

Basic Two Class MI-BCI GUI

1 About

This program serves as interface for classical BCI motor imagery application. Two major programs are used in this application, the first one is the GUI program which is built in python language. Second one is the program that interacts with g.tec USB AMP device to record EEG signal. This one is an exe application generated by program written in C#. In this document the first program (GUI) may be referred as 'master', while the second one (recording) is referred as 'slave'.

All codes will be available in the following GitHub repository:

<https://github.com/orvindemtsy/BCI-MI-sequence>

2 Hardware and Software Requirements

These specification/ tools are required to conduct the experiment:

- Windows 10 OS (g.tec USB AMP only compatible to this OS)
- PyCharm (or equivalent python IDE)
- Program dependencies (explain in the next chapter)
- g.tec USB AMP device and its power supply & USB cable
- g.tec USB AMP driver & g.tec C API driver installer

3 How it Works

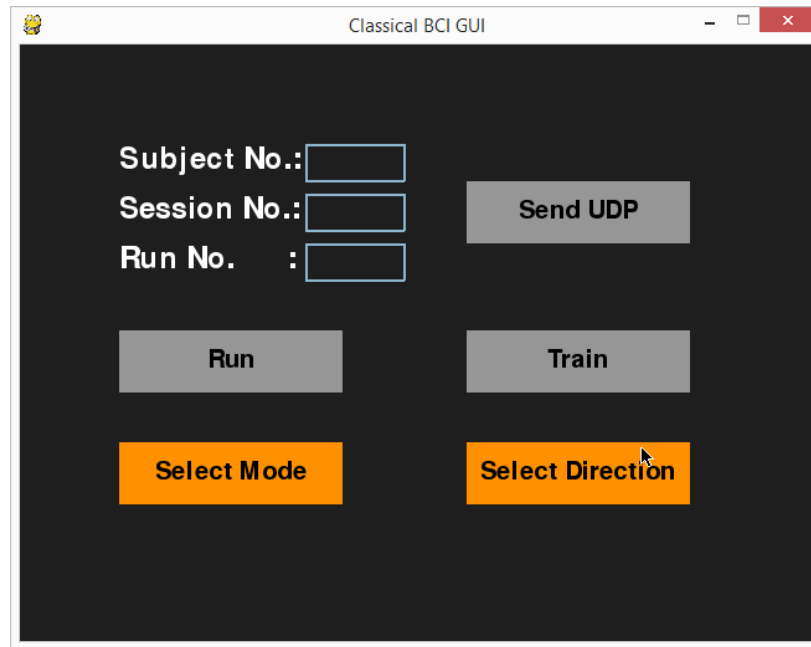
This chapter will explain in details how the standard experiment is carried out by this program. It will mainly be divided into two sections, input window and run window:

3.1 Running main (master) program

Open and run the main.py program under the project folder using python IDE (PyCharm is suggested but not compulsory). As per September 29th, 2020, this project is store in github repository named 'basic-two-class-mi-bci-gui-v1'

3.2 Input Window

You should see the following GUI upon executing the main.py



3.2.1 Input subject, session, run number

There are three items that the subject is required to fill in to satisfy data recording. Those are subject number, session number, run number. User is expected to be able to input their data easily as these input boxes operates like common input boxes.

User need to click inside the box, once the box is clicked the border color will be changed. This is when the box is active and user is able to type in character. Keep in mind that certain format has to be met in order for the data to be properly recorded. Press enter once you want to input the data. The character should be deleted from the box once entered is pressed.

To confirm the input data, user could press 'send UDP' button, this command will print the data to the console. If you found your data is wrong, you can fix it by re-inputting the information in the boxes.

3.2.2 Input format

The following input format is required in order for DataRec.exe (slave) to properly record the data

	Subject	Session	Run
Format	XX	X	X
Example	01	4	9
Allowed range	01 ... 99	0 ... 9	0 ...9

3.2.3 Selecting mode and direction

After all the subject data is properly input, we can proceed to mode and direction selection. Two dropdown menus are available to select which mode and direction to be run.

3.2.3.1 Mode

There are two modes in this application which are:

1. Calibration

Calibration mode is used to calibrate the EEG recording amplifier to specific subject. *This mode should be done prior to test mode especially to new subject who has never done experiment before (confirm to David).* This mode enables EEG recorder to develop subject-specific model, which enables it to properly record EEG. In this mode, subject is asked to perform motor-imagery task that corresponds to certain arrow direction shown on the screen. The details will discuss later in following chapter.

2. Test

Test mode is when the experiment is actually being evaluated by the program. In this mode, the pre-generated model in calibration mode is utilized as user performs several motor-imagery tasks. In the last 4 seconds of each trial of the test a bar will be shown. This bar indicates how well a subject performs motor-imagery task.

3.2.3.2 Direction (horizontal & vertical)

This dropdown menu is selected to determine which orientation each mode is run. Two directions are available, horizontal means there will be five left and five right arrows appearing in random order. The same rule applies to vertical mode.

3.2.4 Send to UDP

This button servers as a checker of the input message so that it can be corrected if user input the information incorrectly. It will show the subject, session, and run number, preceded by 'T' and ended with 'xx'. e.g.: T0532xx

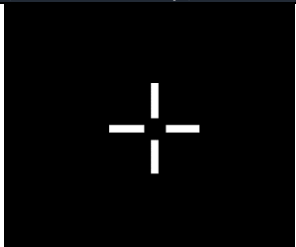
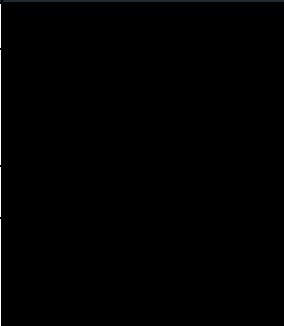




3.3 Run

After inputting the subject information, session and run number, selecting mode and direction. We are ready to run the program. Clicking 'Run' button will execute the sequence/ trial corresponding to selected mode and direction. This section will explain two types of the modes in detail. The recording will start automatically when the sequence runs. The detail of recorded data can be found in the next section.

