EOG con *Neurobit Optima 4*

Juan Hernández García,
Departamento de Arquitectura y Tecnología de Computadores,
Universidad de Granada

27 de marzo de 2014

Resumen

Guía básica y tutoriales sobre el desarrollo de aplicaciones EOG para *Neurobit Optima 4*.

Índice

1. Introducción

En esta guía encontrará los conceptos básicos necesarios para comenzar a desarrollar aplicaciones con *Neurobit Optima 4* de una forma rápida. Se dan pinceladas generales para la utilización de todas las características del dispositivo, aunque este documento se centra en aplicaciones EOG.

2. Características

Neurobit Optima 4 es un dispositivo capaz de optener multiples medidas del cuerpo humano. Entre las señales más importantes que podemos obtener tenemos:

- EEG
- EMG
- EHRV
- GSR
- TEMP

Podemos ver que tenemos un amplio espectro de posiblidades. Algunos dispositivos se centran sólo en las señales EEG, sin embargo *Neurobit Optima 4* puede recoger múltiples lecturas y combinarlas para obtener datos globales. Se puede obtener una descripción más precisa en la web del fabricante. **Hacer referencia**.

3. Canales y Mediciones

EL dispositivo tiene cuatro canales principales nombrados como **A**, **B**, **C**, y **D**. Todos los canales sirven para obtener cualquier tipo de medida, aunque habrá que **configurar mediante software** cada canal antes de su utilización.

Otro canal importante es el **VG**. Este canal hace de tierra y siempre tendremos que utilizarlo. **No es necesario configurarlo**.

Hay que tener en cuenta que la mayoría de mediciones se pueden obtener con los sensores de voltaje que vienen con el dispotivo, sin embargo, si se quieren capturar medidas no relacionadas con el voltaje (temperatura, por ejemplo), **se necesitarán sensores resistivos** no incluidos en el paquete de *Neurobit Optima 4*.

3.1. Configuración de Canales

Todas las aplicaciones que trabajan con *Neurobit Optima 4* constan de un apartado o menú de configuración del dispositivo. Siempre que se abre, se hace una llamada a la API que devuelve la ventana de configuración estándar que se ve en la figura1

