**MEDUSA-ANALYZER: Documentación de los botones**

***1. MAINWINDOW (main\_window.ui)***

La *MainWindow* (*QMainWindow*) es el contenedor principal de la interfaz gráfica de MEDUSA-ANALYZER. En ella se aloja un *QStackedWidget*, donde se cargan dinámicamente los distintos widgets (pantallas) que representan cada etapa del análisis: Preprocessing, Segmentation y Signal Analysis.

A parte de este *stackedWidget*, hay un *centralwidget* (*QWidget*) que contiene un *horizontalLayout* (*QHBoxLayout*) con los siguientes botones.

***nextButton***

* **Función:** Permite avanzar a la siguiente pantalla del flujo de trabajo.
* **Lógica:** Cambia la página activa del *QStackedWidget* al siguiente widget en la secuencia. Se actualiza también el *progressLabel.*
* **Ejemplo de flujo:** Preprocessing → Segmentation → Signal Analysis.

***backButton***

* **Función:** Retrocede a la pantalla anterior en el flujo.
* **Lógica:** Similar al *nextButton* pero en sentido inverso, disminuye el índice del QStackedWidget.

Por último, en la *QMainWindow (fuera del QStackedWidget y del QWidget)* está el siguiente botón:

***progressLabel***

* **Función:** Indica en qué paso del flujo de trabajo se encuentra el usuario.
* **Lógica:** Muestra texto como "Paso 1 de 3: Preprocesamiento (33%)", que se actualiza automáticamente con el avance o retroceso entre pantallas.

***2. PREPROCESSING (preprocessing.ui)***

Pantalla encargada de definir y aplicar las opciones de preprocesamiento sobre los archivos de señal EEG. Se trata de un *Form* (*QWidget*) compuesto por dos *QGroupBox* llamados *dataloading* y *datapreprocessing* respectivamente.

***GROUPBOX: DATALOADING***

A su vez, el *groupbox* de *dataloading* está compuesto por un QHBoxLayout (horizontalLayout) que alberga dos botones (*QPushButton*) y una label (*QLabel*).

***browseButton***

* **Función:** Abre un diálogo para seleccionar múltiples archivos desde uno o varios directorios. IMPORTANTE ESTO. Se tienen que poder seleccionar archivos de varias carpetas.
* **Lógica:** Permite múltiples selecciones. La información se guarda para ser preprocesada más adelante.

***deletefilesButton***

* **Función:** Permite eliminar alguno de los archivos seleccionados.
* **Lógica:** Se actualiza la información y el contenido de *selectLabel*.

***selectLabel***

* **Función:** Muestra el número total de archivos seleccionados.
* **Ejemplo:** "3 archivos seleccionados".

***GROUPBOX: DATAPREPROCESSING***

Por otro lado, el *GruopBox* *datapreprocessing* contiene seis *QHBoxLayouts*.

**LAYOUT: *preprocessing\_selection***

Primero está el *preprocessing\_selection*, que contiene dos botones (*QRadioButton*):

***yesRButton / noRButton***

* **Función:** Botones tipo radio (mutuamente excluyentes) para indicar si se quiere aplicar el preprocesamiento.
* **Síntesis:**
  + *yesRButton*: Habilita el resto de las opciones de preprocesamiento.
  + *noRButton*: Desactiva los controles de preprocesamiento, pasando directamente a la siguiente fase.

**LAYOUT: *resample***

Luego está el *resample* con los siguientes botones (*QCheckBox* y *QSpinBox*):

***resampleCBox***

* **Función:** *Checkbox* para activar/desactivar la función de re-muestreo.
* **Cuando está activo, se habilitan:**
  + *resamplefsBox*: selección de la nueva frecuencia de muestreo.

**LAYOUT: *notch\_filtering***

A continuación, el *notch\_filtering* con varios botones (*QCheckBox* y *QSpinBox*):

***notchCBox***

* **Función:** Checkbox para activar/desactivar el filtro notch.
* **Cuando está activo, se habilitan:**
  + cfnotchBox: selección frecuencia de corte central.
  + bwnotchBox: selección ancho de banda.
  + ordernotchBox: selección orden del filtro.

**LAYOUT: *bandpass\_filtering***

Después, el *bandpass\_filtering* con varios botones (*QCheckBox* y *QSpinBox*):

**bpCBox (bandpass)**

* **Función:** Checkbox para activar un filtro paso banda.
* **Controles habilitados al seleccionarlo:**
  + *minfreqbpBox*: selección frecuencia mínima de la banda de paso.
  + *maxfreqbpBox*: selección frecuencia máxima de la banda de paso
  + *orderbpBox*: selección orden del filtro.