3.3.1 Calibration

In this mode, subject is only required to perform motor imagery task according to arrows shown on the screen. And as previously mentioned, subject's performance is not evaluated.

The sequence in calibration mode with respective duration and events are given below:

	White fixation cross (+ 200ms beep)	Red arrow (horizontal or vertical)	Black screen
Graphic		Horizontal	
		 	
		Vertical	
		 	

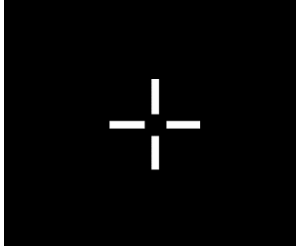
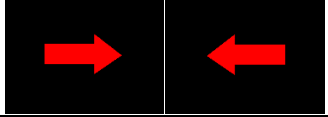
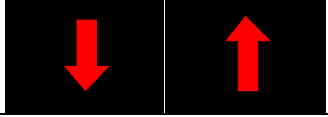
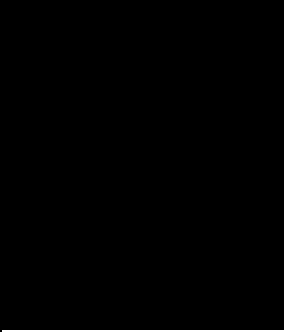
Order of appearance	1th	2th	3th
Time	2s	2s	2s
Expected action from subject	Focus on screen, anticipate beep sound	Imagine hand movement in the direction of given arrow	Relax, getting ready for next trial
Program task	Record EEG once the crosshair appears		

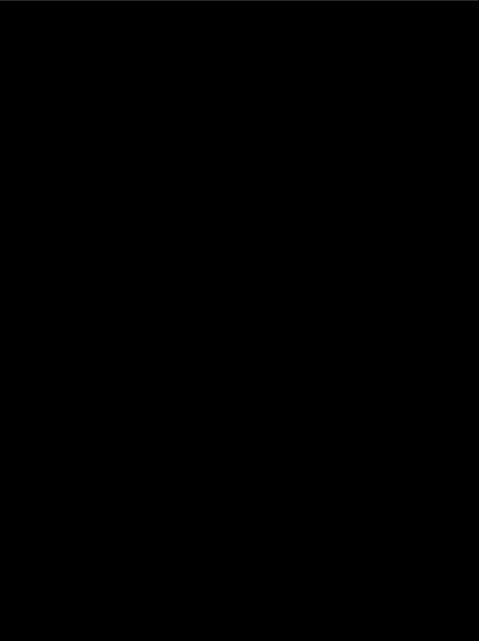
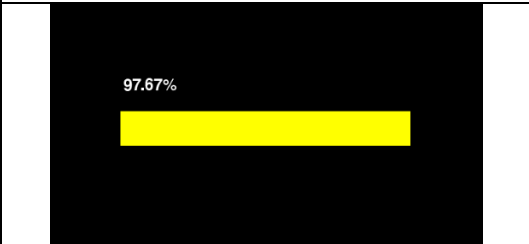
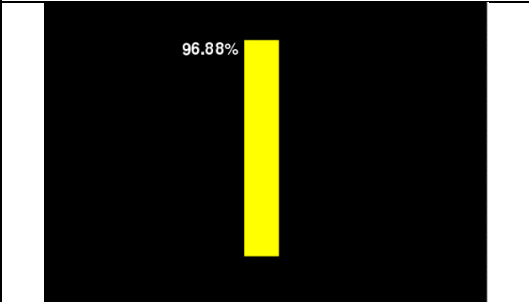
There will be ten sequence or trial in run. Each trial last for 6s, totaling approximately one minute each run.

3.3.2 Test

Test mode has the same sequence as calibration mode plus the bar in the end which acts as performance evaluation. The only difference between test and calibration mode is user performance

Test mode contains the following sequence and events:

	White fixation cross (+ 200ms beep)	Red arrow (horizontal or vertical)	Black screen
Graphic		<div>Horizontal</div>  <div>Vertical</div> 	
Order of appearance	1th	2th	3th
Time	2s	2s	2s
Expected action from subject	Focus on screen, anticipate beep sound	Imagine hand movement in the direction of given arrow	-
Program task	Record EEG once the crosshair appears		Calculate subject' performance

Graphic	Feedback bar	Black screen
	Horizontal	
		
	Vertical	
		
Order of appearance	4th	5th
Time	2s	2s
Expected action from subject	-	Relax, getting ready for next trial
Program task	Display performance evaluation as bar length or width in regard with arrow direction.	

3.4 Recording Data

When 'Run' button is clicked, sequence will run. Simultaneously, the program will interact with externally available DataRec.exe application to record EEG and save to predetermined folder.

The recorded data will be saved in the same directory of main program. A 'Data' folder will be created then inside this folder you shall find 'T' folder, inside this folder there will be folder with one digit indicating number of sessions. Inside this folder there will be a file with three digits, indicating, number of runs, arrow direction, and number of trials. The details of name file will be discussed further in DataRec section.

During the sequence, if program is recording properly the following text should appear on console.

```

=== Running Calibration Mode Horizontal Direction ===

[1, 2, 2, 2, 1, 2, 1, 1, 2, 1]
Received a broadcast from 127.0.0.1:64568
T0532x21
Opening and initializing devices...
    invalid hardware version
Stopping devices, closing them and cleaning up...
Received a broadcast from 127.0.0.1:64568
T0532x22
Opening and initializing devices...
    invalid hardware version
Stopping devices, closing them and cleaning up...
Received a broadcast from 127.0.0.1:64568
T0532x23
Opening and initializing devices...
    invalid hardware version
Stopping devices, closing them and cleaning up...
Received a broadcast from 127.0.0.1:64568
T0532x14
Opening and initializing devices...
    invalid hardware version
Stopping devices, closing them and cleaning up...
Received a broadcast from 127.0.0.1:64568

```

The image above is a screenshot of a console in PyCharm when sequence is running, the first, we can observe that the first four data is being recorded. The explanation of naming of each file will be explained in DataRec section.