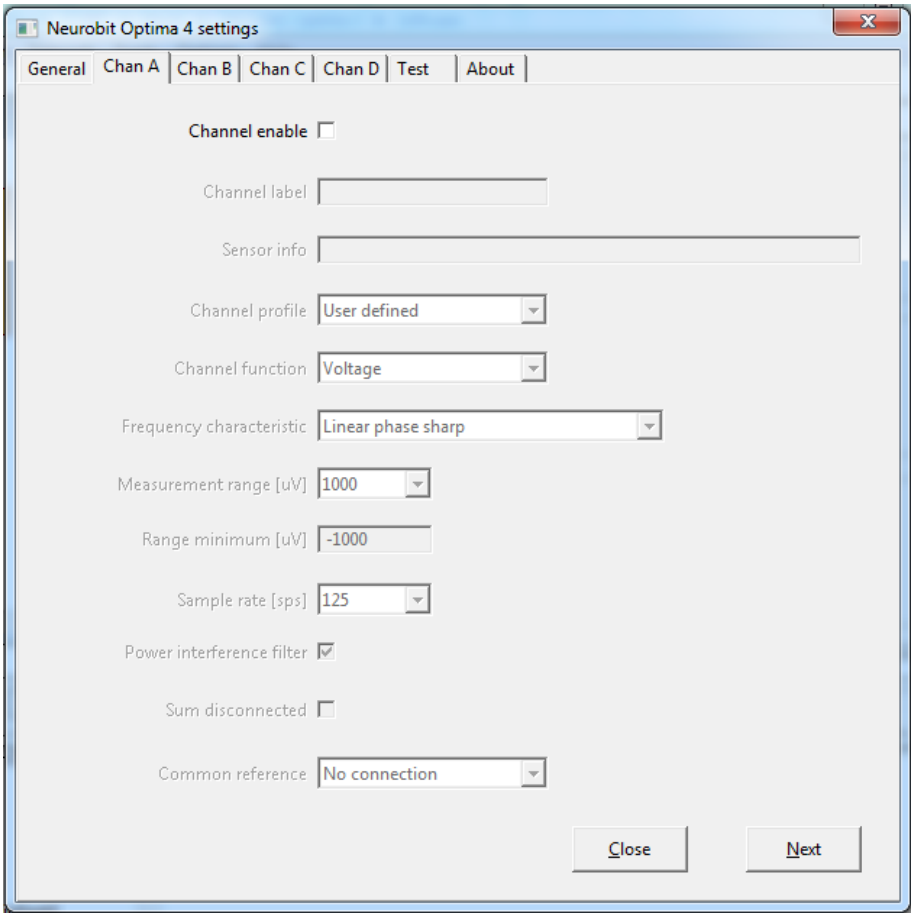

Figura 1: Ventana de configuración

En esta ventana podemos configurar cada uno de los canales para la recepción de las medidas requeridas. Los canales que no vayan a usarse deben estar deshabilitados. Aunque hay configuraciones estándar para cada uno de los tipos de lecturas, el usuario puede configurar estos parámetros manualmente, si piensa que no son adecuados.

Adicionalmente hay una pestaña de Test, donde se puede comprobar la correcta colocación de los sensores y la calidad de señal. Se recomienda siempre hacer este test antes de ejecutar cualquier aplicación, para detectar posibles errores en la colocación o ajuste de canales.

3.2. Correcta colocación de sensores

La correcta colocación de los sensores es fundamental a la hora de recibir datos correctamente. Hay que dedicar tiempo a ello, o todo el proceso posterios será errático. Cabe decir que no hay una forma de colocar siempre los sensores, dependiendo del tipo de medición a realizar se necesitará una disposición de los mismos u otra. Como normal general, se pueden utilizar las disposiciones de la figura2, según la medición sea con sensores de voltaje o con sensores resistivos.

Los sensores necesitan ser conectados a los canales e imbuidos con un gel conductivo. El gel conductivo debe sobresalir un milímetro del sensor, pero no más. Un exceso del mismo podría provocar malas lecturas.

Para obtención de medidas de voltaje para EEG, el canal VG(tierra) debe estar siempre conectado a uno de los lóbulos de las orejas. Es aconsejable utilizar el sensor blanco, cuya pinza ayudará a la colocación. Por otro lado, la parte negativa de cualquier otro canal se enganchará al lóbulo de la oreja contraria. La parte positiva de los canales se repartirá por la cabeza, apartando el pelo para maximizar el contacto directo con la superficie.[1]

Aunque la colocación descrita funciona para casi todas las medidas de voltaje, para obtener medidas EMG es necesario variarla. El canal VG(tierra) sigue yendo al lóbulo de una de las orejas, pero la parte positiva y negativa de cada uno del resto de canales, debe ponerse a cada extremo del músculo a estudiar. Por ejemplo, si quisiéramos estudiar EMG en los músculos horizontales del ojo, deberíamos colocar la parte negativa del canal muy cerca del tabique de la nariz, y el positivo en el rabillo del ojo correspondiente.

3.3. Interpretación de las Medidas

El dispositivo es capaz de obtener 2000 muestras por segundo. Ademas, puede ser forzado a recibir 5000 por segundo, durante algunos segundos. Este nivel de muestreo es más que suficiente para realizar aplicaciones de tiempo real responsivas. Ahora bien, aparte de las amplicaciones correspondientes que se realizan dentro del dispositivo,

[1] Se aconseja utilizar el sistema 10-30 para el espaciado entre sensores

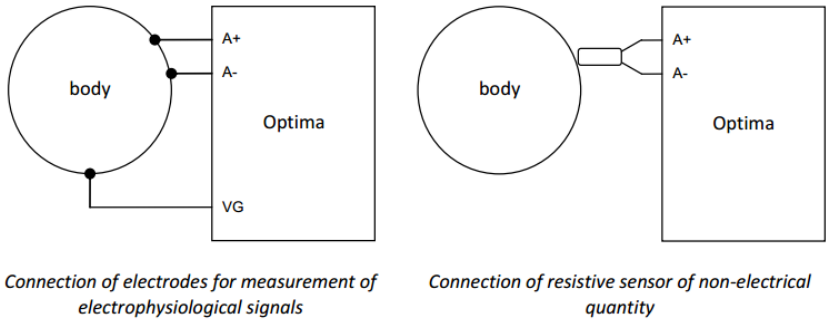

Figura 2: Colocación Genérica de Sensores

no hay más tratamiento interno de los datos (al contrario que sucede con *MindWave Mobile*). Se nos devolverán valores de voltajes (absolutos o relativos), que debemos interpretar nosotros mediante nuestras aplicaciones. Si recibimos un número negativo del orden de -30.000, será indicador de que hay error en la señal.