**LAYOUT: *CAR\_filtering***

Luego, el *CAR\_filtering* con un botón (*QCheckBox*):

***carCBox (Common Average Reference)***

* **Función:** Checkbox para aplicar la referencia promedio común (CAR) al conjunto de datos.

**LAYOUT: *thresholding\_filtering***

*Por último****,*** el *thresholding\_filtering* con varios botones (*QCheckBox*, *QSpinBox y QToolButton*):

***thresCBox (Thresholding)***

* **Función:** Checkbox que habilita un umbral estadístico para el descarte automático de épocas ruidosas en la siguiente fase (Segmentation).
* **Controles asociados:**
  + *threskBox*: Valor de 'k', múltiplo de la desviación estándar usado como umbral.
  + *thressampBox*: Número mínimo de muestras consecutivas que deben superar el umbral.
  + *threschanBox*: Número mínimo de canales afectados simultáneamente.
* ***threshelpButton*:** *QToolButton* de ayuda que explica el significado de los parámetros de thresholding (puede mostrar una ventana emergente o *tooltip*).

3.  **SEGMENTATION (segmentation.ui)**

La sección **Segmentation** permite dividir las señales preprocesadas en épocas o segmentos, en función de condiciones experimentales o eventos anotados o simplemente con una longitud fija definida por el usuario.

La interfaz es un *Form* (*QWidget*) que está organizada en ***dos QGroupBoxes*** llamados *signal\_markers* y *segmentation\_properties* respectivamente.

***GROUPBOX: SIGNAL\_MARKERS***

El *groupbox* de *signal\_markers* se divide en dos *QHBoxLayout* llamados *available\_conditions* y *available*\_*events*, respectivamente.

**LAYOUT: Available Conditions**

***conditionLabel***

* **Función:** Etiqueta informativa, por ejemplo: "Available Conditions:"
* **Contenido:** Se actualiza dinámicamente al cargar los datos para mostrar las condiciones detectadas.

***conditionList***

* **Función:** Lista desplegable que contiene los nombres de las condiciones experimentales detectadas.
* **Ejemplos:** "eyes\_open", "eyes\_closed".

**LAYOUT: Available Events**

***eventLabel***

* **Función:** Etiqueta informativa, por ejemplo: "Available Events:"

***eventList***

* **Función:** Lista de eventos detectados en los archivos.
* **Ejemplos:** "blink", "step", "movement".

***GROUPBOX: SEGMENTATION\_PROPERTIES***

Por otro lado, el *groupbox* *segmention\_properties* se divide también en dos *QHBoxLayout* llamados *segmentation\_selection* y *trials*, respectivamente. El *layout* de *segmentation\_selection* contiene los siguientes botones de tipo *QRadioButton*.

**LAYOUT: Segmentation selection**

***conditionRButton***

* **Función:** Permite al usuario activar o desactivar la segmentación **por condiciones**.
* **Comportamiento:** Si se selecciona, se habilita el control de longitud del segmento (trialBox).

***eventRButton***

* **Función:** Activa la segmentación **por eventos**.
* **Comportamiento:** Si se selecciona, se activan los controles para definir una ventana temporal relativa al evento.

**NOTA: No es posible la combinación de segmentaciones, es decir, no se pueden marcar estos dos botones a la vez. Condiciones y eventos son MUTUAMENTE EXCLUYENTES.**

**LAYOUT: Trials**

Finalmente, en el *layout* de *trials* tenemos los siguientes botones de tipo *QSpinBox*. Este groupbox permiten seleccionar qué tipo de segmentación se quiere aplicar, y sus controles se activan dinámicamente según lo que elija el usuario.

***trialLabel***

* **Texto mostrado:** "Segment Length (ms):"
* **Función:** Etiqueta que acompaña a la *spinbox* para elegir la longitud del segmento que se tomará a partir del *onset* de cada condición seleccionada.

***trialBox***

* **Tipo:** QSpinBox
* **Función:** Permite definir la duración del segmento en milisegundos.
* **Ejemplo:** Si se elige 2000, se generará una época de 2 segundos a partir del inicio de la condición.