4 Dependencies/ Modules Description

Several built-in, open-source, and self-built modules are required in order to conduct the experiment properly. The role of each module will be discussed. Please make sure you have either install or include the modules inside your project folder.

Throughout this section, pygame will sometimes be referred as 'pg'. The names of attributes in each section will not include keyword 'self', readers are assumed to already have an understanding of the syntax of constructor in class.

4.1 Pygame

This program relies heavily on pygame as its basic building block. Pygame can be installed by doing the following in the command prompt

```
pip install pygame
```

Pygame should be installed properly, if there's an error regarding pygame installation, most likely there is also a problem with python installation or to know more about pygame model please refer to the [pygame documentation](#)

4.2 Colors

Colors.py is a self-built module to assist editing color in pygame module. In this module, each color is already defined in tuple format for example black is defined as (0, 0, 0). This tuple format is the desired format in pygame module when defining color, so instead of inserting (0, 0, 0) in pygame to define black, we can import colors.py to pygame and just type in 'black'.

Here's an example of running the main program with and without colors.py, in this example we want a color black of the screen/window named 'win':

To define the color of the screen named 'win'	
<code>win.fill((R, G, B))</code>	
where R, G, B defines the intensity of red, green and blue within 0 - 255	
Without colors.py	With colors.py
<code>win.fill((255, 0, 0))</code>	<code>win.fill(red)</code>

This can be done due to the fact variable of '`red = pygame.colors(255, 0, 0)`' is defined in the colors.py

4.3 Dropdowndir

Dropdowndir.py is a self-built module that consist of drop-down menu for direction selection, horizontal and vertical. Each menu contains a flag that is raised whenever respective choice is selected. The name "Horizontal" and "Vertical" are hard-coded in the program file.

Some constants defined in this class

Variable Name	Description
COLOR_MAIN_INACTIVE	Color, mouse doesn't hover on 'Select Direction' button
COLOR_MAIN_ACTIVE	Color, mouse hover on 'Select Direction' button
font	Constant declaring font-type to print text, font size is 30

4.3.1 class DropDown()

class DropDown() manage the behavior of the button, it has three methods which are handle_event, draw, and option. Each will be explained further in the following section

4.3.1.1 attribute and parameters

Here are some attributes that belongs to this class

Attributes	Data type	Description
rect_main	pg.Rect	This attribute is an object of pg.Rect, will obtained the passed x, y, w, h) to initialize rect object
rect_list1	pg.Rect	This attribute is an object of pg.Rect, will obtained the passed x, y, w, h) to initialize rect object
rect_list2	pg.Rect	This attribute is an object of pg.Rect, will obtained the passed x, y, w, h) to initialize rect object
active_main	boolean	Initialize as True, act as a flag for main selection (Select Direction), will output False once the button is pressed
active_list1	boolean	Initialize as False, will act as flag for list1 (Vertical), will output True if this option is shown and selected.
active_list2	boolean	Initialize as False, will act as flag for list2 (Horizontal), will output True if this option is shown and selected.

color_main	pg.Color	A pygame-specific color format, written in format (Red, Green, Blue), please refer to Colors module. This will determine the color of the main selection (select direction).
color_list	int	A pygame-specific color format, written in format (Red, Green, Blue), please refer to Colors module. This will determine the color of the list1 and list2 (vertical and horizontal)
main_surface	font.render	will use font constant to write 'Select Direction' within the rect_main button
list1_surface	font.render	will use font constant to write 'Vertical' within the rect_list1 button
list2_surface	font.render	will use font constant to write 'Horizontal' within the rect_list2 button

Parameters		
win	pg.display.set_mode	The screen onto which given parameters will be displayed. Initialize as class attribute by defining self.screen
x	int	x coordinate the starting point of pygame.rect object for button
y	int	y coordinate the starting point of pygame.rect object for button
w	int	the height of pygame.rect object for button
h	int	the height of pygame.rect object for button

4.3.1.2 *def handle_event(event)*

This function will handle the behavior and the value of each option when selected.

	Variable	Data type	Description
--	----------	-----------	-------------

Parameters	event	pg.event.type	<p>pg.event will be passed into this function, pg.event is an pygame object that represents event. there are two types of event that will be handled here, they are:</p> <ul style="list-style-type: none"> - pg.MOUSEMOTION This function of pygame will monitor the movement of the mouse and will return the coordinate of the mouse, this function is used to detect when certain button is hovered by a mouse - pg.MOUSEBUTTONDOWN This function of pygame will return True whenever a mouse is clicked, each button of mouse return specific value of integer, but this will be ignored in this program.
Returns	-	-	-

4.3.1.3 *def draw()*

Initially 'Select Direction' will written as the selected option, but when either 'Vertical' or 'Horizontal' is selected, this will overwrite the 'Select Direction', indicating that currently a certain option is selected.

No parameter is passed, this function will draw respective selected option based on the Boolean value of each option stated in handle_event function.

4.3.1.4 *def option()*

Initially 'Select Direction' will written as the selected option, when this button is clicked, two options 'vertical' and 'horizontal' will be drawn onto screen. This function handles this behavior.

No parameter is passed, this function will draw respective selected option based on the Boolean value of each option stated in handle_event function.

4.4 Dropdownmode

Dropdownmode.py is a self-built module that consist of drop-down menu for mode selection, test and calibration. Each menu contains a flag that is raised whenever respective choice is selected. The name "Test" and "Calibration" are hard-coded in the program file.

