

INSTITUTO TECNOLÓGICO DE CHIHUAHUA
DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

***“INTERFAZ CEREBRO COMPUTADORA PARA
SISTEMAS DE COMPUTACIÓN EN EL BORDE”***

TESIS

QUE PARA OBTENER EL GRADO DE

***MAESTRO EN CIENCIAS
EN INGENIERÍA ELECTRÓNICA***

PRESENTA:

VICENTE GONZÁLEZ NAVARRO

**DIRECTOR DE LA TESIS:
*DR. JUAN ALBERTO RAMÍREZ QUINTANA***

CHIHUAHUA, CHIH., AGOSTO 2021



EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA



TECNOLÓGICO
NACIONAL DE MÉXICO®





Chihuahua, Chih. 20 de agosto de 2021

M.C. ROGELIO ENRIQUE BARAY ARANA

JEFE DE LA DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

PRESENTE

En cumplimiento con los requerimientos para la obtención del grado de Maestro en Ciencias en Ingeniería Electrónica, le notificamos que el documento de tesis del alumno C. VICENTE GONZÁLEZ NAVARRO, titulado *"Interfaz cerebro computadora para sistemas de computación en el borde"* dirigido por Dr. Juan Alberto Ramírez Quintana ha sido aprobado y aceptado para su impresión.

Por lo anterior, proponemos le sea concedida la autorización de impresión correspondiente.

Agradeciendo la atención a la presente, quedamos de usted:

ATENTAMENTE

*Excelencia en Educación Tecnológica®
"La Técnica por el Engrandecimiento de México"*



Juan Ramírez

DR. JUAN ALBERTO RAMÍREZ QUINTANA
DIRECTOR DE TESIS

Enr

MTRO. ENRIQUE GERARDO HERNÁNDEZ VEGA
MIEMBRO DEL COMITÉ TUTORIAL

Alfredo

MTRO. ALFREDO CHACÓN ALDAMA
MIEMBRO DEL COMITÉ TUTORIAL

Graciela Ramirez A

DRA. GRACIELA MARÍA DE JESÚS
RAMÍREZ ALONSO
MIEMBRO DEL COMITÉ TUTORIAL



Chihuahua, Chih. 25 de agosto de 2021

**C. VICENTE GONZÁLEZ NAVARRO
PRESENTE**

En cumplimiento con los requerimientos para la obtención del grado de Maestro en Ciencias en Ingeniería Electrónica y a propuesta de su Comité Tutorial, la División de Estudios de Posgrado e Investigación le concede la autorización para imprimir la tesis titulada *"Interfaz cerebro computadora para sistemas de computación en el borde"* dirigida por Dr. Juan Alberto Ramírez Quintana con el siguiente contenido de capítulos:

- I Antecedentes
- II Sistema BCI conectado a IoT
- III Red neuronal de reconocimiento en el borde
- IV Dispositivos
- V Resultados
- VI Conclusiones y trabajo futuro

ATENTAMENTE

Excelencia en Educación Tecnológica®
"La Técnica por el Engrandecimiento de México"

M.C. ROGELIO ENRIQUE BARAY ARANA
JEFE DE LA DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN



Av. Tecnológico No. 2909 Col. 10 de Mayo C.P. 31310, Chihuahua, Chih.,
Tel. (614) 201 2000, (614)413 5187, Ext. 2157 e-mail: dir_chihuahua@tecnm.mx

tecnm.mx | itchiihuahua.edu.mx



CARTA CESIÓN DE DERECHOS

En la ciudad de Chihuahua el día 25 de agosto de 2021 1 de septiembre de 2021, el que suscribe C. VICENTE GONZÁLEZ NAVARRO con número de control G19061417, de la Maestría en Ciencias en Ingeniería Electrónica, adscrita a la División de Estudios de Posgrado e Investigación del Instituto Tecnológico de Chihuahua, manifiesta que es autor intelectual del presente trabajo de Tesis bajo la dirección del Dr. Juan Alberto Ramírez Quintana y cede los derechos del trabajo titulado "Interfaz cerebro computadora para sistemas de computación en el borde", al Tecnológico Nacional de México y/o Instituto Tecnológico de Chihuahua para su difusión, divulgación, transmisión, reproducción, así como su digitalización con fines académicos y de investigación.

C. VICENTE GONZÁLEZ NAVARRO



DECLARACIÓN DE ORIGINALIDAD

En la ciudad de Chihuahua el día 25 de agosto de 2021 1 de septiembre de 2021, el que suscribe C. VICENTE GONZÁLEZ NAVARRO de la Maestría en Ciencias en Ingeniería Electrónica, con número de control G19061417, adscrito a la División de Estudios de Posgrado e Investigación del Instituto Tecnológico de Chihuahua, manifiesta que es autor intelectual de la tesis titulada "Interfaz cerebro computadora para sistemas de computación en el borde" bajo la dirección Dr. Juan Alberto Ramírez Quintana; que el contenido es original y que las fuentes de información consultadas para su fundamentación están debidamente citadas y referenciadas.

C. VICENTE GONZÁLEZ NAVARRO

Agradecimientos

A mi familia

En especial a mi abuela, que gracias a todo su apoyo desde que era pequeño he podido llegar hasta donde estoy hoy.

A mis amigos

Gracias a mis amigos, tanto de la carrera como de la maestría: Felipe, Miguel, Xavier, Alonso, Félix, Fidel, Iván, Edgar y Uriel. Todos formaron parte de este viaje y aportaron algo en el camino.

A mi comité de tesis

Agradezco al Dr. Juan Alberto Ramírez Quintana por haber sido mi asesor y guiarme a lo largo de todo este proceso, tanto académicamente como personalmente. A la Dra. Graciela María de Jesús Ramírez Alonso por siempre dar consejos certeros que ayudaron a formar esta investigación. Al M.C. Alfredo Chacón Aldama por sus aportaciones y conocimientos a lo largo de mi maestría. Al M.C. Enrique Gerardo Hernández Vega quien siempre siguió muy de cerca el desarrollo de este proyecto. Y se ha convertido en mi mentor profesional y personal, además de un gran referente de vida.

Al Tecnológico Nacional de México y CONACYT

Agradezco al Consejo Nacional de Ciencia y Tecnología por el apoyo a la investigación que me brindó estos 2 años. Al Tecnológico Nacional de México por el apoyo a este trabajo bajo el proyecto “Interfaz Cerebro Computadora para Sistemas de Computación en el Borde”,
Clave 7598.20-P.

RESUMEN

INTERFAZ CEREBRO COMPUTADORA PARA SISTEMAS DE COMPUTACIÓN EN EL BORDE

Ing. Vicente González Navarro
Maestría en Ciencias en Ingeniería Electrónica
División de Estudios de Posgrado e Investigación del
Instituto Tecnológico de Chihuahua
Chihuahua, Chih. 2021
Director de Tesis: Dr. Juan Alberto Ramírez Quintana

Internet de las Cosas (IoT) es un tema de importancia actual debido al alto impacto y crecimiento a lo largo de los últimos años por sus aplicaciones médicas, automovilísticas, de aprendizaje, uso recreativo, etc. De esta manera, IoT ha mejorado y facilitado todo tipo de productos y servicios, volviendo más eficiente las comunicaciones, procesos de control, etc. Entre las aplicaciones potenciales de IoT se encuentran las Interfaces Cerebro Computadora ya que al asignar la transferencia de información vía Internet mucho del cableado es eliminado del sistema y el post procesamiento de la señal se vuelve más ágil, permitiendo que se desarrollen integraciones de tecnologías cerebro computadora con mayor potabilidad para conseguir con eficiencia la respuestas o comportamientos buscados en la lectura de señales.

Contribuyendo con este esfuerzo, en esta tesis se propone el desarrollo de una interfaz cerebro computadora bajo el paradigma de computación en el borde para procesar y transmitir señales electroencefalográficas (EEG) procesadas con una red neuronal artificial. Como estrategias para computación en el borde, se utiliza un esquema tradicional con la tarjeta UP-Board y se explora la posibilidad de realizar aceleramiento por hardware con descripciones de bajo nivel por medio de un FPGA.

Como resultado, se obtiene un sistema BCI embebido portable que procesa señales EEG y las trasmite a una red IoT enfocada en computación en el borde, la red en software obtuvo una precisión del 100% junto a una exactitud del 98.33%, la red en hardware presentó una precisión del 78.33% y una exactitud del 81.67%. El sistema completo opera con una conexión

inalámbrica de 20 MB de bajada, 2 MB de subida y una latencia de entre 15ms a 16ms obteniendo un tiempo de operación de entre 15.5s a 17s desde el arranque del equipo hasta el despliegue del resultado en la web. Además, se exponen los resultados de la investigación relacionada al aceleramiento por hardware reconfigurable, haciendo énfasis en las posibilidades, desafíos y posibles opciones que pueden tomarse en el transcurso de este proceso.

CONTENIDO

LISTA DE FIGURAS	xi
LISTA DE TABLAS	xii
INTRODUCCIÓN	xiii
CAPÍTULO I	1
ANTECEDENTES	1
1.1 Interfaz cerebro computadora.....	1
1.1.1 Sistema	1
1.1.2 BCI	1
1.1.3 EEG	2
1.1.4 Potencial evocado.....	2
1.1.5 ERS / ERD	3
1.1.6 Habla imaginada.....	3
1.2 Internet de las cosas.....	4
1.2.1 IoT	4
1.2.2 Cloud Computing (computación en la nube)	4
1.2.2.1 Fog Computing (computación de niebla).....	5
1.2.2.2 Edge Computing (computación en el borde).....	5
1.3 Estado del arte	6
1.3.1 Interfaces cerebro computadora aplicadas a Internet de las cosas	6
1.3.1.1 Un vistazo a sistemas BCI aplicado a IoT.....	7
1.3.1.2 Ciberseguridad en sistemas BCI embebidos	8
1.3.1.3 Sistemas BCI aplicados a la medicina.....	9
1.3.1.4 Sistemas BCI con integración directa a IoT	10
1.3.1.5 Sistemas BCI con fines recreativos	11
1.3.2 Computación en el borde (Edge Computing) basado en Internet de las cosas (IoT) ...	11
CAPÍTULO II	15
SISTEMA BCI CONECTADO A IOT	15
2.1 Dispositivos OpenBCI.....	15
2.1.1 Cyton	15

2.1.2 WiFi Shield	16
2.2 Configuración de los dispositivos (Cyton + WiFi Shield)	17
2.3 Metodología de obtención de datos.....	23
2.4 Transmisión a la red	27
CAPÍTULO III	29
RED NEURONAL DE RECONOCIMIENTO EN EL BORDE	29
3.1 Red Neuronal Artificial.....	29
3.2 Entrenamiento	33
CAPÍTULO IV	39
DISPOSITIVOS	39
4.1 UP Board	39
4.1.1 Especificaciones del dispositivo.....	40
4.1.2 Adecuación del dispositivo	42
4.1.3 Implementación del software diseñado e integración del sistema	44
4.2 Experimentación con FPGA.....	46
4.2.1 Características de hardware (Cora Z7).....	47
4.2.2 Diseño e implementación de la RNA para dispositivos reconfigurables	49
CAPÍTULO V	53
RESULTADOS	53
5.1 Sistema de computación en el borde	53
5.1.1 Métricas del sistema de computación en el borde.....	56
5.1.2 Computación en el borde contra computación en la nube.....	57
5.2 Experimentación con FPGA.....	59
5.2.1 RNA implementada en hardware	59
5.2.2 Desempeño de la RNA	62
CAPÍTULO VI	63
CONCLUSIONES Y TRABAJO FUTURO	63
REFERENCIAS	66

LISTA DE FIGURAS

Figura 2.1. Tarjeta OpenBCI Cyton y dongle USB.....	16
Figura 2.2. Módulo OpenBCI WiFi Shield.	17
Figura 2.3. OpenBCI Cyton para operar con el módulo WiFi Shield.....	17
Figura 2.4. Dispositivos conectados a la fuente de alimentación.	18
Figura 2.5. Localización del interruptor de alimentación para la tarjeta Cyton.....	18
Figura 2.6. Localización del interruptor de encendido del módulo en conjunto.	19
Figura 2.7. Configuración principal de Wifi en Windows.	19
Figura 2.8. Dispositivo OpenBCI WiFi Shiled detectado por el sistema de cómputo.....	20
Figura 2.9. Página de configuración principal del dispositivo OpenBCI WiFi Shield.	20
Figura 2.10. Selección para acceder al administrador de conexión Wifi del dispositivo.	21
Figura 2.11. Administrador de conexión Wifi.....	21
Figura 2.12. Opciones de selección de redes.....	22
Figura 2.13. Mensaje de conexión exitosa.	22
Figura 2.14. Sistema internacional 10-20.....	23
Figura 2.15. Ejemplificación de distribución de electrodos.	24
Figura 2.16. Ejemplo de ingreso de parámetros para la ejecución del programa de adquisición de datos.	25
Figura 2.17. Respuesta del filtro Bessel pasa banda en frecuencia y fase.	26
Figura 2.18. Protocolo de transferencia de archivos.....	27
Figura 2.19. Ejemplo de resultado de detección de artefactos mostrado en la web correspondiente a $y(k)$	28
Figura 3.1. Estructura de neurona abstracta.	30
Figura 3.2. Arquitectura resultante de la RNA.	32
Figura 3.3. Neurona p como un nodo oculto de la red.	36
Figura 4.1. Dispositivo UP Board.	40
Figura 4.2. Módulo Wifi para la tarjeta UP Board.	43
Figura 4.3. Puerto de conexión CN7 de la tarjeta UP Board.....	44
Figura 4.4. Recorte de la sección de ruido de la señal adquirida.....	45
Figura 4.5. Diagrama a bloques del proceso general del sistema.	46
Figura 4.6. Digilent Cora Z7-07S.....	47
Figura 4.7. Representación del número decimal -0.4072834 en binario.	50
Figura 4.8. Representación del número decimal 104.85 en binario.....	51
Figura 4.9 . Diagrama ilustrativo del comportamiento de la neurona en hardware.	52
Figura 5.1. Proceso del sistema completo BCI de computación en el borde.....	55
Figura 5.2. Despliegue de información en la página web.....	55

Figura 5.3. Gráfica de utilización de recursos. 61

LISTA DE TABLAS

Tabla 3.1. Funciones de activación. 33

Tabla 4.1. Especificaciones técnicas del dispositivo UP Board. 40

Tabla 4.2. Especificaciones técnicas del dispositivo Digilent Cora Z7-07S. 48

Tabla 5.1. Matriz de confusión. 56

Tabla 5.2. Métricas de desempeño. 56

Tabla 5.3. Métricas del sistema. 57

Tabla 5.4. Métricas de comparación de computación en el borde y computación en la nube. 58

Tabla 5.5. Estadísticas de consumo de hardware. 60

Tabla 5.6. Recomendaciones de recursos de FPGA para ML. 61

Tabla 5.7. Métricas de desempeño en hardware. 62

INTRODUCCIÓN

La idea de dispositivos electrónicos interconectados ha existido desde la década de 1970, pero fue hasta el año de 1999 que el concepto de Internet de las Cosas (IoT) tomó forma. Con el paso de los años se ha visto el verdadero potencial de las tecnologías IoT y las posibles aplicaciones en el beneficio de la industria, salud y sociedad. Estos beneficios se deben a las ventajas inmensas de IoT en el área de la comunicación y conectividad informática, tales como el manejo remoto de dispositivos, la transmisión de análisis y diagnóstico médico, la comodidad de relegar procesamiento local a procesamiento en la nube, uso recreativo, etc. IoT es algo que hoy día pasa desapercibido por algunos, pero es indispensable para millones de dispositivos y procesos en la vida cotidiana.

