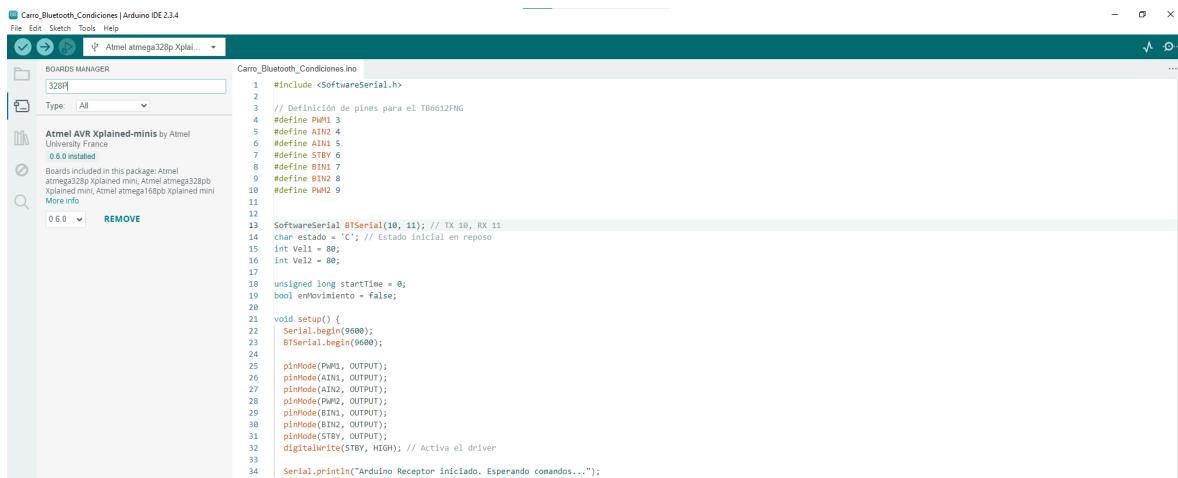


ARDUINO

Nota: Para reprogramar se requiere tener Arduino IDE instalado

Nota: Este código está preinstalado en los Arduinos Nano, este proceso solo se hace una vez, si se requiere modificar y cambiar el código se necesita reprogramar estos Arduinos.

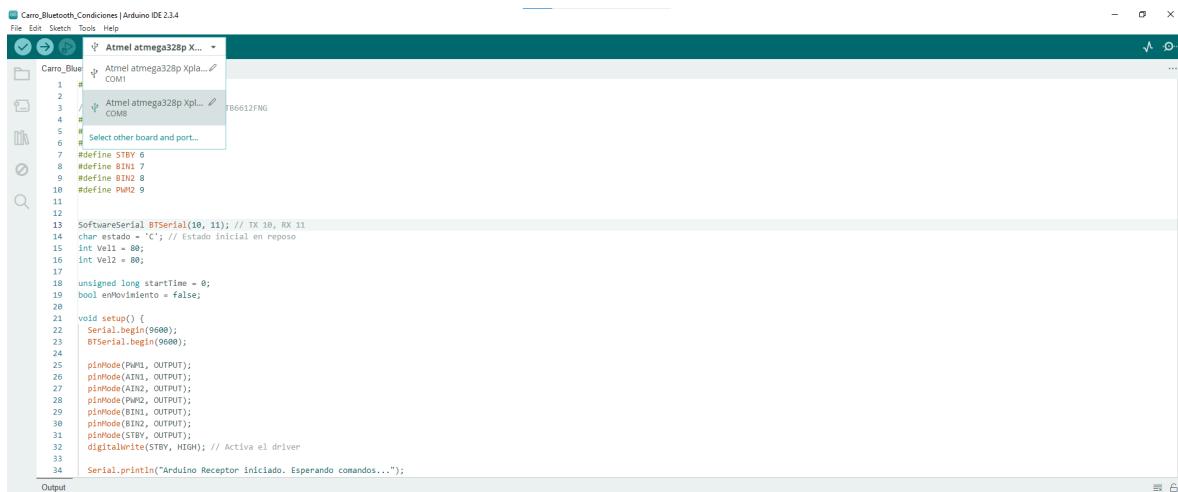


The screenshot shows the Arduino IDE interface. The top menu bar includes File, Edit, Sketch, Tools, Help, and a dropdown for boards. The main window displays the code for 'Carro_Bluetooth_Condiciones.ino'. On the left, the Boards Manager is open, showing the 'Atmel AVR Xplained-minis by Atmel' package version 0.6.0 installed. The code itself is as follows:

```
1 #include <SoftwareSerial.h>
2
3 // Definición de pines para el TB6612FNG
4 #define AIN1 4
5 #define AIN2 5
6 #define STBY 6
7 #define BIN1 7
8 #define BIN2 8
9 #define BIN2_9
10 #define PWM2 9
11
12
13 SoftwareSerial BTSerial(10, 11); // TX 10, RX 11
14 char estado = 'C'; // Estado inicial en reposo
15 int Vell = 80;
16 int Velz = 80;
17
18 unsigned long startTime = 0;
19 bool envMovimiento = false;
20
21 void setup() {
22   Serial.begin(9600);
23   BTSerial.begin(9600);
24
25   pinMode(PWM1, OUTPUT);
26   pinMode(AIN1, OUTPUT);
27   pinMode(AIN2, OUTPUT);
28   pinMode(PWM2, OUTPUT);
29   pinMode(STBY, OUTPUT);
30   pinMode(BIN1, OUTPUT);
31   pinMode(BIN2, OUTPUT);
32   pinMode(STBY, OUTPUT);
33   digitalWrite(STBY, HIGH); // Activa el driver
34   Serial.println("Arduino Receptor iniciado. Esperando comandos...");
}
```

Figura 1 Código Arduino Carro_Bluetooth_Condiciones.

Instalar la compatibilidad para cargar el programa al arduino nano, se recomienda si no es original para poder cargar el código en estos arduinos yendo al apartado de Boards Manager y instalar **Atmel AVR Xplained minis package** for Arduino IDE Boards Manager!



The screenshot shows the Arduino IDE interface. The top menu bar includes File, Edit, Sketch, Tools, Help, and a dropdown for ports. The main window displays the code for 'Carro_Bluetooth_Condiciones.ino'. A port selection dialog is open, showing 'Atmel atmega328p Xpl...' as the selected port. The code is identical to Figure 1:

```
1 #include <SoftwareSerial.h>
2
3 // Definición de pines para el TB6612FNG
4 #define AIN1 4
5 #define AIN2 5
6 #define STBY 6
7 #define BIN1 7
8 #define BIN2 8
9 #define BIN2_9
10 #define PWM2 9
11
12
13 SoftwareSerial BTSerial(10, 11); // TX 10, RX 11
14 char estado = 'C'; // Estado inicial en reposo
15 int Vell = 80;
16 int Velz = 80;
17
18 unsigned long startTime = 0;
19 bool envMovimiento = false;
20
21 void setup() {
22   Serial.begin(9600);
23   BTSerial.begin(9600);
24
25   pinMode(PWM1, OUTPUT);
26   pinMode(AIN1, OUTPUT);
27   pinMode(AIN2, OUTPUT);
28   pinMode(PWM2, OUTPUT);
29   pinMode(STBY, OUTPUT);
30   pinMode(BIN1, OUTPUT);
31   pinMode(BIN2, OUTPUT);
32   pinMode(STBY, OUTPUT);
33   digitalWrite(STBY, HIGH); // Activa el driver
34   Serial.println("Arduino Receptor iniciado. Esperando comandos...");
}
```

Figura 2. Puertos Arduino Nano.

Ahora al conectar el arduino al equipo podemos observar que genera un puerto COM, si en el equipo tiene vinculados más COM aparecerán en la lista, si no sabes cuál es, la

recomendación es desconectar el COM que conectaste y volverlo a conectar, en este caso se tienen dos COM y al desconectarlo aparece solo el COM 1 lo que me dice que mi arduino es el COM 8, eso si siempre se conecta al mismo puerto USB, si se conecta a otro, el puerto COM cambia.

```
Carro_Bluetooth_Condiciones | Arduino IDE 2.3.4
File Edit Sketch Tools Help
Atmel atmega328p Xpl...
Carro_Blu
1
2
3 Select other board and port...
4 #define PWM1 3
5 #define PWM2 4
6 #defineAIN1 5
7 #define STBY 6
8 #define BIN1 7
9 #define BIN2 8
10 #define PWM2 9
11
12
13 SoftwareSerial BTSerial(10, 11); // TX 10, RX 11
14 char estado = 'C'; // Estado inicial en reposo
15 int VelL = 80;
16 int VelR = 80;
17
18 unsigned long startTime = 0;
19 bool enMovimiento = false;
20
21 void setup() {
22   Serial.begin(9600);
23   BTSerial.begin(9600);
24
25   pinMode(PWM1, OUTPUT);
26   pinMode(AIN1, OUTPUT);
27   pinMode(AIN2, OUTPUT);
28   pinMode(PWM2, OUTPUT);
29   pinMode(STBY, OUTPUT);
30   pinMode(BIN1, OUTPUT);
31   pinMode(BIN2, OUTPUT);
32   pinMode(STBY, OUTPUT);
33   digitalWrite(STBY, HIGH); // Activa el driver
34   Serial.println("Arduino Receptor iniciado. Esperando comandos...");

```

Figura 3. Verificación del Puerto COM.

Al seleccionar el COM 8 podemos buscar nuestra tarjeta, en este caso es la que instalamos y la podemos buscar como 328p, seleccionamos el puerto y la tarjeta como se muestra a continuación.

