# **FROM 25st June TO 1st July**

# **Project ID:**

# **2021J\_BV01\_BCI Browser**

# **Project Title:**

# **Design and development of Brain Computer Interface Browser on Web and Mobile**

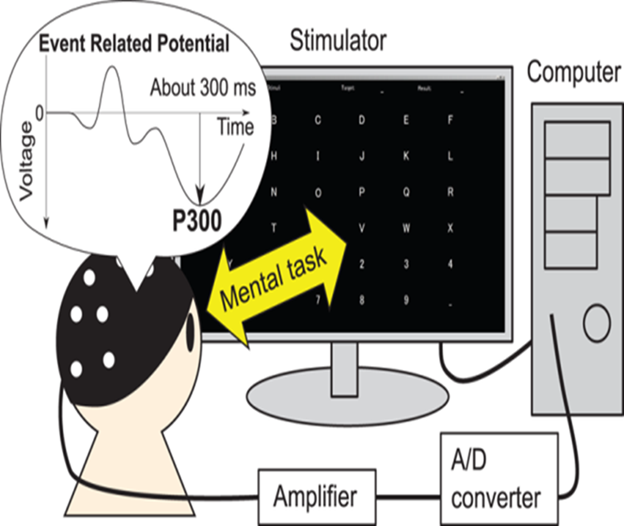
# **Summary:**

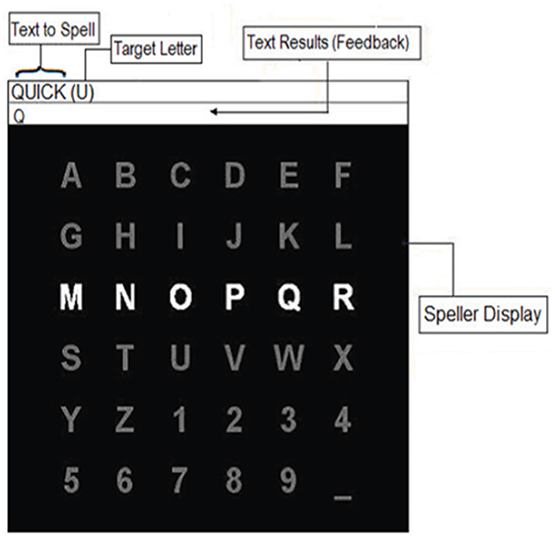
* P300 speller uses different event-related potentials (ERPs) including the P300 evoked response. In this system, the user attends to a character in a matrix while each row or column flashes rapidly and pseudo-randomly. The brain produces a response to the row or column that contains the intended character (i.e., the oddball); this response is different for the other rows or columns. The BCI can detect the desired character by determining the row and column that produces the largest evoked response.
* The P300 wave is a positive peak of an event related potential (ERP) that occurs 300ms recorded by EEG.
* The P300-speller is a widely used BCI system that allows users to communicate characters by focused attention. In our study, we are focusing on BCI system architecture, different applications of BCI, specifically P300 speller which classifies P300 waves and recognizes targeted character, on the datasets.
* P300 Speller is implemented using Web Development Languages such as HTML,CSS, JavaScript(Front-End) part.

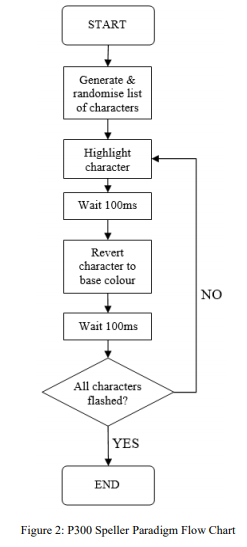
# **Detail:**

**P300 SPELLER**

* The P300 speller is a type of brain writer that types characters using a P300 component derived from characters that flash randomly in a letter matrix (Farwell and Donchin, 1988).
* A visual paradigm based P300 speller system consists of several stages: stimulating a subject by presenting a P300 display paradigm matrix; recording the EEG; signal preprocessing; feature extraction and classification.
* The **P300** wave is an event-related brain potential measured using electroencephalography (EEG). P300 refers to a spike in activity approximately 300ms following presentation of the target stimulus.





****

* P300 Speller is implemented using HTML,CSS,JavaScript(Front-End).
* There are some elements of the P300 Speller that do not vary with time. These time-static elements such as background colour and button positions can be realized using HTML and CSS. However, the P300 Speller also has time-dynamic elements such as element colour changes and timing delays. The only client-side language capable of expressing time-dynamic behaviour is JS. This work also uses JQuery which is a JS Library capable of shorthand coding conventions.