Por ejemplo, una de las aplicaciones de IoT con mayor campo de exploración y mejoría es la integración con las Interfaces Cerebro-Computadora (BCI por sus siglas en inglés). Estas interfaces se definen como sistemas que miden y detectan las características de las señales cerebrales para luego ser procesadas e interpretadas por una computadora. Uno de sus usos comunes es conocer la intención del usuario interpretando en tiempo real para definir comandos que logran describir dicha intención. Existen trabajos donde el IoT beneficia el funcionamiento de BCI, por ejemplo, la transmisión de señales procesadas de manera automática a hospitales y clínicas [1], el aumento de la independencia de las personas con graves discapacidades motoras [2], la clasificación automática de los distintos estados cerebrales de personas saludables, pacientes con deterioro cognitivo o Alzheimer [3], ciberseguridad [4], entre otras. Aún y cuando los trabajos se inclinan hacia el aspecto médico de clasificación de las señales [5] o detección de ataques epilépticos [6], también se tienen algunos otros que se enfocan de manera un poco más recreativa [7]. En [8] se reportan parte de los beneficios de integrar directamente las tecnologías de BCI e IoT para aprovechar la computación en la nube y el aprendizaje profundo en un sistema BCI para predecir ataques epilépticos y su origen cerebral a través de señales electroencefalográficas y electrocorticográficas (ECoG).

Sin embargo, es evidente la falta de trabajos en la literatura donde se exploren o implementen los sistemas BCI enfocados a tecnologías de computación en el borde de una red IoT. Estas

tecnologías son infraestructuras de computación distribuida que forman centros de datos al borde de la red, permitiendo que el procesamiento de datos sea de forma local (cercano a la fuente) en vez de utilizar la nube [9]. Por lo tanto, en esta tesis se propone el desarrollo de un sistema BCI embebido basado en EEG bajo el esquema de computación en el borde para procesar y transmitir señales obtenidas de una estrategia mental, para desarrollar aplicaciones BCI portables y ubicuas. La integración de computación en el borde y los sistemas BCI implicará que los datos serán recolectados, analizados y procesados en la periferia (borde de la red) para cuando sea requerido ponerse en contacto con el mundo físico (dispositivos que requieran la información). Lo anterior, se traduce en mejoras en la velocidad y facilidad de actualizaciones en el software, reducción de latencia, y el aprovechar el procesamiento de datos en forma local para mejorar la seguridad de los datos.

El resto de la tesis se organiza de la siguiente manera: el capítulo 1 contiene los antecedentes necesarios para conocer los fundamentos literarios que justifican este proyecto. El capítulo 2 muestra los dispositivos y procesos para desarrollar una interfaz cerebro computadora operando como dispositivo de Internet de las Cosas. En el capítulo 3 se explica la forma en la que opera la red neuronal diseñada para procesar señales EEG en una BCI conectada a IoT bajo computación en el borde. En el capítulo 4 se destina la explicación de cómo se integran todos los módulos diseñados para completar el sistema BCI – Computación en el Borde en el dispositivo UP board y se complementa con la experimentación de implementar la RNA en un FPGA (Xilinx Cora Z7). Dentro del capítulo 5 son expuestos los resultados obtenidos de esta tesis y, por último, el capítulo 6 presenta las conclusiones y las sugerencias para las posibles aportaciones del trabajo futuro.

CAPÍTULO I

ANTECEDENTES

En este capítulo se presentan los conocimientos y antecedentes para poder comprender esta tesis. Por ello, este capítulo se organiza de siguiente manera: la sección 1.1 contiene los conceptos básicos de interfaces cerebro computadora, en la sección 1.2 se detallan las diversificaciones que otorga seguir una estrategia de Internet de las Cosas. Por último, la sección 1.3 presenta el estado del arte sobre sistemas de interfaz cerebro computadora bajo algún esquema de IoT.

1.1 Interfaz cerebro computadora

1.1.1 Sistema

Según la real academia española, un sistema es un conjunto de elementos que se relacionan de manera ordenada entre sí, contribuyendo a un determinado objetivo. Los sistemas de interés para esta tesis son las interfaces humano-máquina (HMI por sus siglas en inglés), las cuales son comprendidas como el ambiente donde el usuario tiene una interfaz para interactuar con el equipo de cómputo, software o sistema.

1.1.2 BCI

En [10] se define una BCI como un sistema HMI que mide y detecta las características de las señales del sistema nervioso central (SNC). El SNC es responsable de las respuestas a las incidencias en el medio ambiente o en el cuerpo, produciendo las salidas adecuadas. Una BCI le da posibilidad al SNC de ser reinterpretada como alguna salida que no es neuromuscular ni hormonal, produciendo así una respuesta SNC artificial. Para conseguir esto, un sistema BCI cuenta con cuatro componentes fundamentales:

1. Adquisición de señales
2. Extracción de características
3. Clasificador de características
4. Respuesta del sistema (control del sistema)

Además, los sistemas BCI deben contar con un protocolo operativo que especifica el tiempo de operación, parámetros, la naturaleza de las respuestas producidas por el SNC (en caso de ser requerido) y cómo se manejan los errores de interpretación. Un protocolo operativo exitoso permite que el sistema BCI sea flexible y satisfaga las necesidades particulares de cada uno de sus usuarios.

Para medir la actividad cerebral, los BCI utilizan diversas técnicas tales como: electroencefalografía, magnetoencefalografía (MEG), tomografía por emisión de positrones (PET), imagen por resonancia magnética funcional (fMRI), espectroscopia del infrarrojo cercano (fNIRS), electrocorticografía (ECoG) o implantes de microelectrodos a través de neurocirugía. Entre las cuales, EEG es la más utilizada por su portabilidad, accesibilidad y por tratarse de un método no invasivo. Por ello, esta forma de medición es la que se utilizará en esta tesis.

1.1.3 EEG

A partir de [11] se entiende que EEG es la representación del diferencial de potencia eléctrica entre uno o varios electrodos colocados sobre la superficie del cuero cabelludo. EEG es una medida dinámica que se representa en función del tiempo y mediante la diferencia de potencial entre diferentes áreas del cuero cabelludo, se refleja la corriente que fluye en la corteza cerebral durante la activación sináptica de las dendritas de las neuronas piramidales, que se encuentran justo debajo de la superficie craneal.

Las señales EEG suelen tener grandes cantidades de ruido, junto a un comportamiento estocástico, de tal manera que puede resultar complicado medir la actividad cerebral. Para reducir el ruido y mejorar la calidad de la información de la actividad cerebral, se puede recurrir al uso de diversas estrategias que se utilizan para modelar esta actividad, las más populares en la literatura son: potenciales evocados, sincronización/desincronización relacionada a evento (ERS/ERD) y habla imaginada [12], [13].

1.1.4 Potencial evocado

En [14] se define un potencial evocado (EP) como la actividad eléctrica que proviene del sistema nervioso central y periférico, que surge de un estímulo externo espontáneo. Estos,

siguen un periodo de tiempo fijo en respuesta a un estímulo sensorial discreto. Los potenciales evocados son principalmente de dos tipos:

1. Relacionados a estímulos: se producen con latencia corta (mayor de 1.50 ms) y son provocados por estímulos estereotipados de modalidad específica. Por ejemplo, potenciales evocados auditivos, visuales, somatosensoriales y motores del tronco encefálico.
2. Relacionados a eventos: estos son potenciales tardíos (por ejemplo, P300), que dependen del contenido de información del estímulo. Aparecen cuando el sujeto reacciona a un estímulo significativo. Sin embargo, estos EP no son específicos del estímulo, la equivalencia en la relevancia de la tarea es el factor determinante para su obtención, y son comúnmente utilizados con fines de diagnóstico médico.

1.1.5 ERS / ERD

Los procesos sensoriales, cognitivos y motrices del sistema nervioso y el cerebro pueden provocar cambios en las señales EEG en forma de desincronización relacionada con eventos (ERD) o sincronización relacionada con eventos (ERS). Ambos fenómenos son fijos en el tiempo, pero modifican la fase de los componentes oscilatorios de la señal y son pertenecientes a ciertas bandas de frecuencia. La ERD se interpreta como la correlación del área cortical activada con una excitabilidad aumentada, mientras que la ERS en las bandas alfa y beta inferiores pueden ser interpretadas (en ciertas circunstancias) como una correlación del área cortical activa. El mapeo espacial de ERD / ERS puede usarse para estudiar la dinámica de los patrones de activación cortical [15].

1.1.6 Habla imaginada

Esta estrategia se refiere a la medición de mensajes con señales EEG mediante un proceso en el que un sujeto imagina fonemas, palabras o letras [16]. Las tecnologías actuales de BCI relacionadas con el habla imaginada proporcionan estrategias efectivas de comunicación vocal para controlar los dispositivos a través de comandos de voz interpretados a partir de señales cerebrales. Estas tecnologías proporcionan ayuda neuroprotésica para personas con discapacidades del habla, trastornos y enfermedades neuromusculares [17].

1.2 Internet de las Cosas

1.2.1 IoT

El concepto de IoT es presentada en [18], y consiste en permitir que las cosas se conecten a Internet para propiciar una generación e intercambio de información para que el mundo virtual interactúe con el mundo físico. IoT encapsula una visión de un mundo en el que miles de objetos con inteligencia embebida, medios de comunicación y capacidades de detección y actuación se conecten a través de protocolos de redes.

El IoT requiere entornos abiertos y una arquitectura integrada de plataformas interoperables que sea capaz de manejar objetos inteligentes, sistemas ciberfísicos o simplemente cosas, que se definen como los objetos de la vida cotidiana, aumentados con microcontroladores, transceptores ópticos y/o de radio, sensores, actuadores y protocolos para la comunicación en entornos restringidos, donde el hardware objetivo tiene recursos limitados. De esta manera, una red IoT contiene objetos y una red cuya forma de funcionamiento se basa en procesar, transmitir y almacenar la información en la nube, la niebla y el borde.

1.2.2 Cloud Computing (computación en la nube)

Cloud Computing en [19] es un paradigma que permite a los consumidores o usuarios autoabastecerse de sistemas de software basados en servidores remotos, plataformas de desarrollo masivas y servicios de aplicaciones comerciales. El modelo de la nube se compone de cinco características esenciales, tres modelos de servicio y cuatro modelos de implementación.

Las cinco características de la nube son:

- Autoservicio a pedido
- Amplio acceso a la red
- Agrupación de recursos
- Elasticidad rápida
- Servicio medido

Los tres modelos de servicio de la nube incluyen:

- Software como servicio (SaaS)
- Plataforma como servicio (PaaS)
- Infraestructura como servicio (IaaS)

Cada uno de estos modelos de servicio pueden ser implementados como cualquiera de los cuatro modelos de implementación:

- Privado
- Comunitario
- Híbrido
- Público

Las ventajas de la computación en la nube son: fácil accesibilidad debido a que solo es requerida una conexión a Internet para acceder a estos servicios, reducción de costos (principalmente para empresas), recuperación rápida ante desastres o accidentes, además de disponer de almacenamiento extra para dispositivos con memoria limitada. Las desventajas son: dependencia del proveedor, al perder el acceso a Internet se pierde el acceso a la información, vulnerabilidad de la privacidad y probables tiempos de latencia lenta al tener mala conexión.

1.2.2.1 Fog Computing (computación de niebla)

En [20] se describe la computación en la niebla como redes IoT basadas en dispositivos de red tradicionales como enrutadores, conmutadores, decodificadores, servidores proxy, nano servidores dedicados y los centros de microdatos. Estos dispositivos crean servicios de procesamiento, transmisión y almacenamiento de datos de manera local.

De esta manera, la niebla funciona como múltiples redes que comparten datos de manera local, mientras que la nube es una red con sistemas remotos que comparten información masiva. Por lo tanto, la niebla reduce los tiempos de latencia y aumenta la seguridad en el manejo de los datos.

1.2.2.2 Edge Computing (computación en el borde)

El paradigma genérico de Edge Computing [20] distribuye las tareas a los dispositivos y computadoras ubicadas conjuntamente en la misma subred de los dispositivos conectados a la red IoT. Dichas tareas pueden ser la clasificación de datos, el filtrado o la conversión de señales.

Debido a que estos sistemas son capaces de realizar tareas de computación no tan exigentes, se utilizan en tareas livianas.

La computación en el borde ha evolucionado rápidamente para solventar las problemáticas de latencia, ancho de banda y la calidad de servicio (QoS por sus siglas en inglés). Aun así, no todos los dispositivos IoT tienen la capacidad, o los recursos computacionales locales para ejecutar las diferentes aplicaciones a gran escala de manera simultánea, lo que afecta al tiempo de respuesta del servicio y a la QoS.

1.3 Estado del arte

En la literatura se encuentran varios trabajos correspondientes a BCI siendo embebidas en aplicaciones de IoT. Sin embargo, no todos ellos destacan la verdadera importancia del IoT en el BCI, algunos meramente mencionan el uso de IoT, dejando de lado el análisis y la forma de cómo se llevó a cabo dicha implementación. Consecuentemente, este apartado brinda una introducción a trabajos encontrados sobre sistemas BCI e IoT relacionados con:

- Sistemas BCI en la industria.
- Sistemas que parten de la premisa de querer integrar IoT.
- Ciberseguridad en los sistemas.
- Aplicaciones a la industria de la medicina.
- Aplicaciones con integración directa de los sistemas.
- Aplicaciones en el ámbito recreativo.
- Información sobre la computación en el borde basada en IoT.