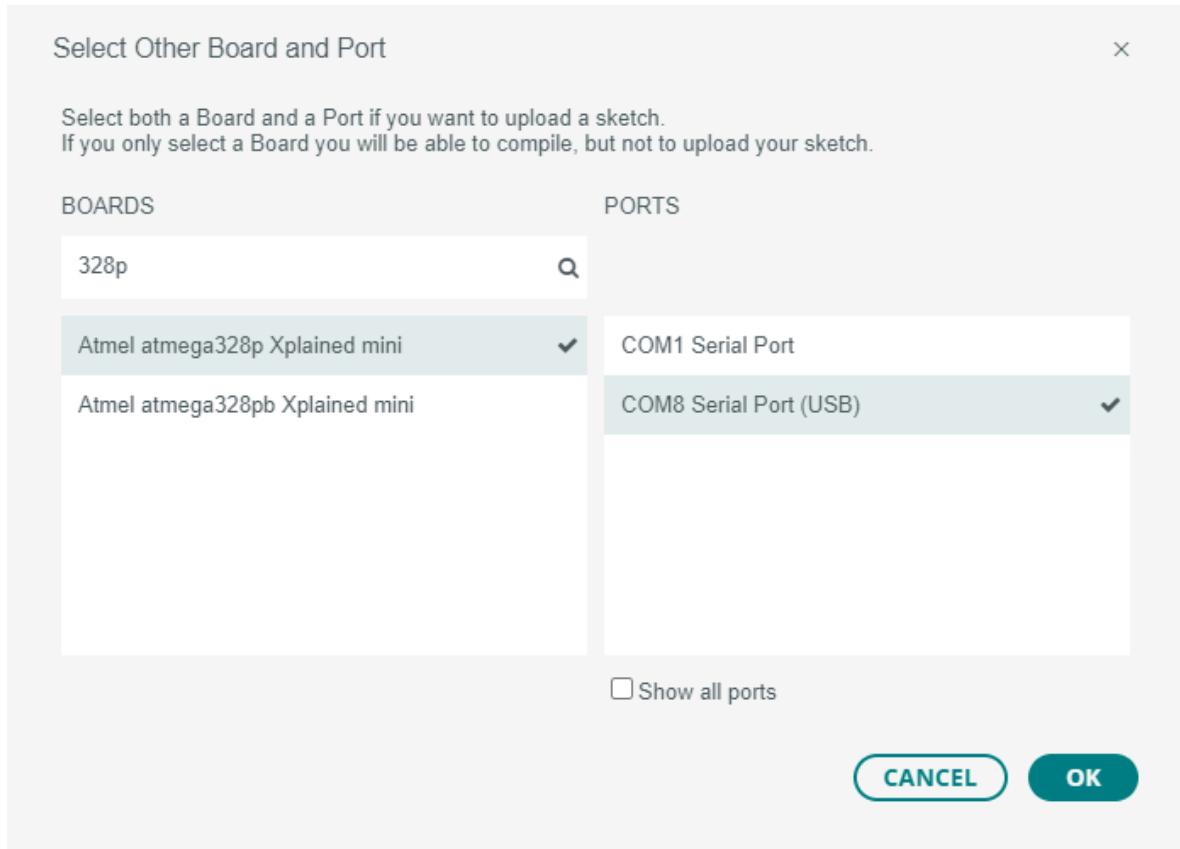


Figura 4. Selección de Tarjeta a Programar.

Después de seleccionar el Atmel atmega 328p podemos subir el código, en este caso vamos a cargar el código para el el carro bluetooth, este tendrá las condiciones requeridas después de recibir el comando para su activación, este código ejecuta una función de movimiento por un tiempo determinado según el carácter que reciba por bluetooth, las funciones del carro están predeterminadas para ejecutarse con los comandos **W, S, A, D** los cuales ejecutan funciones de movimiento distintas.

```

Carro_Bluetooth_Condiciones | Arduino IDE 2.3.4
File Edit Sketch Tools Help
Atmel atmega328p Xplai... Upload
Carro_Bluetooth_Condiciones.ino
1 #include <SoftwareSerial.h>
2 // Definición de pines para el TB6612FNG
3 #define PWM1 3
4 #defineAIN1 4
5 #defineAIN2 5
6 #defineSTBY 6
7 #defineBIN1 7
8 #defineBIN2 8
9 #defineBIN3 9
10 #definePWM2 9
11
12
13 SoftwareSerial BTSerial(10, 11); // TX 10, RX 11
14 char estado = 'C'; // Estado inicial en reposo
15 intVelL = 80;
16 intVelR = 80;
17
18 unsigned long startTime = 0;
19 bool envovimiento = false;
20
21 void setup() {
22   Serial.begin(9600);
23   BTSerial.begin(3600);
24
25   pinMode(PWM1, OUTPUT);
26   pinMode(AIN1, OUTPUT);
27   pinMode(BIN1, OUTPUT);
28   pinMode(PWM2, OUTPUT);
29   pinMode(BIN2, OUTPUT);
30   pinMode(STBY, OUTPUT);
31   pinMode(STBY, OUTPUT);
32   digitalWrite(STBY, HIGH); // Activa el driver
33
34   Serial.println("Arduino Receptor iniciado. Esperando comandos...");
}

```

Figura 5. Subir el Código al Arduino Nano.

Después de subir el código vamos a cargar el otro código de la segunda tarjeta, la cual mandará instrucciones desde el PC.

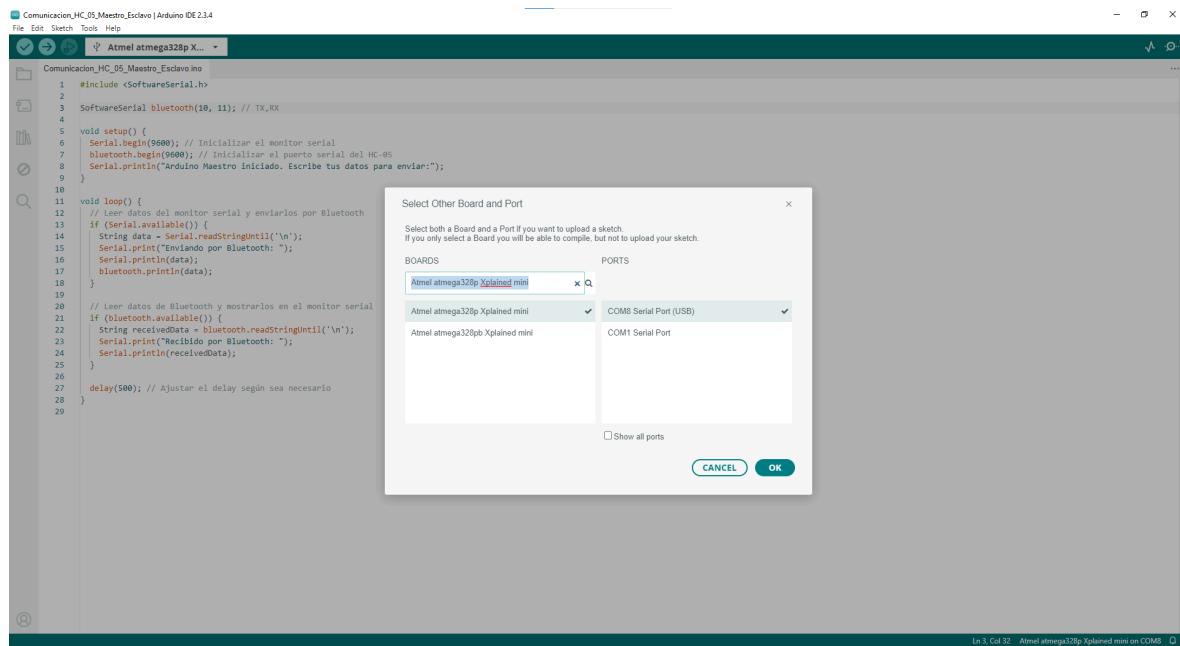


Figura 6. Código de Comunicación Bluetooth.

Configuramos nuevamente el otro bluetooth y cargamos el programa de comunicación maestro esclavo, en este caso los módulos HC-05 ya están configurados para estas funciones.

ahora para realizar la pruebas del sistema podemos abrir el monitor serial y testear el carro enviando información por el monitor serial

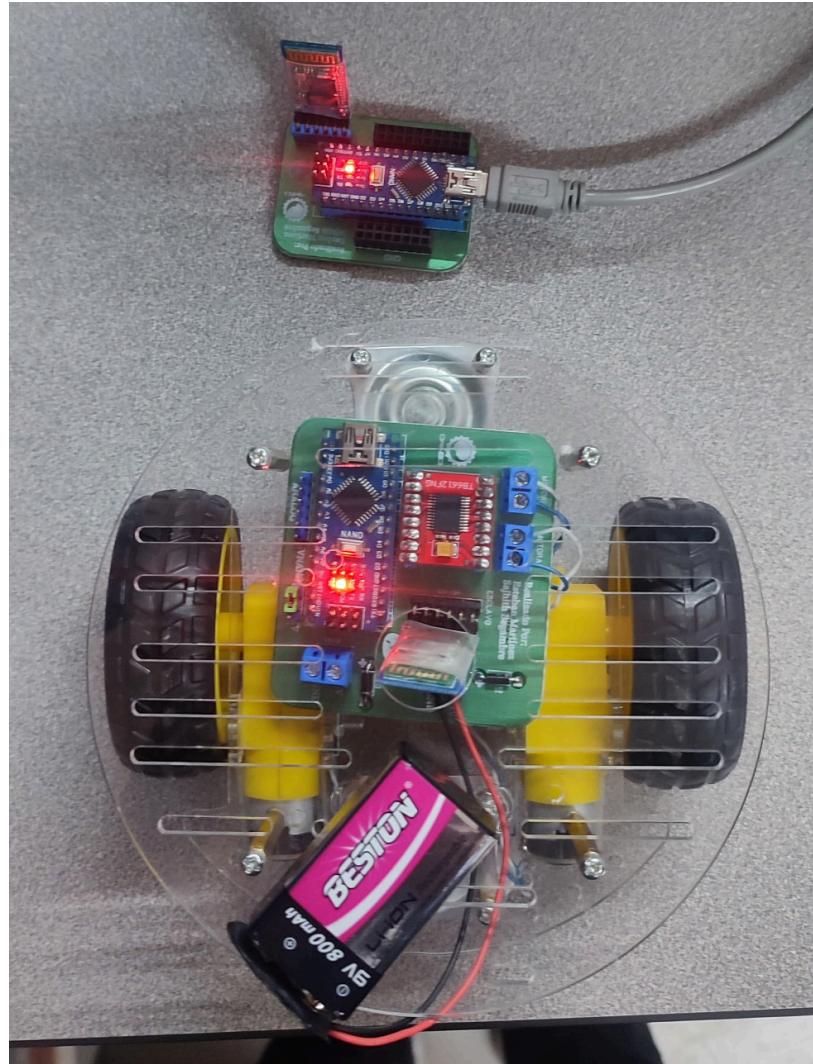


Figura 7. Montaje del Sistema.

Para la prueba del sistema se requiere que las dos tarjetas estén alimentadas, ya que se requiere probar su funcionamiento por monitor serial.

The screenshot shows the Arduino IDE interface with the following details:

- Title Bar:** Comunicación_HC_05_Maestro_Esclavo | Arduino IDE 2.3.4
- Menu Bar:** File, Edit, Sketch, Tools, Help
- Sketch Area:** A code editor window titled "Atmel atmega328p X...". The code is for a HC-05 Master Esclavo sketch, utilizing SoftwareSerial and SoftwareSerial libraries.

```
File Edit Sketch Tools Help
Atmel atmega328p X...
Comunicación_HC_05_Maestro_Esclavo.ino
1 #include <SoftwareSerial.h>
2
3 SoftwareSerial bluetooth(10, 11); // TX,RX
4
5 void setup() {
6   Serial.begin(9600); // Inicializar el monitor serial
7   bluetooth.begin(9600); // Inicializar el puerto serial del HC-05
8   Serial.println("Arduino Maestro iniciado. Escribe tus datos para enviar:");
9 }
10
11 void loop() {
12   // Leer datos del monitor serial y enviarlos por Bluetooth
13   if (Serial.available()) {
14     String data = Serial.readStringUntil('\n');
15     Serial.print("Enviendo por Bluetooth: ");
16     Serial.println(data);
17     bluetooth.println(data);
18   }
19
20   // Leer datos de Bluetooth y mostrarlos en el monitor serial
21   if (bluetooth.available()) {
22     String receivedData = bluetooth.readStringUntil('\n');
23     Serial.print("Recibido por Bluetooth: ");
24     Serial.println(receivedData);
25   }
26
27   delay(500); // Ajustar el delay según sea necesario
28 }
29
```

- Serial Monitor:** A terminal window showing the text "Arduino Maestro iniciado. Escribe tus datos para enviar:"
- Status Bar:** Indicating Line 3, Column 32, Atmel atmega328p Xplained mini on COM8

Figura 8. Envío de Datos Prueba Monitor Serial.