4. Brain Bay

Existen múltiples aplicaciones que permiten tratar los datos recolectados con *Neurobit Optima 4*. Los datos en bruto no son de mucha utilidad, por eso es tan importante el uso de las mismas. En este manual se centrará en una de ellas, *BrainBay*, seleccionada para su potencia y licencia libre.

BrainBay es un entorno de programación gráfico. Programamos el tratamiento de las señales recibidas añadiendo objetos(cajas) y realizando las conexiones pertinentes entre los mismos. Es rápido y sencillo. Entre los objetos que se pueden añadir encontramos: fuentes, transformadores, y displays.

4.1. Fuentes

BrainBay no se centra únicamente en *Neurobit Optima 4*, sino que puede utilizar casi todas las herramientas EEG del mercado. Para ello dispone de un conjunto de fuentes donde están incluidos casi todos los dispositivos comerciales. Al ser una herramienta de software libre y no anclada a un dispositivo concreto, sigue creciendo conforme lo hace el mercado y nos permite la integración de varios dispotivos distintos en una misma aplicación.

Aunque, para los fines de este manual, nos centraremos únicamente en la fuente: EEG Amplifier ->Neurobit Biosignal, no está demás saber que tenemos muchas otras posibilidades de fuentes, incluidas la cámara del ordenador, documentos EDF, etc...

4.2. Transformadores

Estos objetos son los encargados (como su nombre bien indica) de transformar las señales que les llegan de fuentes u otros transformadores. Entre ellos tenemos filtros de señal, umbrales, módulos para expresiones aritmético-lógicas y mucho más. Se recomienda mirar uno a uno los transformadores en la documentación de *BrainBay*.

4.3. Displays

Estos objetos son los encargados de obtener los datos tratados y actuar de mil maneras distintas. Lo más utilizados son los osciloscopios y analizadores de espectro de señal, no obstante, *BrainBay* cuenta con un abanico mucho mayor y, por qué no decirlo, divertido. Entre las opciones con las que contamos, tenemos un display de música midi que toca distintos instrumentos y notas según la señal recibida. También podemos mandar señales a videos y actuar en consecuencia. Las posibilidades son bastante altas.

Como se pretende trabajar a nivel de EOG en tiempo real, un display que puede presentarse de alta utilidad es el **FileWriter**, que nos permite mandar datos ya tratados a un archivo de texto plano.

En la figura 3 se puede ver un ejemplo completo de aplicación con los tres tipos de objetos mencionados.

5. EOG en tiempo real con *Neurobit Optima 4*

Aunque *Neurobit Optima 4* no cuenta directamente con una opción para la captura de EOG, esto no es más que EMG aplicado a los músculos del ojo. Eso es lo que vamos a hacer. Vamos a seguir el proceso siguiente:

- Colocación de Sensores.
- Analizar la señal para obtener patrones.
- Filtrar señal para eliminar ruido.
- Hacer uso de autómata que sea capaz de interpretar dichos patrones.
- Desarrollar una aplicación que use el autómata.

5.1. Colocación de Sensores para EOG

Necesitaremos 5 sensores para la completa captación del EOG. Dos colocados a la parte positivas de A y B, y uno más para VG(tierra). En la figura4 se muestra la forma óptimo de colocación de los sensores. En ella quedan representados los sensores conectados a positivo en rojo y los negativos en negro.

