I + T + C Investigación, Tecnología y Ciencia **ISSN: 1909-5775** **e-ISSN: 2805-7201**

Artículo de Investigación

Implementation of Open-Source Tools with Brain-Computer Interfaces

Implementación De Herramientas Open-Source Con Interfaces Cerebro-Computador

Daniel Eduardo Estevez Cárdenas [1], Nayibe Chio Cho [2*], Johann Barragán Gómez [3]

[1] Universidad Autónoma de Bucaramanga; destevez@unab.edu.co
[2] Universidad Autónoma de Bucaramanga; nchio@unab.edu.co
[3] Universidad Autónoma de Bucaramanga; jbarragan262@unab.edu.co

* Correspondencia: nchio@unab.edu.co

Citación: Estevez, D .; Chio, N .; Barragán, J. Implementación De Herramientas Open-Source Con Interfaces Cerebro-Computador. I + T + C Investigación, Tecnología y Ciencia. Vol 1. Num. 17. 2023.

Nota del editor: Sello editorial Unicomfacauca se mantiene neutral con respecto a los reclamos derivados de los resultados de este trabajo.

Resumen: El continuo avance de las tecnologías es un hecho innegable en la actualidad, los cambios generados en la medicina, el sector automotriz, la educación, se presentan cada vez con mayor frecuencia. Una de las tecnologías que resalta en estos últimos años es la relacionada con el análisis de las señales cerebrales, su clasificación y utilización o también llamadas y conocidas como 'Brain-Computer Interfaces' (BCI). Pese al gran avance en esta tecnología sus aplicaciones están centradas mayoritariamente en el área de salud, más específicamente en la rehabilitación de pacientes, y es por esta misma razón que se generó este proyecto, el cual tiene el objetivo expandir el área de utilización de esta tecnología, esto mediante el desarrollo de un sistema de comunicación y manejo a través de señales cerebrales de un dispositivo externo, todo a su vez siendo implementado en softwares open source. Este proceso se generó por medio de tres etapas principales: lectura de señales, procesamiento/clasificación, y control del dispositivo externo. Para llevar a cabo el proyecto se hizo uso del dispositivo 'MindWave Headset' de NeuroSky para lectura de señales cerebrales, el software 'Open ViBE' para el procesamiento y clasificación de señales, y los softwares 'Python', 'Arduino IDE' y la placa 'Arduino UNO' para el control de la representación de un dispositivo externo. Como producto de todo el proceso del proyecto se obtuvo un sistema completamente funcional y diseñado a partir de softwares y herramientas open source, capaz de controlar un dispositivo simple mediante una interfaz cerebro-computador. Este producto logra actuar como base para seguir avanzando en nuevos procesos implementando estas tecnologías.

Palabras clave: BCI; NeuroSky; Mindwave; Señales cerebrales; OpenViBE; Control; Arduino; Python; Open Source.

Abstract: The continuous advancement of technologies is an undeniable fact in today's world, with changes occurring more frequently in fields such as medicine, the automotive industry, and education. One of the technologies that has stood out in recent years is related to the analysis of brain signals, their classification, and utilization, often referred to as 'Brain-Computer Interfaces' (BCI). Despite significant progress in this technology, its applications are primarily focused on the healthcare sector, specifically in the rehabilitation of patients. It is for this reason that this project was initiated, with the goal of expanding the use of this technology into other areas. This is achieved through the development of a communication and control system using brain signals to interact with an external device, all implemented through open-source software. This project involved three main stages: signal reading, processing/classification, and control of the external device. To carry out the project, the 'MindWave Headset' from NeuroSky was used for reading brain signals, 'Open ViBE' software for signal processing and classification, and 'Python,' 'Arduino IDE,' and the 'Arduino UNO' board for controlling the external device's representation. As a result of the entire project process, a fully functional system was obtained, designed using open-source software and tools, capable of controlling a simple device through a brain-computer interface. This product serves as a foundation for further advancements in implementing these technologies into new processes.

Keywords: BCI; NeuroSky; Mindwave; Brain Signals; OpenViBE; Control; Arduino; Python; Open Source.

1. Introducción

El documento presentado a continuación recolecta los resultados obtenidos a lo largo de la investigación, tratando a profundidad los procesos de capacitación y familiarización con los softwares (Python, OpenViBE, Arduino IDE) y dispositivos utilizados (Mindwave Headset 2 y Arduino UNO), además del proceso de diseño de BCI, con la creación de medios para la adquisición de los datos, su procesamiento y análisis, clasificación de señales, y finalmente, el control del dispositivo externo. De igual manera a lo largo del documento se evidenciará los escenarios diseñados y utilizados para la adquisición de datos, entrenamiento del clasificador, prueba online y prueba de datos de manera offline, además de las funcionalidades de los diferentes softwares en el proceso de comunicación y control de la representación del dispositivo externo, al igual que el montaje físico de toda la implementación.

