fs)
58
59   # Do not use srb1 pin for all 8 channels
60   for i in range(8):
61     shield.set_channel_settings(channel=i+1, use_srb1=False)
62
63   print("WiFi Shield Instantiated")
64   shield.start_streaming(printData)
65
66   shield.loop()
```

Listing A.8: Python test_wifi.py

References

[1] "Cyton biosensing board (8-channels)." [Online]. Available: https://shop.openbci.com/collections/frontpage/products/cyton-biosensing-board-8-channel?variant=38958638542

[2] "Wifi shield." [Online]. Available: https://shop.openbci.com/collections/frontpage/products/wifi-shield?variant=44534009550

[3] R. W. Homan, J. Herman, and P. Purdy, "Cerebral location of international 10–20 system electrode placement," *Electroencephalography and clinical neurophysiology*, vol. 66, no. 4, pp. 376–382, 1987.

[4] S. J. Luck and E. S. Kappenman, *The Oxford handbook of event-related potential components*. Oxford university press, 2011.

[5] "Pakistan's 10 million people are paralysed, country lacks awareness, basic facilities: expert," Jul 2017. [Online]. Available: https://dailytimes.com.pk/997/pakistans-10-million-people-are-paralysed-country-lacks-awareness-basic-facilities-expert/

[6] S. Herculano-Houzel, "The human brain in numbers: a linearly scaled-up primate brain," *Frontiers in Human Neuroscience*, vol. 3, p. 31, 2009. [Online]. Available: https://www.frontiersin.org/article/10.3389/neuro.09.031.2009

[7] D. J. McFarland, L. A. Miner, T. M. Vaughan, and J. R. Wolpaw, "Mu and beta rhythm topographies during motor imagery and actual movements," *Brain topography*, vol. 12, no. 3, pp. 177–186, 2000.

[8] M. J. Black, E. Bienenstock, J. P. Donoghue, M. Serruya, W. Wu, and Y. Gao, "Connecting brains with machines: the neural control of 2d cursor movement," in *Neural Engineering, 2003. Conference Proceedings. First International IEEE EMBS Conference on*. IEEE, 2003, pp. 580–583.

[9] H. Aurlien, I. Gjerde, J. Aarseth, G. Eldøen, B. Karlsen, H. Skeidsvoll, and N. Gilhus, "Eeg background activity described by a large computerized database," *Clinical Neurophysiology*, vol. 115, no. 3, pp. 665–673, 2004.

[10] W. O. Tatum, "Ellen r. grass lecture: Extraordinary eeg," *The Neurodiagnostic Journal*, vol. 54, no. 1, pp. 3–21, 2014.

[11] J. Decety and D. H. Ingvar, "Brain structures participating in mental simulation of motor behavior: A neuropsychological interpretation," *Acta psychologica*, vol. 73, no. 1, pp. 13–34, 1990.

[12] K. D. Markman, W. M. Klein, and J. A. Suhr, *Handbook of imagination and mental simulation*. Psychology Press, 2012.

[13] J. Decety, "Do imagined and executed actions share the same neural substrate?" *Cognitive brain research*, vol. 3, no. 2, pp. 87–93, 1996.

[14] S. Luck, "An introduction to the event-related potential technique. 2005," *Massachusetts: The MIT Press Google Scholar*.

[15] A. Haider and R. Fazel-Rezai, "Application of p300 event-related potential in brain-computer interface," in *Event-Related Potentials and Evoked Potentials*. InTech, 2017.

[16] R. Johnson Jr, "The amplitude of the p300 component of the event-related potential: Review and synthesis," *Advances in psychophysiology*, vol. 3, pp. 69–137, 1988.

[17] Y. Wang, X. Gao, B. Hong, C. Jia, and S. Gao, "Brain-computer interfaces based on visual evoked potentials," *IEEE Engineering in medicine and biology magazine*, vol. 27, no. 5, 2008.

[18] D. Zhu, J. Bieger, G. G. Molina, and R. M. Aarts, "A survey of stimulation methods used in ssvep-based bcis," *Computational intelligence and neuroscience*, vol. 2010, p. 1, 2010.

[19] E. C. Lalor, S. P. Kelly, C. Finucane, R. Burke, R. Smith, R. B. Reilly, and G. Mcdarby, "Steady-state vep-based brain-computer interface control in an immersive 3d gaming environment," *EURASIP Journal on Advances in Signal Processing*, vol. 2005, no. 19, p. 706906, 2005.

[20] K. E. Misulis, "Spehlmann's evoked potential primer," *Visual, auditory and somatosensory evoked potentials in clinical diagnosis*, 1994.

[21] X. Gao, D. Xu, M. Cheng, and S. Gao, "A bci-based environmental controller for the motion-disabled," *IEEE Transactions on neural systems and rehabilitation engineering*, vol. 11, no. 2, pp. 137–140, 2003.

[22] A. Rezeika, M. Benda, P. Stawicki, F. Gembler, A. Saboor, and I. Volosyak, "Brain–computer interface spellers: A review," *Brain sciences*, vol. 8, no. 4, p. 57, 2018.

[23] L. A. Farwell and E. Donchin, "Talking off the top of your head: toward a mental prosthesis utilizing event-related brain potentials," *Electroencephalography and clinical Neurophysiology*, vol. 70, no. 6, pp. 510–523, 1988.

[24] J. R. Wolpaw, N. Birbaumer, W. J. Heetderks, D. J. McFarland, P. H. Peckham, G. Schalk, E. Donchin, L. A. Quatrano, C. J. Robinson, and T. M. Vaughan, "Brain-computer interface technology: a review of the first international meeting," *IEEE transactions on rehabilitation engineering*, vol. 8, no. 2, pp. 164–173, 2000.

[25] I. Volosyak, H. Cecotti, D. Valbuena, and A. Graser, "Evaluation of the bremen ssvep based bci in real world conditions," in *2009 IEEE International Conference on Rehabilitation Robotics*. IEEE, 2009, pp. 322–331.

[26] I. Volosyak, F. Gembler, and P. Stawicki, "Age-related differences in ssvep-based bci performance," *Neurocomputing*, vol. 250, pp. 57–64, 2017.

[27] E. Yin, Z. Zhou, J. Jiang, Y. Yu, and D. Hu, "A dynamically optimized ssvep brain–computer interface (bci) speller," *IEEE Transactions on Biomedical Engineering*, vol. 62, no. 6, pp. 1447–1456, 2015.

[28] E. Yin, Z. Zhou, J. Jiang, F. Chen, Y. Liu, and D. Hu, "A novel hybrid bci speller based on the incorporation of ssvep into the p300 paradigm," *Journal of neural engineering*, vol. 10, no. 2, p. 026012, 2013.

[29] Z. Lin, C. Zhang, W. Wu, and X. Gao, "Frequency recognition based on canonical correlation analysis for ssvep-based bcis," *IEEE transactions on biomedical engineering*, vol. 53, no. 12, pp. 2610–2614, 2006.

[30] J. N. da Cruz, F. Wan, C. M. Wong, and T. Cao, "Adaptive time-window length based on online performance measurement in ssvep-based bcis," *Neurocomputing*, vol. 149, pp. 93–99, 2015.