Some constants defined in this class

Variable Name	Description
---------------	-------------

COLOR_MAIN_INACTIVE	Color, mouse doesn't hover on 'Select Mode' button
COLOR_MAIN_ACTIVE	Color, mouse hover on 'Select Mode' button
font	Constant declaring font-type to print text, font size is 30

4.4.1 class DropDown()

class DropDown() manage the behavior of the button, it has three methods which are handle_event, draw, and option. Each will be explained further in the following section

4.4.1.1 attribute and parameters

Here are some attributes that belongs to this class

Attributes	Data type	Description
rect_main	pg.Rect	This attribute is an object of pg.Rect, will obtained the passed x, y, w, h) to initialize rect object
rect_list1	pg.Rect	This attribute is an object of pg.Rect, will obtained the passed x, y, w, h) to initialize rect object
rect_list2	pg.Rect	This attribute is an object of pg.Rect, will obtained the passed x, y, w, h) to initialize rect object
active_main	boolean	Initialize as True, act as a flag for main selection (Select Mode), will output False once the button is pressed
active_list1	boolean	Initialize as False, will act as flag for list1 (Test), will output True if this option is shown and selected.
active_list2	boolean	Initialize as False, will act as flag for list2 (Calibration), will output True if this option is shown and selected.
color_main	pg.Color	A pygame-specific color format, written in format (Red, Green, Blue), please refer to Colors module. This will determine the color of the main selection (select mode).
color_list	int	A pygame-specific color format, written in format (Red, Green, Blue), please refer to Colors module. This will determine the color of the list1 and list2 (test and calibration)
main_surface	font.render	will use font constant to write 'Select Mode' within the rect_main button
list1_surface	font.render	will use font constant to write 'Test' within the rect_list1 button
list2_surface	font.render	will use font constant to write 'Calibration' within the rect_list2 button

Parameters		
win	pg.display.set_mode	The screen onto which given parameters will be displayed. Initialize as class attribute by defining self.screen
x	int	x coordinate the starting point of pygame.rect object for button

y	int	y coordinate the starting point of pygame.rect object for button
w	int	the height of pygame.rect object for button
h	int	the height of pygame.rect object for button

4.4.1.2 `def handle_event(event)`

This function will handle the behavior and the value of each option when selected.

	Variable	Data type	Description
Parameters	event	pg.event.type	<p>pg.event will be passed into this function, pg.event is an pygame object that represents event. there are two types of event that will be handled here, they are:</p> <ul style="list-style-type: none"> - pg.MOUSEMOTION This function of pygame will monitor the movement of the mouse and will return the coordinate of the mouse, this function is used to detect when certain button is hovered by a mouse - pg.MOUSEBUTTONDOWN This function of pygame will return True whenever a mouse is clicked, each button of mouse return specific value of integer, but this will be ignored in this program.
Returns	-	-	-

4.4.1.3 `def draw()`

Initially 'Select Mode' will written as the selected option, but when either 'Vertical' or 'Horizontal' is selected, this will overwrite the 'Select Mode, indicating that currently a certain option is selected.

No parameter is passed, this function will draw respective selected option based on the Boolean value of each option stated in handle_event function.

4.4.1.4 `def option()`

Initially 'Select Mode' will written as the selected option, when this button is clicked, two options 'vertical' and 'horizontal' will be drawn onto screen. This function handles this behavior.

No parameter is passed, this function will draw respective selected option based on the Boolean value of each option stated in handle_event function.

4.5 Pggraph

Pggraph.py is a self-built module that is responsible for creating graphics used during each mode execution. This module contains function to draw left arrow, right arrow, up arrow, down arrow, fixation cross, horizontal bar, vertical bar.

4.5.1 `def arrow_right(screen, color), def arrow_left(screen, color), def arrow_down(screen, color), def arrow_up(screen, color)`

All these functions have the same mechanism, and are passed with the same parameters as well, that's why this section will simultaneously describe how they work.

	Variable	Data type	Description
Parameters	screen	pg.display.set_mode	The screen onto which each graphic is drawn
	color	pg.Color	A pygame-specific color format, written in format (Red, Green, Blue), please refer to Colors module. This will determine the color of each graphic.
Returns	-	-	-

4.5.2 `horizontal bar (screen, color, w, h), vertical bar (screen, color, w, h)`

Due to the same reason as above, both these functions will be describe in this section

	Variable	Data type	Description
Parameters	screen	pg.display.set_mode	The screen onto which each graphic is drawn
	color	pg.Color	A pygame-specific color format, written in format (Red, Green, Blue), please refer to Colors module. This will determine the color of each graphic.
	w	int	will define the width of the bar
	h	int	will define the height of the bar
Returns	-	-	-

4.6 Sequence

Sequence is a self-built module that manages the sequence of display during test and calibration mode. It contains the following classes and functions:

4.6.1 `def send2UDP(message, iter, dir_idx)`

This function will send the input message via UDP to call DataRec.exe to initiate recording

	Variable	Data type	Description
Parameters	message	string	This will contain message to call DataRec.exe to start recording, this message has to obey certain format, please refer to DataRec.exe section
	iter	int	This denotes the iteration of trial in each mode
	dir_idx	int	This parameter represents directions in numbers. There are four direction for both horizontal and vertical orientation Horizontal (1 for left, 2 for right) Vertical (3 for down, 4 up)
Returns	-	-	-

4.6.2 `def disp_timer(win, t0, x, y)`

This function will help calculate how many seconds have passed since the t0 and return how many seconds passed

	Variable	Data type	Description
Parameters	win	pygame.display.set_mode	The screen onto which the time is printed
	t0	pg.time.get_ticks()	A time definition from pygame module that was declared at some other time in the program, it functions as initial/start time
	x, y	int	The coordinate on win screen, at which the time will be printed
Returns	dt	int	Return the time elapsed from t0 to declaration of this function. The time will be defined in second.

4.6.3 `def arrow(dir_list, screen, color, idx)`

This function will receive a number that corresponds to direction of the arrow that will be drawn on the screen.