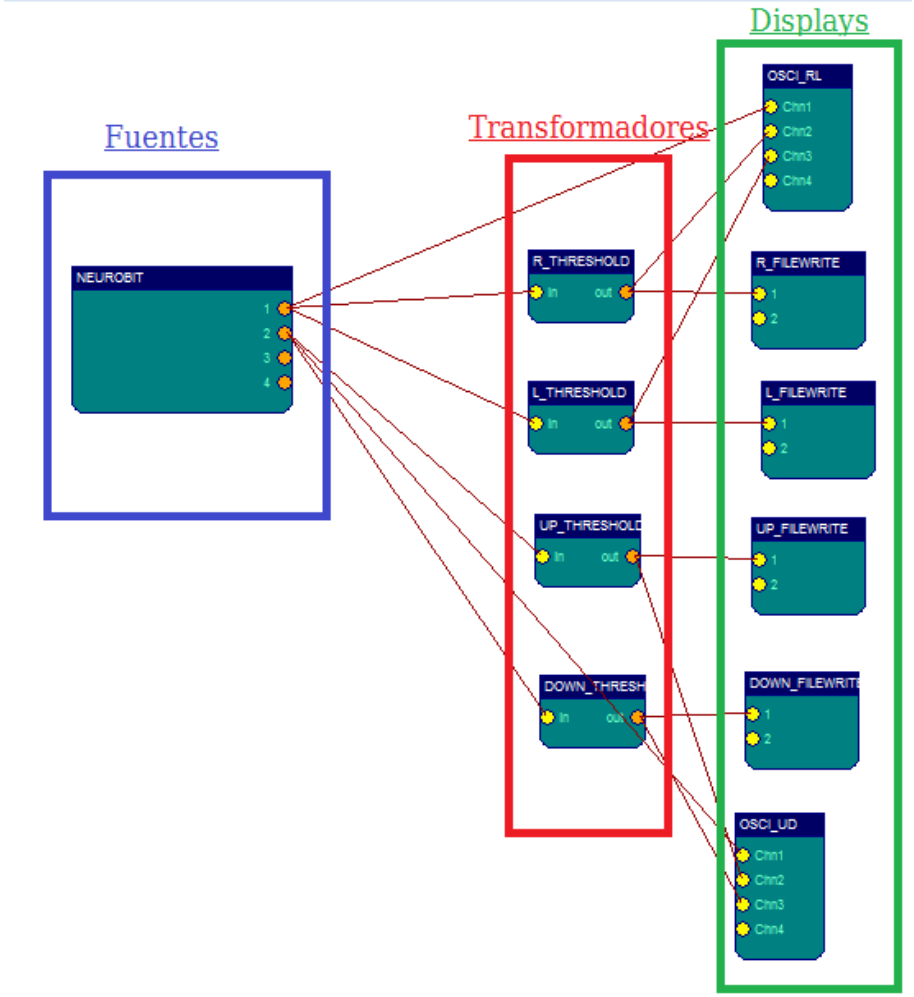

Figura 3: Fuentes, transformadores, y Displays

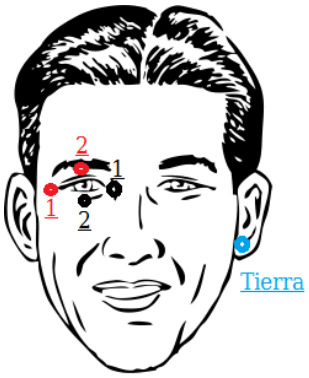

Figura 4: Colocación de Sensores para EOG

5.2. Análisis de Señal

En la gran mayoría de la literatura actual, se nos indica de que forma se comporta el votaje con respecto a los movimientos oculares. Sin embargo estos datos han sido tomados con dispositivos de altas prestaciones. Sería adecuado realizar un análisis de la señal recibida con *Neurobit Optima 4* para comprobar si se ajusta al comportamiento esperado, y en qué umbrales lo hace.

Utilizando la herramienta **Recorder.exe** que viene con el paquete del dispositivo se han hecho capturas sobre la variación del voltaje en el eje horizontal y vertical. Los resultados obtenidos, mostrados por la figura5, parecen adecuarse a la forma clásica representada en la literatura.

La señal se mueve entre [-700, 700] uV. Más en concreto, la figura mostrada es la variación respecto a un movimiento horizontal de ojo hacia la derecha, con la colocación de sensores descrita. Un movimiento hacia la izquierda tiene el comportamiento inverso, haría que la señal primero bajara hasta -700 y luego subiera cerca de los 700 de nuevo.

Si medimos la señal respecto al movimiento vertical del ojo el coportamiento es el mismo, con una excepción, los parpadeos también provocan un incremento en las mediciones recibidas, como puede verse en la figura6.

En ella podemos detectar un patrón. En los movimientos ascendentes la señal sube y luego baja, mientras que en los parpadeos el rebote hacia abajo es apenas inexistente. Podemos utilizar este patrón para diferenciar parpadeos y movimientos ascendentes o descendentes.