1.3.1 Interfaces cerebro computadora aplicadas a Internet de las Cosas

Dentro de la literatura se habla poco de la importancia del desarrollo de nuevas tecnologías, estándares e iniciativas que ofrece la industria en la evolución de los sistemas BCI. Por ello, la Iniciativa del Cerebro de la IEEE (IEEE Brain Initiative) y la Asociación de Tecnología del Consumidor (CTA), tratan de subsanar estos puntos, promoviendo la investigación del cerebro, el desarrollo de tecnologías y su comercialización [21]. Con la estandarización de los datos en las señales de electroencefalografía (EEG) impartida por la CTA, se ha facilitado la interoperabilidad de datos para el desarrollo de tecnología cerebral en el entorno de IoT. Esto

ha permitido que los productos de BCI que han sido abiertos al consumidor aceleren el desarrollo de nuevas herramientas para integrarlos en áreas como la rehabilitación, la educación y el entretenimiento. Con esto reduce la importancia de que la industria estandarice la recolección de datos de las señales cerebrales, para formar un banco de datos abierto que permita intercambiar datos a través de Internet. A partir de estas iniciativas, de la estandarización de los datos de los sistemas BCI, así como del surgimiento del paradigma Industria 4.0, surge la necesidad de integrar los sistemas BCI a las redes IoT para intercambiar información a diferentes sistemas y dispositivos electrónicos. Sin embargo, en la literatura se carece de información al respecto, por lo que esta sección de antecedentes presenta algunos de los trabajos que se acercan más a las iniciativas de IEEE Brain Initiative y la CTA.

1.3.1.1 Un vistazo a sistemas BCI aplicado a IoT

En [1] se expone un marco de predicción automática de edad y género de los usuarios, su uso está en función de las señales capturadas con sensores EEG que han sido posicionados en los sujetos durante el estado de reposo, con los ojos cerrados. Se extrajeron características de las actividades cerebrales de 60 sujetos (hombres y mujeres) que se encontraban en edades de entre 6 y 55 años. Se aplicó un clasificador de bosque aleatorio (Random Forest) para el modelado de las señales cerebrales. Para obtener la precisión se comparó contra una máquina de vectores de soporte (SVM por sus siglas en inglés) y clasificadores de redes neuronales. El desempeño del sistema presenta una precisión del 88.33% al 96.66% en predicción de edad y género respectivamente. Se plantea que este modelo de predicción se puede escalar a distintas aplicaciones de IoT en el sector de salud, donde la edad y el género pueden ser transmitidos de manera automática a hospitales y clínicas.

Leeb et al. [2] presentan un análisis para aumentar la independencia de las personas con discapacidades motoras graves, mediante el uso de BCI y aprovechando el poder de IoT. Se analiza la estabilidad de las señales cerebrales de los usuarios finales con discapacidades motoras al ser puestos a prueba con tareas básicas de interacción con dispositivos en el mundo real. Finalmente, se presentan los resultados de nueve usuarios finales con alguna discapacidad motora que fueron capaces de controlar exitosamente las tareas de navegación de un robot de

telepresencia (en un entorno desconocido para ellos). Sus resultados fueron similares a los de 10 usuarios totalmente sanos y que estaban familiarizados con el ambiente de navegación.

En [3] se reporta un novedoso método de aprendizaje multimodal basado en aprendizaje de máquina (ML) para integrar las características de las señales EEG a la clasificación automática de los distintos estados cerebrales de pacientes saludables (PS) y pacientes con algún tipo leve de deterioro cognitivo (LDC) o Alzheimer (AD). Para los experimentos, se usaron 19 canales EEG dentro del dominio del tiempo y frecuencia (TF) lo que implica la implementación de la Transformada Continua de Wavelet (CWT), un conjunto de características (δ , θ , $\alpha 1$, $\alpha 2$, β) extraídas de las sub-bandas EEG. Además, para complementar la información de acoplamiento de fase no lineal de las señales EEG, se extraen estadísticas de orden superior (HOS) de la representación biespectral (BIS), esto genera un segundo conjunto de características que también son evaluadas en las 5 sub-bandas EEG. Tanto como CWT y BIS son las entradas del clasificador (tanto de 2 como de 3 canales). Como punto de referencia experimental, se analizaron conjuntos de datos EEG de 63 sujetos por cada grupo: PS, LDC, AD. Los resultados comparativos mostraron que cuando la concatenación de características multimodales CWT y BIS se usan como entrada, el clasificador de perceptrón multicapa (MLP) supera a todos los demás modelos, específicamente a: el codificador automático (AE), regresión logística (LR) y máquinas de soporte vectorial (SVM). En consecuencia, se puede considerar el esquema de ML propuesto como una alternativa viable a los enfoques de aprendizaje profundo intensivos computacionalmente avanzados. Adicionalmente, se planteó que el sistema resultante se puede conectar a una red IoT y los beneficios que implica.

1.3.1.2 Ciberseguridad en sistemas BCI embebidos

En [22] se presenta un análisis de la vulnerabilidad de los sistemas BCI que utilizan técnicas de transferencia de información por medio de IoT, para conocer la seguridad del manejo de datos en redes IoT de uso doméstico que manejan EEG. Se utilizó un novedoso modelo de aprendizaje profundo para inferir las actividades de un usuario basado en los datos de características reducidas que evitan el robo de información de dispositivos EEG integrados al IoT. Para mitigar los ataques remotos propuestos se sugiere un framework basado en OAuth (estándar que permite el flujo de actualización para aplicaciones informáticas y sitios web) que

utiliza un kit de desarrollo de software (SDK) y el protocolo de socket ThinkGear (TGSP) para evitar aplicaciones no autorizadas. Para mitigar los ataques inmediatos se habla sobre una solución basada en el cifrado para eliminar posibles robos de datos y ataques de repetición. Los datos del EEG indican que la precisión de inferencia de la RCNN propuesta puede alcanzar el 70.55%.

Otro enfoque de seguridad es visto en [4] donde se propone el uso de una BCI inalámbrica basada en neuroactividad como una fuente segura de entropía para generar claves de seguridad. El enfoque propuesto está orientado para hacer frente al paradigma de IoT centralizado en las personas. Se usó una BCI basada en P300 y se determinó experimentalmente que las mejores posiciones para colocar los electrodos (para su objetivo) son CP1, FC1, Fz, CP2 y Cz. Básicamente, se obtuvo un enfoque novedoso para proporcionar seguridad a comunicaciones inalámbricas de red de área corporal basado en la generación segura de claves.

1.3.1.3 Sistemas BCI aplicados a la medicina

R. Chatterjee et al. [5] proponen un método basado en una nueva versión difusa de la matriz de discernibilidad (matriz de confusión) para determinar un subconjunto de características que proporcionan la mejor precisión de clasificación de señales EEG de imágenes motoras en el entorno de Internet de las Cosas médicas (IoMT). Para ello, se usó la base de datos de BCI Competition-II Dataset-III y para extracción de características se aplicó el algoritmo de extracción autorregresivo adaptativo (AAR). Basado en el orden (número de coeficientes AR), se generaron dos variantes de conjuntos de datos: 12 (orden = 6 por electrodo) y 24 (orden = 12 por electrodo) presentados con conjuntos de datos AAR. Para encontrar el mejor subconjunto de características se usaron varios tipos de medidas de disimilitud en contra del propuesto. Los accionamientos de los subconjuntos seleccionados son evaluados utilizando Support Vector Machine (SVM). Los resultados muestran un desempeño de 80% y 75% de precisión para los 12 y 24 conjuntos de datos AAR respectivamente, esto muestra un mejor comportamiento que el resto de los algoritmos encontrado en la literatura.

Otra aplicación médica puede verse en [23], donde se propone un marco cognitivo de salud inteligente basado en la nube de IoT (IoT-Cloud), la cual se comunica con dispositivos

inteligentes tales como sensores para monitoreo de pacientes. Como caso de estudio, se propone un método de detección de ataques epilépticos por medio de EEG para definir la viabilidad del marco cognitivo de salud. El sistema propuesto utiliza movimientos, gestos y expresiones faciales de los pacientes para determinar su estado actual. Los resultados experimentales muestran que el modelo alcanza una precisión y sensibilidad del 99.2% y 93.5% respectivamente.

1.3.1.4 Sistemas BCI con integración directa a IoT

Los trabajos previos son sistemas BCI que tienen relacionan a IoT, pero no fueron integrados realmente al Internet de las Cosas. Por lo tanto, esta subsección reporta trabajos que integran la tecnología IoT a BCI de forma directa.

Singh y Malhotra [6] presentan un sistema de detección automática de ataques epilépticos con señales EEG utilizando aprendizaje automático y computación en la nube a través de redes 4G o Wi-Fi. Las señales EEG son procesadas en la nube utilizando la transformada rápida Walsh Hadamard, mientras que la extracción de características es realizada con espectros de orden superior (HOS), entropía y correlación. Para clasificar las tres etapas de convulsiones, se utilizaron algoritmos como Random Forest, Bayes Net, Naïve Bayes, Multilayer Perceptron, Radial Basis function neural network y C4.5 Decision Tree. Los resultados de las simulaciones muestran que el clasificador Random Forest provee una precisión del 99.40%, sensibilidad 99.40% y de especificación de 99.66%, el estimado del error cuadrático medio mínimo es de 0.0871 y un entrenamiento de 20 ms, lo que lo hace ideal para trabajos en tiempo real.

Zhang et al. [24] proponen un framework unificado basado en aprendizaje profundo por refuerzo que permite la interactividad cognitiva de los seres humanos para unir individuos y los objetos de IoT. Para extracción de características de las señales EEG, se utiliza el Mecanismo de Atención Selectiva (SAM) y una memoria de largo plazo. El experimento contó únicamente con 7 sujetos (4 hombres y 3 mujeres). El set de entrenamiento cuenta con 31,104 muestras y el de prueba por su parte tiene 3,456 muestras. Los resultados de la evaluación son determinados por las métricas de precisión, recall, F1 Measure, matriz de confusión, ROC y AUC (Area Under Curve).

En [8] se muestra un buen ejemplo de aprovechamiento de la computación en la nube y el aprendizaje profundo en un sistema BCI para predecir ataques y su localización a través de señales EEG y electrocorticográficas (ECoG). Son presentados tres contribuciones importantes:

- Un nuevo método para la predicción de ataques epilépticos.
- El enfoque de optimización en estructuras de aprendizaje profundo existentes, para lo cual, se utilizan redes neuronales convolucionales (CNN), análisis de componentes principales (PCA), análisis de componentes independientes (ICA), y el algoritmo de búsqueda diferencial (DSA).
- Se presenta una solución de computación en la nube (IoT) para definir las estructuras propuestas en tiempo real, computación automática y almacenamiento de macrodatos.

Se demuestra la superioridad del sistema BCI propuesto ante los datos clínicos de 11 pacientes que fueron tomados a través de ECoG, se menciona como una alternativa a la metodología actual para el soporte de pacientes con epilepsia focal intratable.

1.3.1.5 Sistemas BCI con fines recreativos

Coogan and He [7] plantean un método de integración para sistemas de realidad virtual en conjunto con redes IoT para ofrecer al usuario un control en tiempo real del entorno físico y virtual en el que se encuentra. Está implementado con la ayuda del motor gráfico Unity (es común en este tipo de trabajos), y se acopla a un paquete de software BCI2000. Se demostró la viabilidad de controlar un entorno de realidad virtual con base en el motor Unity y varios dispositivos comerciales de IoT a través de una interfaz neuronal directa procesada por BCI2000.

1.3.2 Computación en el borde (Edge Computing) basado en Internet de las Cosas (IoT)

La sección anterior presentó un resumen breve del estado del arte de los sistemas BCI que se relacionan con IoT, lo que permite hacer foco a los temas de mayor interés para el desarrollo de esta tesis, los cuales son los sistemas de computación en el borde, y serán abordados en esta subsección.

En [25] se presenta la definición de computación en el borde (Edge Computing) y las ventajas de utilizar alguna plataforma con este tipo de técnica. También es presentado un resumen de las tecnologías clave para construir una plataforma de computación en el borde, y es propuesto un framework para plataformas. Se desarrolla en detalle el papel de los sistemas de gestión de almacenamiento distribuido en la construcción de la plataforma informática de borde. Además, se proporcionan algunas aplicaciones para ilustrar cómo utilizar plataformas informáticas de borde de terceros para crear aplicaciones específicas. Finalmente, se describen los problemas actuales de la plataforma de computación perimetral encontrados en la literatura.

Chen et al. [26] detallan un análisis de tópicos de interés de la transmisión de respuestas desde la perspectiva de la capa de control de acceso medio (MAC) para proponer un protocolo de canal reservado MAC (ChRMAC), que reduzca la colisión y latencia de respuestas en los procedimientos del servicio de computación en el borde (Edge Computing). Para ello, se diseña un esquema que toma en cuenta la restricción de latencia en el ChRMAC que mejora la efectividad de los canales reservados. También se diseña un mecanismo de recuperación para evitar el aumento de latencia y la colisión del proceso de tareas computacionales después de las reservas de canales. Además, se propone un marco de capas cruzadas para la implementación de ChRMAC. Para las evaluaciones de este marco, se utiliza el simulador de redes basado en eventos discretos ns-3. Los resultados de la simulación indican que ChRMAC puede reducir la latencia promedio y la respuesta, aumentando el servicio de rendimiento en EdgeIoT.

Xhafa et al. [27] investigan los desafíos de procesar flujos de datos de IoT en una capa de computación en el borde. Al usar la transmisión de datos de la vida real provenientes del flujo de datos de un automóvil (datos utilizados para la detección de baches a partir de la velocidad del vehículo), así como una infraestructura real que usa el dispositivo Raspberry Pi y un servidor Node-Red. La infraestructura real para la implementación del sistema se divide en las siguientes categorías: la integración de Raspberry Pi con Ubuntu Machine en RDLab UPC para almacenar y ejecutar los scripts del lado del servidor, un servidor IBM Node-Red para visualizar los resultados entrantes en la vista de escritorio y móvil. El protocolo de transporte de telemetría de Message Queue Server (MQTT) se integra en Raspberry Pi para ejecutar el procesamiento de Edge y enviar datos al servidor a través de MQTT Broker.

En [28] se estudia y aborda el concepto de computación en el borde integrado para el seguimiento colaborativo de la tecnología IoT. Además, fue desarrollada una arquitectura de seguimiento colaborativo habilitada para el proceso de computación en el borde con la intención de reducir la exigencia de recursos de hardware para el cálculo y la toma de decisiones. Es presentado un método novedoso de aprendizaje supervisado para garantizar precisión y efectividad del seguimiento. Por último, el sistema resultante de la investigación fue desarrollado e implementado en un parque industrial para su verificación. Los resultados muestran que el método de seguimiento propuesto tiene una precisión de mejora constante hasta un 96,14% después del aprendizaje, en comparación con el método de identificación por radiofrecuencia (RFID), lo que garantiza una rápida capacidad de respuesta y de escalabilidad.