	Variable	Data type	Description
--	----------	-----------	-------------

Parameters	dir_list	list	List containing 10 numbers consisting either combination of 1 and 2 (for horizontal) or 3 and 4 (for vertical)
	screen	pygame.display.set_mode	The screen onto which the arrow is printed
	color	pygame.Color	A pygame-specific color format, written in format (Red, Green, Blue), please refer to Colors module. This will determine the color of the arrow.
	idx	int	This will serve as index of dir_list, this idx will access each element of dir_list, so that function knows which arrow to print
Returns	-	-	-

4.6.4 `def text_disp(text, win, x, y)`

This function will display the parameter text on to screen

	Variable	Data type	Description
Parameters	text	string	This text will be printed on the screen
	win	pygame.display.set_mode	The screen onto which the text will be printed.
	x, y	int	The coordinate on win screen, at which the given parameter will be printed
Returns	-	-	-

4.6.5 `def count time(t0)`

Similar to `disp_timer` this function will help calculate how many seconds have passed since the `t0` and return how many seconds passed, the only difference is that this function doesn't print onto the screen

	Variable	Data type	Description
Parameters	<code>t0</code>	<code>pg.time.get_ticks()</code>	A time definition from pygame module that was declared at some other time in the program, it functions as initial/start time
Returns	<code>dt</code>	<code>int</code>	Return the time elapsed from <code>t0</code> to declaration of this function. The time will be defined in second.

4.6.6 `def bar(screen, width, height, id)`

This function will print horizontal or vertical bar for test mode. In horizontal mode, the height of the bar represents the accuracy of imagery task, whereas the same rule applies for bar width in vertical mode.

	Variable	Data type	Description
Parameters	<code>screen</code>	<code>pygame.display.set_mode</code>	The screen onto which the bar will be printed.
	<code>width</code>	<code>int</code>	The width of the bar
	<code>height</code>	<code>int</code>	The height of the bar
	<code>id</code>	<code>int</code>	Similar to arrow, in bar function, a number will represent the orientation of the bar, 1 or 2 will display horizontal bar, 3 or 4 will display vertical bar.
Returns	-	-	-

4.6.7 `class Sequence()`

`class Sequence` handles what sequence will be run. Remember that modes and directions can make up four combinations, calibration horizontal, calibration vertical, test horizontal, test vertical.

4.6.7.1 *attributes & parameters*

Here are some attributes that belongs to this class

Attributes	Data type	Description
------------	-----------	-------------

start time	list	This list contains the start time of each event in each trial, e.g. [0, 2, 4, 6, 8, 10] will execute the first event at 0s, second event at 2s, etc. Each number in the list represent seconds.
end time	list	<p>This list contains the start time of each event in each trial, e.g. [2, 4, 6, 8, 10, 12] will end the first event at 0s, second event at 4s, etc. Each number in the list represent seconds.</p> <p>Now it becomes clear that each event duration will be defined between the start and end time. In the give example there are 6 events each lasting 2 seconds.</p>
pg.clock	pg.time.Clock()	A pygame definition for clock, this will define the at what FPS program should run
playedOnce	boolean	Initialize as False, will act as a flag for beep sound to play once each trial
dataRecOnce	boolean	Initialize as False, will act as flag for DataRec.exe to be executed only once
n	list	List that serves as iteration for ten trials in each run
x_n	int	Indicate x coordinate of text position
y_n	int	Indicate y coordinate of text position In this program, the text will be number of iteration

Parameters		
win	pygame.display. set_mode	The screen onto which given parameters will be displayed. Initialize as class attribute by defining self.win

4.6.7.2 *def cal_sequence*

This function will run the sequence for calibration mode, this function will be called the horizontal or vertical calibration sequence i number of times, where i is number of trial.

	Variable	Data type	Description
Parameters	hv_list	list	This list contains either combination of 1 and 2 for horizontal or 3 and 4 for vertical orientation. This list will be declared in horizontal_cal and vertical_cal function
	iter_idx	int	The iteration that will look up each value in hv_list one by one
Returns	-	-	-

4.6.7.3 *def horizontal_cal*

This function will generate the list of 1 and 2, there should be five '1' and five '2' that are declared at random order. This function will call `cal_sequence` method to run the sequence. The list generated in this function will be passed as `hv_list` while the iteration `i` will be passed as `iter_idx`.

4.6.7.4 *def vertical_cal*

This function will generate the list of 3 and 4, there should be five '3' and five '4' that are declared at random order. This function will call `cal_sequence` method to run the sequence. The list generated in this function will be passed as `hv_list` while the iteration `i` will be passed as `iter_idx`.

4.6.7.5 *def test_sequence*

This function will run the sequence for test mode, this function will be called the horizontal or vertical test sequence `i` number of times, where `i` is number of trial. The only thing that separates test with calibration mode is that test sequence ends with evaluation bar at each trial.

	Variable	Data type	Description
Parameters	<code>hv_list</code>	list	This list contains either combination of 1 and 2 for horizontal or 3 and 4 for vertical orientation. This list will be declared in <code>horizontal_cal</code> and <code>vertical_cal</code> function
	<code>iter_idx</code>	int	The iteration that will look up each value in <code>hv_list</code> one by one
	<code>bar_w</code>	int	<code>bar_w</code> will define bar width, in horizontal orientation <code>bar_w</code> will remain at fix value
	<code>bar_h</code>	int	<code>bar_h</code> will define bar height, in vertical orientation <code>bar_h</code> will remain at fix value
	<code>hv_id</code>	int	<code>hv_id</code> indicates the bar orientation, 1 or 2 will output horizontal bar, 3 or 4 will output vertical bar
Returns	-	-	-

4.6.7.6 *def horizontal_test*

This function will generate the list of 1 and 2, there should be five '1' and five '2' that are declared at random order. This function will call `cal_test` method to run the sequence `i` times. The list generated in this function will be passed as `hv_list` while the iteration `i` will be passed as `iter_idx`. Bar height will also be defined here. Currently, it is set to output random value each iteration.