Las diferentes partes del proyecto fueron establecidas y logradas teniendo como referente varios estudios e investigaciones realizadas por otros actores, sin embargo, se resaltan dos proyectos en especial, primero la "Implementación de interfaz cerebro-máquina para el control de movimiento de un brazo manipulador robótico UR3 para aplicaciones de pick and place" [1], proyecto de grado del cual se obtuvieron las bases en cuanto a objetivos y softwares necesarias para realizar la conexión y comunicación en conjunto entre Python y Arduino, comunicación utilizada en el control del dispositivo , y las bases conceptuales y teóricas requeridas para el análisis de señales cerebrales su interpretación y manejo. Y como segundo proyecto "Design of classification methods for Brain Computer Interfaces: An application using the OpenViBE software" [2], en base al cual se logró definir las estrategias de clasificación y análisis de las señales cerebrales EEG, más óptimas para implementar en base a los requerimientos propios.

Como convergencia de todos los procesos se resultó en un proyecto referente para futuras aplicaciones de BCI en control de dispositivos eléctricos, electrónicos, y a grandes rasgos, en multitud de aplicaciones de ingeniería.

2. Materiales y métodos

2.1. Materiales

Los materiales utilizados para la realización del proyecto de investigación se dividen en dos grupos, los softwares implementados para la creación de BCI, escenarios y los códigos necesarios para interconectar todo, y los implementos físicos utilizados para realizar la implementación real. A continuación, se pasará a profundizar en los elementos que componen estos dos grupos.

2.1.1. Software

- OpenViBE (V 3.4.0)

Como principal software de trabajo se encuentra OpenViBE, el cual puede ser definido como *"una plataforma de software libre y de código abierto para el diseño, prueba y uso de interfaces cerebro-computadora, la cual consta de un conjunto de módulos de software que se pueden integrar de manera fácil y eficiente para diseñar BCI, para aplicaciones reales y de realidad virtual"* [3]. Este software, como la descripción anterior indica, fue utilizado para la creación de los diferentes escenarios requeridos para los procesos de adquisición y procesamiento de los datos, y fue elegido por su facilidad de uso, dado que cuenta con programación de procesos por medio de bloques de trabajo y ejemplos explicativos, como por su licencia libre.

- Python (V 3.11)

Mas que un software Python, fue el lenguaje de programación mediante el cual se logró y realizo todo el proceso de comunicación desde el software 'OpenViBE' hasta la tarjeta Arduino UNO. El código creado fue generado dentro del software 'Visual Code' (V 1.81), el cual cumple la funcionalidad de ser un editor de código fuente, con herramientas como soporte de depuración y control integrado de Git, lo que facilita la creación, retroalimentación y corrección de códigos [4].

- Arduino IDE (V 1.8.57.0)

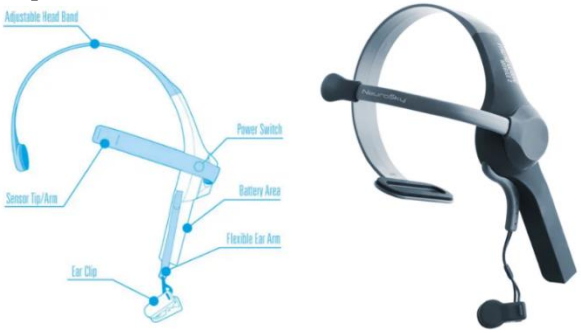
De último recurso se hizo uso de Arduino IDE, software por medio del cual se realizó el código de comportamiento del Arduino, estableciendo todos los procesos que debía realizar internamente y el control de los dispositivos eléctricos que debía ejercer. Además de ser el medio de creación de código, también fue el medio para cargar el código en la tarjeta. Es el elemento final tanto en el proceso de comunicación, como de funcionamiento en conjunto del proyecto.

2.1.2. Hardware

- NeuroSky MindWave Mobile Headset

"El MindWave Mobile 2 de NeuroSky, es una diadema auricular EEG que mide y transfiere con seguridad los datos del espectro de potencia (ondas alfa, ondas beta, etc.) mediante Bluetooth Low Energy (BLE) o Bluetooth Classic para comunicarse de forma inalámbrica con su dispositivo receptor. Consiste únicamente en un auricular con diadema en forma de Tun clip para la oreja más amplio y un brazo sensor flexible. Los electrodos de referencia y de tierra del auricular están en el clip de la oreja, mientras que el electrodo de EEG se encuentra en el brazo del sensor flexible, descansando en la frente" [5].

Así como es indicado en la referencia anterior, el dispositivo MidWave es el medio de adquisición de datos de todo el proceso, por medio del electrodo que maneja en su brazo flexible logra captar los potenciales de las señales eléctricas y transmitirlas por medio de la comunicación Bluetooth al equipo de trabajo y al software OpenViBE. El dispositivo maneja detección de tres diferentes estados del usuario, concentración, relajación y parpadeo, este último siendo la detección trabajada para todo el proceso. En base a la detección de los momentos de parpadeo y descanso del usuario se logró crear el sistema de control del dispositivo externo.



A B

Figura 1. (A) Partes Neurosky MindWave Mobile Headset; (B) Neurosky MindWave Mobile Headset. Extraído de Neurosky. (2023). MindWave Mobile 2 [6]

- Arduino UNO

Como fue indicado anteriormente, la placa Arduino es el elemento final de todo el proceso, siendo el dispositivo al que llega la información interpretada de las señales cerebrales y mediante la cual se realiza el control de los dispositivos eléctricos.

Fue elegido tanto por su microcontrolador ATmega328P, como por la distribución de 14 pines de entrada/salida digital que maneja (de los cuales 6 pueden ser usando con PWM), 6 entradas analógicas, además de los tipos de conexiones que posee, conexión USB, conector jack de alimentación, terminales para conexión ICSP y un botón de reseteo [7].