Zeng et al. [29] proponen un esquema de adición de datos multidimensionales y multidireccionales (MMDA) para la conservación de la seguridad en comunicaciones mejoradas de IoT enfocadas en computación en el borde. En este esquema, los datos de cada dispositivo IoT se describen como un vector dimensional (n) y los datos de IoT (m) se enumeran como una matriz (D) del orden $m \times n$. MMDA hace posible que un dispositivo perimetral agregue los datos multidimensionales de los dispositivos IoT en dos direcciones: agregación de filas y agregación de columnas. Estos datos se pueden emplear para calcular la suma de datos en cada fila y cada columna de la matriz D , a manera de preservar la privacidad (a diferencia de los esquemas de agregación de datos multidimensionales existentes que tienen solo la agregación de columnas). Esto permite la capacidad de proporcionar más información estadística a un centro de control de IoT para análisis y procesamiento. MMDA también adopta la tecnología de verificación por lotes para reducir los costos de autenticación. Por último, es reportado un amplio análisis que muestra que MMDA es factible en términos de costo de cómputo, seguridad y tolerancia a fallas.

Kuang et al. [30] exponen un modelo de sistema para entornos con múltiples usuarios, múltiples puntos de descarga y tareas estructuradas. Se formaliza un problema de decisión de descarga en un entorno de minimización de costos, esto se conoce como un problema NP-hard. Con base en lo anterior, se propuso un método basado en retroceso para obtener la solución exacta. Debido a la complejidad del método, el desempeño con respecto al tiempo es

desafortunadamente exponencial con respecto al total de puntos de descarga. Por lo tanto, para reducir la complejidad se diseñó un método basado en un algoritmo genético mejorado y otro basado en un algoritmo voraz (Greedy algorithm). Finalmente, se reporta la validación y comprobación contra otros tres métodos, en término del costo total de los usuarios, la utilización de recursos de los puntos de descarga y el tiempo de ejecución.

CAPÍTULO II

SISTEMA BCI CONECTADO A IOT

Los sistemas de interfaz cerebro computadora permiten monitorear la actividad cerebral para diagnóstico médico o procesos de control, haciendo la traducción y caracterización con base en las intenciones del usuario (esta es la aplicación de interés para el desarrollo de esta tesis). Con el paso de los años, estos sistemas se han ido popularizando y expandiendo en más aplicaciones, por lo que diversas compañías colocan en el mercado dispositivos que realizan este tipo de tareas a un precio relativamente contenido. Tal es el caso de la compañía OpenBCI, quien ofrece dispositivos de adquisición de datos electroencefalográficos, y dispositivos capaces de tratar o transmitir esta información por Bluetooth o Wifi. Por ello, OpenBCI se puede justificar como una de las mejores opciones para desarrollar una BCI embebido que esté conectado a IoT. Por tales motivos, se utilizaron los dispositivos de OpenBCI en este proyecto.

Este capítulo se organiza de la siguiente manera. La sección 2.1 habla sobre los dispositivos OpenBCI que fueron necesarios para este trabajo. La sección 2.2 muestra las configuraciones necesarias para su correcta operación. En la sección 2.3 se encuentra la metodología de obtención de datos a seguir. La sección 2.4 detalla la configuración para la transmisión de datos y satisfacer el concepto de IoT.

2.1 Dispositivos OpenBCI

El dispositivo OpenBCI es una plataforma de interfaz cerebro-computadora de código abierto creada en el año 2013. Se especializa en la creación de hardware y biosensor de alta calidad y bajo costo para las interfaces cerebro-computadora. Sus dispositivos de adquisición ofrecen capturas de señales en alta resolución a la par de grabación de señales electromiografía (EMG), electrocardiograma (ECG), y electroencefalograma (EEG).

2.1.1 Cyton

La tarjeta Cyton es una interfaz neuronal de 8 canales con un procesador de 32 bits. Implementa un microcontrolador PIC32MX250F128B, este le proporciona una alta capacidad

de memoria local y un alto rendimiento al momento de procesar la información. Los datos tienen una tasa de muestreo de 250 Hz en cada uno de los ocho canales.

Esta tarjeta puede ser usada para muestrear la actividad cerebral (EEG), la actividad muscular (EMG) y la actividad cardíaca (ECG). El dispositivo se comunica de forma inalámbrica con una computadora a través del dongle USB OpenBCI mediante módulos de radio RFDuino. También puede comunicarse de forma inalámbrica con cualquier dispositivo móvil o tableta compatible con Bluetooth Low Energy (BLE). En la Figura 2.1 se muestra a la derecha la tarjeta Cyton, mientras que a su izquierda se encuentra el dongle de comunicación.

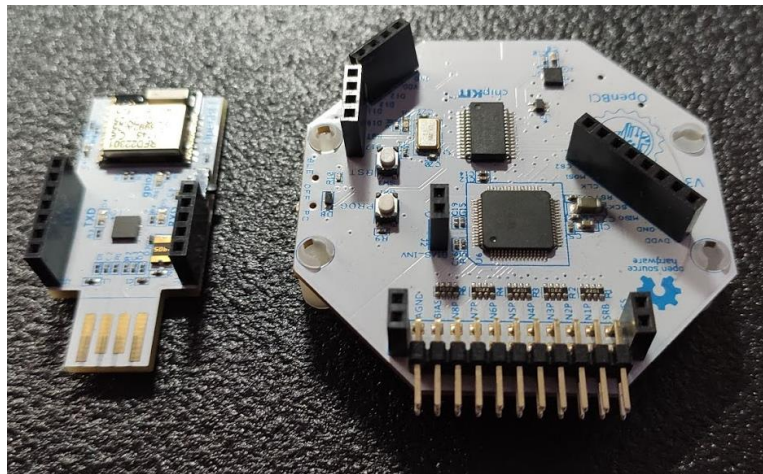


Figura 2.1. Tarjeta OpenBCI Cyton y dongle USB.

2.1.2 WiFi Shield

El módulo WiFi Shield está diseñado para funcionar con el software OpenBCI GUI y emparejarse con cualquier tarjeta OpenBCI para permitir la transmisión de biodatos a través de Wifi. Esto ofrece una ventaja frente al dongle Bluetooth que usa por defecto la tarjeta Cyton debido a que se puede alcanzar mayor tasa de transmisión de datos. El Wifi Shield se puede controlar a través de solicitudes HTTP y puede enviar objetos JSON con datos en nano voltios. En la Figura 2.2 se muestra el módulo OpenBCI Wifi Shield.



Figura 2.2. Módulo OpenBCI WiFi Shield.

2.2 Configuración de los dispositivos (Cyton + WiFi Shield)

Es necesario la adaptación y configuración adecuada de los dispositivos Cyton y WiFi Shield para garantizar una transmisión de datos óptima, de esto dependerá la fiabilidad del manejo de los datos que serán adquiridos, procesados y transmitidos. Para alcanzar este objetivo serán listados y explicados los 10 puntos fundamentales de este proceso.

1. Adaptación de módulos: se inserta el módulo OpenBCI WiFi Shield en los pines de conexión correspondientes en la tarjeta OpenBCI Cyton, el resultado de esto se visualiza en la Figura 2.3.

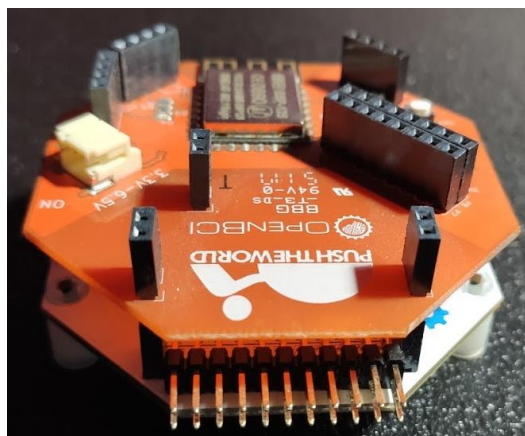


Figura 2.3. OpenBCI Cyton para operar con el módulo WiFi Shield.

2. Alimentación: se conecta la fuente de alimentación al puerto correspondiente en el módulo WiFi Shield. Ver Figura 2.4.

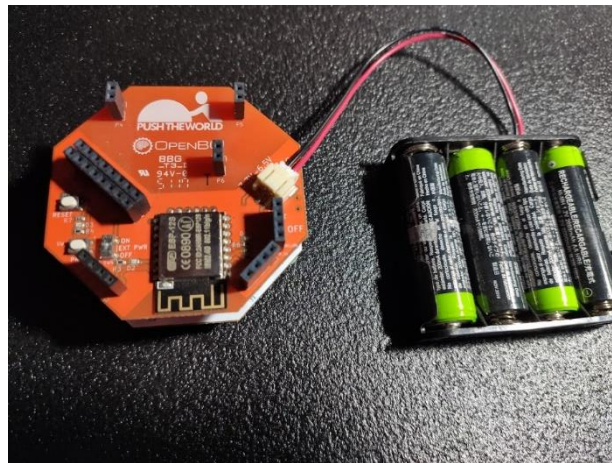


Figura 2.4. Dispositivos conectados a la fuente de alimentación.

3. Configuración de interruptores: el módulo OpenBCI WiFi Shield dispone de dos interruptores, el primero localizado en la parte lateral (encargado de alimentar dicho módulo y el dispositivo Cyton), el segundo está ubicado en la parte superior (de este depende que la tarjeta Cyton sea alimentada), para establecer conexión con la red será requerido mantener el segundo interruptor apagado (Figura 2.5), después será necesario encender el módulo WiFi (Figura 2.6).

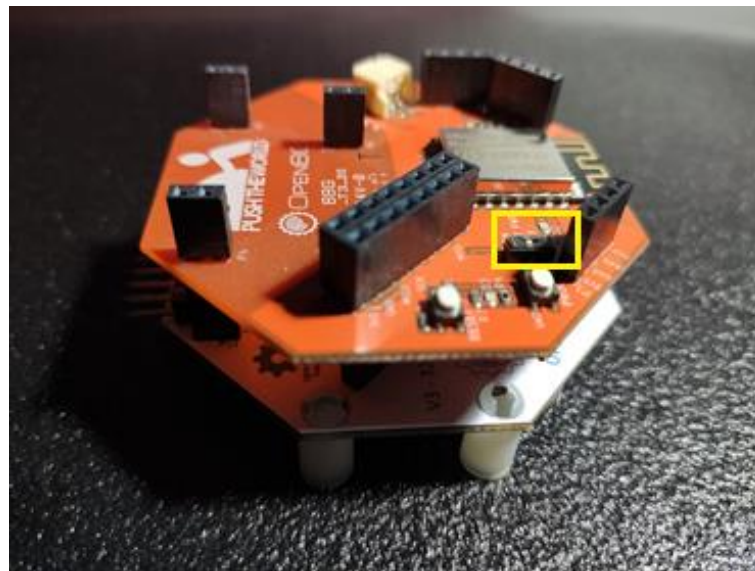


Figura 2.5. Localización del interruptor de alimentación para la tarjeta Cyton.

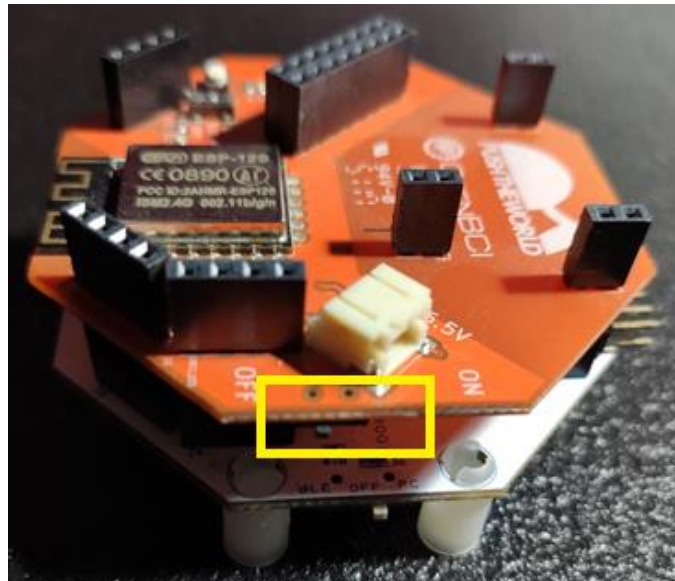


Figura 2.6. Localización del interruptor de encendido del módulo en conjunto.

4. Conexión a la red: con los dispositivos energizados, se accede a la configuración de la red Wifi y se selecciona el apartado de redes disponibles, el dispositivo OpenBCI debe aparecer entre los dispositivos disponibles para establecer conexión con él (Figura 2.7 y Figura 2.8)

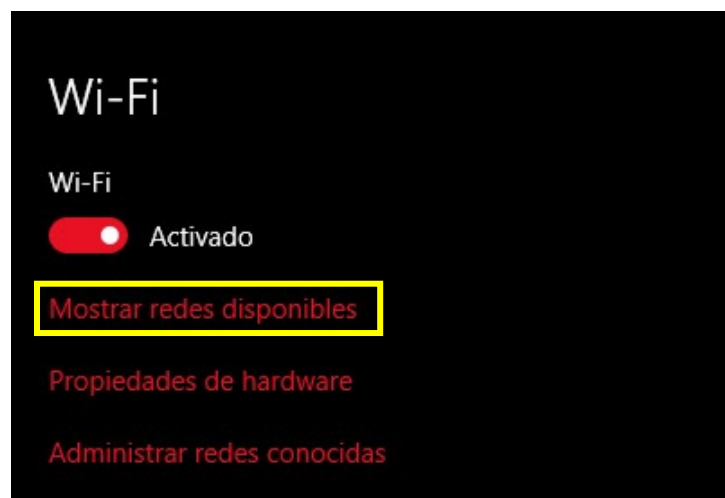


Figura 2.7. Configuración principal de Wifi en Windows.

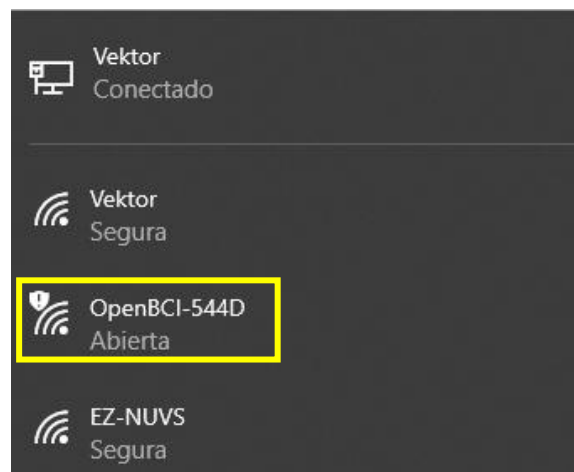


Figura 2.8. Dispositivo OpenBCI WiFi Shiled detectado por el sistema de cómputo.