Antes de enviar el dato por el monitor serial debemos observar que el módulo no esté parpadeando de manera intermitente, porque esto indica que el módulo aún no se ha emparejado, si esto sucede posiblemente el otro módulo no tenga alimentación suficiente o el otro módulo esté apagado. Una vez que el módulo no parpadee o parpadee cada 3 o 4 segundos está vinculado y listo para enviar información, si se envía la letra en mayúscula el carro debería moverse por petición del puerto serial.

OPENVIBE

Ahora vamos al apartado de Openvibe, en este caso vamos a utilizar los esquemáticos y conexiones que se encuentran en el drive este será para EOG, vamos a realizar la conexión mediante la tarjeta Cyton y hacemos uso de un canal diferencial.

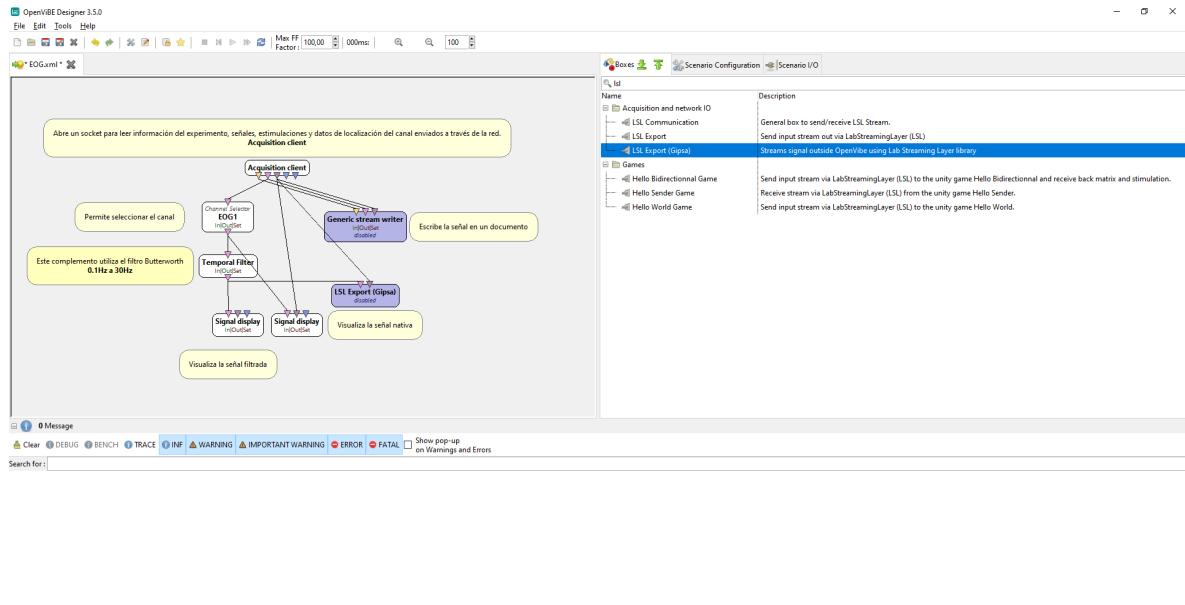


Figura 9. Diagrama de Bloques OpenVibe EOG.

Este es un escenario de OpenVibe, la función de este escenario es servir de intermediario entre la Cyton y por medio del bloque Adquisition client poder recibir la señal del bioamplificador y la función del Channel selector es selecciona el canal, ahora desde esta interfaz podemos manipular esta trama de datos utilizando bloques, ahora podemos filtrar la señal para tener una mejor interpretación del EOG, dandole doble click podemos acceder a las configuraciones y los Signal display sirven de visualizadores, uno muestra la señal nativa y el otro muestra la señal filtrada, ahora podemos deshabilitar los bloques si no se van a usar en este caso tenemos dos deshabilitados por el momento, uno es Generic stream writer este bloque lo que hace es escribir la trama de datos en un documento y guardarla en el lugar que se asigne, estos datos son los correspondientes a las pruebas adquiridas por el Adquisition client cuando se da play al designer, estos datos sirven para después observarlos o hacer pruebas sin el Adquisition client, pruebas simuladas con señales guardadas.

El bloque LSL Export, exporta la señal para manipular la trama de datos en otro programa como puede ser Python, Matlab, etc...

Ahora vamos a realizar una prueba de EOG lo primero es realizar la conexión típica en este caso se tiene en cuenta la siguiente:

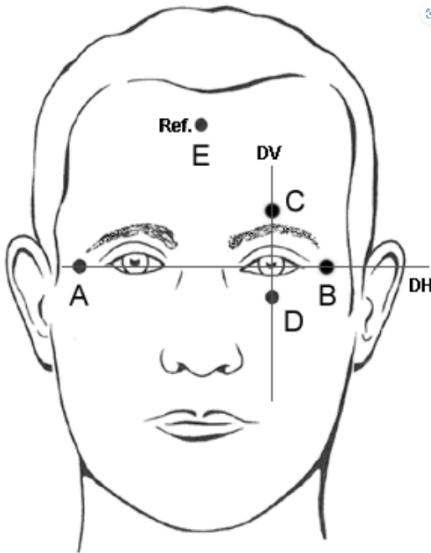


Figura 10. Posición sugerida de los electrodos para EOG [23].

Color electrodo	Cyton Pin	Ubicación del cuerpo	Objetivo
Morado	Pin N1P Inferior	Ojo Derecho (A)	Medición de la diferencia de potencial para el canal 1
Gris	Pin N1P Superior	Ojo Izquierdo (B)	Medición de la diferencia de potencial para el canal 1
Blanco	Pin BIAS	Frente (E)	Referencia

Tabla 1. Conexión electrodos EOG Cyton.

Con esto conectamos los electrodos a la Cyton, no es necesario seguir los colores sugeridos, pero es mejor tener en cuenta siempre la misma conexión para que las pruebas sean prácticamente idénticas, ahora es aconsejable aislar los equipos los cuales están conectados a la Cyton y generen ruido al igual que tocar los equipos que estén conectados a la red eléctrica, limpiar bien las zonas de conexión con alcohol y utilizar cinta para fijar los electrodos y tener una mejor sujeción.

Ahora debemos conectar el Dongle de la Cyton un pequeño USB que es el receptor de la trama de datos y despues entramos a Adquisition Server un programa de OpenVibe, este programa nos permite configurar la Cyton primero que todo vamos a Driver Properties.

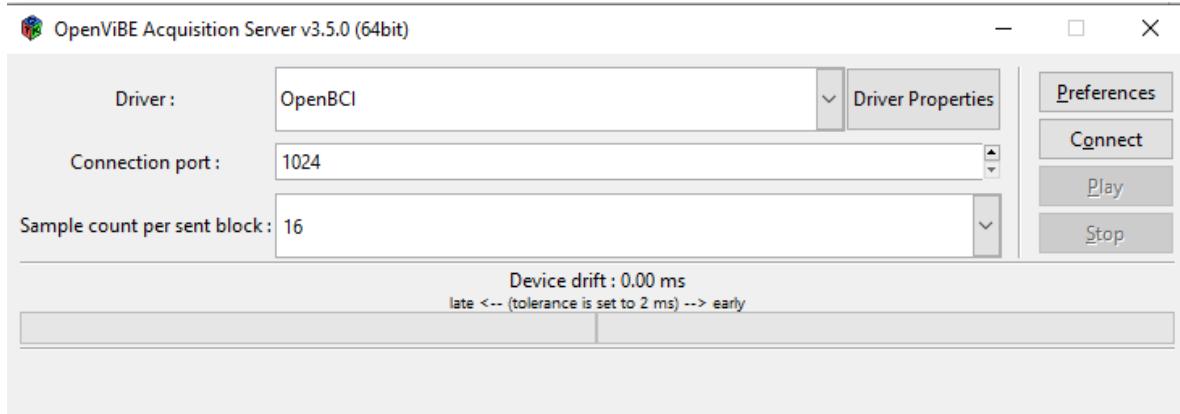


Figura 11. OpenVibe Acquisition Server.

En este apartado podemos seleccionar el puerto en este caso el nuevo COM que se vincula en mi caso el 7 y otras opciones, en esta ventana también podemos configurar los canales, seleccionamos Change channel name y podemos modificar el nombre de cada canal según corresponda

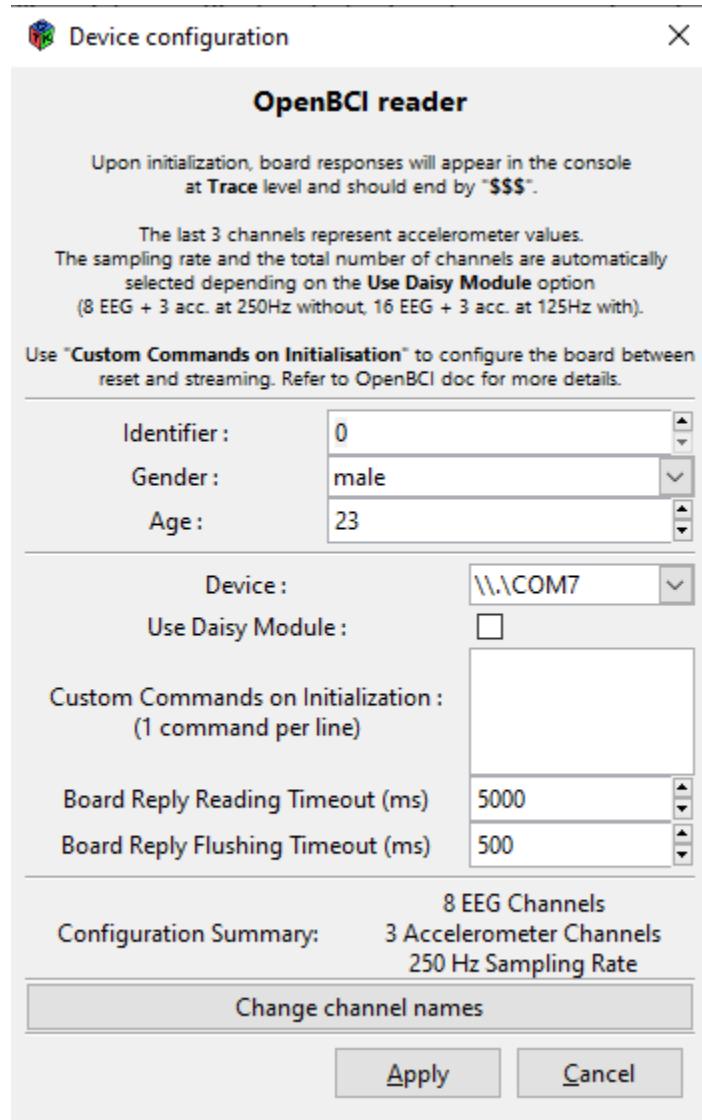


Figura 12. Driver Properties.