**METHODOLOGY**

**HTML CODE:-**

*The matrix is represented using HTML code.*

index.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1">

<title>P300-speller</title>

<link rel="shortcut icon" type="image/x-icon" href="brain.ico" />

<link href="custom.css" rel="stylesheet">

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">

<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css" integrity="sha384-Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q9Ifjh" crossorigin="anonymous">

<link href="https://cdnjs.cloudflare.com/ajax/libs/jqueryui/1.12.1/jquery-ui.min.css" rel="stylesheet">

</head>

<body class="container mt-3">

<div class="row">

<div class="col-md-6 col-sm-12 mb-3" id="speller\_matrix">

<table>

<colgroup>

<col class="a5">

<col class="a6">

<col class="a7">

<col class="a8">

</colgroup>

<tr class="a1">

<td class="5 1 11" id="A">A</td>

<td class="6 1 12" id="B">B</td>

<td class="7 1 13" id="C">C</td>

<td class="8 1 14" id="D">D</td>

</tr>

<tr class="a2">

<td class="5 2 15" id="E">E</td>

<td class="6 2 16" id="F">F</td>

<td class="7 2 17" id="G">G</td>

<td class="8 2 18" id="H">H</td>

</tr>

<tr class="a3">

<td class="5 3 19" id="I">I</td>

<td class="6 3 20" id="J">J</td>

<td class="7 3 21" id="K">K</td>

<td class="8 3 22" id="L">L</td>

</tr>

<tr class="a4">

<td class="5 4 23" id="M">M</td>

<td class="6 4 24" id="N">N</td>

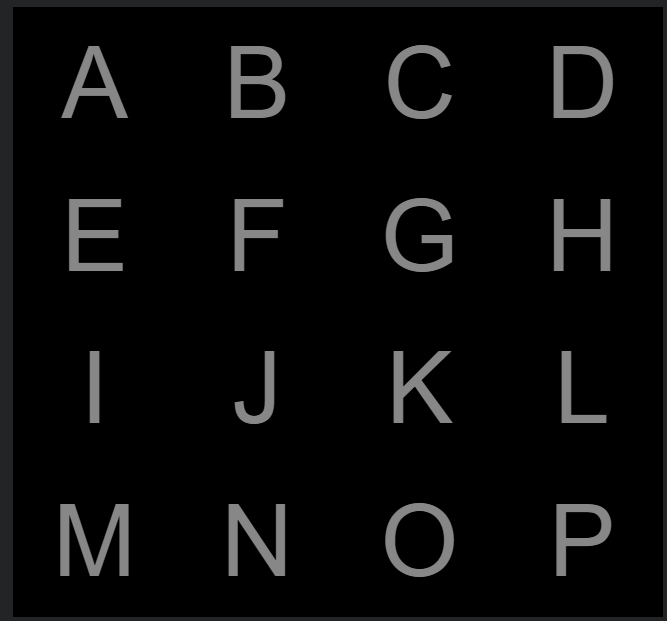
<td class="7 4 25" id="O">O</td>

<td class="8 4 26" id="P">P</td>

</tr>

</table>

</div>



<div class="col-md-6 col-sm-12">

<div class="n\_t">

<label for="trials">Enter the number of trials, please!</label>

<select name="select" id="number\_of\_trials">

<option value="2">2</option>

<option value="5">5</option>

<option value="6">6</option>

<option value="7">7</option>

<option value="8">8</option>

<option value="9">9</option>

<option value="10" selected>10</option>

<option value="11">11</option>

<option value="12">12</option>

<option value="13">13</option>

<option value="14">14</option>

<option value="15">15</option>

<option value="16">16</option>

<option value="17">17</option>

<option value="18">18</option>

<option value="19">19</option>

<option value="20">20</option>

<option value="25">25</option>

<option value="30">30</option>

</select>

</div>



<div class="d\_s">

<label for="trials">Choose the value of stimulation frequency, please!</label>

<select name="select" id="duration\_of\_stimulus">

<option value="980">1 Hz</option>

<option value="480">2 Hz</option>

<option value="313">3 Hz</option>

<option value="230" selected>4 Hz</option>

<option value="180">5 Hz</option>

</select>

</div>



<div class="d\_s">

<label for="trials">Choose the color of stimulus, please!</label>

<select name="select" id="s-color">

<option value="blue">blue</option>

<option value="white" selected>white</option>