5. Página de configuración: cuando se tiene la conexión establecida se ingresa desde un navegador Web a la dirección IP 192.168.4.1, esta es la dirección predeterminada del dispositivo y mostrará las configuraciones principales del mismo, Figura 2.9.



Figura 2.9. Página de configuración principal del dispositivo OpenBCI WiFi Shield.

6. Configuración Wifi 1: se hace selección sobre la opción de configuración Wifi, es posible que no se redireccione automáticamente a la página de configuraciones siguiendo este método (esto depende de la estabilidad de la conexión y el navegador que se esté utilizando). En caso de no ser redireccionado, otra opción es simplemente ingresar la dirección de configuración directamente en el navegador (<http://192.168.4.1/wifi/config>). Y será necesario seleccionar la opción señalada en la Figura 2.10.



Figura 2.10. Selección para acceder al administrador de conexión Wifi del dispositivo.

7. Configuración Wifi 2: lo anterior redirige al usuario hacia el menú que puede ser visto en la Figura 2.11, en dado caso de que la página siga cargando simplemente se debe seleccionar el botón de retroceso en el navegador que se esté utilizando (debido a un error de carga que es ocasionado por algunos navegadores basados en Chrome).



Figura 2.11. Administrador de conexión Wifi.

8. Configuración Wifi 3: en este punto se tiene la posibilidad de seleccionar la opción de configuración Wifi con o sin escaneo de red, en este caso se selecciona la primera opción. Deben aparecer las redes disponibles que estén al alcance del dispositivo (Figura 2.12). Cuando se hayan ingresado los datos necesarios para acceder a una red (contraseña e identificador) aparece un mensaje de verificación de conexión (Figura 2.13).

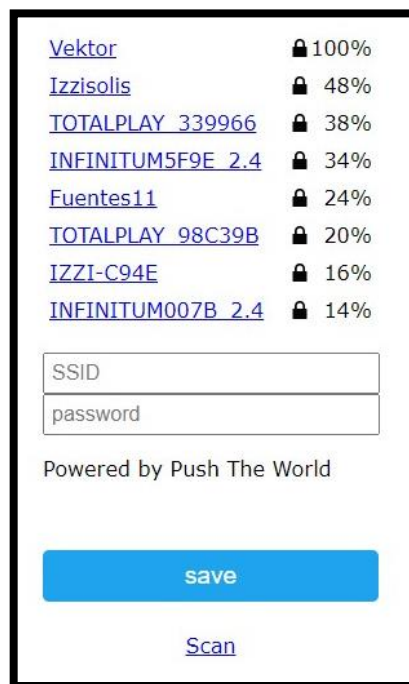


Figura 2.12. Opciones de selección de redes.



Figura 2.13. Mensaje de conexión exitosa.

9. Arranque con la conexión establecida: se debe interrumpir momentáneamente la alimentación eléctrica a los dispositivos, se cambia de posición el interruptor de la parte superior del módulo OpenBCI Wifi Shield, y se energiza nuevamente el dispositivo (la tarjeta debe establecer conexión automáticamente con la red Wifi que seleccionó, de no ser así, se debe reintentar establecer conexión desde el panel de configuración Wifi, si esto no soluciona el problema será necesario repetir el proceso desde el comienzo).

10. Finalización de configuración: con la conexión a la red Wifi la tarjeta habitualmente es asociada a una nueva dirección IP, es importante conocer esta dirección, de ella dependerá la posibilidad de adquirir los datos de las señales EEG correctamente. Existen varias formas de consultar dicha IP, la primera es consultarla directamente desde el centro de control de conexiones de Windows (o el sistema operativo que se esté utilizando), otra opción es ingresar a la configuración del módem con el que hizo conexión el dispositivo, comúnmente se puede

acceder por medio de la IP 192.168.0.1, una vez dentro de su configuración es necesario dirigirse a la sección de dispositivos inalámbricos, allí se encontrará un dispositivo llamado ESP, a este estará ligado la dirección IP con la que se debe trabajar (esto puede variar, pero en esencia es lo mismo para la mayoría de módems comerciales).

2.3 Metodología de obtención de datos

Esta metodología se refiere al proceso requerido para obtener los datos de las señales que serán manipuladas para el ingreso a la red, desde la colocación de los electrodos hasta los procesos matemáticos que son utilizados para obtener los resultados.

Para esta tesis, fue seleccionada como estrategia mental la detección de artefactos oculares en señales electroencefalográficas. Para la obtención de los datos se consideró el sistema internacional 10-20 [31], ya que en la literatura se ha utilizado ampliamente, lo que genera mayor certidumbre y fiabilidad para trabajar con él [32] [33] [34] (Figura 2.14). En este caso, se utilizan las lecturas obtenidas del sector frontal FP2.

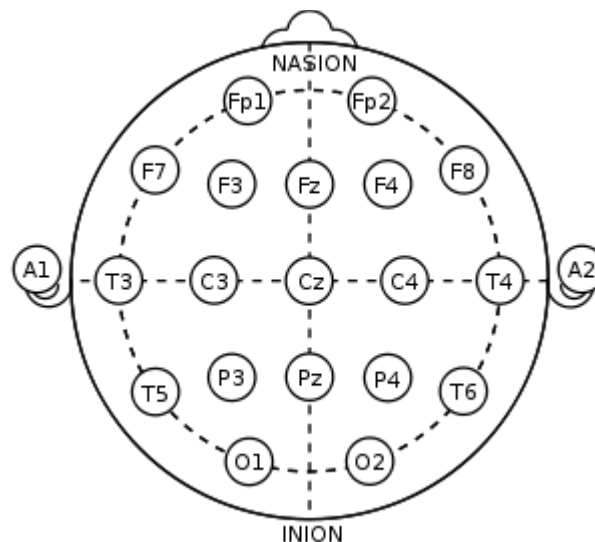


Figura 2.14. Sistema internacional 10-20.

Como fue mencionado en temas anteriores, se trabaja con el dispositivo OpenBCI Cyton y los puertos utilizados de esta plataforma son: BIAS, N2P y SRB. Esto significa que se utilizaron 3 electrodos en total. Para el lóbulo de la oreja izquierda (A1) fue posicionado el electrodo del puerto de BIAS, en el lóbulo de la oreja derecha (A2) se encuentra el electrodo de SRB, y por

último, 2.5 cm sobre el parpado derecho (FP2) se coloca el electrodo N2P. La Figura 2.15 ilustra el posicionamiento de los electrodos.

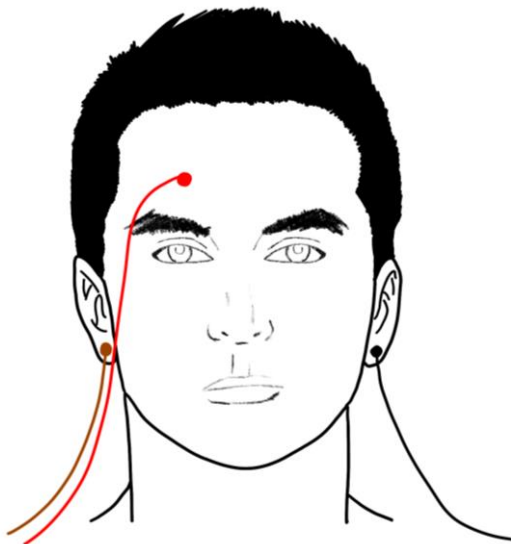


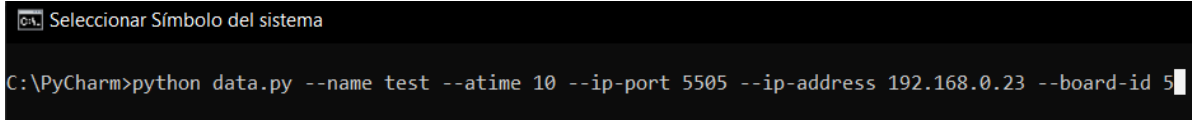
Figura 2.15. Ejemplificación de distribución de electrodos.

El electrodo SRB2 (ubicado en la parte inferior de las conexiones SRB del dispositivo Cyton) actúa como referencia midiendo la diferencia de potencial para el resto de los electrodos que sean colocados en el cuero cabelludo o en alguna otra parte del área craneal. El puerto BIAS es usado como supresor de ruido, es similar a la conexión de tierra que establece una tierra común entre el dispositivo Cyton y el cuerpo del sujeto [35].

Las lecturas adquiridas por el dispositivo Cyton son transmitidas por medio de la tarjeta OpenBCI WiFi Shield. Para ello, un programa diseñado en Python permite recibir y luego procesar esta información. El programa se apoya en una librería llamada BrainFlow, que se destina para la obtención, inspección y análisis de señales EEG, EMG, ECG y otros tipos de datos de biosensores. Entre los dispositivos compatibles se encuentra un amplio catálogo por parte de OpenBCI, lo que la vuelve ideal para trabajar con el dispositivo Cyton.

El programa es llamado desde la consola de comandos del sistema que se esté utilizando, en este caso CMD o Windows PowerShell. A él llegan varios parámetros para poder ser ejecutado de manera exitosa. Los parámetros están conformados por: llamada al script principal, nombre que se le designará al archivo (--name), tiempo de adquisición de datos en segundos (--atime),

puerto IP de comunicación (--ip-port), dirección IP (--ip-address) y el identificador de dispositivo con el que se está trabajado (--board-id). Un ejemplo de estas instrucciones es ilustrado en la Figura 2.16. Para la tarjeta Cyton el identificador corresponde al número 5, como se puede ver en [36] donde se explican los diversos ajustes de API que se logran con la librería BrainFlow.



```

Selecc
C:\PyCharm>python data.py --name test --atime 10 --ip-port 5505 --ip-address 192.168.0.23 --board-id 5

```

Figura 2.16. Ejemplo de ingreso de parámetros para la ejecución del programa de adquisición de datos.

2.3.1 Filtrado de la señal EEG

El siguiente paso es el filtrado de las lecturas en crudo que se obtienen del dispositivo, hay gran variedad de opciones para esta tarea. Sin embargo, con base en la experimentación de los datos encontrados en [37], se optó por utilizar un filtro pasa banda tipo Bessel, ya que los resultados obtenidos se asemejan fielmente a los registrados de manera estándar por la GUI proporcionada por OpenBCI.

Recordando, el dispositivo Cyton en conjunto con el módulo WiFi Shield transmiten una señal digital EEG $St(k)$ donde k es el índice de tiempo de la señal. Esta señal puede presentar altos niveles de ruido debido al entorno donde se toman las muestras, el estado de los electrodos o simplemente por interferencia cerebral de otras regiones que no son de interés. Por ello, es necesario definir un filtro a implementar (Bessel en este caso).

El filtro Bessel seleccionado está caracterizado por la Ecuación 2.1, donde $B_N(s)$ es el orden N del polinomio y s es el resultado de multiplicar un número imaginario j por la frecuencia angular ω , Ecuación 2.2. Donde los coeficientes del filtro a_k son expresados en la Ecuación 2.3.

$$H(s) = \frac{1}{B_N(s)} \quad (2.1)$$

$$B_N(s) = \sum_{k=0}^N a_k s^k \quad (2.2)$$

$$a_k = \frac{(2N - k)!}{2^{N-k} k! (N - k)!} \quad k = 0, 1, \dots, N \quad (2.3)$$

Los polinomios de Bessel pueden generarse de forma recursiva a partir de la siguiente expresión:

$$B_N(s) = (2N - 1)B_{N-1}(s) + s^2 B_{N-2}(s) \quad (2.4)$$

Con $B_0(s) = 1$ y $B_1(s) = s + 1$. La señal $St(k)$ hace una operación en diferencias con el filtro $H(s)$ obteniendo la señal filtrada $X_c(k)$ y puede ser visto en la Ecuación 2.5. Donde M es el tamaño del filtro (en este caso de cuarto orden) y ri son los coeficientes de la respuesta al impulso del filtro.

$$X_c(k) = \sum_{ri=0}^{M-1} St(k - ri) \cdot H(ri) \quad (2.5)$$

El filtro fue diseñado con los siguientes parámetros: centro de frecuencia de 15 Hz, ancho de banda de 6 Hz, y corresponde a un filtro de cuarto orden. El comportamiento del filtro en frecuencia y fase se aprecia en la Figura 2.17.

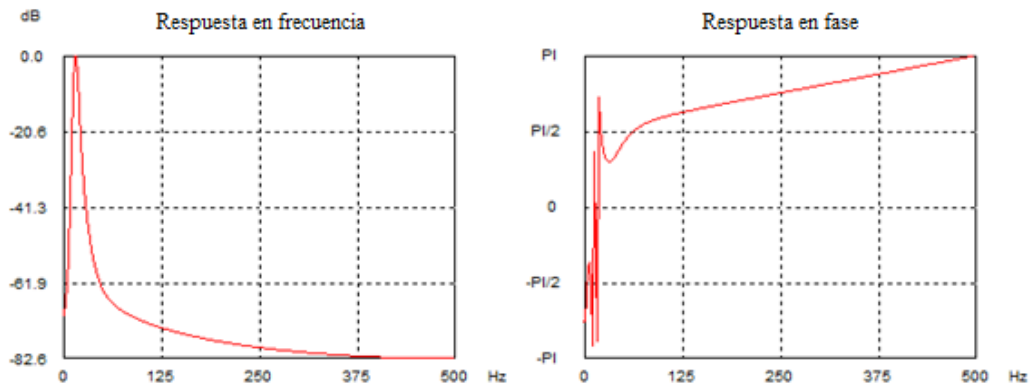


Figura 2.17. Respuesta del filtro Bessel pasa banda en frecuencia y fase.

Cuando se tienen los resultados de la señal filtrada $X_c(k)$ se continua a la siguiente etapa de procesamiento, este paso corresponde al recorte de datos, para este trabajo se estandarizó la

lectura en un lapso de 10 segundos, a una frecuencia de lectura de 1,000 Hz, lo que implica que se obtendrán señales con 10,000 datos, del total de datos se suprime el primer segundo de lecturas, debido a que durante estas primeras 1,000 muestras se presentan alteraciones de ruido indeseadas en la señal, esto da como resultado final una señal filtrada recortada a 9,000 muestras llamada $y(k)$. Por último, el recorte de esta señal es almacenado en un archivo de valores separados por comas (CSV), para su posterior conversión al dominio de la frecuencia y su evaluación en la red neuronal artificial.

2.4 Transmisión a la red

Para satisfacer el paradigma de computación en el borde es necesario realizar la transmisión de los resultados a la red, de esta manera estarán a disposición de los dispositivos de IoT que requieran alguno de estos datos.

