

Manual de usuario de herramienta para la evaluación de la fisiología visual.



En el marco del proyecto:

**Desarrollo de una herramienta para la evaluación de la fisiología visual usando
electroencefalografía portable y de bajo costo**

**Proyecto de Investigación
Pregrado Bioingeniería
Verónica Henao Isaza**

**Programa de Bioingeniería
Medellín, Antioquia
2020**



INTRODUCCIÓN

El objetivo principal de este proyecto es desarrollar una herramienta para la evaluación de la fisiología visual mediante el análisis de señales electroencefalográficas obtenidas con un dispositivo portable y de bajo costo, con el fin de lograr registrar de manera efectiva los potenciales visuales en estado estacionario con una herramienta portable, una interfaz amigable, con un protocolo de montaje sencillo, eficiente y asequible, favoreciendo de gran manera la ejecución de pruebas en diferentes entornos, además, el desarrollo y aplicación de medidas de procesamiento cada vez más especializadas pueden permitir el diagnóstico de patologías y la rehabilitación de pacientes con deficiencias visuales partiendo desde los parámetros más simples y que entregan más información para este tipo de sujetos.

La herramienta se entrega como un ejecutable pero adicionalmente se comparten todos los archivos para su manipulación y uso académico, por lo tanto en el siguiente manual se habla de la parte técnica y funcional de la herramienta.

GLOSARIO

- Usuario: Persona que manipula la herramienta, puede ser o no la misma persona que le realiza el montaje al sujeto.
- Sujeto: Persona que es registrada por uno o más usuarios, puede ser un paciente o una persona sin ninguna patología asociada. Será a quien se le realice el montaje de los electrodos y se le muestre la estimulación visual.

BIBLIOTECAS Y DEPENDENCIAS

La herramienta fue desarrollada en python 3.7 con el IDE Spyder de Anaconda en Windows 10.

Instalaciones previas

Python, para evitar mayor cantidad de instalaciones se recomienda instalar Anaconda

- pygame -> conda install -c cogsci pygame
- wmi -> conda install -c primer wmi
- pymongo -> conda install -c anaconda pymongo
- serial -> pip install serial
- pyserial -> conda install -c anaconda pyserial
- pylsl -> pip install pylsl
- pyOpenBCI -> pip install pyOpenBCI
- pip install bitstring
- pip install xmlltodict

Bibliotecas utilizadas

- PyQt5
- Matplotlib
- scipy
- numpy
- pandas
- wmi
- os
- pygame
- time
- traceback
- sys
- pylsl
- datetime
- csv
- errno
- serial
- subprocess
- pymongo
- spectrum
- pyOpenBCI

COMPONENTES

ARCHIVOS .PY	Definición
model.py	Modelo
controller.py	Controlador
View.py	Vista
Stimulation_Acuity.py	Estimulación agudeza de Vernier
dataprocessing.py	Procesamiento multitaper
Plot_stfft.py	Procesamiento tiempo-frecuencia
randData.py	Simulador de datos
Server.py	Conexión con OpenBCI
linearFIR.py	Filtro FIR
nonlinear.py	Filtro no lineal

ARCHIVOS .UI	Definición
ViAT	Inicio
Agregardatos	Permite agregar o elegir un sujeto de la base de datos
Adquisicion	Permite realizar la conexión con el openBCI y medir la impedancia
Adquisicion_accion	Permite visualizar los datos adquiridos en tiempo real
Buscardatos	Permite buscar y observar los datos en la base de datos
Visualizacion	Permite visualizar señales guardadas previamente

MODELO

Contiene los datos y la funcionalidad de la aplicación, es decir, se encarga de realizar las funciones de actualizar, búsqueda, consulta, procesamiento de datos, etc.

model.py

Funciones

1. **init**

Recibe: Nombre de la base de datos, nombre de la colección de la base de datos, datos para grafica.

Función: Realiza la conexión con la base de datos de mongo, diseña los filtros llamando a *filtDesign()* y define las ubicaciones de almacenamiento de los archivos.

2. **newLocation**

Recibe: La nueva ubicación

Función: Seleccionar la ubicación de los nuevos registros adquirir

3. **location**

Recibe:

Función: Redefine la ubicación de los nuevos registros adquirir

4. **startDevice**

Recibe:

Función: Inicia el envío de los datos, utiliza la función *subprocess.Popen* para iniciar un proceso nuevo sin detener el existente.

5. **stopDevice**

Recibe:

Función: Detener el envío de los datos y cerrar el proceso.

6. **startData**

Recibe:

Función: Define una matriz de ceros para los datos con la cantidad de canales y la cantidad de muestras.

Utiliza la función *resolve_stream* de la biblioteca *pysl* para recibir los datos enviados por el servidor *Server.py* o el simulador de datos *randData.py*.

Utilizar el objeto *StreamInlet* para recibir datos de transmisión (y metadatos) desde la red del laboratorio, toma las trasmisiones disponibles y finalmente se hace un *pull_chunk()* que extrae un trozo de muestras de la entrada.

7. **stopData**

Recibe:

Función: Detiene la acción de recibir los datos cerrando el stream con *close_stream()* y llama los módulos de procesamiento *dataprocessing.py* y *plot_stft.py*

8. **startStimulus**

Recibe:

Función: Llama el módulo de estimulación *Stimulation_Acuity.py* y lo inicia.

9. **stopStimulus**

Recibe:

Función: Detiene la estimulación al cerrar el *pygame*.

10. **startZ**

Recibe:

Función: Inicia la recepción de datos con *StreamInlet* para posteriormente hallar la impedancia de los datos.

11. **stopZ**

Recibe:

Función: Cierra la recepción de los datos con *close_stream* al finalizar la lectura de la impedancia.

12. **readZ**

Recibe:

Función: Hace un *pull_sample()* para tomar un valor y realizar la operación por ley de ohm. V = voltaje rms recibido. i = corriente del dispositivo = 6 nA

$$Z = \frac{V * \sqrt{2}}{i}$$

El valor de Z es dividido por 1000 antes de presentarse en la vista.

13. **readData**

Recibe:

Función: Hace un *pull_chunk()* para tomar una cantidad de valores y realizar una diferencia entre el canal de referencia y cada uno de los canales de interés.

Crea un Dataframe de cada resultado para almacenarlo en un archivo .csv

14. **filtDesign**

Recibe:

Función: Diseña un filtro pasa-bajas y un filtro pasa-altas

15. **filtData**

Recibe:

Función: Llama la función *readData()* y utiliza la función *filtfit* de *scipy.signal* para aplicar un filtro a los datos recibidos en tiempo real.

16. **Pot**

Recibe:

Función: Aplica el método de Welch al canal o configuración definida por el usuario en la interfaz

17. **laplace**

Recibe: Del menú desplegable de la interfaz gráfica, toma los valores de los canales de interés.

Función: Crea una matriz para realizar la operación de Laplace con los valores entregados.

18. **returnLastData**

Recibe:

Función: Ejecuta *Pot()* y retorna los valores a la vista para graficarlos.

19. **returnLastZ**

Recibe:

Función: Ejecuta *readZ()* y retorna los valores de la impedancia.

20. **returnLastStimulus**

Recibe:

Función: Ejecuta *readData()*

21. **add_into_collection_one**

Recibe:

Función: Agrega un sujeto a la base de datos

22. **search_one**

Recibe:

Función: Busca una persona de la base de datos

23. **search_many**

Recibe:

Función: Entrega la lista de integrantes de la base de datos

24. **delete_data**

Recibe:

Función: Elimina un sujeto de la base de datos

25. **assign_data**

Recibe:

Función: Entrega los datos a graficar de un archivo .csv

26. **return_segment**

Recibe:

Función: Permitir el avance en el tiempo de la señal

27. **signal_scale**

Recibe:

Función: Permitir realizar la ampliación o disminución de la señal

28.file_location

Recibe:

Función: Permite encontrar un archivo de un sujeto determinado.

CONTROLADOR

controller.py

Determina que procesos debe realizar el modelo cuando el usuario interacciona con el sistema, para luego comunicarle a la vista los resultados. Simplemente toma una orden de la vista y se la envía al controlador, por esto las funciones en este modulo hacen referencia a las mismas funciones que hay en el modelo.