Como se va a utilizar el canal 1 a ese canal le asignamos el nombre de EOG1 qué es la conexión que vamos a realizar

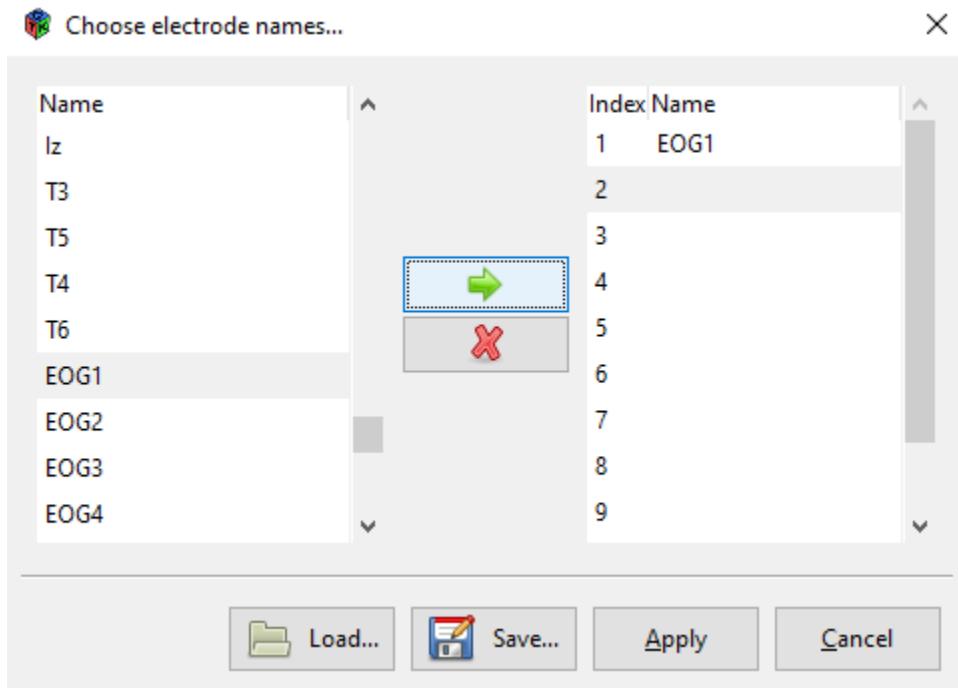


Figura 13. Change Channel Names.

Ya que tengamos el canal configurado podemos seleccionar Apply y procede a energizar la Cyton, en este caso hay varias maneras de energizarla

PYTHON

Nota: SE REQUIERE PYTHON

Nota: El editor de texto que se utilizó para el código python fue visual studio code, tener instalado para después instalar las librerías correspondientes.

Figura 14. Script Python LSL (EOG).

Este es el código propuesto para probar el sistema, al abrir el código de python en visual studio code observa el editor de código y una restricción en donde tendrán que darle permisos a visual studio code para que pueda utilizar correr el script ya sea abriendo el archivo con permisos de administrador o dándole a Manage.

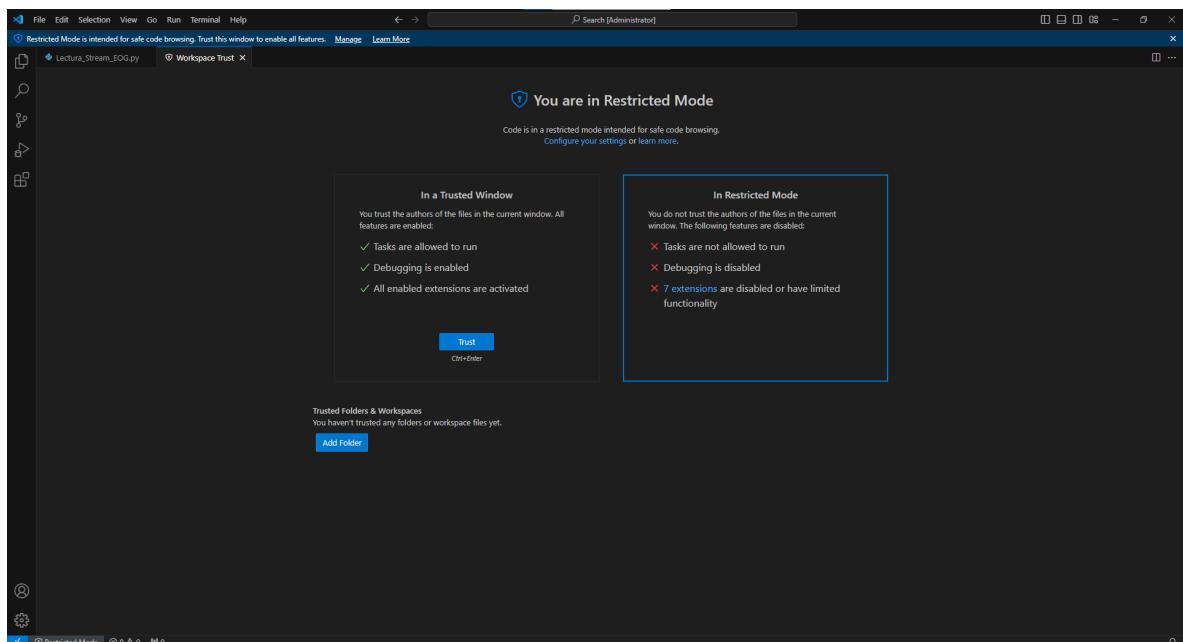


Figura 15. Manage Permisos de Python.

Tendrán que habilitar los permisos seleccionando Trust, esto para que el programa pueda gestionar la información del código y se pueda ejecutar.

Ahora se requiere tener python y agregar una versión para utilizar el script y se requieren instalar las librerías que estan subrayadas en amarillo

The screenshot shows a Windows terminal window titled "Lectura_Stream_EEG.py 3". The code reads EEG streams from a serial port (COM3) at 9600 baud. It defines a StreamInlet object for the EEG stream and then enters a loop to read data. The terminal window also shows the command "python -V" and "pip install pylsl" being run.

```
RUN AND DEBUG
...> RUN
Run and Debug

To customize Run and Debug, open a
folder and create a launch.json file.
Show all automatic debug
configurations.

File Edit Selection View Go Run Terminal Help
F1 > Comunicación > Lectura_Stream_EEG.py > ...
1 from pylsl import StreamInlet, resolve_stream
2 import keyboard
3 import time
4 import serial
5
6 # Configuración del puerto serial
7 serialPort = serial.Serial("COM3", 9600, timeout=1)
8 print("Puerto serial configurado correctamente.")
9
10 # Resolver streams de EEG y EOG
11 print("Buscando streams de EEG y EOG...")
12 stream_EEG = resolve_stream('type', 'EEG')
13
14 # Verificar que se hayan encontrado ambos streams
15 if len(streams_EEG) == 0:
16     raise RuntimeError("No se encontró ningún stream de EEG.")
17
18 # Crear inlets para leer de cada stream
19 inlet_EEG = StreamInlet(streams_EEG[0], max_buflen=5)
20
21 sample_block_size = 32 # Bloques de 32 muestras
22
23 while True:
24     if keyboard.is_pressed('q'):
25         print("Programa terminado por el usuario.")
26         break
27
28     # Leer datos de EEG
29     chunk_EEG, _ = inlet_EEG.pull_chunk(timeout=0.5, max_samples=sample_block_size)
30     if chunk_EEG:
31         for sample_value in chunk_EEG:
32             # Aplicar condiciones a la señal EEG
33             if 400 < sample_value[0] < 450:
34                 serialPort.write(b'D')
35             elif -450 < sample_value[0] < -400:
```

Figura 16. Primera Opción Instalación de Librerías en Terminal Python.

estos lo pueden hacer desde CMD o desde la consola de visual studio code:

```
pip install pylsl  
pip install keyboard  
python -m pip install PySerial
```

```
C:\ Administrador: Símbolo del sistema
Microsoft Windows [Versión 10.0.19045.5131]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\Administrador>pip install pylsl■
```

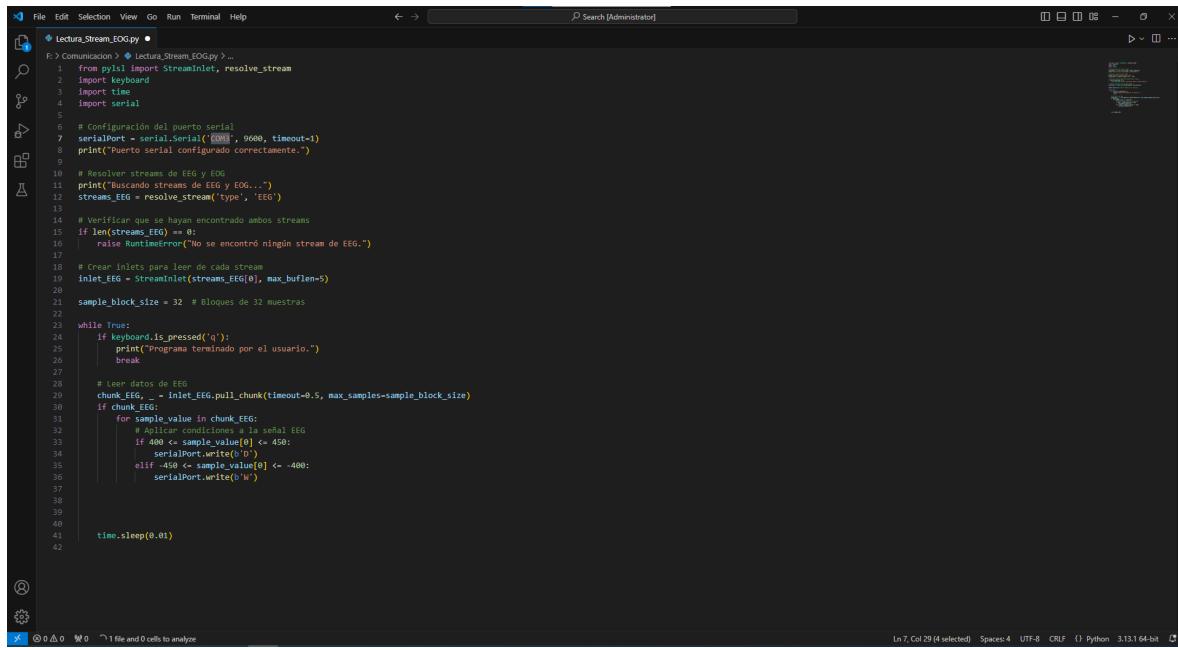
Figura 17. Segunda Opcion Instalacion de librerias en CMD.