Nuevamente, basándose en un programa diseñado en Python, se ideó la forma para realizar esta transmisión. Utilizando el Protocolo de Transferencia de Archivos (FTP), este es uno de los protocolos más antiguos utilizados en Internet, este protocolo está diseñado para transferir archivos de una ubicación a otra. Es bidireccional, lo que significa que se puede utilizar tanto para cargar como para descargar archivos [38]. Este protocolo de red está diseñado para transferir archivos entre sistemas conectados a una red con Protocolo de Control de Transmisión (TCP). En la Figura 2.18 se muestra un esquema general del funcionamiento de este protocolo.

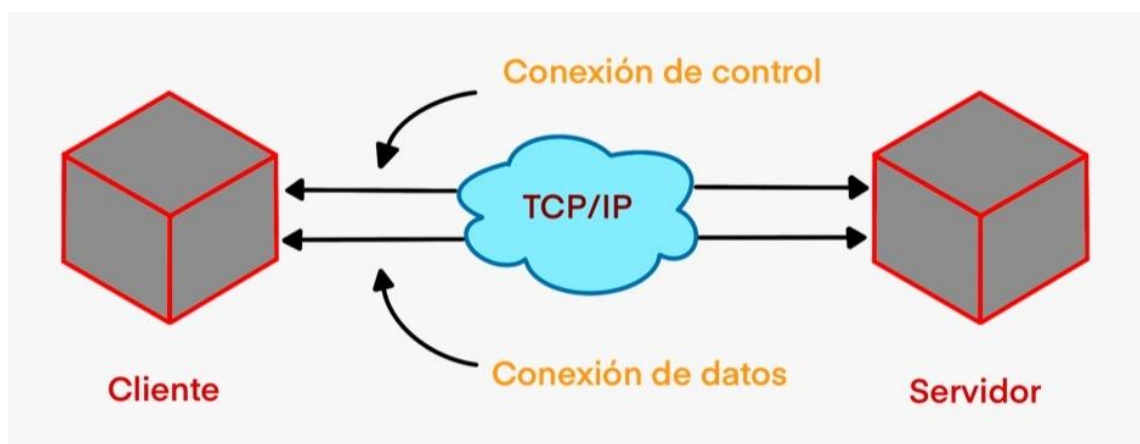


Figura 2.18. Protocolo de transferencia de archivos.

Para satisfacer el apartado del servidor es empleado el panel de control de administración de servidores *cPanel*, este panel provee de herramientas de automatización y una interfaz gráfica basada en páginas web.

La librería de Python *ftplib* utiliza el protocolo de transferencia de archivos ingresando los parámetros de: dirección IP del servidor (107.180.41.70, en este caso), nombre de acceso de la cuenta FTP, y la contraseña de esta cuenta. A partir de esto fue completado el resto del programa con la selección de archivos y la escritura de estos en el servidor.

La visualización de resultados es sostenida por una página web escrita en PHP alojada en el servidor, para acceder a esta página se tiene que ingresar a la dirección web <http://proyectoservicio2018.info/Galahad/>, la página muestra los resultados que fueron adquiridos, procesados y finalmente subidos al servidor, también mostrando la ilustración de la señal con los artefactos detectados y el dato de la cantidad de estos. La Figura 2.19 contiene un fragmento de esta página junto a los resultados que son exhibidos.

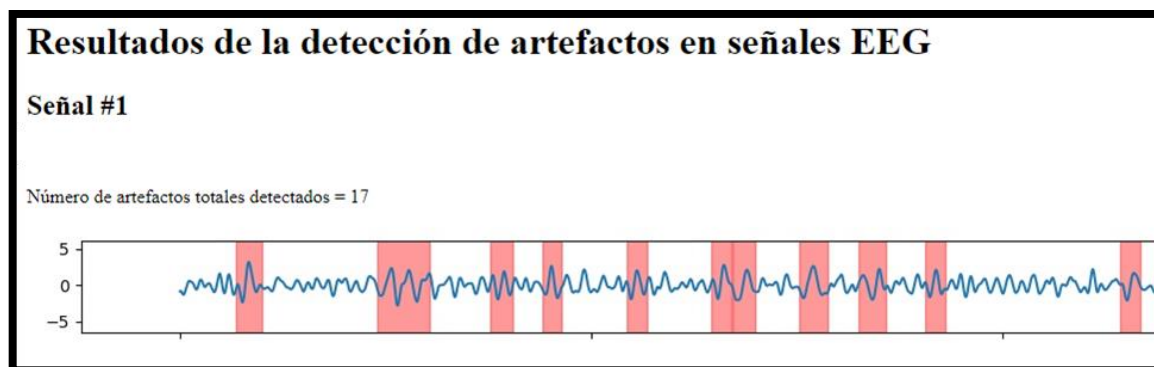


Figura 2.19. Ejemplo de resultado de detección de artefactos mostrado en la web correspondiente a $y(k)$.

CAPÍTULO III

RED NEURONAL DE RECONOCIMIENTO EN EL BORDE

Las redes neuronales artificiales (RNA) es el nombre que se le da a una rama de la inteligencia artificial (IA) y su objetivo es simular funcionamiento de las redes neuronales biológicas. En 1943 Warren McCulloch y Walter Pitts propusieron un modelo matemático en el que los cambios sinápticos son continuos en la neurona, este modelo es la base de las redes neuronales artificiales contemporáneas. Para el año de 1957 surge el primer perceptrón artificial, creado por Frank Rosenblatt. Este algoritmo de reconocimiento de patrones está basado en una red de aprendizaje por computadora de dos capas, el algoritmo utiliza operaciones matemáticas básicas, como la adición y substracción [39]. No obstante, debido a la publicación de Marving Minsk [40], las RNA fueron consideradas poco útiles durante una década. Sin embargo, desde 1974 han progresado en el diseño y desarrollo de estos algoritmos, siendo hoy día de los más utilizados para propósitos generales como tareas de análisis de datos de reconocimiento de patrones [41].

Entre los tipos de redes neuronales existentes se destacan las redes supervisadas y no supervisadas. Las redes no supervisadas requieren de datos sin etiquetar para realizar el aprendizaje. Por otro lado, las redes supervisadas aprenden mediante un entrenamiento controlado por datos ya etiquetados [42].

Debido a las características de identificar artefactos oculares identificados para el diseño de este proyecto se optó por seguir un aprendizaje de red supervisado. Por ello, este capítulo de organiza de la siguiente forma: la primera sección 3.1 explica los conceptos básicos de una RNA y el diseño que se siguió para la implementación de la red. La sección 3.2 da los fundamentos necesarios para comprender el método de entrenamiento que fue utilizado en la red de este proyecto.

3.1 Red Neuronal Artificial

La estructura de una neurona abstracta de c entradas se muestra en la Figura 3.20. Cada canal de entrada i transmite un valor real X_i y tienen un peso asociado además de un sesgo b , esto

implica que la información entrante de X_i se multiplica por el peso W_i correspondiente. La sumatoria de sesgos y pesos o la salida de la neurona se llama f . La información transmitida se integra en la neurona y luego se evalúa la función primitiva, generalmente agregando las diferentes señales [43].

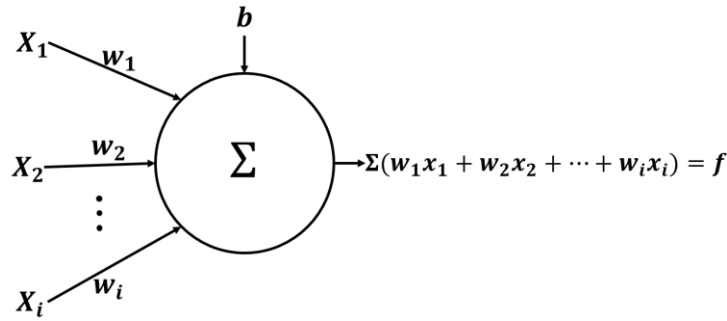


Figura 3.20. Estructura de neurona abstracta.

El tipo de red utilizada en este proyecto es de una topología de red neuronal prealimentada [44]. Esta RNA se distingue por las conexiones entre los nodos, debido a que no forman una retroalimentación. Este tipo de red fue la primera y más simple de las redes neuronales artificiales que fueron diseñadas. En esta red, la información va en una únicamente hacia adelante, desde los nodos de entrada, a través de los nodos ocultos hacia los nodos de salida. Esta particularidad hace que sean utilizadas principalmente para el aprendizaje supervisado en casos donde los datos que serán aprendidos no son secuenciales y no dependen del tiempo.

Para el diseño de la red de este proyecto se consideran los conjuntos de muestras $y(k)$ de 9,000 datos tomados después del recorte de ruido y guardados en archivos CSV. Para hacer una estimación de datos que ocupa un artefacto ocular se realizó un estudio con 200 muestras EEG extraídas del mismo sujeto en condiciones ideales mientras se realizaban 3 parpadeos voluntarios, de esto se identificaron manualmente los artefactos voluntarios y la cantidad de datos que ocupan en la señal, para finalmente obtener un resultado promedio de 170 datos cada vez que se presentaba un artefacto ocular de interés. Para facilitar el entrenamiento e identificación de los artefactos oculares por parte de la RNA estas señales son convertidas al dominio de la frecuencia con la transformada rápida de Fourier. Esto quiere decir que se utilizarán los primeros 85 datos de cada señal en el dominio de la frecuencia con los que se

alimenta a la RNA para el entrenamiento. Fueron adquiridas y recortadas manualmente 300 señales de muestra para el entrenamiento de la red, 150 de esas muestras correspondientes a artefactos oculares voluntarios, y las 150 muestras restantes corresponden a situaciones donde no se presentaron artefactos voluntarios o de ningún tipo. Por lo tanto, las entradas de la RNA se definen por:

$$X(l) = \sum_{k=0}^{N-1} y(k) \cdot e^{-j2\pi k/N} \quad (3.1)$$

Donde $X(l)$ son los componentes de frecuencia de $y(k)$ (estos se mencionan al inicio de la sección 2.3.1).

La RNA con mejores resultados que se obtuvo mediante la experimentación está compuesta por 3 capas, 85 neuronas de entrada, 35 neuronas en la capa oculta, y una neurona de salida, una representación de la arquitectura se muestra en la Figura 3.21. La agregación de la capa oculta de la red está dada por f , donde p es el índice de la neurona, esta ecuación se expresa como:

$$f_{1p} = \sum_{R=0}^{34} W_{1p} X(l) + b_{1p} \quad (3.2)$$

El ingresar los valores f_{1p} a la función de activación sigmoideal usada en la capa oculta resulta en la siguiente expresión:

$$u_{1p} = \varphi_{1p}(f) = \frac{1}{1 + e^{-f_{1p}}} \quad (3.3)$$

El valor de la neurona de la capa de salida se describe con la ecuación:

$$f_2 = W_2 u_{1R} + b_2 \quad (3.4)$$

Finalmente, la expresión correspondiente al paso de la salida f_2 por la función de activación u se representa como:

$$u_2 = \varphi_2(f) = \frac{1}{1 + e^{-f_2}} \quad (3.5)$$

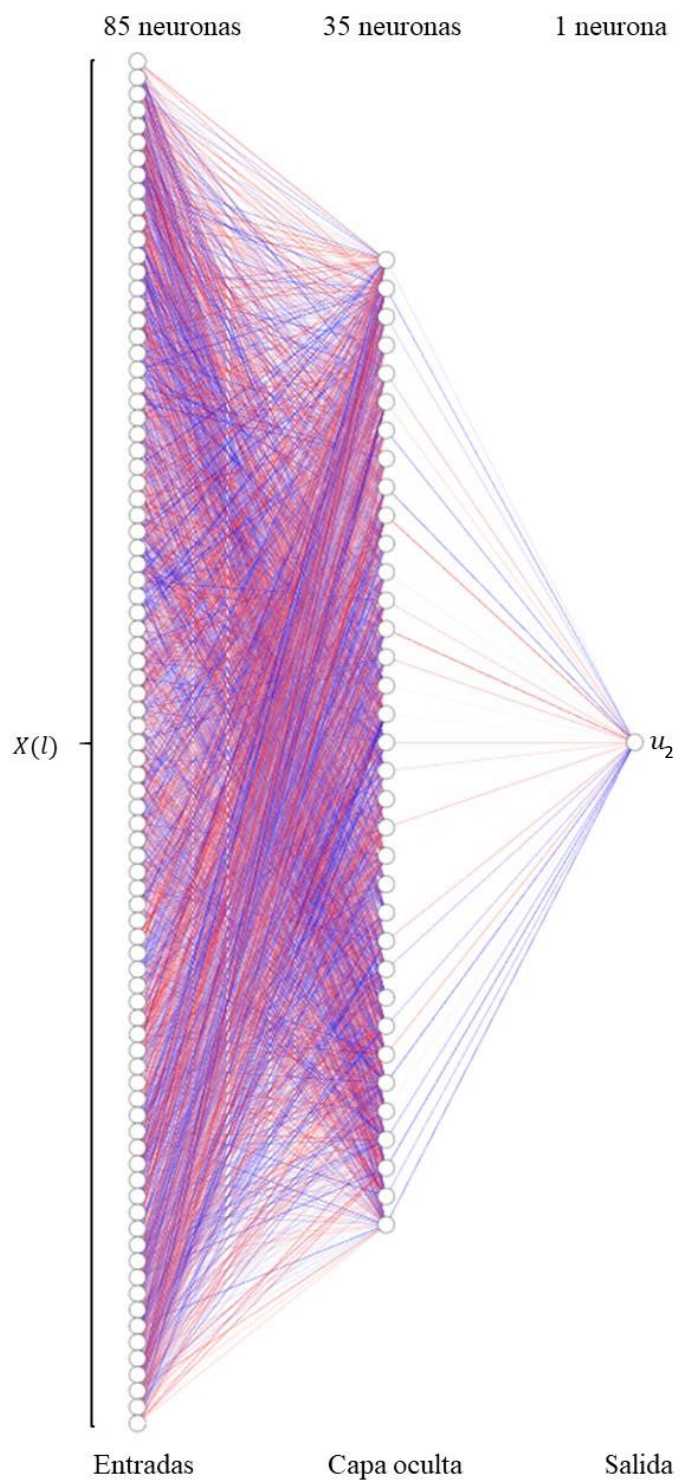


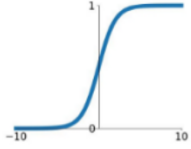
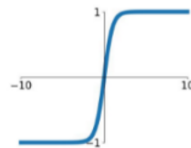
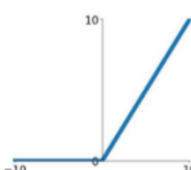
Figura 3.21. Arquitectura resultante de la RNA.

3.2 Entrenamiento

Como algoritmo de aprendizaje se emplea la retropropagación (mayormente conocido en inglés como *backpropagation*). Este es un método de cálculo del gradiente que es utilizado en algoritmos de aprendizaje supervisado para el entrenamiento de redes neuronales artificiales, ejemplos de este tipo de implementación se presentan en [45], [46] y [47].