El controlador esta compuesto por 3 clases:

- Principal: Inicia el servidor de la base de datos MongoDB en un subproceso de la aplicación, inicia la aplicación y define las variables de la vista y el modelo. Ingresa la vista y el modelo a cada uno de los controladores y asigna los controladores para utilizarlos en la vista.
- Controller: Maneja todas las funciones relacionadas con los datos recibidos del dispositivo y las señales graficadas en la interfaz.
- Controlador: Maneja todas las funciones relacionadas con la base de datos de los sujetos y la interacción con MongoDB.

VISTA

view.py

Gestiona como se muestran los datos en la interfaz gráfica por medio de las ordenes que le envía al controlador y los datos que este le devuelve de las operaciones realizadas en el modelo.

Está compuesto 6 vistas diseñadas en QtDesigner y por 8 clases que controlan cada una de las vistas y dos adicionales para el control de hilos para los procesos que se llevan acabo en simultaneo

Clases:

- **WorkerSignals:** Define las variables que controlan los procesos como:
 - Finalizar, cuando el proceso llega a su fin
 - Error, cuando se presentan tareas combinadas
 - Resultados, los datos esperados en cada proceso
 - Progreso, el seguimiento de proceso.
- **Worker:** Realiza la ejecución de los hilos
- **ViAT (Vista 1):** Ejecuta la vista principal **ViAT.ui** (Figura 1) con el menú para desplazarse por la aplicación. Contiene una breve descripción de la aplicación y cuenta con 3 botones:
 1. Iniciar registro: Permite dirigirse a la vista 2.
 2. Base de datos pacientes: Permite dirigirse a la vista 6

3. Salir: Permite salir de la aplicación.

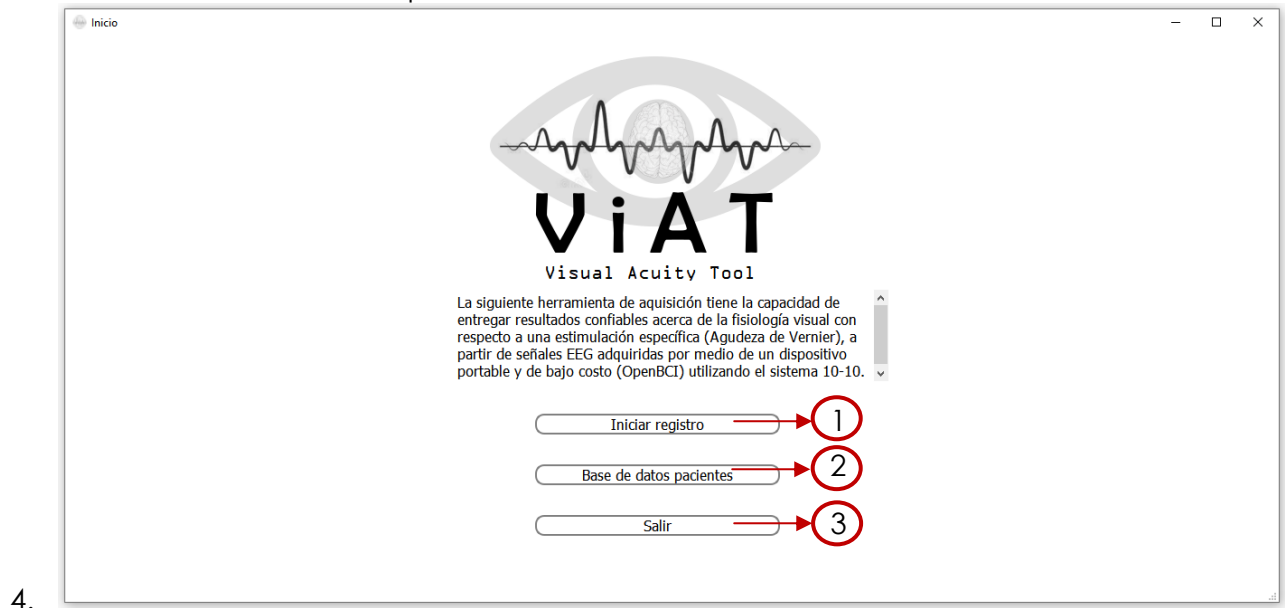


Figura 1. ViAT.ui

- **LoadRegistration (Vista 2):** Ejecuta la vista **Agregardatos.ui** (Figura 2). Tiene la función de crear y administrar una base de datos de cada sujeto registrado con la herramienta ViAT.
 1. Campos de libre escritura: Son campos de los datos comunes como: Nombre, Apellido, Cédula (CC), Agudeza de Snellen (Valor que debe encontrarse en la escala de Snellen es decir desde 20/16 hasta 20/200, se debe evaluar sin gafas), Agudeza de Snellen corregida (hace referencia a la agudeza de un sujeto al usar gafas), Edad, ¿Tiempo desde el accidente visual? (Es un dato tomado para sujetos que hayan perdido funciones visuales debido a un accidente cráneo encefálico) y responsable (Persona que realiza el registro de un sujeto, es decir el usuario actual de la aplicación)
 2. Botón de verificación: Permite al usuario reducir tiempos de registro al autocompletar los campos de un sujeto que se encuentre previamente registrado en la base de datos (Figura 3).
 3. Botones de menú desplegable: Permiten al usuario responder de manera concisa preguntas cortas como: Sexo (Femenino o masculino), Ojo dominante (Derecho o izquierdo), ¿Usa gafas? (Si o No), Estímulo (Vernier) por el momento es el único estímulo implementado en la herramienta.
 4. Permite al usuario tener la autonomía de elegir el lugar de ubicación de los archivos resultantes de la estimulación (Registro, marcas, procesamiento con multitaper y procesamiento con tiempo frecuencia)
 5. Permite al usuario desplazarse a la base de datos para buscar un sujeto vista 5
 6. Permite agregar un sujeto, luego de completar todos los campos.
 7. Al verificar un sujeto de la base de datos, los campos de la columna 2 (Figura 3) permiten ser manipulados y actualizados con el botón “Actualizar” se creara un

nuevo registro de la base de datos con los mismos datos de la columna 1 y los nuevos de la columna 2, en la base de datos se conservan ambos registros como historia clínica, si se desea eliminar alguno de los registros se debe ir a la base de datos y confirmar esta acción.

8. Permite ir a la siguiente vista. (Vista 3)
9. Es el botón de ayuda, contiene el siguiente mensaje como instrucción de uso de la vista: "Diríjase al campo de CC y digite la cédula del sujeto a registrar, para verificar que no se encuentre ya en la base de datos, si no está llene todos los campos presentados a continuación y presione agregar, en el botón 'definir ubicación' podrá modificar la ubicación donde desea que quede guardado el registro, al finalizar presione el botón 'Siguiente'. Si el sujeto ya se encuentra en la base de datos y desea modificar un campo, presione 'Actualizar'. Si desea observar todos los sujetos en la base de datos presione 'Mostrar Pacientes'. Para volver al menú anterior presione 'Atrás'."