<option value="red">red</option>

<option value="green">green</option>

<option value="yellow">yellow</option>

</select>

</div>



<div class="b\_s">

<button class="btn btn-outline-primary dis" id="start1">First Protocol</button>

<button class="btn btn-primary dis" id="start2">Second Protocol</button>

<div class="mt">For random order use the protocols below</div>

<div class="mt">

<button class="btn btn-outline-primary dis" id="start3">Third Protocol</button>

<button class="btn btn-primary dis" id="start4">Fourth Protocol</button>

</div>

<div class="mt">For background flashing use the protocols below</div>

<div class="mt">

<button class="btn btn-outline-primary dis" id="start5">fifth Protocol</button>

<button class="btn btn-primary dis" id="start6">sixth Protocol</button>

</div>

<div class="mt">

<button class="btn btn-outline-primary dis" id="start7">seventh Protocol</button>

<button class="btn btn-primary dis" id="start8">eighth Protocol</button>

</div>

<p class="mt-2">Start Time:<span id="time"></span></p>

<p class="d-none ">First Stimulus:<span id="f\_s"></span></p>

<button class="btn btn-outline-success mt-1" id="showData">Show selected numbers</button>

<button class="btn btn-success mt-1 dis" id="showTime">Show time of stimuli</button>

<div class="numbers">

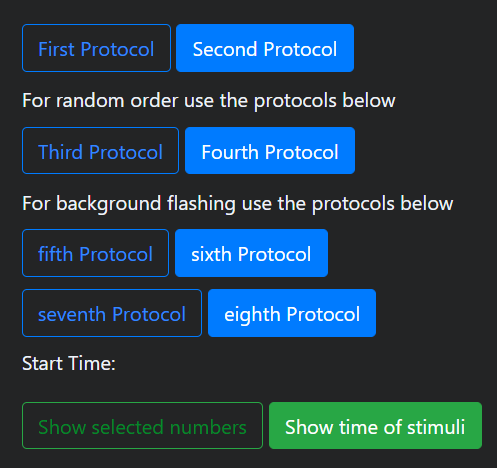
<p id="data" class="mt showMe"></p>

</div>

<div class="numbers">

<p id="data\_time" class="mt showMe"></p>

</div>



<button class="btn btn-outline-success mt-2" id="showDoc">Show documentation</button>

<div class="numbers">

<div id="doc" class="mt showMe">

<ul>For the first, third, fifth and seventh protocols ...

<li>1 = first row</li>

<li>2 = second row</li>

<li>3 = third row</li>

<li>4 = fourth row</li>

<li>5 = first column</li>

<li>6 = second column</li>

<li>7 = third column</li>

<li>8 = fourth column</li>

</ul>

<ul>For the second, fourth, sixth and eighth protocols ...

<li>We start counting from the letter A (number 1) to the letter P (number 16)</li>

</ul>

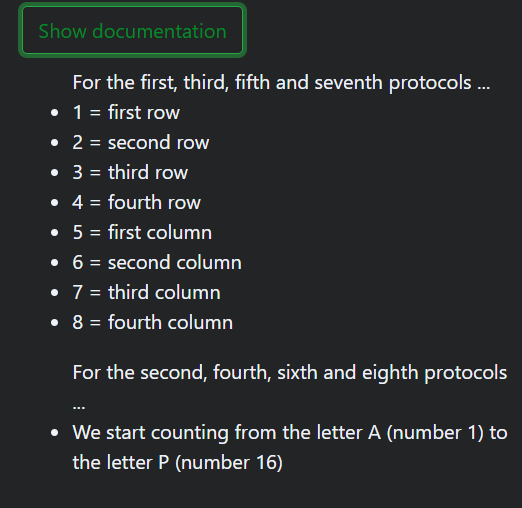
</div>

</div>

</div>

</div>

</div>



<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/jqueryui/1.12.1/jquery-ui.min.js"></script>

<script src="functions1.js"></script>

<script src="functions2.js"></script>

<script src="functions3.js"></script>

<script src="functions4.js"></script>

<script src="functions5.js"></script>

<script src="functions6.js"></script>

<script src="functions7.js"></script>

<script src="functions8.js"></script>

<script src="download.js"></script>

</body>

</html>

**CSS CODE:**

Custom.css

.numbers{

max-width: 400px;

}

#data{

word-wrap: break-word;

}

#data\_time{

word-wrap: break-word;

}

.showMe{

display: none;

}

body {

background-color: #F2F3F4 !important;

}

.b\_s{

margin-top: 20px;

}

.n\_t{

color:grey;