***winLabel\_1 y winBox\_1***

* **Texto mostrado:** "Window start (ms):"
* **Función:** Permite definir el punto inicial de la ventana de segmentación respecto al evento.
* **Ejemplo:** -100 ms → 100 milisegundos antes del evento.

***winLabel\_2 y winBox\_2***

* **Texto mostrado:** "Window end (ms):"
* **Función:** Define el punto final de la ventana respecto al evento.
* **Ejemplo:** 500 ms → 500 milisegundos después del evento.

**NOTA: Caso sin condiciones ni eventos.** En esta situación, aparece la **condición por defecto ‘null’** y se habilita el modo de segmentación por **longitud fija**.

* Se activa únicamente el *trialBox*, que el usuario usa para definir el tamaño de cada segmento a generar a lo largo de toda la señal, dividiéndola en bloques consecutivos de esa duración.
* **Ejemplo:** Señal de 10 minutos → con una segmentación de 2000 ms → se generan automáticamente segmentos de 2 segundos consecutivos hasta agotar la señal.

4.  **PARAMETERS (parameters.ui)**

**Widget principal:** Form (*QWidget*) con un *QToolBox* con cuatro páginas: *bandSegentationPage*, *signalMetricsPage*, *connectivityMetricsPage*, *graphMetricsPage*. Aquí se van a calcular diversas métricas sobre la señal.

**Página: bandSegmentationPage**

***GROUPBOX: BANDS***

Esta página está compuesta por un *QGroupBox* llamado *Bands* que alberga tres *Layouts* horizontales (*QHBoxLayout*) llamados *broadband\_definition*, *band\_segmentation\_selection* y *band\_edition*.

**LAYOUT: *broadband\_definition***

Aparece una label ‘Broadband’ con dos botones QDoubleSpinButtons para indicar el rango de frecuencias de la broadband.

***minbroadBox***

* **Función:** Botón para modificar el valor mínimo del rango de frecuencias de la broadband.

***maxbroadBox***

* **Función:** Botón para modificar el valor máximo del rango de frecuencias de la broadband.

***Lógica:*** por defecto, en estos dos botones siempre van a aparecer unos valores. En el caso de que el usuario haya marcado el botón de *bpCBox*, es decir, haya indicado que va a aplicar un filtro paso banda, se tomarán por defecto los valores de la banda de paso del filtro paso banda (*minfreqbpBox*, *minfreqbpBox*). En caso contrario, se tomarán por defecto los valores de 1 a fs/2 en *minbroadBox* y *maxbroadBox* respectivamente, donde fs es la frecuencia de muestreo.

**LAYOUT: *band\_segmentation\_selection***

Contiene un botón de tipo *QCheckBox*:

***bandCBox***

* **Función:** Checkbox para activar la opción de selección por bandas de frecuencia definidas por el usuario.
* **Lógica:**en el caso de que el usuario pulse, se desplegará el contenido del *band\_edition* layout. Sino no se despliega (o no se hace visible).

**LAYOUT: *band\_edition***

Este *layout* aparece en el caso de que el usuario si que desee segmentar los datos por bandas. Contiene los siguientes elementos:

***bandButton***

* **Función:** Al pinchar sobre el (aparece como texto ‘edit bands’ abre la ventana ***bands\_table.ui*** para permitir al usuario definir nuevas bandas y seleccionar cuáles aplicar.

***bandLabel***

* **Función:** Muestra las bandas de frecuencia seleccionadas por el usuario desde el widget bands\_table.
* **Valor inicial:** “None”.

***bands\_table.ui***

Se trata de un *QWidget* compuesto por un *QTableWidget* llamado *bandsTable* y dos *layouts* horizontales con varios botones. Esta ventana se abre cuando el usuario pincha sobre ‘*edit bands’*, es decir, sobre el botón *bandButton*. Permite al usuario definir bandas de frecuencia personalizadas.

La tabla se compone de cinco columnas: ‘Enabled’, ‘Name’, ‘Min. Freq.’, ‘Max. Freq.’ y ‘Remove’. En la primera columna habrá botones para seleccionar o deseleccionar esa banda. En ‘name’ se indicará el nombre de cada manda. Las dos columnas siguientes son para indicar la frecuencia mínima y máxima de esa banda, respectivamente. Por último, en ‘remove’ habrá un símbolo de una papelera en cada una de las filas de forma que al pulsar se elimine toda esa fila.