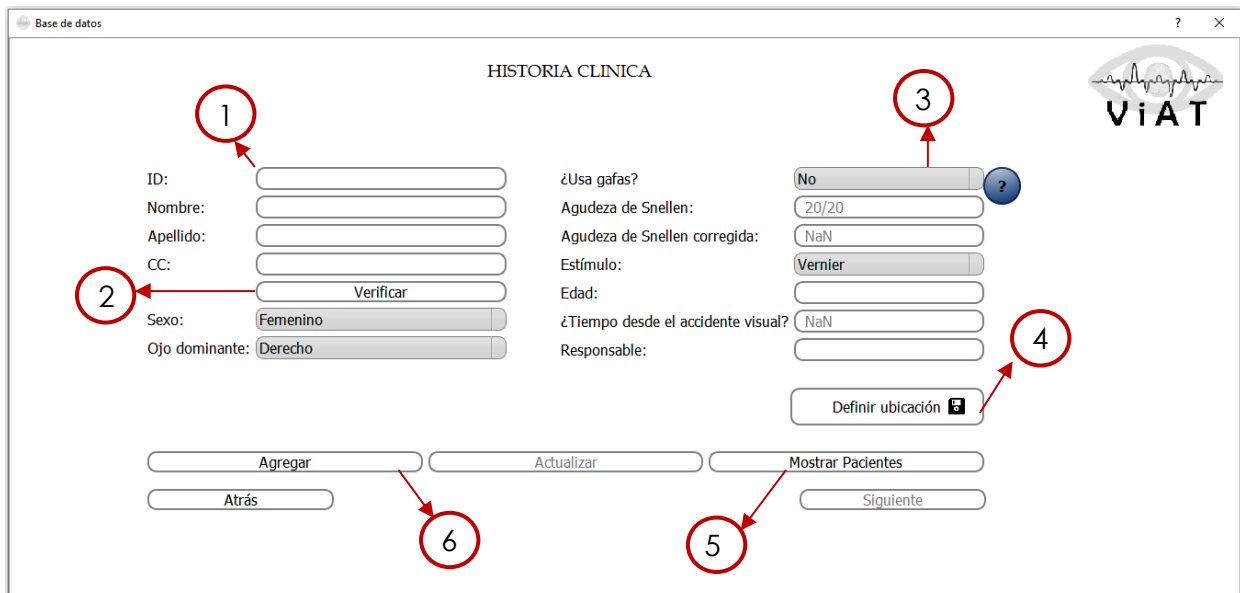
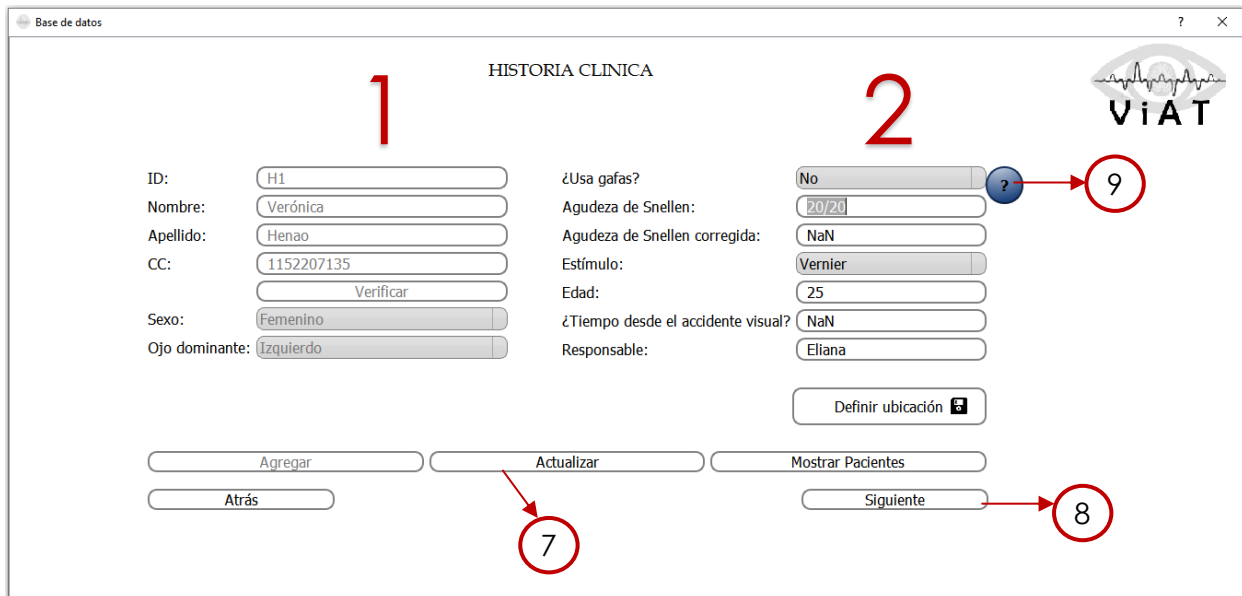


Figura 2. Agregar datos.ui



Base de datos

HISTORIA CLINICA

1

2

3

4

5

6

7

8

9

ID: H1

Nombre: Verónica

Apellido: Henao

CC: 1152207135

Sexo: Femenino

Ojo dominante: Izquierdo

¿Usa gafas? No

Agudeza de Snellen: 20/20

Agudeza de Snellen corregida: NaN

Estímulo: Vernier

Edad: 25

¿Tiempo desde el accidente visual? NaN

Responsable: Eliana

Definir ubicación

Agregar

Actualizar

Mostrar Pacientes

Atrás

Siguiente

Figura 3. Agregardatos.ui en uso

- DataAcquisition (Vista 3):** Ejecuta la vista **Adquisicion.ui** (Figura 4). Tiene la función de preparar el registro, permitiendo al usuario observar el montaje de electrodos sugerido (10-10), medir la impedancia de cada electrodo posicionado, conectar el dispositivo y verificar la segunda pantalla en la que se presentará el estímulo.
 - Montaje de electrodos corteza occipital (10-10): El sistema internacional de colocación de EEG toma cuatro puntos de referencia craneales universales (nasion, inion y ambos puntos pre-auricular), y distribuye proporcionalmente los electrodos del EEG sobre la superficie de la cabeza. Según el porcentaje de la distancia entre los sensores, tenemos el layout 10-20 con un total de 21 sensores, si partimos de una distribución de distancias del 10% y el 20% de las curvas de referencia central sagital y coronal. Si estas líneas centrales se dividen en un 10%, entonces tenemos el layout 10-10 con 81 sensores. Tener un menor porcentaje de distancia nos permite concentrar los 8 electrodos en la zona de interés en este caso, visión. Este esquema permite también presionar los botones de los electrodos para comenzar la medición de la impedancia.
 - Conectar dispositivo: Activa el envío de datos del dispositivo a la “red del laboratorio” en donde se tramite la información de cada electrodo. Este es el primer botón que se debe presionar al interactuar con la vista.
 - Verificar pantalla: Se activa inmediatamente después de presionar “Conectar dispositivo”, el estímulo se debe presentar en una pantalla aparte para poner tener control de la interfaz y observar la señal adquirida durante la estimulación. Al conectar una segunda pantalla por medio de un cable HDMI el software detectara la entrada y permitirá continuar con las demás tareas, si no se tiene una segunda pantalla, el software entregara un mensaje de advertencia solicitando dicha acción, no es posible continuar sin una segunda pantalla conectada.

Esta verificación se hace por medio de la biblioteca wmi:

```
obj = wmi.WMI().Win32_PnPEntity(ConfigManagerErrorCode=0)
displays = [x for x in obj if 'DISPLAY' in str(x)]
```

4. Al verificar la segunda pantalla y con el dispositivo conectado, se pasa a observar la impedancia, que como se menciona anteriormente, se debe presionar alguno de los botones del montaje para activar la recepción de los valores de impedancia. La impedancia permite evaluar el adecuado montaje de los electrodos, ya que, a menor impedancia, menor será la resistencia hecha por el cuero cabelludo al paso de la señal para ser recibida por el electrodo. Se recomienda que la impedancia sea menor a 30kohms y de debe acomodar el electrodo, agregar mayor cantidad de pasta o gel conductores para disminuir la impedancia.
5. Antes de continuar se debe detener la medición de la impedancia para que la adquisición de los datos no entre en conflicto con los procesos siguientes.
6. Permite volver a la vista anterior Vista 2.
7. Permite continuar a la siguiente Vista 4.
8. Es el botón de ayuda, contiene el siguiente mensaje como instrucción de uso de la vista: Comience identificando que el dispositivo está conectado y que puede comenzar a adquirir los datos, se abrirá una ventana negra la cual le anuncia que se ha comenzado la adquisición, presione 'Minimizar' y a continuación verifique que la pantalla de estimulación se encuentra conectada. Para verificar la impedancia presione cualquiera de los electrodos de la configuración. Antes de pasar a la siguiente etapa recuerde detener la medición de la impedancia.