2.2. Metodología

La metodología utilizada para la creación de este proyecto se puede dividir en cuatro fases principales, investigación, interacción, diseño e implementación.

2.2.1. Investigación

La fase de investigación, como su nombre lo indica, encierra todas las actividades de "research" realizadas para entender el funcionamiento de los softwares y de los dispositivos a manejar, el actuar del dispositivo de entrada (Headset Mindwave – Neurosky), y el dispositivo de salida (Arduino UNO), al igual que de los softwares, OpenViBE, Arduino IDE y Python. Para esta etapa se investigó, además de los proyectos mencionados anteriormente en el apartado de introducción, proyectos específicamente relacionados con el uso de OpenViBE en la creación de BCI y las interpretaciones de señales dentro del mismo software para el control de dispositivos, como lo es el proyecto de "Diseño de un sistema de guiado de robots mediante la adquisición de señales EEG" [8], en el cual se diseñó esta interfaz con el objetivo de poder controlar, basándose en señales EEG, un robot industrial de seis grados de libertad.

De igual manera en el transcurso de realización del proyecto de investigación, dentro de la institución educativa se estaban desarrollando proyectos similares en los que estaban involucrados los autores de estos proyectos y en los cuales se pudo apoyar la investigación.

A partir de esta etapa fue posible la selección de detección parpadeo como acción de control del dispositivo externo, lo cual a su vez dio las directrices para la creación de los escenarios de adquisición y procesamiento de datos dentro de OpenViBE.

2.2.2. Interacción

Ya realizada la fase de investigación y poseer las bases teóricas del funcionamiento de dispositivos, y softwares, se continúan con las actividades de interacción con los mismos, realizando la capacitación dentro de OpenViBE, generada de manera independiente por medio del estudio de los ejemplos incluidos y descargados con el software, interactuando con los bloques y diferentes funcionalidades que los mismo poseen. De la misma manera se realizó la capacitación en la manipulación de los dispositivos de entrada estudiando e interaccionando con las aplicaciones desarrolladas para el Mindwave Headset encontradas en el store oficial de NeuroSky [9], al igual que por medio de pruebas de adquisición de datos dentro de OpenViBE y el entendimiento del proceso de detección de parpadeo, realizando la conexión entre los dispositivos.

Todo este proceso se generó con el fin de tener unas bases más sólidas sobre el comportamiento de todos estos sistemas.

2.2.3. Diseño

Como tercera fase del desarrollo se encuentra la fase de diseño, ya conociendo como se interactúan con las BCI y de que componentes deben ser generadas, se puede seguir a

diseñar una que cumpla con los requisitos del proyecto en específico, como la diferenciación de las señales cerebrales captadas por EEG en el rango de frecuencia Alpha y Beta (8-40HZ) y la creación de un sistema de presentación de estímulos al usuario personalizado.

El diseño de esta interfaz transcurrió en su mayoría dentro del software OpenViBE y consistió en la creación de 4 escenarios de trabajo, escenario de adquisición de datos, de entrenamiento del clasificador, de prueba online, y de prueba de datos offline, con los datos capturados de la prueba online y envió de información de manera externa. En estos escenarios se profundizará en la sección de resultados.

2.2.4. Implementación

Tras todos los anteriores procesos se llega a la culminación del proyecto con la última fase. Ya habiendo creado la BCI deseada y haber realizado todas las conexiones tanto con los dispositivos de entrada como con el de salida, se puede proceder a implementar el proyecto completo, comprobando en vivo el funcionamiento del mismo, sus alcances y realizando las correcciones pertinentes de ser necesarios.

La implementación final se realiza de la mano del último escenario generado en OpenViBE, "Prueba Datos", y por medio de comunicación TCP/IP, y de una serie de LEDs indicativos, cuyo montaje con la tarjeta Arduino únicamente requirió de la conexión directa del positivo de los mismos a los puertos 3,4 y 5 y del negativo al ground. Esta conexión puede ser observada en la figura 2.

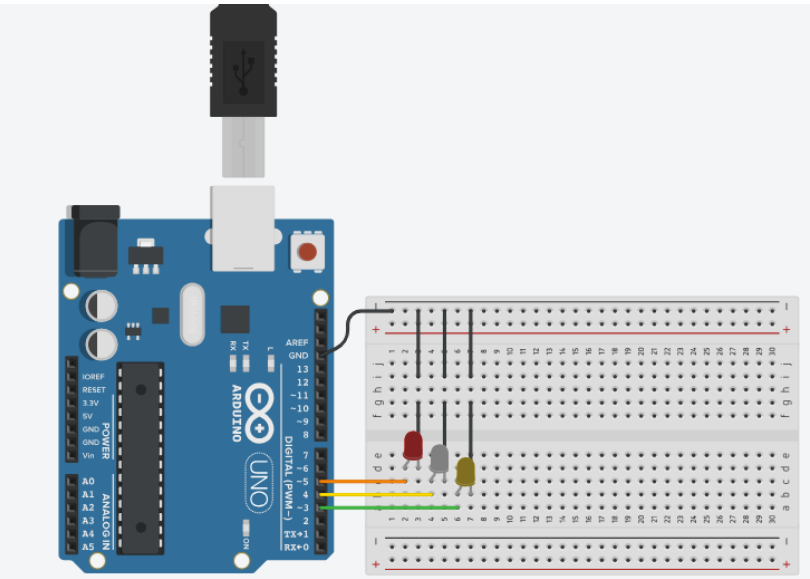


Figura 2. Esquemático Conexión Tarjeta Arduino. Autores

En consecuencia, a que todos los procesos de comunicación y transmisión de datos se realizan automáticamente, gracias al proceso de diseño, solo fue necesario al momento de la implementación cargar el código a la tarjeta Arduino UNO por medio de Arduino IDE, y ejecutar el escenario correspondiente dentro de OpenViBE y el código de Python.