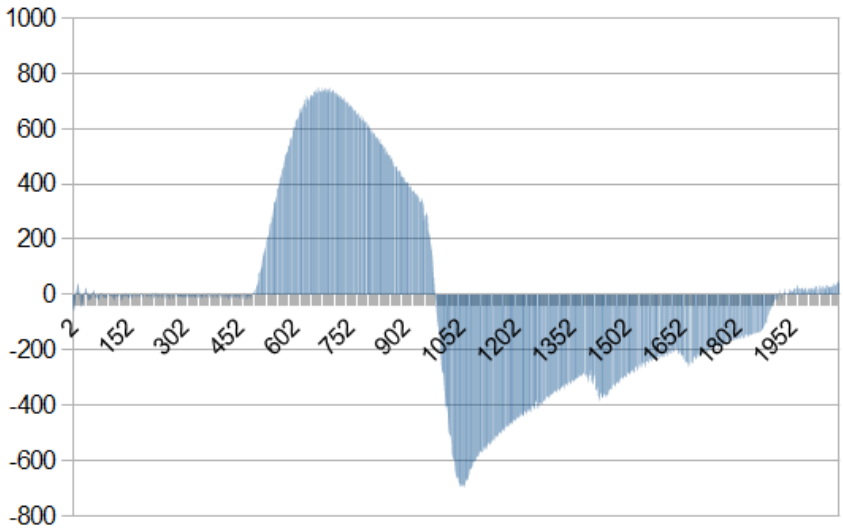

Figura 5: Movimiento a la Derecha

Figura 6: Arriba, abajo, y parpardeo

5.3. Filtrado de señal

Ya que conocemos *BrainBay* y sus cualidades en el tratamiento de la señal, vamos a utilizarlo para filtrar la señal y hacer más sencillo el trabajo de detección de patrones. Básicamente hemos observado que:

- Movimientos Derecha: valores del canal1 por encima de 600.
- Movimientos Izquierda: valores del canal1 por debajo de -600.
- Movimiento Arriba: valores del canal2 por encima de 600 (y posteriormente caida a valores negativos).
- Movimiento Abajo: valores del canal2 por debajo de -600.
- Parpadeo: valores del canal2 por encima de 600 (sin caída posterior).

Se ha creado un programa, llamado EOG.con, que realiza un proceso de filtrado de señal y escribe en 4 ficheros de salida (correspondientes a arriba, abajo, izquierda y derecha). En cada uno de los ficheros sólo se escriben las partes de la señales que son interesantes para dicho canal. No es necesaria la comprensión compleja del programa, pero vamos a intentar explicar las partes más relevantes.

El programa es el mostrado en la figura3. Se puede ver que lee de dos canales del dispositivo *Neurobit Optima 4*, más concretamente de los canales A, y B. A se utiliza para el movimiento horizontal y B para el vertical. Los objetos transformadores que tiene son limitadores de umbral. Cada señal se limita al umbral de interés citado[2]. Finalmente, las señal resultante de los umbrales se escribe en cuatro FileWriter[3] (cuatro ficheros de salida) cada uno correspondiente a señales derecha, izquierda, abajo, arriba. Estos ficheros son los que leerá el autómata de estados para determinar cuando avisar de un movimiento del ojo.

5.4. Autómata EOG

Se ha desarrollado una autómata de estados que es capaz de capturar los datos generados por el programa creado en *BrainBay*. No es necesario que el desarrollador conozca todas las características de implementación del mismo, tan sólo las funciones principales que le permitan utilizarlo e integrarlo en su aplicación. El proyecto en el repositorio: `https://github.com/juanhg/EOGEyeTracking` contiene los ficheros requeridos por el autómata y le servirá de guía y soporte para el desarrollo de aplicaciones que hagan uso de él.

Debemos prestar especial atención a dos ficheros:

- IEOG.java
- EOGModel.java

[2]Es posible que sea necesaria calibración de estos umbrales según la persona que utilice el dispotivo

[3]Asegúrese de que cuando inicia la aplicación los fileWriter están abiertos y en modo escritura. Para ello pulse botón derecho sobre ellos y configúrelos. Es posible que esta configuración se restablezca al cerrar *BrainBay*.