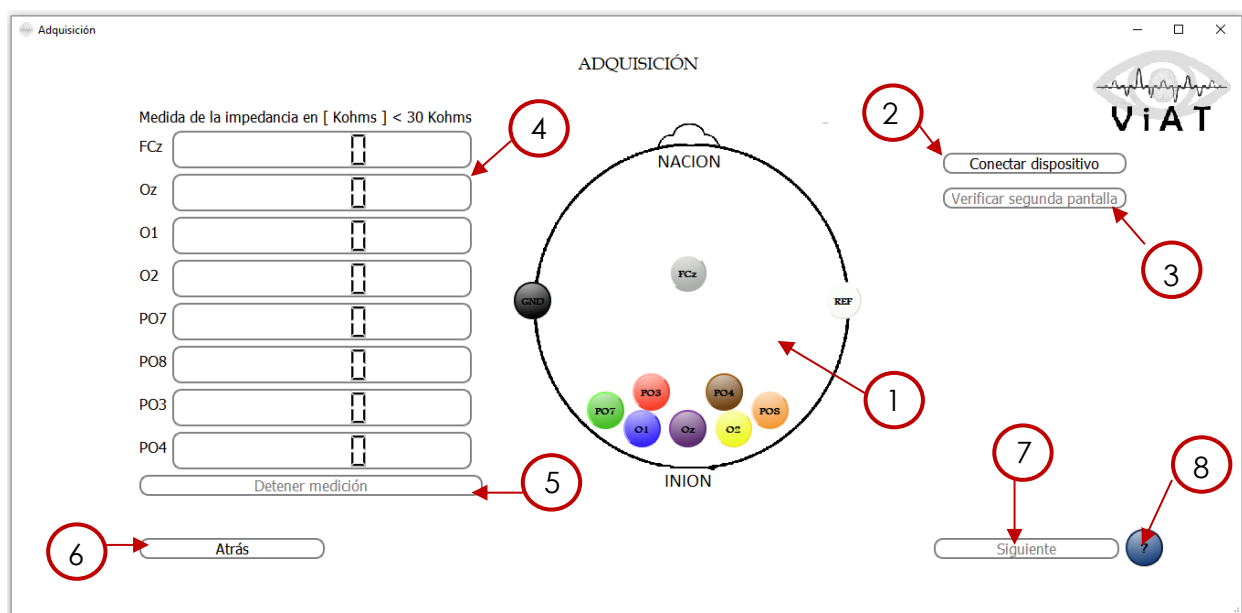


Figura 4. Adquisicion.ui

- **AcquisitionSignal (Vista 4):** Ejecuta la vista **Adquisicion_accion.ui** (Figura 5). Tiene la función de visualizar la adquisición de los datos:
 1. Botón de iniciar: Activa una línea de carga que le da tiempo al usuario de comenzar la visualización (2) en la pestaña "Visualización" antes de iniciar el estímulo.
 2. Permite iniciar la visualización, toma un tiempo aproximado de 15 segundos comenzar a visualizar las señales, este tiempo está considerado en la línea de tiempo llamada "Cargando estimulación" al presionar el botón iniciar.
 3. Reiniciar permite comenzar con la carga del estímulo, se debe parar la visualización antes de iniciar de nuevo el estímulo.
 4. El método de Welch: Se utiliza para estimar la potencia de una señal en diferentes frecuencias: es decir, que es un enfoque de estimación de la densidad espectral. En el menú aparece cada uno de los canales de la adquisición, al seleccionar un canal se le aplica el método de Welch y se observa el resultado en el color del canal seleccionado.
 5. Laplace: El objetivo de esta operación es eliminar el ruido debido al efecto de resistencia ocasionado por el cráneo, actúa como un filtro de pasa alta. Además, eliminan el efecto del electrodo de referencia. Esta grafica permite observar el comportamiento de la señal con la técnica de Laplace, como método adicional de verificación de una correcta adquisición bajo el estímulo.
 6. Se puede detener la estimulación sin desconectar el dispositivo, permitiendo realizar registros continuos durante la ejecución de la aplicación.
 7. Desconectar dispositivo, el dispositivo deja de enviar datos y se hace necesario volver a la vista anterior para iniciarlo nuevamente. Esta acción es necesaria para pasar a la base de datos o visualizar las señales adquiridas previamente.
 8. Permite volver a la vista anterior, para observar las impedancias o conectar el dispositivo.
 9. Cierra el programa.
 10. Permite ir a la vista 6 y visualizar las señales de la base de datos.
 11. Botón de ayuda contiene el siguiente mensaje: En la parte superior encontrará dos pestañas, *cargar estimulación*, en la que se encuentra actualmente y la de *visualización*, en la cual podrá observar la señal adquirida en tiempo real. Primero debe presionar el botón inicio en la pestaña actual, luego, pase a la pestaña de visualización y presione 'visualizar', en la primera pestaña podrá observar una barra de proceso que le anunciara que el estímulo está a punto de presentarse. En esta pestaña podrá interactuar con la señal aplicando el método de Welch a cada uno de los canales o modificando la configuración de Laplace según el interés del registro. Podrá reiniciar el estímulo las veces que sea necesario sin necesidad de detener la adquisición. En la parte inferior podrá observar 4 botones que le permiten ir a la vista anterior, desconectar el dispositivo de adquisición, visualizar las señales adquiridas previamente o salir del programa.



Figura 5. Control vista estímulo.

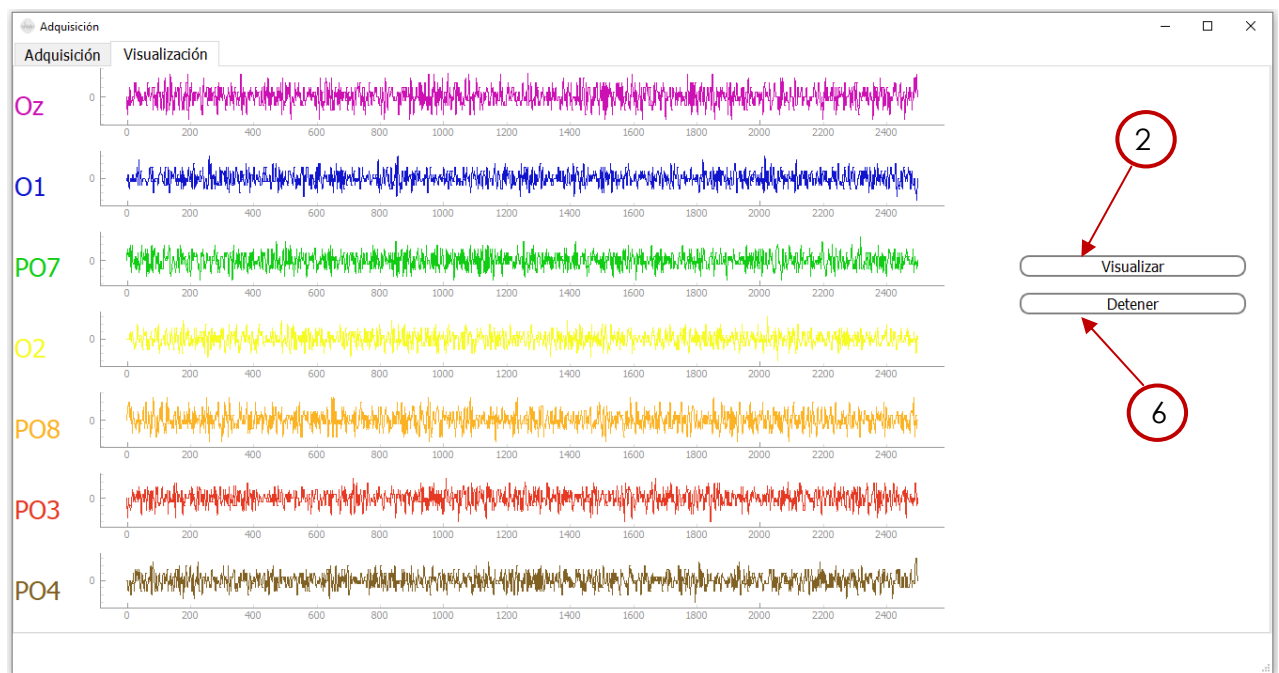


Figura 6. Visualización de señal

- **DataBase (Vista 5):** Ejecuta la vista **Buscardatos.ui** (Figura 7). Tiene la función de visualizar los sujetos registrados en la base de datos:
 1. Permite buscar un sujeto en la base de datos por medio del número de Cédula registrado.
 2. Es un Menú que permite elegir la función que tendrá al presionar doble clic sobre un sujeto en la base de datos, permitiendo eliminar un registro o direccionar a los archivos registrados localmente del sujeto seleccionado.
 3. Este botón permite mostrar todos los sujetos en la base de datos local creada en MongoDB al ejecutar el programa.
 4. Permite direccionar a la vista 6, donde se observan señales previamente registradas.
 5. Botón de ayuda contiene el siguiente mensaje: En el primer campo que encuentra podrá digitar la CC de un sujeto para buscar su información en la base de datos, o presionar 'Mostrar todo' para visualizar todos los sujetos presentes en la base de datos. Existe un menú desplegable debajo de este campo, este permite eliminar un sujeto de la base de datos o direccionar al usuario al registro del sujeto. En el botón 'Visualizar señales' se dirigirá a la vista que le permite buscar y visualizar señales previamente adquiridas. Para volver al menú anterior presione 'Atrás'.
 6. Atrás, este botón permite volver a la vista anterior, esta puede ser; Vista 1 o 2. Según el uso dado por el usuario.