Cualquier método es funcional ahora colocamos los codigos para instalar las librerías

```
Python 3.13.1
Collecting pylsl
  Downloading pylsl-1.16.2-py2.py3-none-win_amd64.whl (353 kB)
Installing collected packages: pylsl
  Successfully installed pylsl-1.16.2
PS C:\Users\Administrador> pip install keyboard
Collecting keyboard
  Downloading keyboard-0.13.5-py3-none-any.whl.metadata (4.0 kB)
  Downloading keyboard-0.13.5-py3-none-any.whl (58 kB)
Installing collected packages: keyboard
  Successfully installed keyboard-0.13.5
PS C:\Users\Administrador> pip serial
ERROR: unknown command "serial"
PS C:\Users\Administrador> pip install serial
Collecting serial
  Downloading serial-0.0.97-py2.py3-none-any.whl.metadata (889 bytes)
  Collecting future<=0.17.1 (from serial)
    Downloading future-1.0.0-py3-none-any.whl.metadata (4.0 kB)
  Collecting pyyaml<3.13 (from serial)
    Downloading pyyaml-5.1-py3-none-any.whl.metadata (4.8 kB)
    Downloading future-1.0.0-py3-none-any.whl.metadata (4.0 kB)
    Downloading future-1.0.0-py3-none-any.whl.metadata (4.0 kB)
  Collecting pyyaml<=3.13 (from serial)
    Downloading PyYAML-6.0.2-cp313-cp313-win_amd64.whl.metadata (2.1 kB)
  Collecting iso8601>=1.12 (from serial)
    Downloading iso8601-2.1.0-py3-none-any.whl.metadata (3.7 kB)
  Downloading serial-0.0.97-py2.py3-none-any.whl (48 kB)
  Downloading future-1.0.0-py3-none-any.whl (491 kB)
  Downloading iso8601-2.1.0-py3-none-any.whl (7.5 kB)
  Downloading PyYAML-6.0.2-cp313-cp313-win_amd64.whl (156 kB)
Installing collected packages: pyyaml, iso8601, future, serial
Successfully installed future-1.0.0 iso8601-2.1.0 pyyaml-6.0.2 serial-0.0.97
PS C:\Users\Administrador>
```

Figura 18. Descarga de Librerías Finalizado.

Podemos cerrar visual studio para actualizar las librerías las cuales se acaban de descargar, despues de iniciar nuevamente el programa no deberian aparecer subrayado en amarillo las primeres lineas



```
File Edit Selection View Go Run Terminal Help ⏎ ⏎ Search [Administrador]
F2 Comunicacion > Lectura_Stream_EOG.py ...
1 from pylsl import StreamInlet, resolve_stream
2 import keyboard
3 import time
4 import serial
5
6 # Configuración del puerto serial
7 serialPort = serial.Serial('COM1', 9600, timeout=1)
8 print("Puerto serial configurado correctamente.")
9
10 # Resolver streams de EEG y EOG
11 print("Buscando streams de EEG y EOG...")
12 streams_EEG = resolve_stream('type', 'EEG')
13
14 # Verificar que se hayan encontrado ambos streams
15 if len(streams_EEG) == 0:
16     raise RuntimeError("No se encontró ningún stream de EEG.")
17
18 # Crear inlets para leer de cada stream
19 inlet_EEG = StreamInlet(streams_EEG[0], max_buflen=5)
20
21 sample_block_size = 32 # Bloques de 32 muestras
22
23 while True:
24     if keyboard.is_pressed('q'):
25         print("Programa terminado por el usuario.")
26         break
27
28     # Leer datos de EEG
29     chunk_EEG_ = inlet_EEG.pull_chunk(timeout=0.5, max_samples=sample_block_size)
30     if chunk_EEG_:
31         for sample_value in chunk_EEG_:
32             # Aplicar filtro pasa bajos a la señal EEG
33             if 400 < sample_value[0] < 450:
34                 serialPort.write(b'D')
35             elif -450 < sample_value[0] < -400:
36                 serialPort.write(b'W')
37
38
39
40
41     time.sleep(0.01)
```

Figura 19. Verificación del Puerto COM Arduino Emisor.

ya estaría listo para empezar las pruebas con el sistema, algo a considerar es tener en cuenta el COM ya que este se comunica con el arduino entonces tiene que ser el mismo, para saber que COM se esta conectando

Para verificar el puerto COM de un Arduino conectado, se puede usar el Administrador de dispositivos de Windows:

1. Presionar Win + X para abrir el menú de Usuario avanzado
2. Seleccionar Administrador de dispositivos
3. Dirigirse a la sección de puertos COM y LPT
4. Desplegar el árbol para verificar el puerto de conexión del Arduino.

Si tiene muchos puertos COM puede desconectar y conectar el Arduino para verificar que puerto COM se está conectando.

PRUEBA DEL SISTEMA

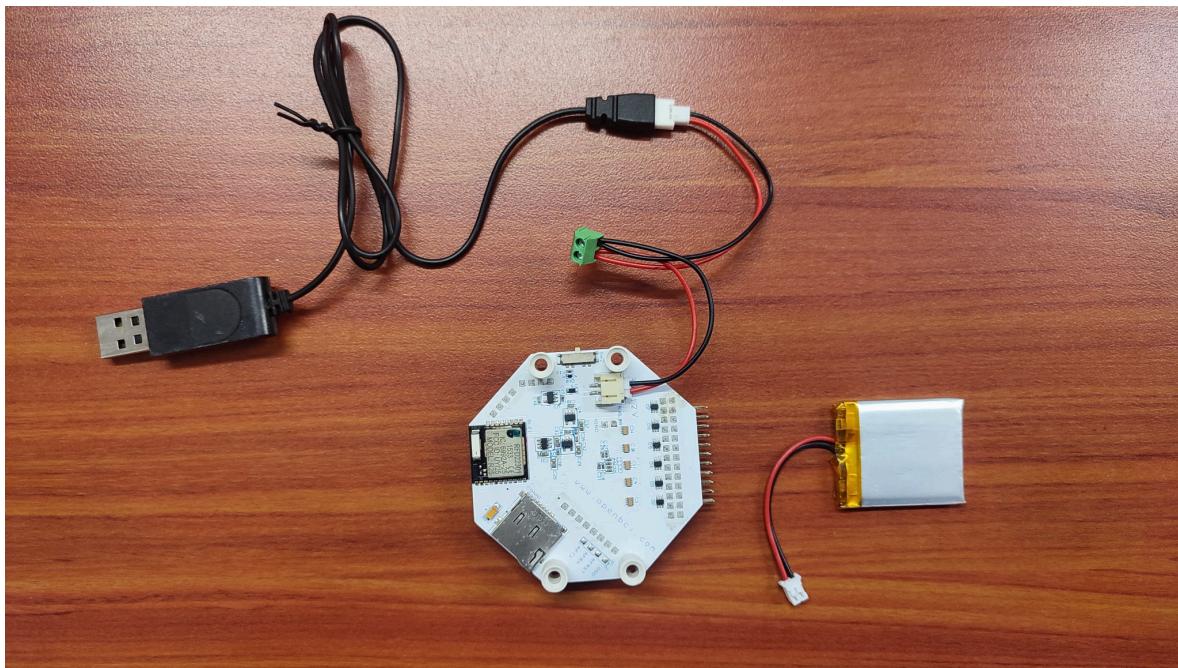


Figura 20. Tipos de Alimentación de la Cyton.

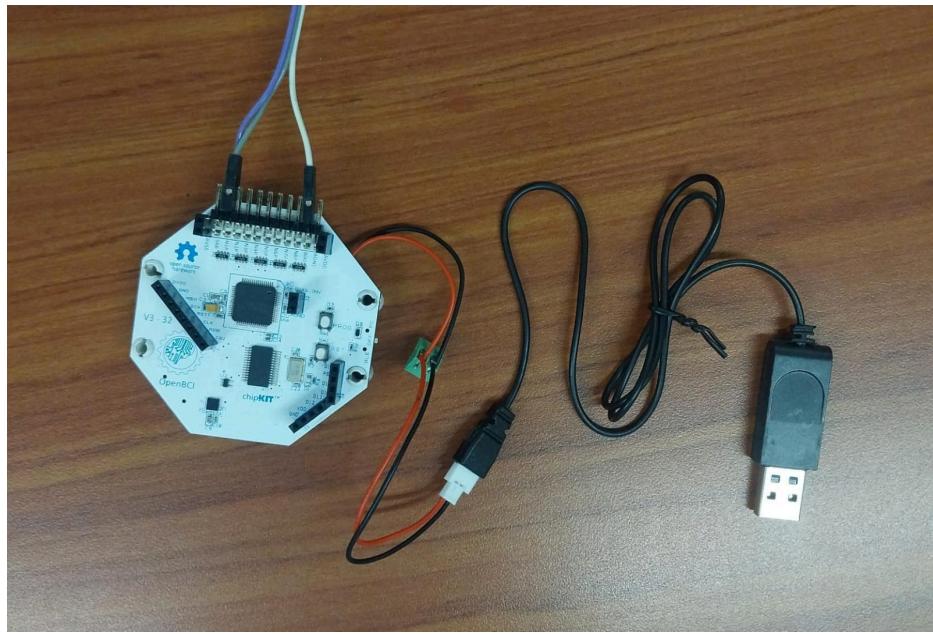


Figura 21. Conexión de los Electrodos Cyton.

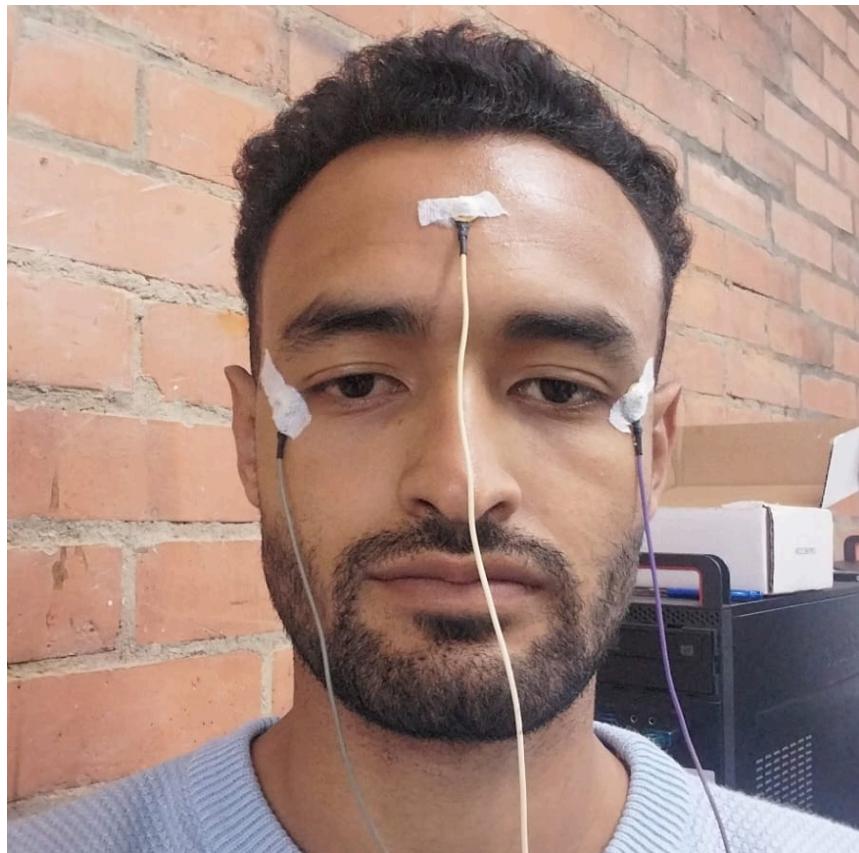


Figura 22. Conexión EOG.



Figura 23. Alimentación de la Cyton y Conexión del Dongle.



Figura 24. Conexión Emisor Completa EOG.