3. Resultados

El producto del proceso de investigación se puede dividir en dos apartados diferentes, los escenarios y programas realizados en los diferentes softwares, y la implementación en físico de la interconexión entre softwares del control del dispositivo externo.

3.1 Escenarios y Programas Realizados

3.1.1 Escenarios OpenViBE

Dentro de OpenViBE se realizaron 4 escenarios como fue mencionado anteriormente, uno para la adquisición de datos, uno para el entrenamiento del clasificador, uno de prueba online, y el ultimo para verificación offline de los datos.

Estos escenarios cumplen diferentes objetivos, como sus nombres lo indican, capturar/adquirir los datos del usuario mediante el simulador de estímulos, entrenar el clasificador de señales por medio de los datos captados, comprobar de modo online el proceso del clasificador, y verificador en modo offline los datos capturados de manera online, respectivamente. Todos los escenarios fueron creados a partir de los diferentes bloques que maneja el software y diseñados desde 0, mas, con guía en los escenarios por default propios de OpenViBE.

Es necesario recordar que estos escenarios fueros creados partiendo de la característica de detección de parpadeo del dispositivo "Mindwave Headset", y que las dos clases entre las que diferenciará las señales fueron, de parpadeo y de detención de parpadeo, indicada dentro de OpenViBE por los 'labels' "OVTK_GDF_Eye_Blink" y "OVTK_GDF_Eyes_Down", respectivamente. Con el fin de poder indicar al usuario en qué momento parpadear y en cual no, se utilizó la figura 3, y se le indico al usuario que en el momento que esta apareciera en el simulador de estímulos, realizará un parpadeo constante y consistente, y cuando esta se retirara de la pantalla, detuviera al parpadeó.

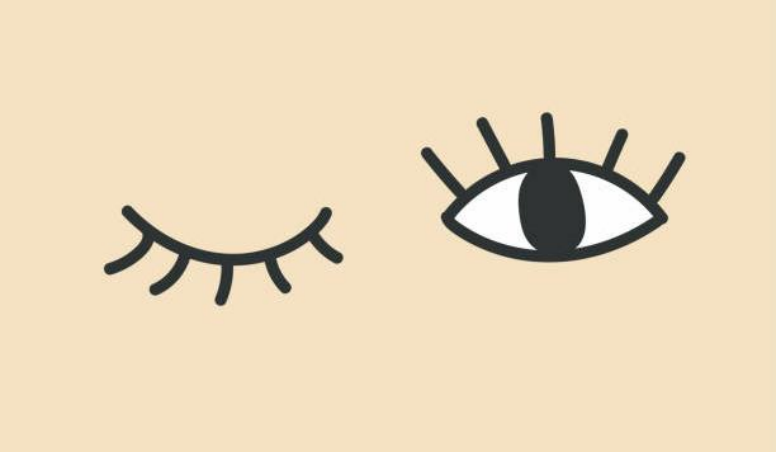


Figura 3. Imagen de Indicación de Parpadeo al Usuario. Tomado de: iStock_iNueng

A continuación, los escenarios realizados

- Escenario Adquisición de Datos

Como se mencionó anteriormente la adquisición de datos se genera a partir de un simulador de estímulos, bloque que se observa en la figura 4, denominado 'Neurosky BCI Stimulator', el cual es el encargado de indicar al usuario los momentos en los que debe ejecutar el parpadeo y los momentos en que no, en otras palabras, es la guía para el usuario. La captación de los datos es lograda por medio del bloque "Acquisition Client", el cual trabaja en conjunto con una sub aplicación denominada "OpenViBE Acquisition Server", la cual hace posible la conexión con el Mindwave Headset. Por ultimo para el guardado de datos se usa el bloque "Generic stream writer", el cual graba toda la información tanto de los estímulos como de la señales generadas durante el experimento en un archivo ".ov", denominación propia del software.