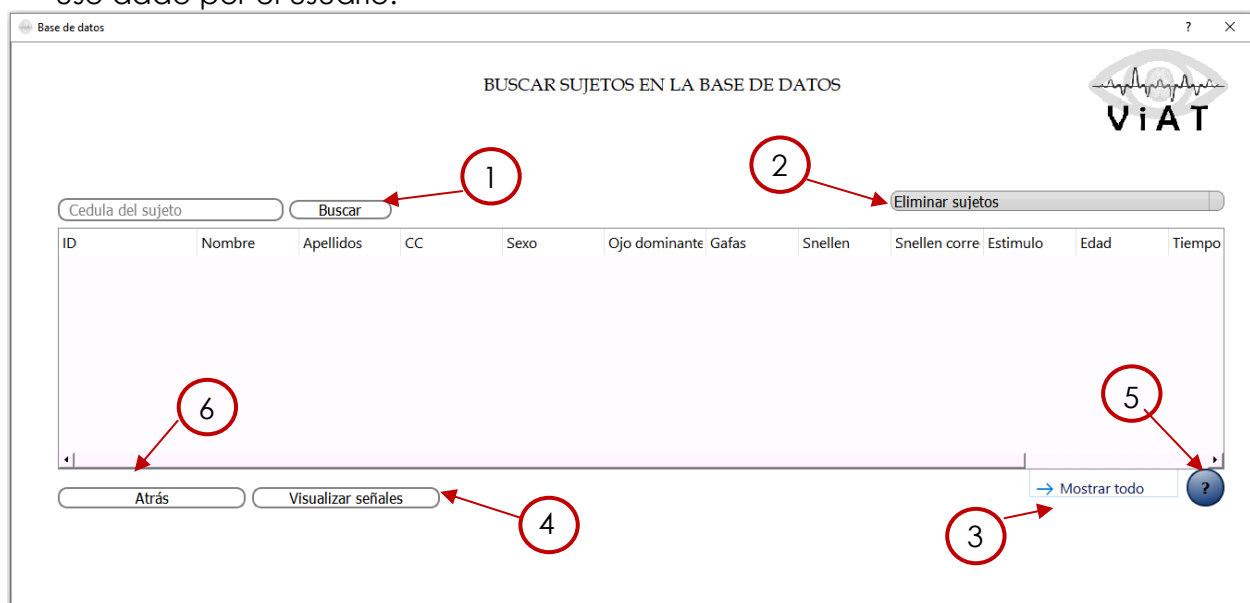
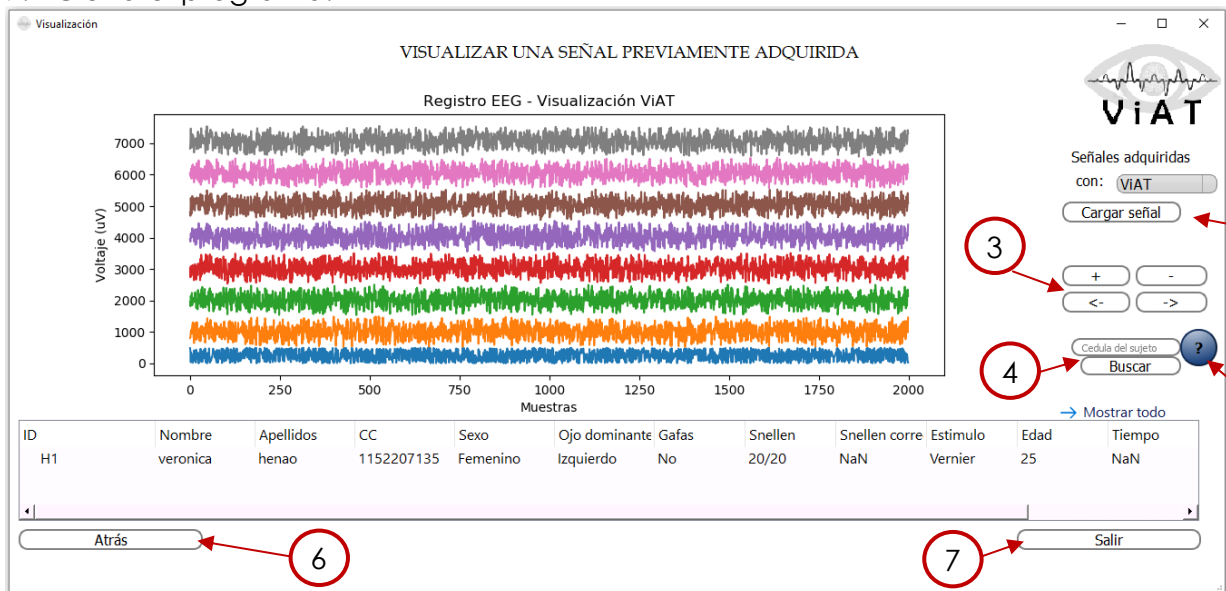


Figura 7. Base de datos

- **GraphicalInterface (Vista 6):** Ejecuta la vista **visualizacion.ui** (Figura 8). Tiene la función de visualizar las señales registradas con el software ViAT o el software OpenViBE:
1. Menú. Las señales adquiridas con OpenViBE tienen una estructura de almacenamiento diferente a la adquirida por ViAT por esta razón es necesario que el usuario especifique de que tipo es la señal antes de cargarla.
 2. Cargar señal, abre el explorador de archivos en la carpeta de la aplicación y permite buscar el registro que se quiera observar. Existe una carpeta llamada "Record" en la carpeta principal, ahí deberá encontrar todos los registros realizados por ViAT a no ser que hace cambiado la ubicación de un registro en la Vista 2.
 3. Son botones para interactuar con la señal visualizada. Permiten mover hacia la derecha o izquierda la señal y hacer un pequeño zoom de la misma.
 4. Permite buscar un sujeto en la base de datos y al aparecer en la tabla y presionar doble clic el usuario se dirigirá a la carpeta de registros del sujeto.
 5. Botón de ayuda contiene el siguiente mensaje: En esta vista podrá visualizar señales previamente registradas por ViAT o en el software OpenViBE, en el menú desplegable a la derecha elija el tipo de archivo que va a buscar y presione 'Cargar señal'. Los botones en la parte inferior de este botón le permiten desplazar la señal en el tiempo y realizar un ajuste de la señal para visualizarla mejor. En el campo siguiente podrá interactuar nuevamente con la base de datos buscando un sujeto con su CC para visualizar la señal de manera más ágil. Si desea observar todos los sujetos en la base de datos presione 'Mostrar Pacientes'. Para volver al menú anterior presione 'Atrás'.
 6. Atrás, este botón permite volver a la vista anterior, esta puede ser; Vista 4 o 5. Según el uso dado por el usuario.
 7. Cierra el programa.



ESTIMULACIÓN AGUDEZA DE VERNIER

Stimulation_Acuity.py

Debido a la versatilidad de los movimientos sacádicos, se ha desarrollado con el tiempo un número importante de tareas sicomotoras para probar distintos mecanismos visuales, entre ellos la agudeza visual. El estímulo (Figura 8) consiste en un conjunto de segmentos verticales blancos sobre fondo negro, separados una distancia determinada para una distancia de 1 metro. Existe una imagen fija que intercala con los segmentos que cambian de distancia en el tiempo. La tarea inconsciente del observador es determinar el desplazamiento máximo horizontal entre las líneas que les permite verlas alineadas.