IEOG Intefaz que deberá implementar cualquier aplicación para recibir correctamente los mensajes del modelo. La interfaz define cinco métodos, que son los que recibirán los eventos del modelo para actuar en consecuencia.

```

    public abstract void rightEvent();
    public abstract void leftEvent();
    public abstract void upEvent();
    public abstract void downEvent();
    public abstract void blinkEvent();
```

EOGModel Contiene la implementación del modelo. Con tan sólo conocer tres de sus métodos podremos trabajar con él. Pasemos a analizarlas:

```

//Constructors
public EOGModel(String rightFile, String leftFile, String
    upFile, String downFile);

public EOGModel(String rightFile, String leftFile, String
    upFile, String downFile,
                            int rightLimit, int leftLimit, int
                                upLimit, int downLimit);
```

Ambos constructores reciben cuatro path de ficheros. Estos ficheros son los generados por la aplicación de *BrainBay* antes citada. El segundo constructor permite además modificar los rangos de los límites establecidos por defecto. Los límites de señal establecidos por defecto, se adecúan a los umbrales establecidos en EOG.con. **Si se hacen cambios de calibración** en el mismo, deberá llamarse al modelo con los nuevos valores escogidos.

```
public void simulate(IEOG listener);
```

Este método avanza un instante de simulación. Para que tenga el efecto deseado, hay que llamarlo dentro de un bucle. No se realiza el bucle internamente el bucle, para permitir al usuario realizar acciones entre instantes de simulación. Con este se consigue una mayor flexibilidad para las futuras posibles aplicaciones. Como se puede observar, simulate recibe un elemento que implemente la interfaz del modelo. Como nuestra aplicación que haga uso del modelo deberá implementar la interfaz, bastará con llamar al simulate de la siguiente manera:

```
while(...){
    simulate(this);
    //Aqui posibles acciones extra
}
```

Por último, añadir que el modelo hace uso de Time.java (también contenida en el repositorio) para llevar cómputo interno de tiempo.

5.5. Construcción de Aplicaciones

Construir aplicaciones que hagan uso del modelo es simple. Hay que seguir los siguientes pasos:

- Crear un nuevo proyecto que incluya tanto la intefaz como el modelo.
- Crear la clase principal de la aplicación y hacer que implemente la interfaz IEOG.
- Implementar el funcionamiento los eventos requeridos por la interfaz.
- Instanciar y lanzar el modelo.

Un posible esqueleto es el siguiente:

```
1  public class EOGBasicApplication implements IEOG {
2
3      EOGModel model;
4
5      public EOGBasicApplication(String rFile, String lFile,
           String dFile, String uFile ) throws IOException{
6          model = new EOGModel(rFile, lFile, dFile, uFile);
7      }
8
9      public void run() throws IOException{
10         while(true){
11             model.simulate(this);
12             //Otras acciones
13         }
14     }
15
16     @Override
17     public void rightEvent() {
18         System.out.println("Derecha");
19         //Acciones cuando se detecta Derecha
20     }
21
22     @Override
23     public void leftEvent() {
24         System.out.println("Izquierda");
25         //Acciones cuando se detecta Izquierda
26     }
27
28     @Override
29     public void upEvent() {
30         System.out.println("Up");
31         //Acciones cuando se detecta Arriba
```

```
32      }
33
34      @Override
35      public void downEvent() {
36          System.out.println("Down");
37          //Acciones cuando se detecta Abajo
38      }
39
40      @Override
41      public void blinkEvent() {
42          System.out.println("Parpadeo");
43          //Acciones cuando se detecta Parpadeo
44      }
45
46  }
```

Recuerde, que el modelo lee datos de cuatro ficheros de entrada. Estos ficheros deben estar rellenándose mientras las aplicación está encendida, lo que quiere decir que se deben estar capturando datos del usuario en ese momento, para poder hacer un lectura en vivo. Antes de ejecutar las aplicaciones que hagan uso del modelo, compruebe:

- Que EOG.con (o programa similar) está funcionando.
- Que los cuatro ficheros de datos se están rellenando correctamente.
- Que se están rellenando mediante escritura no bloqueante (EOG.con permite la lectura mientras se él está escribiendo)

Si todo es correcto, cuando se instancia el modelo, se sincronizará con los datos más recientes de los ficheros y comenzará a evolucionar con respecto a los datos en tiempo real.