}

.d\_s{

margin-top: 15px;

color:grey;

}

button {

margin-right: 10px;

padding:6px 10px 6px 10px;

background-color: #98FB98;

border-color: #006400;

cursor:pointer;

border-radius: 10px 10px 10px 10px;

}

.fa{

margin-right: 40px;

}

.mt{

margin-top: 10px;

}

button:hover {

background: none;

}

table {

color:grey;

background-color: black !important;

}

td {

width:130px;

height:80px;

font-size:5em;

text-align: center;

font-family: Arial;

}

.footer {

position: absolute;

bottom: 0;

height: 100px;

width: 100%;

}

.red{

background-color:red;

}

.blue{

background-color:blue;

}

.green{

background-color:green;

}

.yellow{

background-color:yellow;

}

.white{

background-color:white;

}

.black{

background-color:black;

}

**JS CODE:**

Function1.js

$(document).ready(function() {

$("#start1").click(

function() {

$(".dis").prop('disabled', true);

var flashes = [];

var milis = [];

const s\_color = $("#s-color").val();

const ISI = $("#duration\_of\_stimulus").val();

const d\_s = 100;

const time = d\_s + ISI;

const n\_t = $("#number\_of\_trials").val();

number\_of\_trials = n\_t;

var all\_chars = [1,2,3,4,5,6,7,8];

new\_chars = [1,3,5,7,2,4,6,8];

number\_of\_trials--;

for(a=0; a<number\_of\_trials; a++) {

temp\_chars = [1,3,5,7,2,4,6,8];

new\_chars = new\_chars.concat(temp\_chars);

if(a == number\_of\_trials-1){

document.getElementById("data").innerHTML = new\_chars;

}

}

c=new\_chars.length;

i=0;

var d = new Date();

var h = d.getHours();

var m = d.getMinutes();

var s = d.getSeconds();

var n = d.getMilliseconds();

var startTime = h + ":" + m + ":" + s + " -- " + "you choosed the first protocol";;

var fix\_s = s+5;

var firstStimulus = m + ":" + fix\_s;

document.getElementById("time").innerHTML = startTime;

document.getElementById("f\_s").innerHTML = firstStimulus;

setTimeout(flash,5000);

// 2 second pause before stimulus presentation starts

var flash\_time = d\_s;

function flash() {

if(i<c) {

var flash\_index = new\_chars[i];

requestAnimationFrame(() => {

light\_unlit(flash\_index,1); // highlight element

var d = new Date();

var m = d.getMinutes();

var s = d.getSeconds();

var n = d.getMilliseconds();

//var timer = m + ":" + s;

//document.getElementById("timer").innerHTML = timer;

var mili\_s = m\*60\*1000+1000\*s+n;

milis.push(mili\_s);

new\_time = (m + "," + s + "," + n);

flashes.push(new\_time)

})

setTimeout(

function() {

light\_unlit(flash\_index,0); // revert element to default colour after flash

setTimeout(flash,ISI);

}

,flash\_time);

}

i++;

if(i == c+1 && flashes){

for(i=0;i<milis.length-1;i++){

milis[i] = -milis[i] + milis[i+1] - (time) + 99900

}

var total = 0;

for(j = 0; j < milis.length-1; j++) {

total += milis[j];

}

console.log(milis,total)

var avg = total / (milis.length-1);

flashes.push("Mean Error = " + avg)

document.getElementById("data\_time").innerHTML = flashes.join('\r\n');

$(".dis").prop('disabled', false);

}

}

// recursive function to keep calling setTimeout until all characters have flashed

function light\_unlit(char\_index,state) {

if(state==0) {

stim\_colour = "grey";

} else {

stim\_colour = s\_color;

}

switch(char\_index) {

case 1: $(".1").css("color",stim\_colour); break;

case 2: $(".2").css("color",stim\_colour); break;

case 3: $(".3").css("color",stim\_colour); break;

case 4: $(".4").css("color",stim\_colour); break;

case 5: $(".5").css("color",stim\_colour); break;

case 6: $(".6").css("color",stim\_colour); break;

case 7: $(".7").css("color",stim\_colour); break;

case 8: $(".8").css("color",stim\_colour); break;

default:

}

}

function shuffle(array) {

var currentIndex = array.length, temporaryValue, randomIndex;

// While there remain elements to shuffle...

while (0 !== currentIndex) {

// Pick a remaining element...

randomIndex = Math.floor(Math.random() \* currentIndex);

currentIndex -= 1;

// And swap it with the current element.