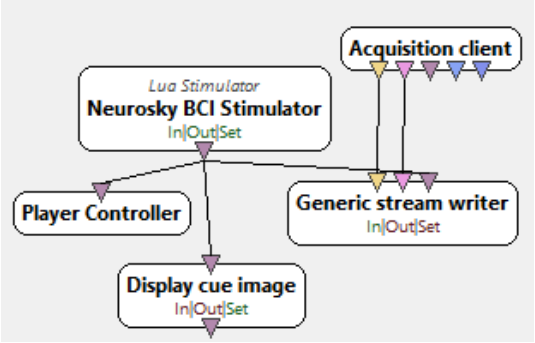


Figura 4. Escenario diseñado en OpenViBE para el proceso de Adquisición de Datos. Autores

- Escenario Entrenamiento Clasificador

Para el entrenamiento del clasificador se dividen las señales dependiendo de cuál de los estímulos, de parpadeo o no parpadeo, estaban siendo presentados al usuario en los diferentes periodos de tiempo, esta separación genera las dos columnas de datos separadas, por la izquierda estarán todos los periodos del experimento en la que se generaba parpadeo, y por la derecha todos los periodos de tiempo en los que no se estaba parpadeando; esta separación es lograda gracias al bloque 'Stimulation based epoching. Posterior a la separación de señales se le aplica a cada rama de datos el bloque "Power Log" con el objetivo aumentar la intensidad de las señales y por último se convierten estas señales en vectores para entrenar al clasificador. El clasificador se entrenó mediante el algoritmo LDA (Linear Discriminant Analysis) dado que solo se debía diferenciar entre dos clases y que las diferencias en las señales eran fáciles de captar y analizar.

Es necesario aclarar que los datos con los que se entrenó el clasificador fueron los mismos capturados en el escenario de adquisición.

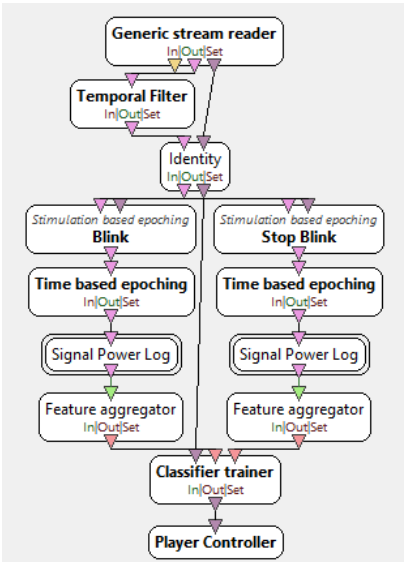


Figura 5. Escenario diseñado en OpenViBE para el proceso Entrenamiento Clasificador. Autores

- Escenario Prueba Online

El pen ultimo escenario es el escenario de prueba online, éste actúa mezclando los funcionamientos y procesamiento de los otros dos anteriores escenarios, el escenario de adquisición de datos y el escenario de entrenamiento del clasificador, trabajando con el simulador de estímulos para indicar al usuario en qué momentos efectuar el parpadeo y también cuando no realizarlo, a la par que analiza las señales y las clasifica, mediante el clasificador entrenado. La clasificación generada en tiempo real se puede visualizar a manera de feedback y también a manera de porcentajes, utilizando el bloque "Graz

visualization", y el bloque "Accuracy measure", respectivamente (Figura 7). El feedback es representado mediante una barra azul, la cual es mostrada los últimos 3 segundos de presentación de cada estímulo, indicando, dependiendo de si crece hacia la izquierda o derecha, que las señales se están clasificando como parpadeo o como periodo de no parpadeo. En lo que respecta a la representación a manera de porcentaje, el bloque logra generar el porcentaje de correcta clasificación comparando los estímulos mostrados al usuario con las clasificaciones generadas por el clasificador, valga la redundancia.

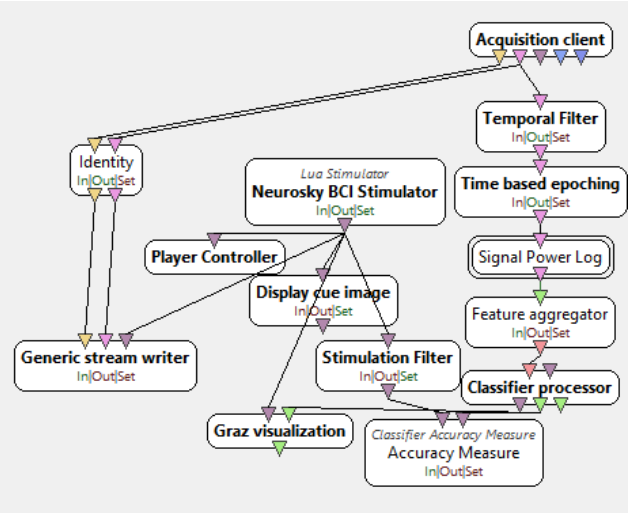


Figura 6. Escenario diseñado en OpenViBE para el proceso Prueba Online. Autores

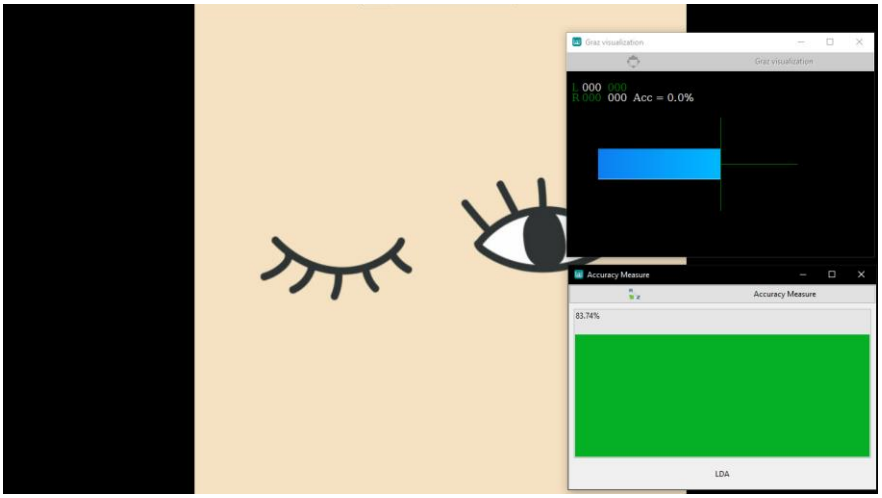


Figura 7. Ejecución Escenario Prueba Online. Autores

- Escenario Prueba de Datos (Offline)