4.6.7.7 *def vertical_test*

This function will generate the list of 3 and 4, there should be five '3' and five '4' that are declared at random order. This function will call `cal_test` method to run the sequence `i` times. The list generated in this

function will be passed as hv_list while the iteration i will be passed as iter_idx. Bar width will also be defined here. Currently, it is set to output random value each iteration.

4.7 Widget

Widget is a self-built module that manages widgets used on display. It contains the declaration of button and input box to input subject number

Some constants defined in this class

Variable Name	Description
FONT1	Constant declaring font-type to print text, font size is 35
FONT2	Constant declaring font-type to print text, font size is 30
COLOR_INACTIVE	Color when the button is not clicked
COLOR_ACTIVE	Color when the button is clicked

4.7.1 class Button()

This class will define the handling of button on pygame screen

4.7.1.1 attributes & parameters

Here are some attributes that belongs to this class, all parameters passed into this button then assign as attributes

Attributes	Data type	Description
-	-	-

Parameters		
color	pygame.Color	A pygame-specific color format, written in format (Red, Green, Blue), please refer to Colors module. This will determine the color of the arrow.
x	int	x coordinate the starting point of pygame.rect object for button
y	int	y coordinate the starting point of pygame.rect object for button
w	int	the height of pygame.rect object for button
h	int	the height of pygame.rect object for button
text	string	the text which will be printed on the button

4.7.1.2 def draw(win, outline = None)

This function draw button on the screen, this function will work with the help of pg.draw.rect, that could draw rectangle object on screen

	Variable	Data type	Description
--	----------	-----------	-------------

Parameters	win	pygame.display.set_mode	The screen onto which the bar will be printed.
	outline	int	the thickness of the button border Initialize as None.
Returns	-	-	-

4.7.1.3 *def isOver(pos)*

This function will control handling of event when mouse position is within button box, in other words when mouse hovers the button

	Variable	Data type	Description
Parameters	pos	(int, int)	The position of mouse. The pos variable that will be passed here is obtain from pygame. MOUSEMOTION, that will return value of x and y.
Returns	-	boolean	This function will return True or False depends on the position of the mouse.

4.7.2 *class InputBox()*

This class will define the handling and behavior of input box on pygame screen

4.7.2.1 *attributes & parameters*

Here are some attributes that belongs to this class, while the ones in attributes are initialized to have initial values, parameters are the one that is defined as attributes,

Attributes	Data type	Description
rect	pg.Rect	This attribute is an object of pg.Rect, will obtained the passed x, y, w, h) to initialize rect object
color	pg.Color	will be initialized as COLOR_INACTIVE
txt_surface	pg.FONT2.render	will use constant of FONT2 to write text on the screen
active	False	initializes as False, will act as a flag indicating input box is active or inactive
text2UDP	string	initializes as empty string, will have a value according to what user input inside the box

Parameters		
x	int	x coordinate the starting point of pygame.rect object for button

y	int	y coordinate the starting point of pygame.rect object for button
w	int	the height of pygame.rect object for button
h	int	the height of pygame.rect object for button
text	string	the text which will be printed on the button

4.7.2.2 *def handle_event(event)*

This function will define the behavior of the input box. Several behaviors that are defined include:

1. Change color border when mouse is clicked within the input box area,
2. Delete a character when backspace is pressed
3. Store the string into text2UDP and delete the string in the input box when enter is pressed
4. Print the a character in the input box according to key pressed on the keyboard

The event obtained from pg.event.get() is passed into this function.

	Variable	Data type	Description
Parameters	pos	(int, int)	The position of mouse. The pos variable that will be passed here is obtain from pygame. MOUSEMOTION, that will return value of x and y.
Returns	text2UDP	string	text2UDP will contain string that will be sent through UDP to DataRec.exe to start recording

4.7.2.3 *def update*

Update serves as the function to widen the length of input box, if the input character reach the default box length. THIS IS WHERE I STOP

	Variable	Data type	Description
Parameters	pos	(int, int)	The position of mouse. The pos variable that will be passed here is obtain from pygame. MOUSEMOTION, that will return value of x and y.
Returns	-	boolean	This function will return True or False depends on the position of the mouse.

4.7.2.4 *def draw(screen)*

	Variable	Data type	Description
--	----------	-----------	-------------

Parameters	pos	(int, int)	The position of mouse. The pos variable that will be passed here is obtain from pygame. MOUSEMOTION, that will return value of x and y.
Returns	-	boolean	This function will return True or False depends on the position of the mouse.

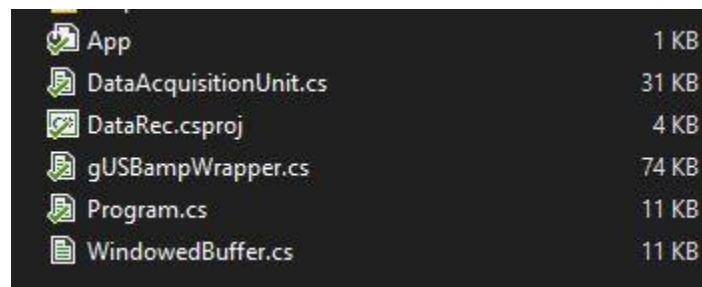
4.8 DataRec.exe

DataRec.exe is an executable application that is generated by the C# program. Its role is to interact with gUSB amplifiers (EEG recorders) to start, stop, save recording of EEG.

In terms of interaction with your main program. DataRec will *listen* to message/command sent by the main program (sometimes referred as master). Keep in mind that they should share common PORT in order to communicate, in this project port is set to 1010.