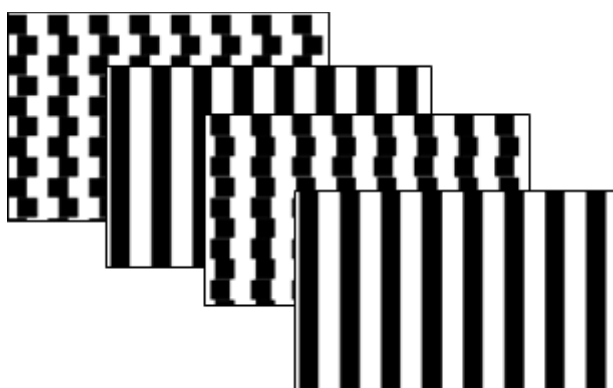


Figura 8. Estímulo de Vernier.

Stimulation_Acuity.py

Bibliotecas

pygame

Definición	permite hacer aplicaciones multimedia como juegos construidos sobre la excelente biblioteca SDL.	Instalación e importación	pip install pygame import pygame
Init	pygame.init() inicializar todos los módulos de pygame importados	Display	pygame.display.set_mode(size, flags, depth, display) Inicializar una ventana o pantalla para mostrar
Out	quit() Finalizar todos los módulos de Pygame	Image	pygame.image.load(filename) cargar nueva imagen de un archivo
Flip	pygame.display.flip()	Pixels	pygame.image.load(filename)

	Actualice la superficie de visualización completa en la pantalla		<code>.get_width()</code> // <code>.get_height()</code> Devuelve el ancho y el largo de la superficie en píxeles
Scale	<code>pygame.transform.scale(update)</code> Cambia el tamaño de la superficie a una nueva resolución.		<code>pygame.display.update()</code> Solo permite actualizar una parte de la pantalla, en lugar de toda el área. Si no se pasa ningún argumento, actualiza toda la superficie como <code>pygame.display.flip()</code> .
Blit	<code>pygame.display.set_mode Event.type e()).blit(img,(width,height))</code> generar imagen fotorrealista a partir de un modelo, copiando los píxeles que pertenecen a dicho objeto en el objeto de destino.		<code>pygame.KEYDOWN</code> and <code>event.key == pygame.K_SPACE</code> Eventos cuando se presionan y sueltan los botones del teclado. En este caso al presionar la tecla SPACE
Event	<code>pygame.event.get()</code> Esto obtendrá todos los mensajes y los eliminará de la cola. Si se proporciona un tipo o secuencia de tipos, solo esos mensajes se eliminarán de la cola.		
Time			
Definición	Permite manejar tareas relacionadas con el tiempo.	Importación	<code>import time</code>
Time	<code>time.time()</code> Devuelve el número de segundos transcurridos desde la época.	Sleep	<code>time.sleep(secs)</code> La cantidad de segundos que el programa Python debería pausar la ejecución. Este argumento debe ser un <code>int</code> o un <code>float</code> .

Funciones

1. Init

Recibe: ID del sujeto registrado en la base de datos, Cédula del sujeto registrado en la base de datos, ubicación del registro.

Función: Ver: función: `start_stimulus` para más detalles.

StreamInfo: el objeto StreamInfo almacena la declaración de un dato flujo, para describir sus propiedades y luego construir un

StreamOutlet con él para crear la transmisión en la red.

StreamOutlet: los puntos de venta se utilizan para hacer streaming de datos (y los metadatos) disponibles en la red de laboratorio.

2. Display

Recibe: Número de la imagen a presentar.

Función: Ajusta el espacio en la pantalla y recibe la imagen diseñada con el grosor o característica de cada nivel de agudeza visual.

0: corresponde a la imagen intermedia entre cada visual estimulación y cambio en el nivel de agudeza visual.

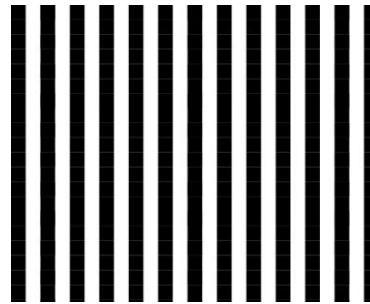


Figura 9. Imagen base. Separación 0.

0.1: corresponde a la imagen intermedia entre cada cambio de estado (binocular o monocular)

3. save

Recibe: Número de la Marca al cambiar imagen.

Función: Le permite guardar la marca de tiempo al cambiar las imágenes. Para este estímulo particular, permite separar las agudezas en 7 niveles

4. start_stimulus

Recibe: Número de la Marca al cambiar imagen.

Función: "Iniciar la estimulación Vernier.

num: niveles transversales del rango de agudeza visual (1,7) para 6 niveles de agudeza visual

marca de tiempo: Devuelve la marca de tiempo POSIX como flotante

push_sample: empuje una muestra en la salida.

Cada entrada en la lista corresponde a un canal.

Argumentos de palabras clave:

x: una lista de valores para insertar (uno por canal).

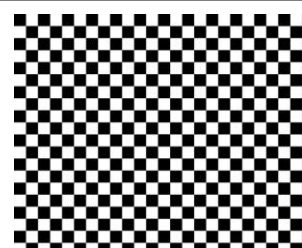
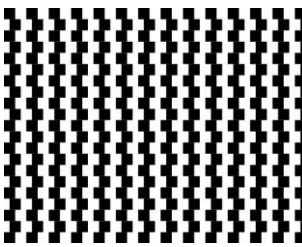
marca de tiempo - Opcionalmente el tiempo de captura de la muestra, de acuerdo con `local_clock ()`; si se omite, se usa la hora actual. (por defecto 0.0)
`event.get ()`: Pygame registrará todos los eventos del usuario en una cola de eventos

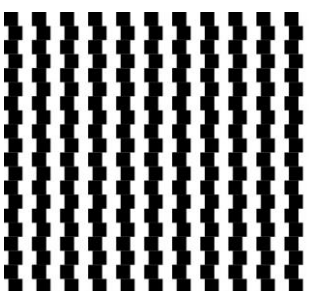
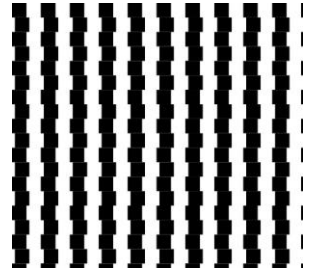
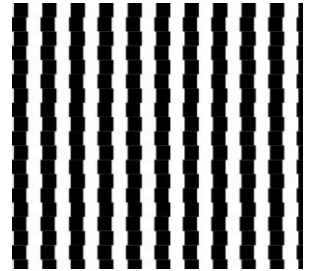
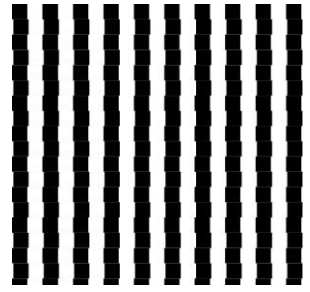
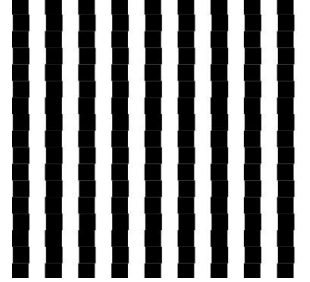
imágenes presentadas

Cada imagen corresponde a un valor de agudeza en la estala de Snellen y se intercambian de la 1 a la 7. Se utilizo la Figura 10 como referente para estimar la separación de los cuadros en la estimulación.

Decimal	Fracción	Snellen (6 m)	Snellen (20 pies)	logMAR
0,10	1/10	6/60	20/200	1,0
0,12	1/8	6/48	20/160	0,9
0,16	4/25	6/37,5	20/125	0,8
0,20	1/5	6/30	20/100	0,7
0,25	1/4	6/24	20/80	0,6
0,32	1/3	6/19	20/63	0,5
0,40	2/5	6/15	20/50	0,4
0,50	1/2	6/12	20/40	0,3
0,63	2/3,2	6/9,5	20/32	0,2
0,80	4/5	6/7,5	20/25	0,1
1,00	1/1	6/6	20/20	0,0
1,25	5/4	6/4,8	20/16	-0,1

Figura 10. Conversión de valores de agudeza visual.