El cierre de los escenarios lo realiza el escenario de verificación de datos, este realiza todo el proceso visto en el escenario de prueba online mas de manera offline. Gracias a que el escenario fue diseñado para trabajarse de manera offline, fue el escenario elegido para implementar la comunicación por TCP/IP, con el fin de enviar la información del clasificador al software Python. Esta comunicación es lograda con el bloque "TCP Writer", a él entran los indicativos de la clasificación generada, también llamados "labels", para ser enviados en formato 'String' hacia Python.

Con el fin de mandar únicamente a Python la información de los periodos en los que eran presentados los estímulos se implementaron los bloques "Switch director" y "Stream Switch", los cuales, mediante la configuración dada, logran este objetivo. Además de todos los procesos previamente indicados, el escenario también cuenta con dos bloques

estadísticos que ayudan conocer mediante más medios que tan bueno fue el proceso de clasificación, el "Confusion matrix" y el "Kappa coefficient", los cuales indican los valores de positivos y falsos positivos de la clasificación por cada clase, y el porcentaje de aleatoriedad de la clasificación.

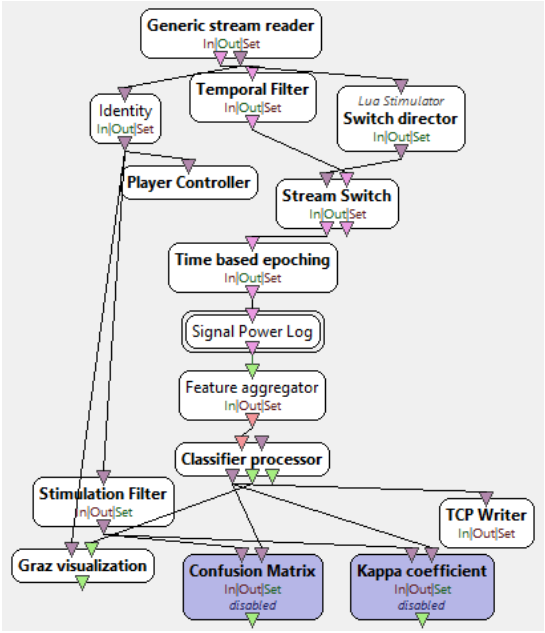


Figura 8. Escenario diseñado en OpenViBE para el proceso Prueba de Datos. Autores

3.1.2 Código Python

Con el objetivo de lograr enviar la información de clasificación de señales de OpenViBE hacia el dispositivo externo, se utilizó de intermediario el software Python. Haciendo uso de las diferentes librerías con las que cuenta se creó un cliente TCP por medio del cual recibir la información desde OpenViBE para posteriormente procesarla y dependiendo de la misma, enviar en binario, ceros y unos a la tarjeta Arduino por medio de la conexión y comunicación serial.

Las partes más relevantes del código pueden ser visualizadas a continuación.

- Configuración Parámetros de Conexiones

```
# Configure the serial port parameters
serial_port = 'COM8'  # Replace 'COMx' with the correct serial port name of your Arduino
baud_rate = 57600

# Create a TCP/IP socket
client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Connect the socket to the server's IP address and port
server_address = ('localhost', 5679)
```

Figura 9. Apartado de Configuración/Conexión Serial y TCP del Código Python. Autores

En la figura anterior se puede visualizar la configuración de los parámetros necesarias para la conexión con el Arduino mediante comunicación serial y también con el servidor TCP de OpenViBE. Esta configuración requiere para el Arduino, la asignación del puerto "COM" al que es conectado, además del "baud rate" de la comunicación, y para la comunicación TCP, la asignación del servidor al que se conecta y el puerto, en este caso como es una conexión en el mismo equipo la dirección del servidor es la local, 'localhost', y el puerto el configurado dentro del bloque "TCP Writer" en OpenViBE, '5679'.

- Establecimiento de Conexiones y Captura y Clasificación de los datos

```
# Create a TCP/IP socket
client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client_socket.connect(server_address)

try:
    print("Connected to OpenViBE")

    # Open the serial port connection to Arduino using the contextlib.closing context manager
    with closing(serial.Serial(serial_port, baud_rate)) as arduino:
```

Figura 10. Apartado de Establecimiento de Conexiones - Código Python. Autores

Para el establecimiento de conexiones se hizo uso de tres comandos, el primero el "socket.socket", el cual es utilizado para la creación del socket TCP/IP necesario para la comunicación entre OpenViBE y Python. El segundo comando es el ".connect" con el cual se establece la dirección del server para poderse interconectar (dirección establecida en la configuración de parámetros de conexión). Como tercer y último comando se encuentra el "serial.serial" con el cual se establece la conexión mediante el serial con Arduino