El algoritmo de retropropagación busca el valor mínimo de la función de error en el espacio de pesos, utilizando el método de descenso del gradiente [43]. La combinación de pesos que minimiza la función de error se considera una solución del problema de aprendizaje. Debido a que este método requiere el cálculo del gradiente de la función de error en cada paso de la iteración, por lo que es necesario garantizar la continuidad y diferenciabilidad de la función de error. Para el cálculo de la retropropagación se usa una función de activación diferente (sigmoidal) a la función escalón que es usada por el perceptrón, porque la función que se produce por la red es discontinua, por lo tanto, la función del error también lo es. En la Tabla 3.1 se muestran las funciones de activación más comúnmente aplicadas en la literatura.

Tabla 3.1. Funciones de activación.

Nombre	Grafica	Ecuación	Derivada
Sigmoidal		$\varphi(f) = \frac{1}{1 + e^{-f}}$	$\varphi'(f) = o(f)(1 - o(f))$
Tangente hiperbólica		$\varphi(f) = \tanh(f) = \frac{(e^f - e^{-f})}{(e^f + e^{-f})}$	$\varphi'(f) = 1 - o(f)^2$
Unidad lineal rectificadora (RELU)		$\varphi(f) = \begin{cases} 0 & \text{para } f < 0 \\ f & \text{para } f \geq 0 \end{cases}$	$\varphi'(f) = \begin{cases} 0 & \text{para } f < 0 \\ 1 & \text{para } f \geq 0 \end{cases}$

La formación de la retropropagación se desarrolla en dos fases:

- En la fase de avance, los pesos sinápticos de la red son fijos y la señal de entrada se propaga a través de la red, capa por capa, hasta llegar a la salida. Por tanto, en esta fase, los cambios se limitan a los potenciales de activación y salidas de las neuronas en la red.
- En la fase inversa, se produce una señal de error al comparar la salida de la red con una respuesta deseada. La señal de error resultante se propaga a través de la red, nuevamente capa por capa, pero esta vez la propagación se realiza en la dirección hacia atrás.

Para las operaciones se considera el campo local inducido $v_p(f)$ que se produce en la entrada de la función de activación de la neurona p , esta expresión se aprecia en la Ecuación 3.6. Donde m es el número total de entradas (excluyendo el sesgo). Por lo tanto, la señal de función $u_p(f)$ se expresa como la Ecuación 3.7.

$$v_p(f) = \sum_{q=0}^m w_{pq}(f) u_q(f) \quad (3.6)$$

$$u_p(f) = \varphi_p(v_p(f)) \quad (3.7)$$

En el algoritmo de retropropagación se aplica una corrección $\Delta w_{pq}(f)$ al peso sináptico $w_{pq}(f)$, que es proporcional a la derivada parcial $\frac{\partial \xi(f)}{\partial w_{pq}(f)}$ (Ecuación 3.8).

$$\frac{\partial \xi(f)}{\partial w_{pq}(f)} = \frac{\partial \xi(f)}{\partial e_p(f)} \frac{\partial e_p(f)}{\partial u_p(f)} \frac{\partial u_p(f)}{\partial v_p(f)} \frac{\partial v_p(f)}{\partial w_{pq}(f)} \quad (3.8)$$

Haciendo una sustitución con las equivalencias de las diferenciales mostradas en las Ecuaciones (3.9) - (3.12), resulta en la expresión matemática de la Ecuación 3.13.

$$\frac{\partial \xi(f)}{\partial e_p(f)} = e_p(f) \quad (3.9)$$

$$\frac{\partial e_p(f)}{\partial u_p(f)} = -1 \quad (3.10)$$

$$\frac{\partial u_p(f)}{\partial v_p(f)} o = \phi_p(v_p(f)) \quad (3.11)$$

$$\frac{\partial v_p(f)}{\partial w_{pq}(f)} = u_q(f) \quad (3.12)$$

$$\frac{\partial \xi(f)}{\partial w_{pq}(f)} = -e_p(f) \phi_p(v_p(f)) u_q(f) \quad (3.13)$$

La corrección del gradiente descendiente $\Delta w_{pq}(n)$ está definida por la regla delta (Ecuación 3.14), con los despejes necesarios se llega a la expresión para el cálculo de las actualizaciones de delta (Ecuación 3.15).

$$\Delta w_{pq}(f) = -\eta \frac{\partial \xi(f)}{\partial w_{pq}(f)} \quad (3.14)$$

$$\delta_p(f) = e_p(f) \phi_p(v_p(f)) \quad (3.15)$$

El comportamiento de una neurona basado en su posición se compone de:

- La neurona es una salida. Cuando la neurona p está ubicada en la capa de salida de la red, recibe una respuesta deseada propia. Habiendo determinado $e_p(f)$ (error de salida), resulta sencillo el cálculo del gradiente local $\delta_p(f)$.
- La neurona está en la capa oculta. Cuando la neurona p se encuentra en una capa oculta de la red, no existe una respuesta deseada específica para esa neurona. En consecuencia,

la señal de error para una neurona oculta debería determinarse de forma recursiva y hacia atrás en términos de las señales de error de todas las neuronas a las que esa neurona oculta está conectada directamente.

Considerando la situación de la Figura 3.22, que muestra la neurona p como un nodo oculto de la red. El gradiente local $\delta_p(f)$ para la neurona oculta p se expresa en la Ecuación 3.16.

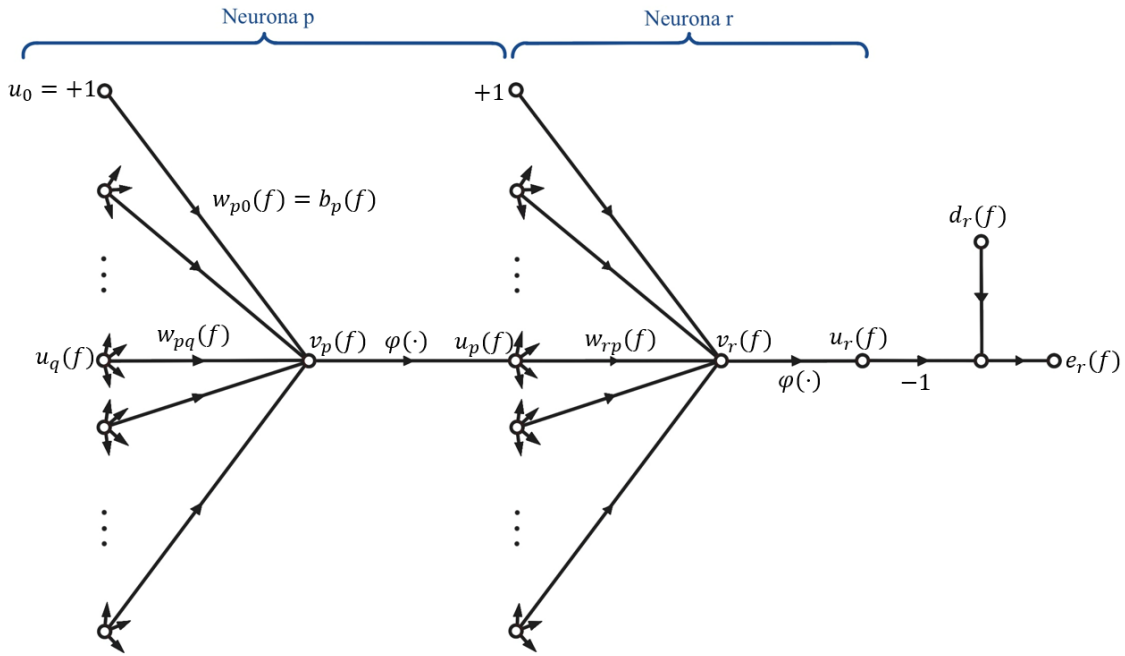


Figura 3.22. Neurona p como un nodo oculto de la red.

$$\delta_p(f) = -\frac{\partial \xi(f)}{\partial u_p(f)} \frac{\partial u_p(f)}{\partial v_p(f)} = -\frac{\partial \xi(f)}{\partial u_p(f)} \phi_p(v_p(f)) \quad (3.16)$$

Para obtener el delta local es necesario tomar la función de costo (Ecuación 3.17). Donde el conjunto $\mathcal{C}$ incluye todas las neuronas en la capa de salida. Diferenciando la expresión anterior con respecto a u_p se obtiene la Ecuación 3.18. Por la regla de la cadena (Ecuación 3.19), al conocer el error $e_r(f) = d_r - \varphi_r(v_r(f))$, donde d es la salida esperada, se consigue la Ecuación 3.20.

$$\xi(f) = \frac{1}{2} \sum_{r \in C} e_r^2(f) \quad (3.17)$$

$$\frac{\partial \xi(f)}{\partial u_p(f)} = \sum_r e_r \frac{\partial e_r(f)}{\partial u_p(f)} \quad (3.18)$$

$$\frac{\partial \xi(f)}{\partial u_p(f)} = \sum_r e_r \frac{\partial e_r(f)}{\partial v_r(f)} \frac{\partial v_r(f)}{\partial u_p(f)} \quad (3.19)$$

$$\frac{\partial e_r(f)}{\partial v_r(f)} = -\phi_r(v_r(f)) \quad (3.20)$$

Por otro lado, tomando $v_r = \sum_{p=0}^m w_{rp}(f)u_p(f)$, m siendo el número total de entradas aplicadas a la neurona r , resulta en la Ecuación 3.21. De esto se puede volver a expresar $\frac{\partial \xi(f)}{\partial u_p(f)}$ como la Ecuación 3.22 o la Ecuación 3.23.

$$\frac{\partial v_k(f)}{\partial u_p(f)} = w_{kp}(f) \quad (3.21)$$

$$\frac{\partial \xi(f)}{\partial u_p(f)} = \sum_r e_r \frac{\partial e_r(f)}{\partial v_r(f)} \frac{\partial v_r(f)}{\partial u_p(f)} = - \sum_r e_r \phi_r(v_r(f)) w_{rp}(f) \quad (3.22)$$

$$\frac{\partial \xi(f)}{\partial u_p(f)} = - \sum_r \delta_r(f) w_{rp}(f), \text{ donde } \delta_r(f) = e_r \phi_r(v_r(f)) \quad (3.23)$$

Para la obtención del gradiente local se utiliza la Ecuación 3.24. Con el conocimiento de las expresiones anteriores se define finalmente la retropropagación como la Ecuación 3.25.

$$\delta_p(f) = - \frac{\partial \xi(f)}{\partial u_p(f)} \phi_p(v_p(f)) \quad (3.24)$$

$$\delta_p(f) = \sum_r \delta_r(f) w_{rp}(f) \phi_p(v_p(f)) \quad (3.25)$$

Como ya fue mencionado, se tienen 150 fragmentos de señales de artefactos y 150 correspondientes a cualquier otra parte de una lectura EEG que no corresponda a un artefacto, 90 señales de cada uno de estos grupos fue utilizado para el entrenamiento, mientras que para la etapa de comprobación de rendimiento fueron usadas 60 señales de cada grupo. Se recorre cada señal a analizar dato por dato creando conjuntos de 170 datos, posteriormente son pasadas al dominio de la frecuencia para finalmente ser evaluadas por la red.

CAPÍTULO IV

DISPOSITIVOS

Para la implementación de la red, se consideraron varios dispositivos de aplicaciones embebidas en tendencia encontrados en la literatura, la disposición de compatibilidad con IoT, y los lenguajes necesarios para su manejo. La Raspberry Pi [48], Arduino [49] e inclusive la Intel Edison [50] fueron consideradas al momento de buscar un sistema de cómputo compacto con buenas prestaciones de hardware, debido a que son de fácil acceso y son muy utilizadas en los trabajos encontrados en la literatura. Por el lado de los FPGAs se experimentó con el dispositivo Xilinx Cora Z7 por ser un dispositivo compacto y de fácil adquisición. De esta manera, en este capítulo se abordarán los tópicos concernientes a la implementación de la red neuronal diseñada para este proyecto. La sección 4.1 habla sobre el dispositivo seleccionado (UP Board), la configuración de este y la implementación de la RNA. En la sección 4.2 se describe la experimentación de implementar el diseño de la RNA en un dispositivo de matriz de puertas lógicas reconfigurables (FPGA), el hardware con el que se contaba y los retos que esto conlleva.

4.1 UP Board

La tarjeta UP Board (Figura 4.1) es un dispositivo de dimensiones contenidas con características de alto rendimiento y bajo consumo de energía. Cuenta con una unidad de procesamiento gráfico (GPU) interna que le otorga un rendimiento gráfico extremadamente alto para un dispositivo de esta gama de precio. Este dispositivo tiene 40 puertos de entrada y salida con la misma definición de pines del dispositivo Raspberry PI2, lo que da mayor versatilidad a la hora de integrar algún módulo externo (en dado caso de ser necesario). Al ser gestionada por un procesador de marca Intel, se ofrecen las funciones de seguridad necesarias para aplicaciones profesionales de IoT, como las instrucciones de Inter AES y la tecnología de protección de identidad [51].

Después de hacer comparaciones técnicas de las especificaciones respecto al abanico de dispositivos disponibles en el mercado, se seleccionó el dispositivo UP Board por las siguientes razones:

- Recursos computacionales adecuados para esta aplicación.
- Facilidad de integración con los sistemas operativos necesarios.
- Sus dimensiones reducidas la hacen ideal para aplicaciones de IoT.
- Disponibilidad inmediata.



Figura 4.1. Dispositivo UP Board.

4.1.1 Especificaciones del dispositivo

El dispositivo UP Board integra un procesador Intel ATOM x5-Z8350 de 64 bits que opera hasta 1,92 GHz. La GPU interna es la Intel Gen 8 HD 400 con 12 unidades de ejecución de hasta 500 MHz para ofrecer un rendimiento gráfico 3D alto considerando que se trata de un dispositivo para aplicaciones embebidas y de IoT. El modelo en específico que se está utilizando cuenta con 4 GB de RAM DDR3L y 64 eMMC.

La Tabla 4.1 contiene las capacidades computacionales y de conexiones que formaron parte de la decantación por la selección de este dispositivo.

Tabla 4.1. Especificaciones técnicas del dispositivo UP Board.