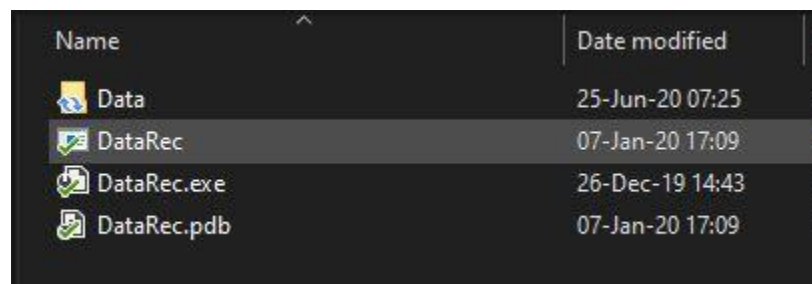
4.8.1 Build and run DataRec.exe

As of 2020, June 25th, the program to build DataRec.exe is written in C++ and is named DataRec.csproj. Then we can open, build, and run this .csproj with Visual Studio. Use 'Start without debugging' to run the file. Once done, executable file (.exe) in bin -> Debug or x64 folder will be created. This file can be used to record data, instead of building and running every time.



App	1 KB
DataAcquisitionUnit.cs	31 KB
DataRec.csproj	4 KB
gUSBampWrapper.cs	74 KB
Program.cs	11 KB
WindowedBuffer.cs	11 KB

These are the files required to build DataRec.exe



Name	Date modified
Data	25-Jun-20 07:25
DataRec	07-Jan-20 17:09
DataRec.exe	26-Dec-19 14:43
DataRec.pdb	07-Jan-20 17:09

Above files will generated once you finished running .csproj



Above is how it will look like once you finish running or opening DataRec.exe. As you can see DataRec.exe is just a console ready to receive some message. The message will determine how file is named.

4.8.2 Illustration of How DataRec.exe and BCI Interface Communicate

As mentioned above DataRec.exe's job is to communicate with gUSBamp and record data. DataRec.exe keep listening to message sent to it. This message should obey some pre-determined format that is explained in next section. This section will give you basic illustration how DataRec.exe and this python program interacts and what it will look like once received some messages.

4.8.2.1 Basic test to see how

1. Open DataRec.exe either by running the project or executing built DataRec.exe

4.8.3 How to run DataRec.exe simultaneously with main

The DataRec.exe is executed simultaneously with main program. DataRec.exe can be run by using this command:

```
subprocess.Popen([DataRec.exe address]\DataRec.exe', shell=False)
```

[Data.Rec.exe address] is where DataRec.exe is located.

4.8.4 Format of Recorded Data

The input format is briefly explained in chapter 'Input format', but that information is only an input format, the full format of data name will be explained in this chapter.

Originally, this is the format of received message specified in the C# program, this is mentioned here, for the sake of clarity, because as of 2020, June 24th SDK for g.USBamp is still written in the C# code.

T/O	Subject	Session	Run	Duration	Label	task
XX	X	X	X	X	X	X

But this format was no longer used, and in my code was modified to this

T/O	Subject	Session	Run	Duration	Direction	Sequence
XX	X	X	X	X	X	X

where 'X' indicates one digit.

Details of each variable

Variable	Description	Allowed Values										
T/O	Indicates the training or online test, for classical BCI application, this part is hard-coded as ‘T’	-										
Subject	Specify the number of test subject, consist of two numbers, so one-digit numbers need zero preceding them.	00 ... 99										
Session	Specify the session number	0 ... 9										
Run	Specify the run number	0 ... 9										
Duration *	Specify the length of duration, this part is hardcoded as ‘x’, which means it is not defined. When it’s not defined the program will record for two seconds.	-										
Direction	<div>Indicates the direction of the arrow, which will come in handy in future data analysis.</div> <div>Will holds values of 1, 2, 3, or 4, each represents different direction of the arrow as follows:</div> <table><tr><th>Value</th><th>Direction</th></tr><tr><td>1</td><td>Left</td></tr><tr><td>2</td><td>Right</td></tr><tr><td>3</td><td>Up</td></tr><tr><td>4</td><td>Down</td></tr></table>	Value	Direction	1	Left	2	Right	3	Up	4	Down	1 ... 4
Value	Direction											
1	Left											
2	Right											
3	Up											
4	Down											
Sequence	Indicates the nth sequence, where n is number between 0 to 9, because our run consists of 10 trials	0 ... 9										

* Please note that 'duration' will not be mentioned in the recorded filename, which means the file name of recorded data will only have SIX digits, which are subject, session, run, direction and sequence.

Here is how the console of an IDE (PyCharm) will look like, when a sequence is being run.

```
=== Running Calibration Mode Horizontal Direction ===
[1, 2, 2, 2, 1, 2, 1, 1, 2, 1]
Received a broadcast from 127.0.0.1:64568
T0532x21
Opening and initializing devices...
    invalid hardware version
Stopping devices, closing them and cleaning up...
Received a broadcast from 127.0.0.1:64568
T0532x22
Opening and initializing devices...
    invalid hardware version
Stopping devices, closing them and cleaning up...
Received a broadcast from 127.0.0.1:64568
T0532x23
Opening and initializing devices...
    invalid hardware version
Stopping devices, closing them and cleaning up...
Received a broadcast from 127.0.0.1:64568
T0532x14
Opening and initializing devices...
    invalid hardware version
Stopping devices, closing them and cleaning up...
Received a broadcast from 127.0.0.1:64568
```

Square colors	Details
	The list containing the order of the arrow direction, please refer to naming format
	The name of the file, this will translate to 'T' mode, subject 05, session 3, run 2, right arrow, trial 1
	Notice the last digits is incrementing, this shows the trial number while the second last digit indicates the arrow direction*

*There is still unresolved bug about this please consult problems and possible bugs section

4.9 Miscellaneous

4.9.1 short_beep_200ms.wav

This .wav file is an 200 milliseconds long audio file. This file must be located in the same directory as main file. This beep sound will be run each trial either in test or calibration mode.