Por otro lado, en los layouts hay los siguientes botones:

***addButton***

* **Función:** Añade una nueva fila en la tabla para definir una banda.
* **Contenido de la fila:** *checkbox* para seleccionar/deseleccionar

***resetButton***

* **Función:** Elimina todas las filas actuales de la tabla, reseteando la definición de bandas.

***acceptButton***

* **Función:** Guarda las bandas definidas y actualiza el *bandLabel* con los nombres de las bandas seleccionadas.
* **Lógica adicional:** Cierra la ventana *bands\_table* y pasa la información al widget de Preprocessing.

***Nota:*** En la primera fila de la tabla va a salir siempre la ***broadband*** con el rango de frecuencia correspondiente (o el de la banda de paso, o el de fs/2). Este rango se puede modificar manualmente por el usuario; **el nombre de la banda NO se puede cambiar. Esta fila NO se puede eliminar**, pero sí que se puede seleccionar/deseleccionar. A parte, saldrán **otras cinco filas por defecto** con las siguientes bandas de frecuencia: de**lta (1-4 Hz), theta (4-8 Hz), Alpha (8-13 Hz), beta (13-30 Hz) y gamma (30 – 70 Hz).** Estas también se pueden editar, tanto el nombre como el rango de frecuencias, seleccionar/deseleccionar y eliminar. TOTAL: 6 filas por defecto.

**Página: signalMetricsPage**

Esta página contiene tres secciones organizadas como *QGroupBox* llamados: *basic\_statistical\_metrics*, *spectral\_metrics*, *nonlinear\_metrics*.

***GROUPBOX: BASIC\_STATISTICAL\_METRICS***

Contiene los siguientes *QCheckBox*, cada uno habilita el cálculo de una métrica:

| **Nombre del objeto** | **Tipo** | **Descripción** |
| --- | --- | --- |
| *meanCBox* | QCheckBox | Calcula la **media** de la señal. |
| *medianCBox* | QCheckBox | Calcula la **mediana**. |
| *varianceCBox* | QCheckBox | Calcula la **varianza**. |
| *kurtosisCBox* | QCheckBox | Calcula la **curtosis**. |
| *skewnessCBox* | QCheckBox | Calcula el **apuntamiento** o **asimetría** (skewness). |

***GROUPBOX: SPECTRAL\_METRICS***

Contiene los siguientes botones:

| **Nombre del objeto** | **Tipo** | **Descripción** |
| --- | --- | --- |
| *rpCBox* | QCheckBox | Activa el cálculo de la **potencia relativa** (relative power). |
| *rpLabel* | QLabel | Muestra el nombre de la banda sobre la que se calcula la potencia. Valor inicial: "None" |
| *rpButton* | QPushButton | Abre la ventana con la tabla de las bandas (**bands\_table.ui**) para seleccionar las bandas en las que se quiere calcular. |

**NOTA: Lógica especial:**

* **Al activar rpCBox**, se despliega una ventana emergente (QDialog) llamada warning\_rp que informa al usuario de que para el cálculo de la potencia relativa **se normaliza sobre la banda "*broadband*"**.

Además, en esta sección se encuentran otras métricas espectrales:

| **Nombre del objeto** | **Tipo** | **Descripción** |
| --- | --- | --- |
| *apCBox* | QCheckBox | Calcula la **potencia absoluta.** |
| *mfCBox* | QCheckBox | Calcula la **frecuencia media**. |
| *seCBox* | QCheckBox | Calcula la **entropía espectral**. |

***GROUPBOX: NONLINEAR\_METRICS***

Métricas avanzadas que requieren parámetros adicionales en algunos casos:

| **Nombre del objeto** | **Tipo** | **Descripción** |
| --- | --- | --- |
| lzcCBox | QCheckBox | Calcula la **complejidad de Lempel-Ziv**. |
| mlzcCBox | QCheckBox | Calcula la **MLZC (complejidad multiescala de Lempel-Ziv)**. |
| mlzcEdit | QLineEdit | Campo de entrada para las escalas a usar (solo visible si mlzcCBox está activo). Ejemplo de entrada: [1, 3, 5, 7, 9]. |
| ctmCBox | QCheckBox | Calcula la **CTM (Complexity of Time Series Mapping)**. |
| ctmrBox | QDoubleSpinBox | Parámetro radio necesario para calcular CTM. Solo visible si ctmCBox está activo. |
| sampenCBox | QCheckBox | Calcula la **Sample Entropy**. |
| sampenmBox, sampenrBox | QSpinBox / QDoubleSpinBox | Parámetros m y r necesarios. Solo visibles si sampenCBox está activo. |
| msampenCBox | QCheckBox | Calcula la **Multi-Scale Sample Entropy**. |
| msampenscaleBox, msampenmBox, msampenrBox | QSpinBox / QDoubleSpinBox | Escala máxima y parámetros m, r. Solo visibles si msampenCBox está activo. |