Número	Separación según Snellen [mm]	Agudeza Snellen	Imagen mostrada
1	5.817764173	>20/200	
2	2.908882087	20/200	

Número	Separación según Snellen [mm]	Agudeza Snellen	Imagen mostrada
3	1.818051304	20/125	
4	1.163552835	20/80	
5	0.727220522	20/50	
6	0.461727315	20/30	
7	0.290888209	>20/20	

PROCESAMIENTO CON MULTITAPERING

dataprocessing.py

Estimación espectral con multitapering

Funciones

1. Init

Recibe: ID del sujeto registrado en la base de datos, Cédula del sujeto registrado en la base de datos, fecha del registro, ubicación del registro y ubicación para guardar.

Función: Establece las variables con los elementos recibidos.

2. run

Recibe:

Función: Crea el directorio para guardar los resultados del procesamiento, lee los datos del registro y de las marcas, compara ambos archivos y separa la señal por cada marca, luego toma cada grupo de marca y aplica el procesamiento de Multitaper así:

```
Sk_complex, weights, _ = pmtm(data[i], NW=2, k=4, show=False)
```

La función *pmtm* es de la biblioteca *spectrum* y contiene las siguientes características **def pmtm(x, NW=None, k=None, NFFT=None, e=None, v=None, method='adapt', show=False)**:

: param array x: los datos

: param float NW: el parámetro de ancho de banda medio del tiempo (los valores típicos son

2.5,3,3.5,4). Debe proporcionarse de lo contrario las ventanas cónicas y se deben proporcionar valores propios (salidas de dpss)

: param int k: utiliza las primeras k secuencias de Slepian. Si no se proporciona * k *,

* k * se establece en * NW * 2 *.

Por lo general, en la estimación espectral, la media para reducir el sesgo es usar la reducción gradual de ventana. Para reducir la varianza, necesitamos promediar diferentes espectros. El problema es que solo tenemos un conjunto de datos. Por lo tanto, necesitamos descomponer un conjunto en varios segmentos. Tales métodos son bien conocido: el periodograma de daniell, método de Welch, etc. El inconveniente de tales métodos es una pérdida de resolución ya que los segmentos utilizados para calcular el espectro son más pequeño que el conjunto de datos. El interés del método multitaper es **mantener una buena resolución** mientras se da la reducción de sesgo y varianza.

¿Como funciona? Primero se calcula un periodograma simple diferente con el conjunto de datos completo (para mantener una buena resolución) pero se

calcula cada periodograma con ventanas diferentes, luego, promediamos todo este espectro para evitar la redundancia y el sesgo debido a los conos mtm.

PROCESAMIENTO CON TIEMPO FRECUENCIA.

plot_stft.py

Este procesamiento se da por medio del cálculo la transformada de Fourier de corto tiempo.

Funciones

1. Init

Recibe: ID del sujeto registrado en la base de datos, Cédula del sujeto registrado en la base de datos, fecha del registro, ubicación del registro y ubicación para guardar.

Función: Definir variables y ubicaciones de los archivos entregados.

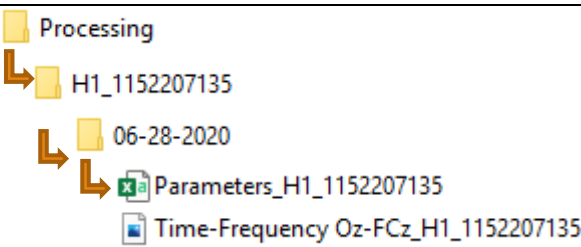
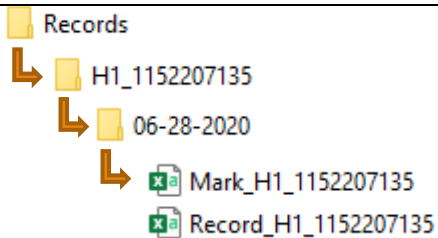
2. plot_stft

Recibe:

Función: Lee el registro en el canal Oz-FCz y aplica transformada de Fourier de corto tiempo *stft* de la biblioteca *scipy.signal*

Las STFT se pueden usar como una forma de cuantificar el cambio de la frecuencia de una señal no estacionaria y el contenido de fase en el tiempo.

LOS RESULTADOS ENTREGADOS POR LA HERRAMIENTA, INCLUYENDO LOS GENERADOS EN: STIMULATION_ACUITY.PY, DATAPROCESSING.PY Y PLOT_STFT.PY

Estructura	Visual
PROCESSING ID_CC FECHA <i>Parameters_ID_CC.csv</i> <i>Time-Frequency Oz-FCz_ID_CC.jpg</i>	
RECORDS ID_CC FECHA <i>Mark_ID_CC.csv</i> <i>Record_ID_CC.csv</i>	

Se elijen los archivos .csv por su facilidad de manipulación.

La estructura de los archivos es la siguiente:

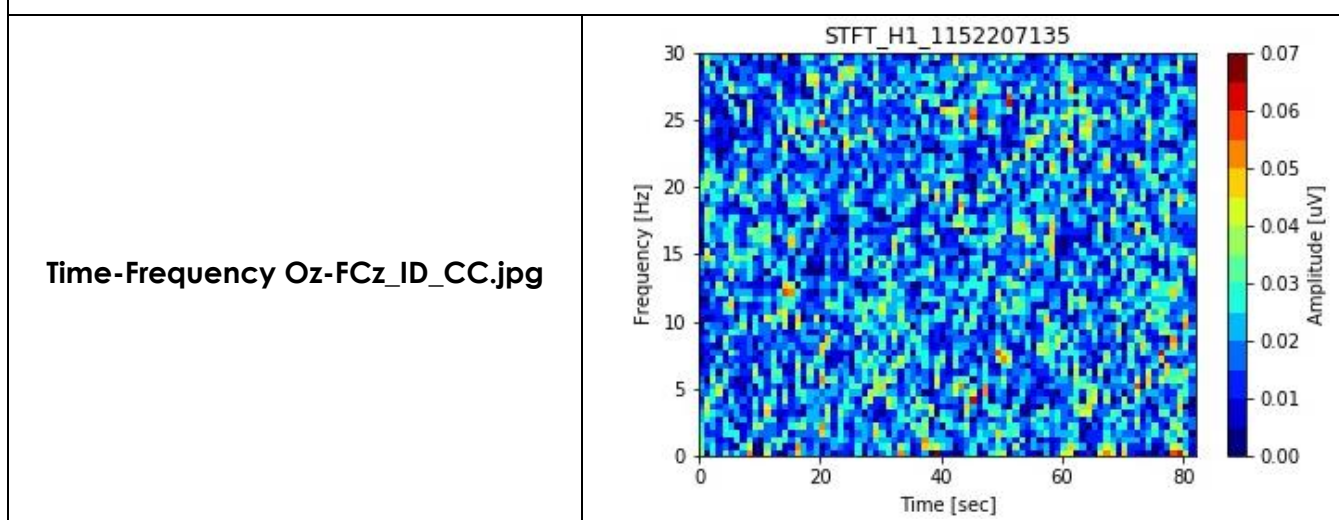
Processing

Parameters_ID_CC.csv	Max	Fre
	0.59804803	7.08699902
	0.10245691	8.30889541
	0.40024617	8.30889541
	0.25079922	7.08699902
	0.07403019	8.30889541
	0.31702204	8.30889541

Max: Hace referencia a la amplitud máxima encontrada en el rango de frecuencia de interés, en este caso entre 7 y 8.5 Hz

Fre: Hace referencia a la frecuencia exacta a la que se encontró el valor máximo.

Estos datos son útiles y necesarios para realizar evaluaciones posteriores de la señal con apoyo de Machine Learning [46] u otras técnicas de análisis estadístico.



La imagen permite observar a lo largo de la señal diferencial Oz-FCz la frecuencia en el tiempo y con esto validar la eficiencia de la estimulación realizada a una frecuencia determinada.

Records

Record_ID_CC.csv	C1	C2	C3	C4	C5	C6	C7	C8	H
	0.73187554	0.1191259	-0.4066793	-0.2687161	-0.393944	-0.29221848	-0.30618772	-0.00962621	11-22-25
	0.52693057	0.33831501	0.45620424	0.24923992	0.23631233	0.08804327	0.11061227	-0.41921276	11-22-26
	0.92276698	-0.13704956	-0.58744523	-0.76322785	-0.19375223	-0.41278476	-0.42801601	-0.88206647	0
	0.30380246	-0.25563728	0.44580564	-0.19913614	0.23976383	-0.03268635	-0.22412168	-0.19581272	0
	0.30664465	0.64603671	0.38948187	-0.15265588	-0.137475	0.64516971	-0.04174143	0.37839356	0
	0.60575074	-0.12673205	0.18656027	-0.4891784	0.23382193	-0.36466084	0.34857655	0.21318001	0

Contiene todos los registros de los sujetos y sus respectivas marcas al realizar la estimulación.