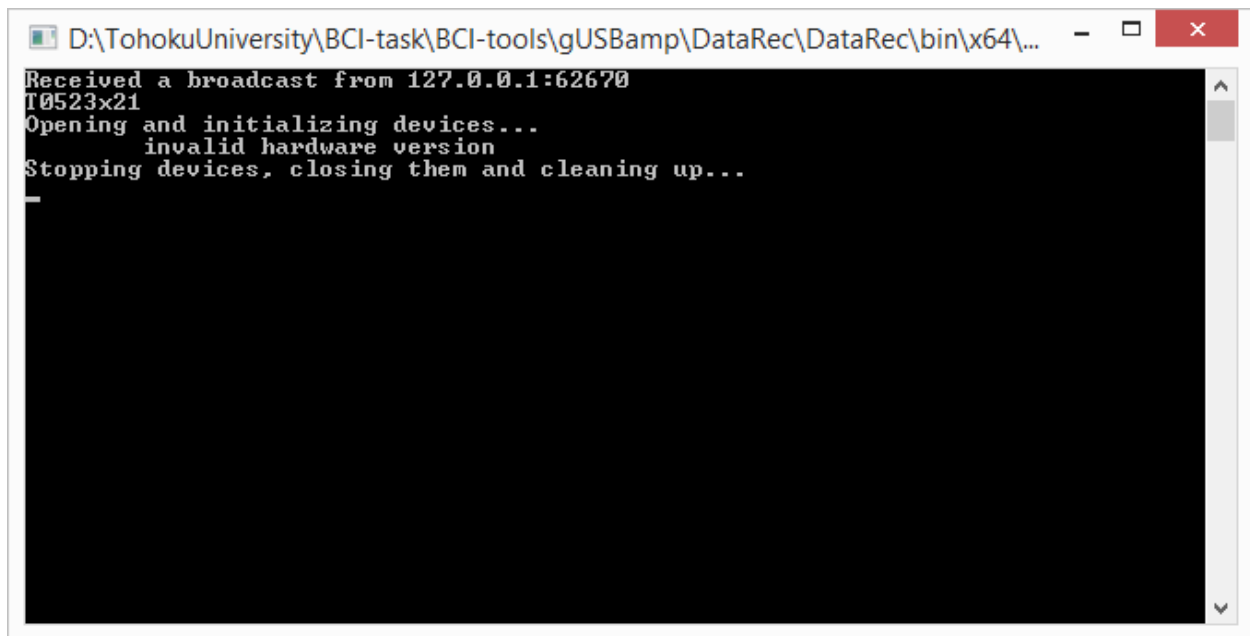
5 Problems & Possible Bugs

5.1 Regarding DataRec.exe Console

This section will discuss several problem and solution encountered during the making of connection to DataRec.exe via UDP.

Why you need UDP Connection, and

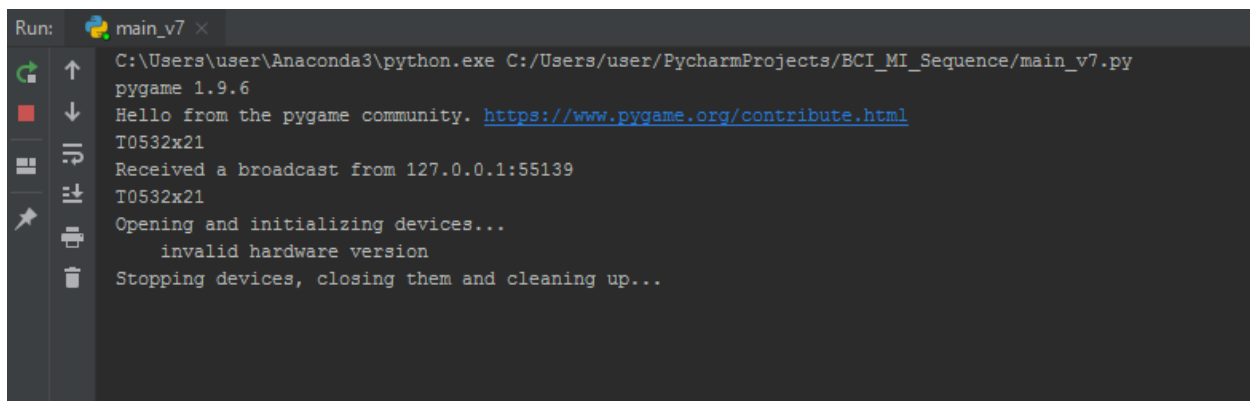
When running DataRec.exe and python separately, DataRec will be run on separate console, such as this



```
D:\TohokuUniversity\BCI-task\BCI-tools\gUSBamp\DataRec\DataRec\bin\x64\...
Received a broadcast from 127.0.0.1:62670
T0523x21
Opening and initializing devices...
invalid hardware version
Stopping devices, closing them and cleaning up...
```

Then when DataRec.exe detect some request the corresponding responds will be shown in the console.

But when simultaneously running DataRec.exe and main program with the command given in the previous chapter, console will not appear, instead the console inside associated IDE will be used. This is an example of the message shown in PyCharm console.



```
Run: main_v7
C:\Users\user\Anaconda3\python.exe C:/Users/user/PycharmProjects/BCI_MI_Sequence/main_v7.py
pygame 1.9.6
Hello from the pygame community. https://www.pygame.org/contribute.html
T0532x21
Received a broadcast from 127.0.0.1:55139
T0532x21
Opening and initializing devices...
invalid hardware version
Stopping devices, closing them and cleaning up...
```

Also, it is important to note that DataRec.exe is highly depended on specification of operating system it is currently run, it is advised that initially to always run the C# code to generate DataRec.exe. The DataRec.exe generated with x32 system won't work in x64 system.

5.2 Regarding Test Button

As of Jan 27th, 2020, no function is assigned to this button.

5.3 Regarding Recorded Data

As of Jan 29th, 2020, there's still a bug regarding the recorded data. Notice in the console above, the initial data is T0532x21. It start with 1 (2nd trial) where it should have started with 0, it happened when first run of the sequence, but oddly, the bug fix itself after the first run, without changing the code or

anything. In other words, this bug where you only record data 2-10 (9 data) only happened once in the initial run. This bug is still being investigated for solution.

6 Author

This code is written by Orvin Demy

7 Contribution

Thanks to David Achancaray for his guidance and direction throughout the making of this program