En lo que respecta a la captura de datos interviene el comando ".recv", el cual establece la cantidad de bytes recibidos desde OpenViBE (2042), datos que a su vez son mostrados al usuario mediante el terminal gracias el comando "print". Y para la clasificación se utiliza la sección de 'If' generada, en la que se leen los bytes provenientes de OpenViBE y según sea el resultado de la clasificación leída, se envía un valor de 0 y 1 en binarios al Arduino, 0 para clasificación de no parpadeo y 1 para parpadeo.

```
    data_bytes = client_socket.recv(2042)

    if data_bytes:
        # Save the data
        data = data_bytes

        # Print the received data
        print(f"Received data: {data}")

        if data == b'OVTK_GDF_Eye_Blink\r\n':
            binary_data = b'1'
        elif data == b'OVTK_GDF_Eyes_Down\r\n':
            binary_data = b'0'
        else:
            binary_data = b'3'  # Set to an appropriate value if needed
```

Figura 11. Apartado de Captura y Clasificación de los datos - Código Python. Autores

- Envío de datos y Cierre del Código

```
                # Send binary data to Arduino through the serial port
                arduino.write(binary_data)
                print(f"Data Send: {binary_data}")

except Exception as e:
    print(f"Error: {e}")
finally:
    # Close the connection and clean up resources
    client_socket.close()
```

Figura 12. Apartado de Envío a Arduino y Finalización - Código Python. Autores

Para la última etapa del código los binarios generados por las sección de 'If' son enviados por medio del comando "arduino.write" al Arduino UNO mediante la conexión serial, y para verificación son mostrados estos mismos al usuario mediante la terminal. Posterior a finalizar el envío de información el "socket" generado es cerrado y se finaliza el código.

De ocurrir algún error en la ejecución del código en cualquiera de sus etapas se muestra al usuario mediante al terminal la letra “e”.

3.1.3 Código Arduino

Como finalización de la comunicación está el código de Arduino el cual cumple el objetivo de leer los binarios provenientes de Python y encender o apagar una serie de LEDs, dependiendo del binario detectado, siendo el LED 1 encendido cuando no se detecta parpadeo, el LED 2 cuando es detectado parpadeo, y el LED 3 únicamente siendo encendido cuando es detectado algo diferente a estas dos opciones.

```
const int ledPin1 = 3; // Pin connected to LED 1
const int ledPin2 = 4; // Pin connected to LED 2
const int ledPin3 = 5; // Pin connected to LED 3
void setup() {
  pinMode(ledPin1, OUTPUT); // Set LED 1 pin as an output
  pinMode(ledPin2, OUTPUT); // Set LED 2 pin as an output
  pinMode(ledPin3, OUTPUT); // Set LED 2 pin as an output
  Serial.begin(57600); // Set the baud rate to match the communication
}

void loop() {
  while (Serial.available()) {
    char c = Serial.read();

    if (c == '0') {
      digitalWrite(ledPin1, HIGH); // Turn on LED 1
      digitalWrite(ledPin2, LOW);  // Turn off LED 2
      digitalWrite(ledPin3, LOW); // Turn off LED 3
    } else if (c == '1') {
      digitalWrite(ledPin1, LOW);  // Turn off LED 1
      digitalWrite(ledPin2, HIGH); // Turn on LED 2
      digitalWrite(ledPin3, LOW); // Turn off LED 3
    } else {
      digitalWrite(ledPin1, LOW);  // Turn off LED 1
      digitalWrite(ledPin2, LOW);  // Turn off LED 2
      digitalWrite(ledPin3, HIGH); // Turn on LED 3
    }
  }
}
```

Figura 13. Código Arduino IDE. Autores

3.2 Implementación en Físico

El montaje en físico únicamente requirió de la tarjeta Arduino UNO, una protoboard y 3 LEDS, y estos componentes fueron interconectados como se visualiza en las siguientes figuras. El LED amarillo se encendería en el momento en que fuera detectado que no se está parpadeando, el blanco en el momento en que se estuviera detectando el parpadeo y el rojo cuando hubiera errores en la comunicación

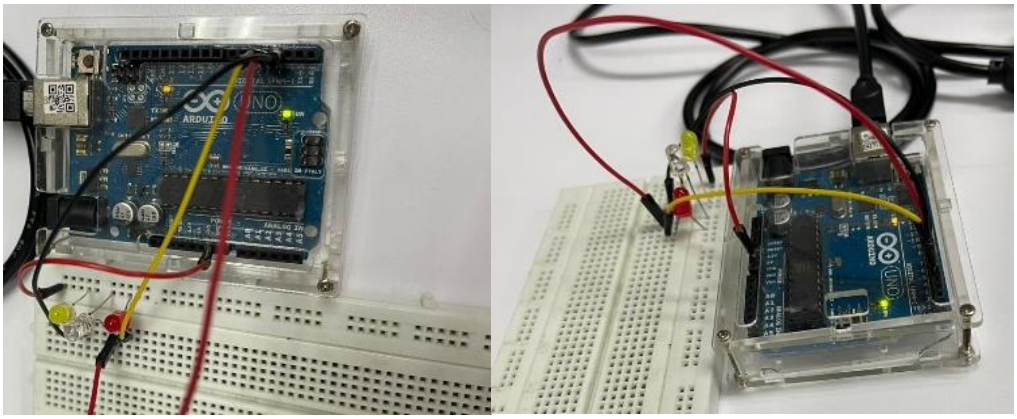


Figura 14. Implementación en Físico. Autores

Los LEDs fueron conectados en orden mediante jumpers a los puertos 3,4,5 de la tarjeta Arduino, y el negativo de los mismos a la tierra.