C1: Representa el canal 1 es decir FCz, canal de referencia.

C2: Representa la diferencia entre el canal 1 y el canal 2 es decir Oz-FCz

C3: Representa la diferencia entre el canal 1 y el canal 3 es decir O1-FCz
 C4: Representa la diferencia entre el canal 1 y el canal 4 es decir PO7-FCz
 C5: Representa la diferencia entre el canal 1 y el canal 5 es decir O2-FCz
 C6: Representa la diferencia entre el canal 1 y el canal 6 es decir PO8-FCz
 C7: Representa la diferencia entre el canal 1 y el canal 7 es decir PO3-FCz
 C8: Representa la diferencia entre el canal 1 y el canal 8 es decir PO4-FCz
 H: La hora en la que se adquirio el segmento de datos. El formato es:

Hora-Minutos-Segundos

Los datos H = 0, quiere decir que hacen parte del mismo chunk de datos adquiridos durante la ultima hora guardada.

Mark_ID_CC.csv

H	M
11-23-09	1
11-23-14	2
11-23-18	3
11-23-23	4
11-23-27	5
11-23-31	6
11-23-36	7
11-23-44	8

H: Representa la hora en la que se hizo el cambio de imagen del estimulo es decir, al llegar la marca, el formato es:

Hora-Minutos-Segundos

M: Representa el numero de marca que llego en ese momento iniciando en **1**, para el estimulo presentado Vernier se evaluando **7** agudezas, una marca por cada una y una **8va** marca al finalizar el registro.

SERVIDOR

Server.py

Un servidor es una aplicación en ejecución capaz de atender las peticiones de un cliente y devolverle una respuesta en concordancia.

Unas bibliotecas útiles para crear este tipo de servidor compatible con OpenBCI son pyOpenBCI y pylsl.

pyOpenBCI proporciona un controlador Python estable para todos los biosensores OpenBCI.

Pylsl proporciona un conjunto de funciones para crear datos transmitidos en tiempo real dentro de una red.

Inicialmente se deben instalar e importar las bibliotecas necesarias:


```
from pyOpenBCI import OpenBCICyton
from pylsl import StreamInfo, StreamOutlet
import numpy as np
from serial.tools import list_ports
from datetime import datetime
```

Para iniciar la comunicación se escribe:

```
board = OpenBCICyton(port='COM3', daisy=False)
```

El valor de “port” se puede encontrar en -Panel de control-Administrador de dispositivos -Otros dispositivos- o usando `serial.tools.list_ports.comports()`

El dispositivo OpenBCI contiene 8 salidas para los canales y 3 salidas auxiliares. Para los canales se recomienda convertir los canales_datos a uVolts multiplicando por **(4500000) / 24 / (2 ** 23 - 1)** actualmente se multiplica únicamente por **(4500000)** para amplificar los datos y para los auxiliares se recomienda multiplicar por **0.002 / (2**4)**

El objeto StreamInfo almacena la declaración de una secuencia de datos.

```
StreamInfo('OpenBCIEEG', 'EEG', 8, 250, 'float32', 'OpenBCItestEEG')
```

En este se incluye la siguiente información: Nombre, Tipo, numero de canales, frecuencia de muestreo formato o tipo de dato, identificador del dispositivo (ID)

Primero creará un StreamInfo para describir sus propiedades y luego se construye un StreamOutlet con él para crear la transmisión en la red.

El objeto StreamOutlet se utilizan para hacer que los datos de transmisión (y los metadatos) estén disponibles en la red del laboratorio.

```
outlet_eeg = StreamOutlet(info_eeg)
```

Requiere del StreamInfo para tomar la información, opcional se puede incluir tamaño del Chunk (muestras en la transmisión), y cantidad máxima de datos (buffered) para almacenar. Predeterminado 6 min.

`streams = resolve_stream('type', 'Markers')` devuelve todas las transmisiones disponibles desde cualquier salida en la red en tiempo real. Y al utilizar el objeto StreamInlet para recibir datos de transmisión (y metadatos) desde la red del laboratorio, toma las transmisiones disponibles y finalmente se hace un `pull_chunk()` que extrae un trozo de muestras de la entrada.

Comenzar a manejar la transmisión de datos, haciendo pull de las marcas, push de los datos y evaluando continuamente la existencia de las marcas.

Pull: Recibe los datos de la red

Push: Envía los datos a la red

```
def lsl_streamers(sample):
    sample_mark, timestamp = inlet.pull_sample(timeout=0.0)
    outlet_eeg.push_sample(np.array(sample.channels_data)*SCALE_FACTOR_EEG)
    outlet_aux.push_sample(np.array(sample.aux_data)*SCALE_FACTOR_AUX)
    if sample_mark is not None:
```

```
print(sample_mark)
```

Para llamar la transmisión se ejecuta:

```
board = OpenBCICyton(port='COM3', daisy=False)
board.start_stream(lsl_streamers)
```

Bibliotecas

OpenBCI: Significa interfaz de cerebro-computadora de código abierto (BCI). Es una herramienta que proporciona a cualquier persona con una computadora, las herramientas necesarias para tomar muestras de la actividad eléctrica de su cuerpo.

Primero, asegúrese de tener las dependencias necesarias.

```
pip install numpy pyserial bitstring xmltodict
```

pyOpenBCI

Definición	Proporcionar un controlador Python estable para todos los biosensores OpenBCI	Instalación e importación	pip install pyOpenBCI from pyOpenBCI import OpenBCICyton
Conexión	OpenBCICyton(port, daisy=False) Maneja la conexión a una placa OpenBCI Cyton. La clase OpenBCICyton interactúa con el Cyton Dongle y la placa Cyton para analizar los datos recibidos y enviarlos a Python como un objeto OpenBCISample.		

pysl

Definición	Proporciona un conjunto de funciones para crear datos de instrumentos accesible en tiempo real dentro de una red. A partir de ahí, las corrientes pueden ser recogido por programas de grabación, programas de visualización o experimentos personalizados para aplicaciones que acceden a flujos de datos en tiempo real.	Instalación e importación	pip install pylsl from pylsl import StreamInfo, StreamOutlet from pylsl import StreamInlet, resolve_stream
------------	--	---------------------------	--

Resolver todas las transmisiones en la red	<code>resolve_stream('type', 'Markers')</code> Esta función devuelve todas las transmisiones disponibles actualmente desde cualquier salida en la red.	<code>StreamInlet</code>	<code>StreamInlet(streams[0])</code> Las entradas se utilizan para recibir datos de transmisión (y metadatos) desde la red del laboratorio.
Pull	<code>StreamInlet().pull_chunk()</code> Saque un pedazo de muestras de la entrada	Iniciar flujo	<code>OpenBCICyton.start_stream(self, callback)</code> Comience a manejar la transmisión de datos desde el tablero. Llame a una devolución de llamada proporcionada por cada muestra procesada.

Funciones

1. Init

Recibe:

Función: Define las variables para *StreamInfo* y para *StreamOutlet*

2. IsStreamers

Recibe: Las muestras desde el dispositivo (Samples)

Función: Realiza la conversión de los datos sugerida por el OpenBCI y realiza un push de los datos. Saca un pedazo de muestra y empuja una muestra por canal a la salida.

3. Port

Recibe: Las muestras desde el dispositivo (Samples)

Función: Evalúa la lista de puertos en uso y establece el puerto a utilizar.

SIMULACIÓN

Permite crear una nueva transmisión de información y establecer un número de serie para el dispositivo o identificador único local para la transmisión (también se podría omitir, pero las conexiones interrumpidas no se recuperarían automáticamente).

Funciones

1. Init

Recibe:

Función: Define las variables para *StreamInfo* y para *StreamOutlet*

2. sample

Recibe:



Función: Crea un arreglo de datos aleatorios en un ciclo continuo y genera un *push_sample*, empujando una muestra por canal a la salida.