temporaryValue = array[currentIndex];

array[currentIndex] = array[randomIndex];

array[randomIndex] = temporaryValue;

}

return array;

}

}

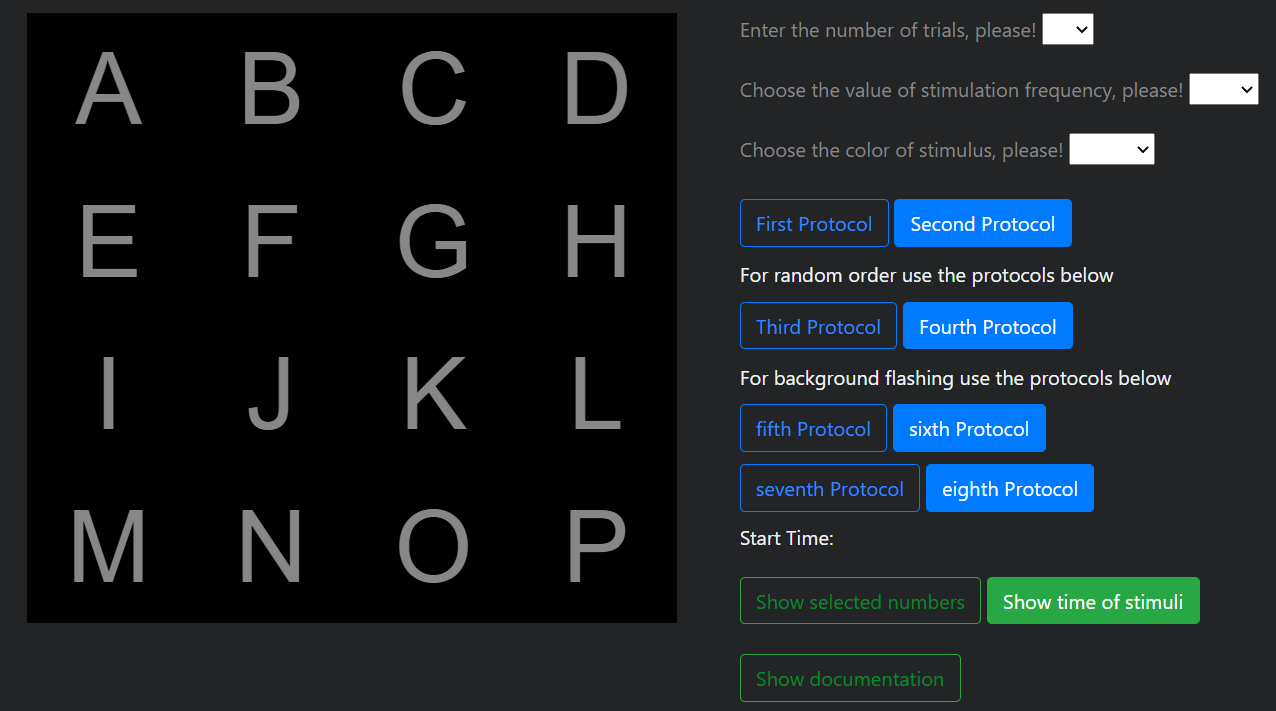
);

});

The ‘new\_chars’ variable with length ‘c’ is defined outside of the function and contains the total list of characters to be flashed based on the number of trials. The “light\_unlit” function flashes a P300 matrix element if the second function input is ‘1’ and reverts the element base colour to white if the second input is ‘0’. The JS language does not contain a standard delay function that halts code execution for a predefined period. The paradigm timing is therefore implemented using the “setTimeout” JS function. The “setTimeout” function waits for a certain time before running some specified code. Recursion is then employed to ensure that the code progresses to highlight and then revert character colours until the total character set is flashed.

*The code for other protocols is similar with little changes.*

**RESULTS:**

****

**CONCLUSION-:**

* The P300 speller is a type of brain writer that types characters using a P300 component derived from characters that flash randomly in a letter matrix .The brain produces a response to the row or column that contains the intended character ; this response is different for the other rows or columns. The BCI can detect the desired character by determining the row and column that produces the largest evoked response.
* The P300 wave is an event-related brain potential measured using electroencephalography (EEG).The P300 wave is a positive peak of an event related potential (ERP) that occurs 300ms recorded by EEG.
* We have focused on BCI system architecture, different applications of BCI, specifically P300 speller which classifies P300 waves and recognizes targeted character, on the datasets.
* P300 Speller is implemented using Web Development Languages such as HTML,CSS, JavaScript(Front-End) part.