4. Discusión

Además de los resultados directos mencionados anteriormente, la obtención de los escenarios, los códigos desarrollados y el sistema completo de control por medio de señales cerebrales de dispositivos, este proyecto investigativo también brinda como resultado la apertura, aún más, de las puertas y áreas de aplicación de las BCI. Dado que fue un proyecto desarrollado netamente en softwares y por medio de herramientas OpenSource su alcance se vuelve mucho más cercano para todas los usuarios, áreas e industrias que quisieran aplicarlo.

A pesar de ser un proyecto implementado para el control de un proceso simple de encendido y apagado de LEDs, es un proyecto referente para futuras aplicaciones de sistema de control eléctrico, electrónico, e industrial, dependiendo hasta el nivel que se quiera llevar, tanto por los softwares que maneja como por los dispositivos físicos de los cuales se hicieron uso, debido a que son dispositivos de gran accesibilidad en el mercado dentro de sus áreas, Arduino UNO en lo referente a tarjetas de desarrollo de electrónica embebida y el Mindwave Headset en lo referente a dispositivos de captación de EEG.

5. Conclusiones

Como resultado de las diferentes partes que componían el proyecto se logró obtener un sistema completo y correcto para el control de dispositivos a través de señales EEG. Por medio de la investigación realizada a fondo se logró plantear las bases para el desarrollo eficiente y correcto del proyecto, conociendo las funcionalidades y alcance que el software OpenViBE y el dispositivo Mindwave Headset poseían, partiendo de estas bases, en el apartado de diseño, fueron desarrollados 4 escenarios finales y dos códigos en los softwares, según las especificaciones deseadas, capaces de evaluar al usuario frente a la instrucción de parpadeo y detención del mismo, a la vez que procesaban, clasificaban, manejaban las señales cerebrales adquiridas de manera satisfactoria, y generaban comandos de control a partir de las mismas.

Posteriormente se comprobó el funcionamiento de todo el proyecto mediante la implementación los escenarios y protocolos diseñados, conectándolos y comunicándolos por medio de la conexión TCP/IP con Python, y por la conexión serial, a la tarjeta Arduino UNO, evaluando su correcta labor en tiempo real y de manera física en la manipulación del dispositivo externo.

El producto es un proyecto con el que, al tenerlo de base, los alcances que se le den a la tecnología solo dependen de la visión que se tenga, porque los elementos para desarrollarlos ya fueron en su mayoría cubiertos.

Contribuciones de autor: Conceptualización, NC y JB; metodología, NC y JB; software, DE, NC y JB; validación, DE; análisis formal, DE; investigación, DE; recursos, NC y JB; curación de datos, DE; redacción — preparación del borrador original, DE; redacción: revisión y edición, NC y JB; visualización, DE, NC y JB; supervisión, NC, JB; administración de proyectos, NC y JB; adquisición de financiación, NC y JB. Todos los autores han leído y aceptado la versión publicada del manuscrito.

Agradecimientos: Agradecemos al programa de Ingeniería de Mercados de la Facultad de Ingeniería de la Universidad Autónoma de Bucaramanga por el préstamo del dispositivo Mindwave Headset de NeuroSky para todo el periodo de la investigación

Conflictos de interés: "Los autores declaran no tener ningún conflicto de intereses".

Referencias

1. H. Y. Espitia & E. A. Sandoval (2022). Implementación de interfaz cerebro-máquina para el control de movimiento de un brazo manipulador robótico UR3 para Aplicaciones de Pick and Place.
2. Díez, I. (2015). Design of classification methods for Brain Computer Interfaces: An application using the OpenViBE software (thesis). Universidad del País Vasco, Leioa.
3. Rennard, Y., Lotte, F., Gilbert, G., Congedo, M., Maby, E., Delannoy, V., Bertrand, O., & Lécuyer, A. (2010, April 28). OpenViBE: An Open-Source Software Platform to Design, Test and Use Brain-Computer Interfaces in Real and Virtual Environments. France.
4. Pyhon.org. Aprende Python (2023). Retrieved from https://es.python.org/aprende-python/
5. Diadema NeuroSky MindWave mobile 2. SANDOROBOTICS. (2021, August 13). Retrieved from https://sandorobotics.com/producto/sen-14758/#:~:text=Este%20es%20el%20MindWave%20Mobile,%2C%20iOS%2C%20o%20dispositivo%20Android.
6. Neurosky. (2023). MindWave Mobile 2. MindWave. Retrieved from https://store.neurosky.com/pages/mindwave
7. Arduino Uno. (2019). Retrieved from https://arduino.cl/arduino-uno/
8. Revera, J. D. (2017). DISEÑO DE UN SISTEMA DE GUIADO DE ROBOTS MEDIANTE LA ADQUISICIÓN DE SEÑALES EEG (thesis). Universidad Politécnica de Valencia, Valencia.
9. NeuroSky Store - Apps. Apps. (2014). https://store.neurosky.com/collections/apps