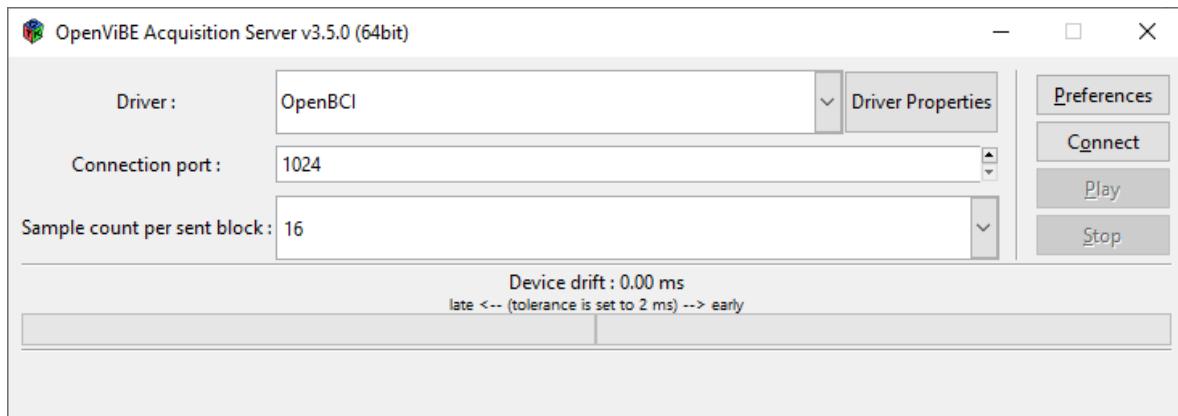


Figura 25. OpenVibe Acquisition Server.

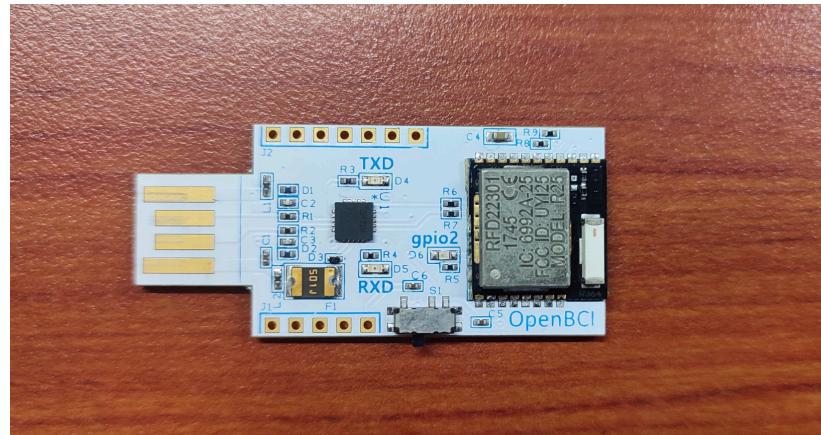


Figura 26. Conexión del Dongle.

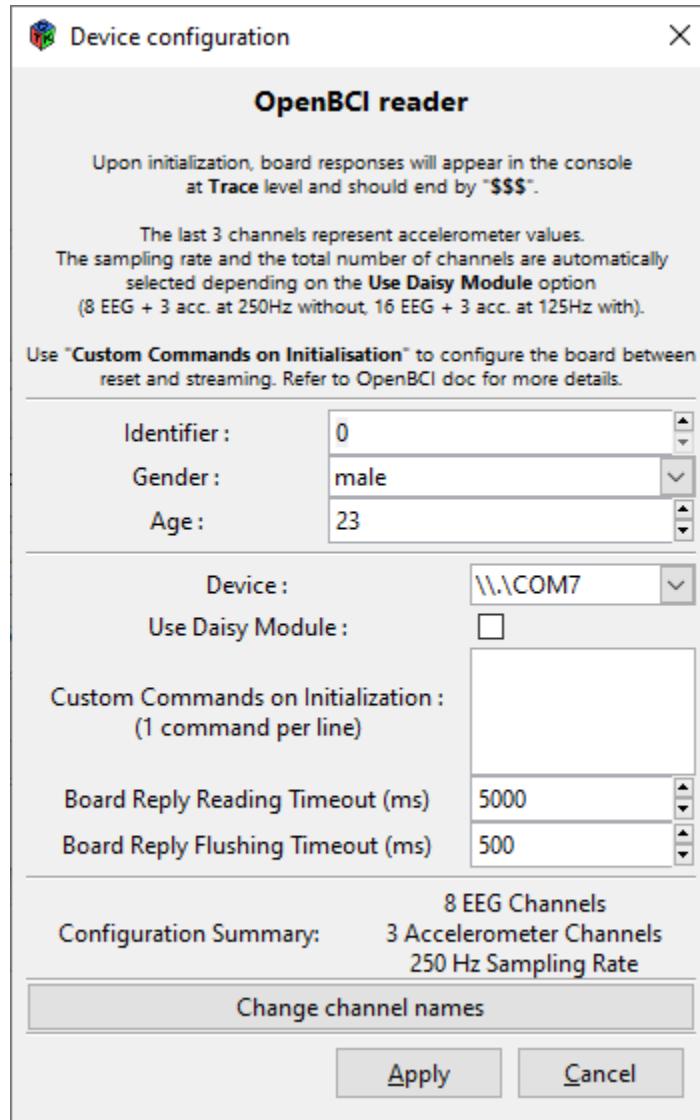


Figura 27. Configuración del Puerto COM del Dongle.