**Página: connectivityMetricsPage**

Se divide en tres QGroupBox:

***GROUPBOX: AMPLITUDE***

| **Nombre del objeto** | **Tipo** | **Descripción** |
| --- | --- | --- |
| aecCBox | QCheckBox | Calcula la **AEC (Envelope Amplitude Correlation)**. |
| aecyesRButton, aecnoRButton | QRadioButton | Seleccionan si se aplica **ortogonalización**. |
| iacCBox | QCheckBox | Calcula la **IAC (Instantaneous Amplitude Correlation)**. |
| iacyesRButton, iacnoRButton | QRadioButton | También seleccionan ortogonalización. |

***GROUPBOX: PHASE***

| **Nombre del objeto** | **Tipo** | **Descripción** |
| --- | --- | --- |
| plvCBox | QCheckBox | Calcula el **PLV (Phase Locking Value)**. |
| pliCBox | QCheckBox | Calcula el **PLI (Phase Lag Index)**. |
| wpliCBox | QCheckBox | Calcula el **wPLI (weighted Phase Lag Index)**. |

(No requieren parámetros adicionales.)

***GROUPBOX: TIMEFREQUENCY***

Contiene una etiqueta QLabel con el texto **"To be done"**, indicando que esta funcionalidad está reservada para versiones futuras.

**Página: *graphMetricsPage***

Contiene un único elemento *QLabel* con el texto **"To be done"**, reservado para desarrollos futuros relacionados con métricas de grafos.

**NOTA**: Todos los widgets que representan **parámetros adicionales** (QSpinBox, QDoubleSpinBox, QLineEdit) se encuentran **ocultos por defecto** (setVisible(False)) y **solo se muestran** cuando el checkbox asociado está activo. Esto se gestiona mediante las señales toggled(bool) de los QCheckBox, conectadas a funciones que controlan la visibilidad de los parámetros relacionados.

**CONEXIÓN ENTRE VENTANAS**

* Al abrir MEDUSA-ANALYZER, se carga el widget de **Preprocessing** en el *QStackedWidget*.
* Con el botón *nextButton*, se cambia al siguiente widget: **Segmentation**.
* El botón *backButton* permite volver a *Preprocessing* si el usuario lo desea.
* El *progressLabel* se actualiza dinámicamente en cada cambio de pantalla con el estado actual del flujo.
* Al pulsar *nextButton* en la **MainWindow**, tras Preprocessing, se llega a la pantalla de **Segmentation**.
* El *progressLabel* se actualiza a algo como "Paso 2 de 3: Segmentación (66%)".
* Si el usuario pulsa *backButton*, se vuelve al widget de Preprocessing conservando las selecciones previas.
* Al pulsar *nextButton* en el widget de **Segmentation** (paso 2), el sistema carga automáticamente la fase de **Parameters** (paso 3) dentro del *QStackedWidget*.
* En ese momento, el *progressLabel* se actualiza dinámicamente con el texto:  
  "Paso 3 de 3: Parámetros (100%)"
* Si el usuario pulsa *backButton* desde Parameters, el sistema retorna al widget de **Segmentation** conservando las selecciones previamente realizadas (por ejemplo: segmentación por eventos, valores de ventana, etc.).
* La interacción con *bands\_table.ui*  se realiza desde *Parameters* al pulsar *bandButton*, o *rpButton* y es una ventana modal que bloquea la interfaz hasta que se pulsa *acceptButton*, momento en el que se devuelve el contenido a *bandLabel*.
* Ver como gestionar bien lo de las tablas, teniendo en cuenta que se puede interaccionar con ella dos veces y no se tiene porque poner lo mismo las dos veces.
* Todos los valores introducidos por el usuario (checkboxes activados, escalas definidas, parámetros numéricos, etc.) se almacenan internamente en el controlador del flujo y no se pierden al navegar hacia atrás y volver.
* Todos los datos se van a ir guardando en un diccionario con diccionarios anidados. Pondremos none si no hay datos sobre algo.