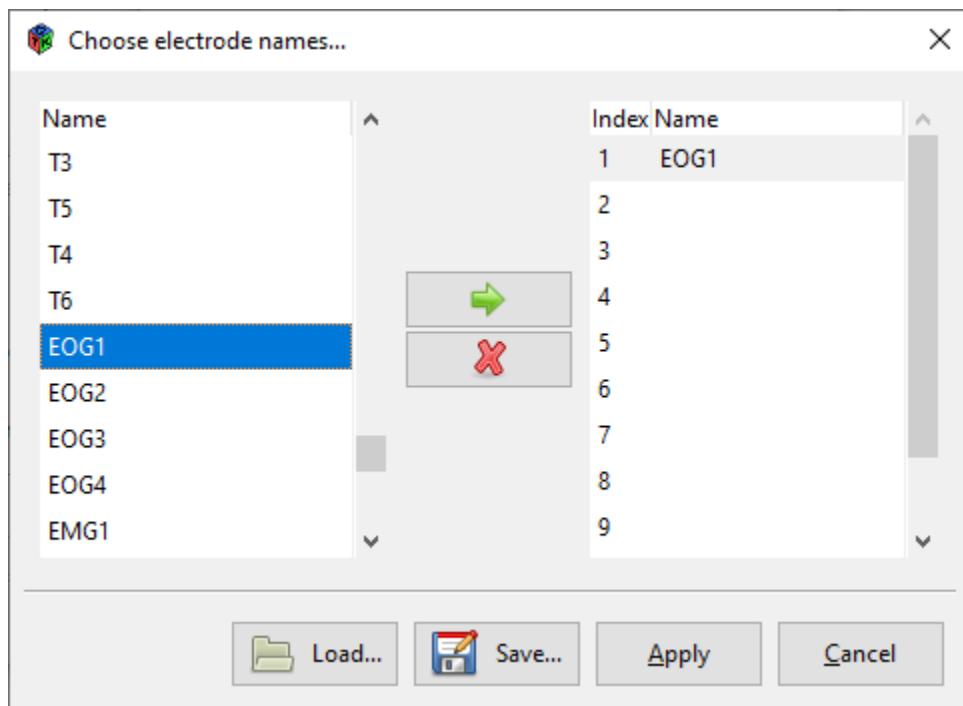
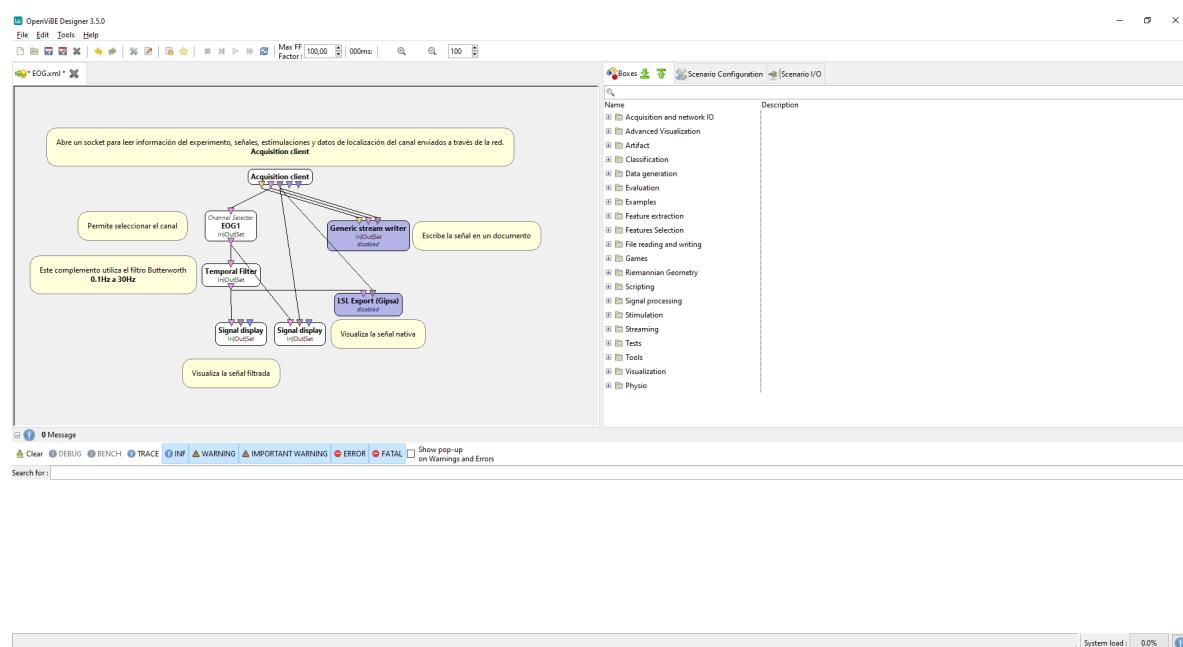
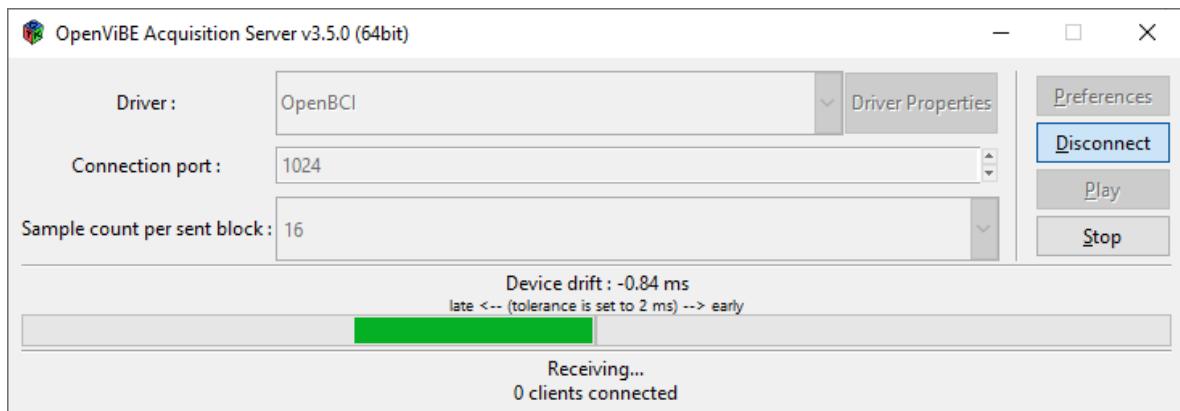
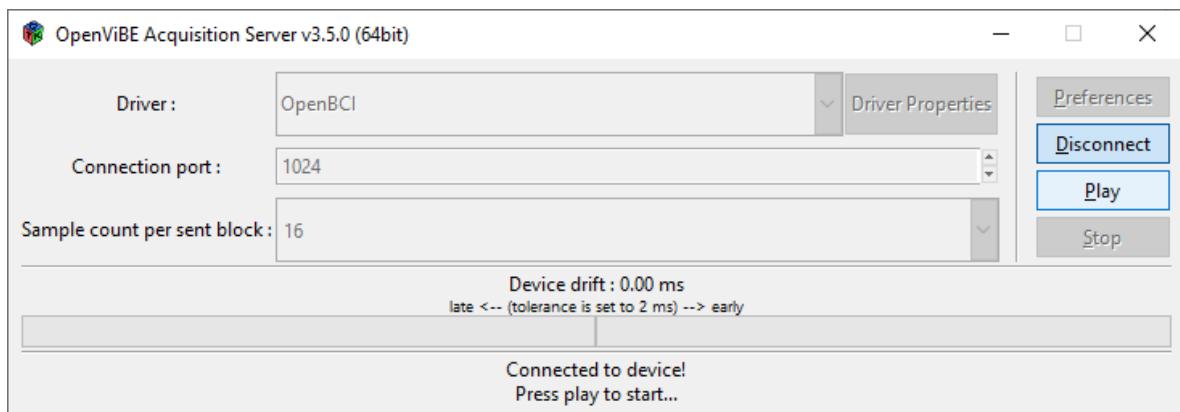
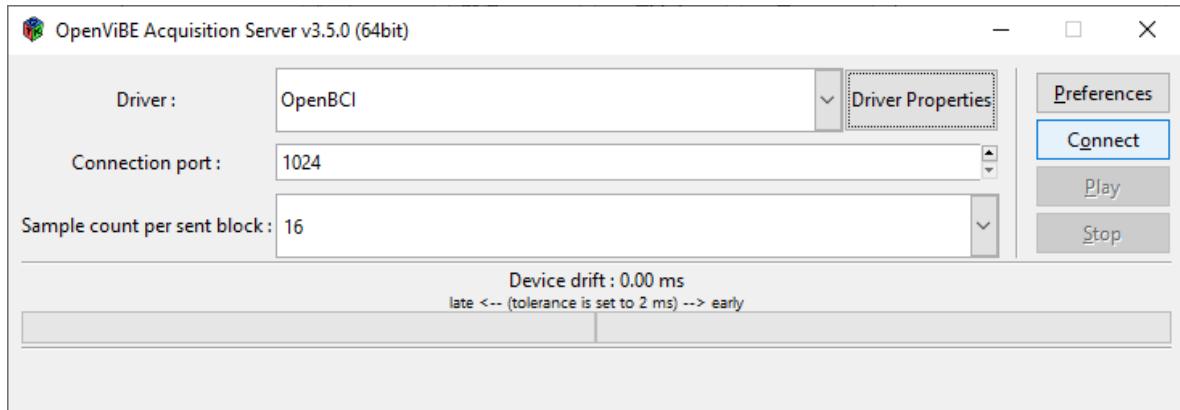
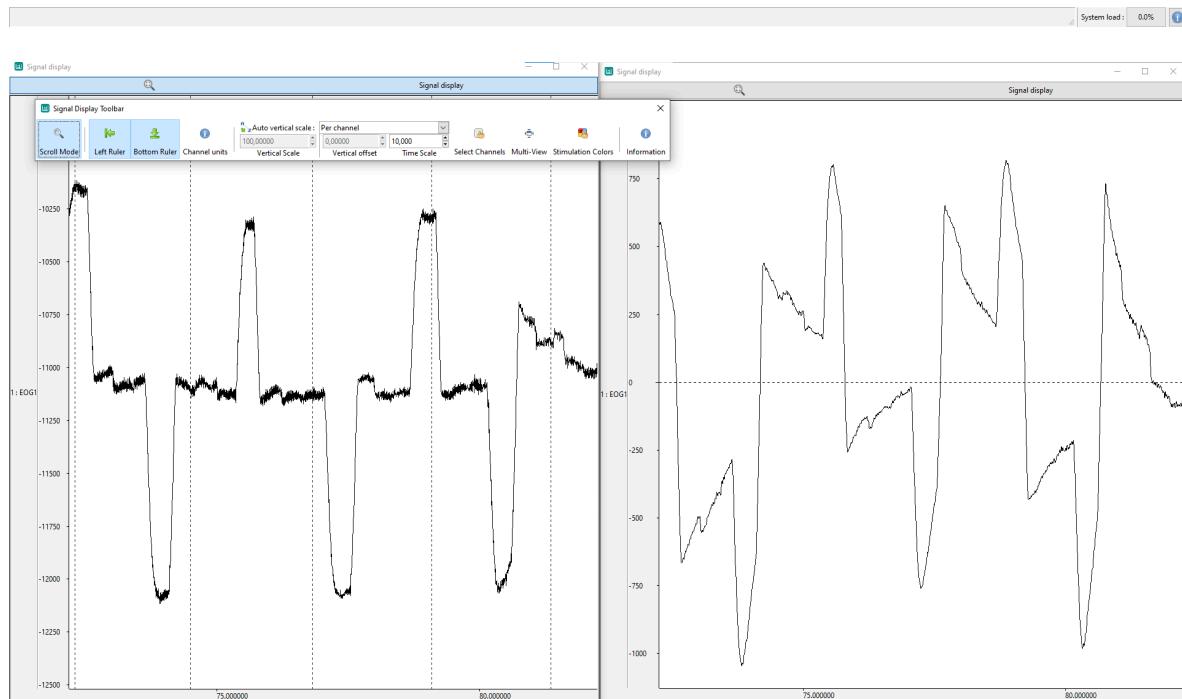
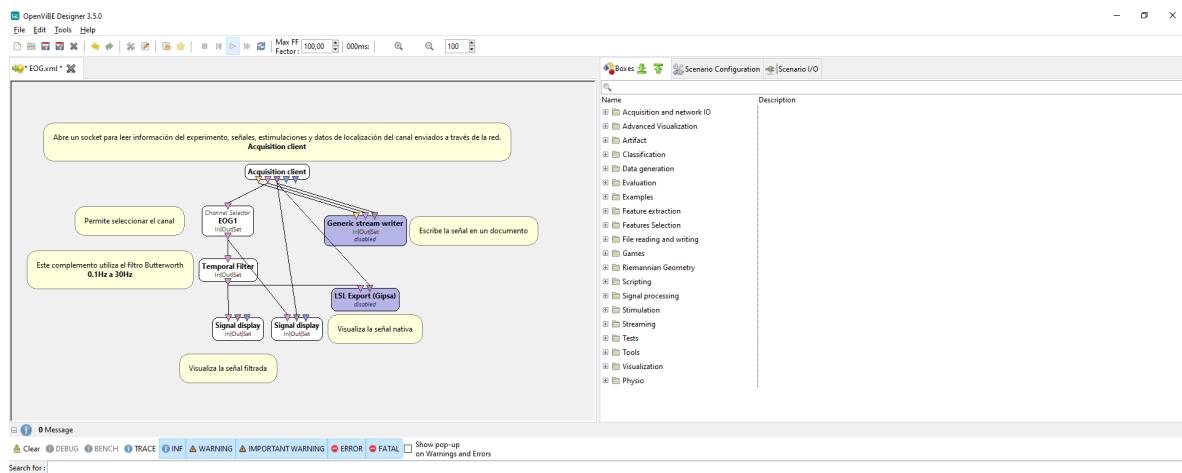
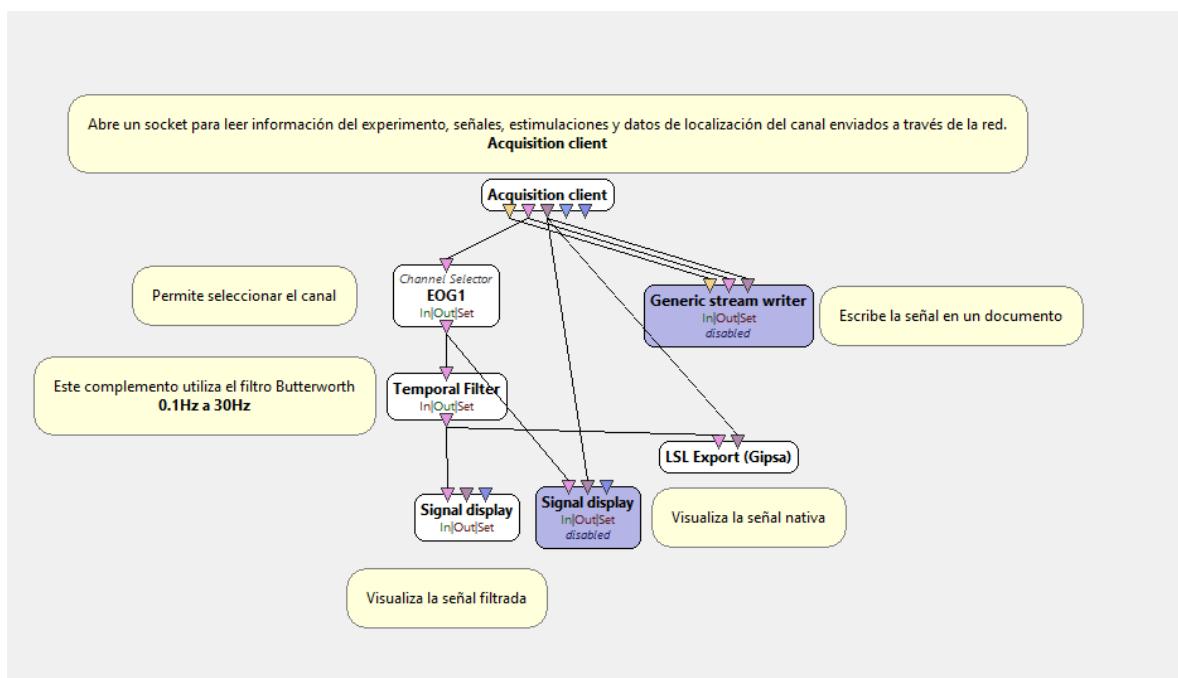


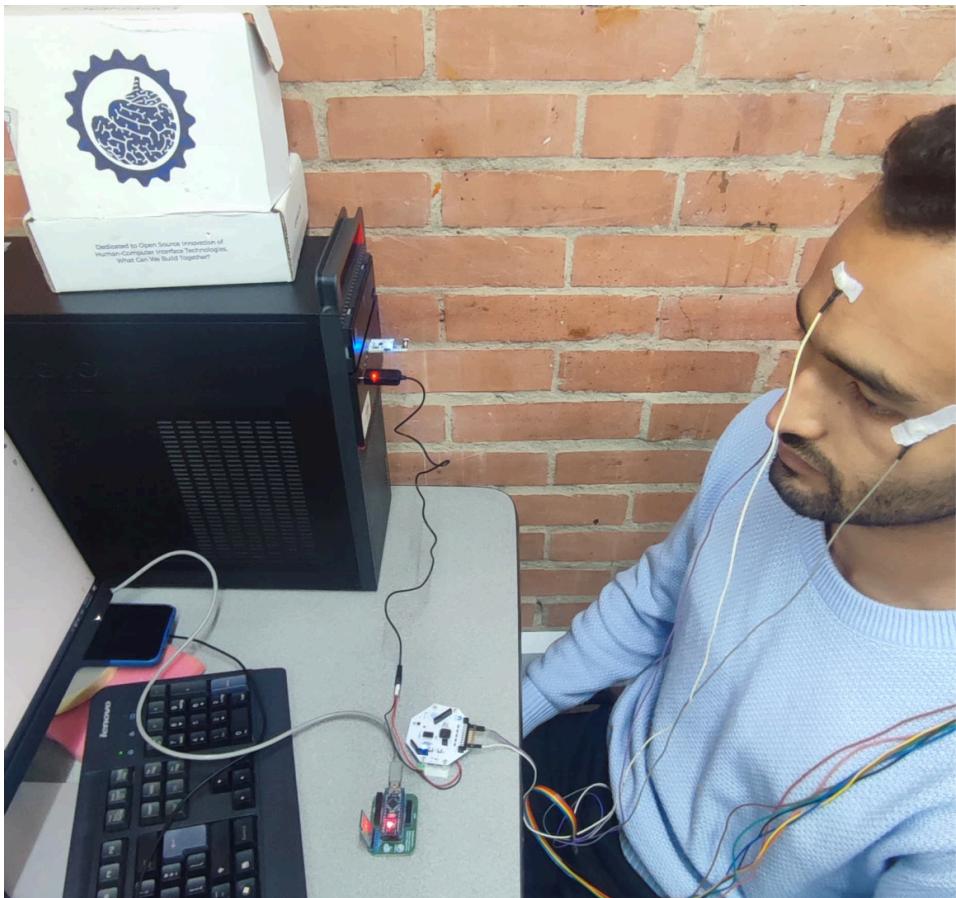
Figura 28. Configuración del Canal Utilizado Cyton.

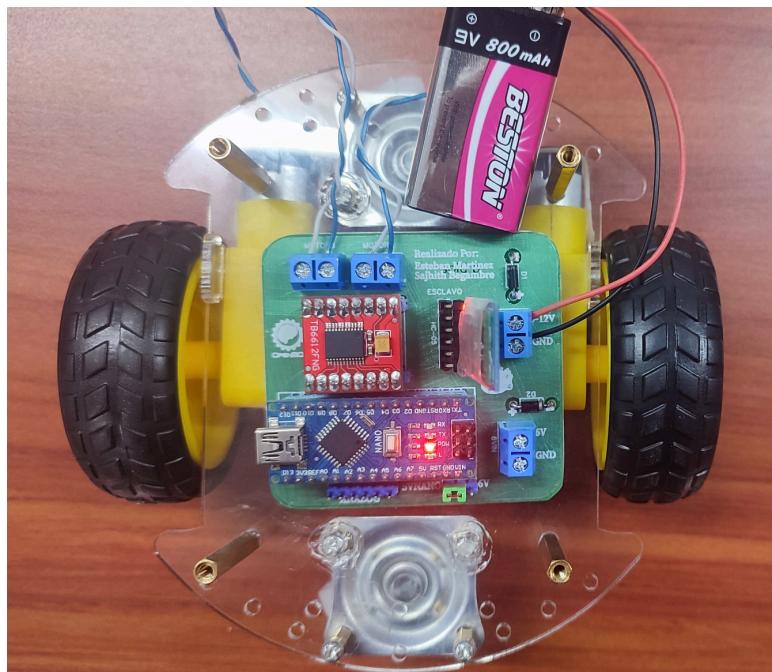
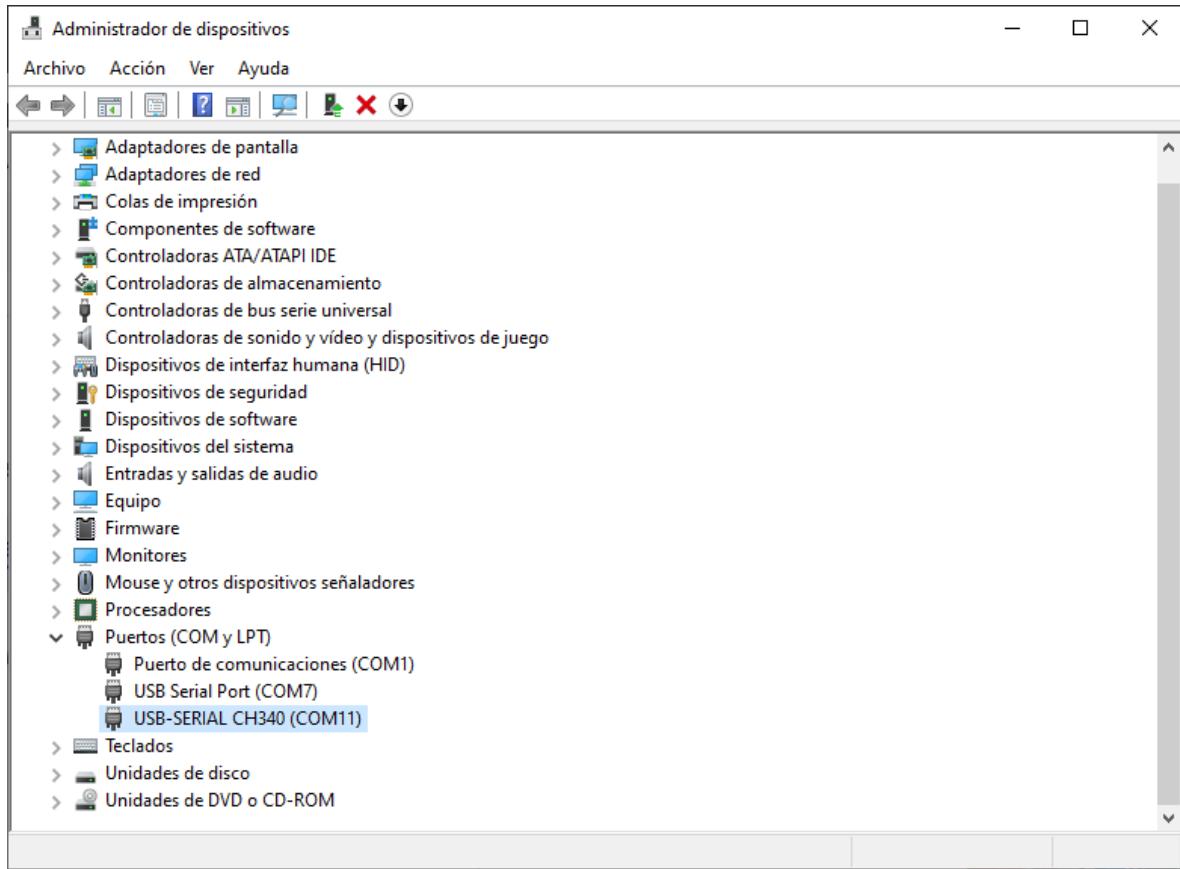


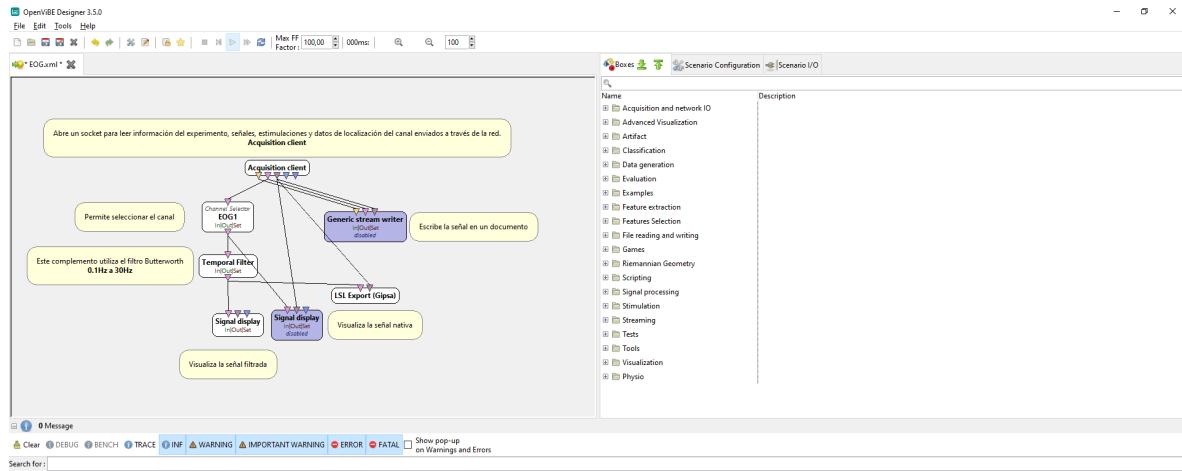












R: > Comunicacion > Lectura_Stream_EOG.py ...

```

File Edit Selection View Go Run ... ← → ⌂ Search [Administrator] ⌂ System load: 0.0% ⓘ
Run Python File

Lectura_Stream_EOG.py ●
F: > Lectura_Stream_EOG.py > ...
1 from pylsl import StreamInlet, resolve_stream
2 import keyboard
3 import time
4 import serial
5
6 # Configuración del puerto serial
7 serialPort = serial.Serial('COM11', 9600, timeout=1)
8 print("Puerto serial configurado correctamente.")
9
10 # Resolver streams de EEG y EOG
11 print("Buscando streams de EEG y EOG...")
12 streams_EEG = resolve_stream('type', 'EEG')
13
14 # Verificar que se hayan encontrado ambos streams
15 if len(streams_EEG) == 0:
16     raise RuntimeError("No se encontró ningún stream de EEG.")
17
18 # Crear inlets para leer de cada stream
19 inlet_EEG = StreamInlet(streams_EEG[0], max bufsize=5)
20
21 sample_block_size = 32 # Bloques de 32 muestras
22
23 while True:
24     if keyboard.is_pressed('q'):
25         print("Programa terminado por el usuario.")
26         break
27
28     # Leer datos de EEG
29     chunk_EEG, _ = inlet_EEG.pull_chunk(timeout=0.5, max samples=sample_block_size)
30     if chunk_EEG:
31         for sample_value in chunk_EEG:
32             # Aplicar condiciones a la señal EEG

```

Run Python File

0 0 0 1 file and 0 cells to analyze

Ln 7, Col 29 (5 selected) Spaces:4 UTF-8 CRLF {} Python 3.13.1 64-bit