Sistema en Chip (SoC)	Intel® Atom™ x5-z8350 (hasta 1.92 GHz)
Número de núcleos	Quad Core
Gráficos	Intel® HD 400 Graphics
Memoria del sistema	4 GB
Capacidad de almacenamiento	64 GB

BIOS	UEFI
RTC	Disponible
Activación de la LAN	Disponible
Temporizador de vigilancia	Disponible
Entorno de ejecución de prearraque	Disponible
Sistemas operativos compatibles	Microsoft Windows 10, Linux (ubilinux, Ubuntu, Yocto), Android
Alimentación eléctrica necesaria	5V DC-in @ 4A 5.5/2.1mm jack
Tipo de fuente de alimentación	AT/ATX (AT predeterminado)
Dimensiones (L x A)	85.6mm x 56.5mm
Temperatura de operación	0°C ~ 60°C
Temperatura de almacenamiento	-40°C ~ 80°C
Humedad de funcionamiento	0% ~ 90% de humedad relativa, sin condensación
Tiempo medio entre fallos (horas)	270,599 horas
Certificación eléctrica	CE, FCC Clase A
HDMI	1 (HDMI 1.4a)
MIPI DSI/ eDP	MIPI DSI/eDP
Audio	I2S puerto de audio
USB	4 x USB2.0 puerto tipo A 2x USB2.0 pin 1 x USB 3.0 OTG Micro B
UART	2 puertos de depuración UART (Tx / Rx)
40-pin GP-bus	1
MIPI-CSI	1x MIPI-CSI 2 carriles
Ethernet	1 x GbLAN (Realtek RTL8111G-CG)

Wifi	Posibilidad de implementar módulo
------	-----------------------------------

4.1.2 Adecuación del dispositivo

El proceso para volver el dispositivo UP Board un entorno de desarrollo favorable para el proyecto constó de los siguientes puntos:

- Selección del sistema operativo.
- Instalación del software de diseño necesario.
- Optimización de recursos.

El mercado de los sistemas operativos (SO) es variado y creciente, pero son destacables 3 de ellos: GNU/Linux, Windows y macOS. En el campo de Linux se puede encontrar un catálogo de distribuciones que se va enriqueciendo día a día, pero, en esencia Linux es gratuito de código abierto, multiplataforma, multiusuario y multitarea. Windows es un sistema operativo de pago que domina el mercado de los sistemas de cómputo personal, por lo que muchas de las aplicaciones existentes están disponibles o son diseñadas para este SO.

Tanto Linux (Ubuntu, Debian, Red Hat, etc.) como Windows cumplen las características deseadas para el diseño e implementación del producto de Edge Computing y EEG sobre la tarjeta UP Board. Sin embargo, con base en las consideraciones de accesibilidad y facilidad se trabajó con una versión estable de Windows 10 pro (con una depuración manual de software de relleno). No obstante, debido a la forma en la que fue diseñado este proyecto los programas resultantes pueden transferirse y ser escalados a cualquier otro de los sistemas operativos que soporten la versión de Python 3.4 en adelante.

Como fue mencionado en capítulos previos, Python es parte clave de la programación del proyecto, en específico se operó con la versión 3.8 de 64 bits y fue complementado con paquetes o módulos gratuitos que facilitan la codificación del diseño. Las instalaciones externas de módulos junto a una breve descripción de su funcionalidad se listan como:

- *BrainFlow* en su versión 3.9.2 se encarga de analizar las señales EEG que adquiere el dispositivo biosensor Cyton OpenBCI.
- *Matplotlib* se destina para la generación de gráficos a partir de datos que estén contenidos dentro de listas o arreglos.

- *NumPy* da soporte para la creación de vectores y matrices multidimensionales comúnmente de un gran número de datos, además de disponer de un gran repertorio de funciones matemáticas de alto nivel.
- *Pandas* es una biblioteca de software escrita con la extensión NumPy para manipular y analizar datos. Brinda estructuras de datos y operaciones para la manipulación de tablas numéricas y series temporales.
- *Pathlib* es un módulo que ofrece clases que representan las rutas del sistema de archivos para su manejo rápido.
- *SciPy* contiene módulos que son usados para optimización, álgebra lineal, integración, interpolación, funciones especiales, y transformadas de Fourier al igual que procesamiento de señales e imágenes.

Como herramienta adicional, fue instalado por medio de los pines USB (puerto CN7) del dispositivo UP Board un módulo Wifi (diseñado por la misma compañía), lo que facilita el manejo del dispositivo para aplicaciones de IoT. La Figura 4.2 muestra los componentes que acompañan al módulo y la Figura 4.3 señala dentro de un rectángulo rojo el puerto de conexión del módulo.



Figura 4.2. Módulo Wifi para la tarjeta UP Board.

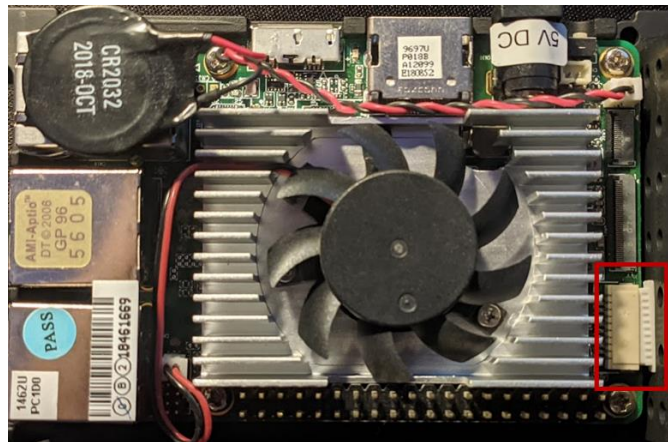


Figura 4.3. Puerto de conexión CN7 de la tarjeta UP Board.

4.1.3 Implementación del software diseñado e integración del sistema

Con los resultados de pesos y sesgos obtenidos del entrenamiento usando las Ecuaciones (3.6) - (3.25) de la red descrita en el capítulo 3, se procede a la creación de la red de detección de artefactos. Para ello, se vuelve a utilizar Python como lenguaje de desarrollo y la misma arquitectura implementada en el entrenamiento (85 entradas, 35 neuronas en la capa oculta y 1 neurona de salida). Debido a que la RNA diseñada en este proyecto no se apoya en ninguna biblioteca o framework especializado en redes neuronales se tiene la oportunidad de crear un sistema embebido escalable con mejor rendimiento en dispositivos de cómputo pequeños o con hardware limitado.

Con la RNA de detección finalizada se tienen todos los módulos necesarios para crear la BCI para el sistema de computación en el borde. El proceso del sistema completo consta de:

- Adquisición de datos: Con los electrodos conectados al sujeto, la tarjeta Cyton recibe las lecturas EEG para después ser transmitidas al UP Board por medio del OpenBCI Wi-Fi Shield.
- Filtrado de la señal: Las lecturas obtenidas son procesadas por un filtro Bessel (dado por las Ecuaciones (2.1) – (2.4) vistas en la sección 2.3) para reducir posibles alteraciones de ruido en la señal.
- Recorte del primer segundo de muestras: Después de un gran número de pruebas se llegó a la conclusión de que el primer segundo de adquisición de señales

(correspondiente a 1,000 datos) presenta un pico de ruido externo, el cual se suprime para la optimización de tiempos de procesamiento en la RNA de detección. La Figura 4.4 señala la aproximación del recorte de datos registrados como ruido.



Figura 4.4. Recorte de la sección de ruido de la señal adquirida.

- **Conversión del dominio del tiempo al dominio de la frecuencia:** El manejo de los datos se hace en el dominio de la frecuencia (Ecuación 3.1) debido a la mejora sustancial que se obtuvo al entrenar la RNA con señales en el dominio de la frecuencia, ya que ofrece una mejor separabilidad a la hora de ser clasificados. Esta conversión se hace con un ventaneo de 170 datos, lo que implica una salida de 85 datos por cada conversión.
- **Ingreso de los datos para su evaluación en la RNA:** Las 85 entradas de la RNA son los datos de salida de la conversión a Fourier $X(l)$ explicada en el punto anterior. La red cuenta con los pesos W y sesgos b que alcanzaron un mejor comportamiento en el entrenamiento. La información pasa a la capa oculta de la red expresada en la Ecuación 3.2 y el resultado ingresa a la función de activación (Ecuación 3.3). Después se analiza el valor de la capa de salida (Ecuación 3.4) para finalmente pasar por una función de activación sigmoideal (Ecuación 3.5). Cuando un artefacto es detectado se guarda la información de apertura y cierre dónde se presentó y al terminar de analizar la señal se hace un conteo de artefactos detectados y una imagen es creada resaltando en zonas naranjas las detecciones positivas percibidas por la red.

- Subida de resultados al servidor: Los datos obtenidos por la RNA son subidos a un servidor cPanel para luego estar disponibles para su monitoreo en una página web.

Cada uno de los procesos anteriores están contenidos dentro de algún script de diseño específico para esta aplicación, en la Figura 4.5 se puede apreciar un diagrama a bloques general de la relación que guardan cada uno de los pasos en el proceso.

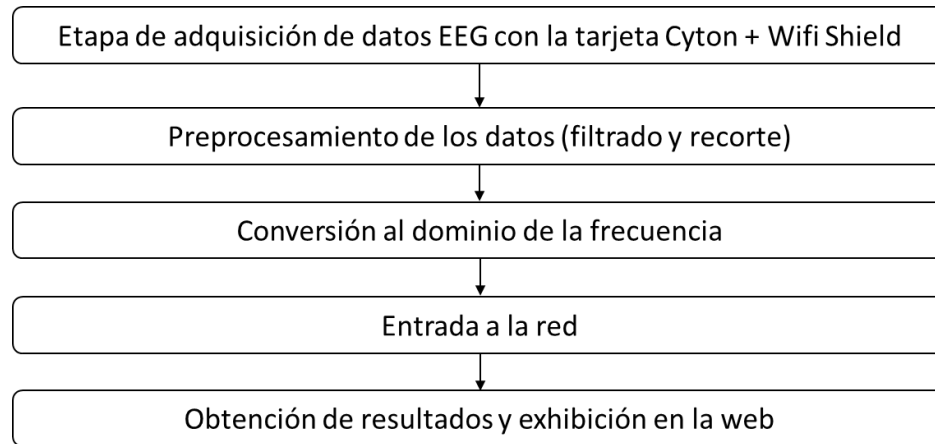


Figura 4.5. Diagrama a bloques del proceso general del sistema.

4.2 Experimentación con FPGA

Los FPGAs son dispositivos que contienen bloques lógicos reconfigurables entre sus interconexiones. Los bloques lógicos contienen memoria para la configuración del bloque mediante estructuras de datos *Lookup table* (LUT) para implementar circuitos de lógica combinacional. *Flip-flops* (FF) y multiplexores a la entrada y salida de cada bloque lógico. También, existen bloques dedicados para las entradas y salidas del FPGA identificados como IO. La reconfiguración se logra haciendo uso de algún lenguaje descriptivo de la circuitería. De esta manera, los FPGA son un tipo circuito integrado que tiene la particularidad de poder ser utilizado para implementar cualquier circuito digital [52], esto los clasifica como un dispositivo en tendencia para múltiples aplicaciones, ejemplos como: la industria de la salud [53], industria automovilística [54], aplicaciones de IoT [55], y aplicaciones de inteligencia artificial [56]. Esta última fue el detonante de la decisión de experimentar con ellos, con la intención de comprobar la factibilidad de crear algún sistema de aceleramiento por hardware para este proyecto o futuras

aportaciones que se quieran realizar, por lo que no necesariamente debía formar parte del producto entregable final.

4.2.1 Características de hardware (Cora Z7)

Actualmente Xilinx (adquirida por AMD) y Altera (adquirida por Intel) son las empresas con mayor foco cuando se habla de desarrollo de dispositivos FPGA. Ambas ofrecen un amplio catálogo de dispositivos en un gran rango de precios, por lo que podría parecer una tarea difícil seleccionar el dispositivo correcto para esta aplicación. No obstante, como se planteó en la sección anterior, la intención tras esta experimentación es comprobar la viabilidad de diseño, entrenamiento e implementación de la RNA de detección de artefactos oculares en un FPGA de bajo consumo, pequeño y de una accesibilidad económica para sistemas de computación en el borde.

Considerando las necesidades y posibilidades de este proyecto para diseñar una RNA en hardware, fue elegido el dispositivo Digilent Cora Z7-07S (Figura 4.6) el cual, es una plataforma de desarrollo, de bajo costo y con facilidad de integración para los diseños de sistema en chip totalmente reconfigurables (APSoC) de Zynq-7000 ofrecida por Xilinx. La arquitectura Zynq-7000 integra un procesador ARM Cortex-A9 de 667 MHz de un núcleo con un FPGA Xilinx de serie 7. Esta combinación otorga la capacidad de integrar al procesador un conjunto particular de periféricos y controladores definidos por software, diseñados para la aplicación que se destine [57].

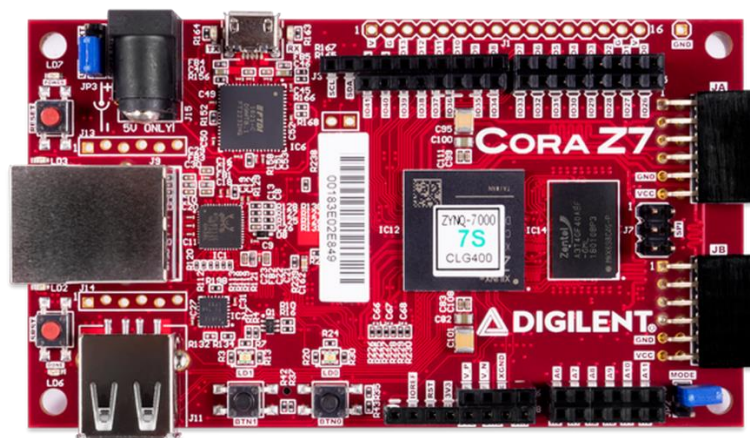


Figura 4.6. Digilent Cora Z7-07S.

La Tabla 4.2 contiene las especificaciones técnicas de mayor importancia en el dispositivo de desarrollo de Digilent. Es necesario conocerlas para comprender el límite al que se puede llevar al dispositivo, y tener conciencia de un objetivo realista con tales capacidades computacionales.

Tabla 4.2. Especificaciones técnicas del dispositivo Digilent Cora Z7-07S.

Procesador ZYNQ	Cortex-A9 de 1 núcleo a 667MHz
	FPGA: Xilinx serie 7 3,600 porciones lógicas programables 60 porciones DSP Bloque de 225 KB de RAM
	Controlador de memoria DDR3 con 8 canales DMA y 4 puertos esclavos AXI3 de alto rendimiento
	Controladores periféricos de gran ancho de banda: Ethernet 1G, USB 2.0, SDIO
	Controladores periféricos de ancho de banda baja: SPI, UART, CAN, I2C
	Convertidor analógico-digital interno de doble canal, 1 MSPS
	Programable desde JTAG y tarjeta microSD
Memoria	DDR3 de 512 MB con bus de 16 bits a 1050 Mbps
	Ranura microSD
Alimentación	Por USB o cualquier fuente de alimentación externa de 4.5V-5.5V
USB y Ethernet	Gigabit Ethernet PHY con identificador único mundial de 48 bits compatible con EUI-48/64
	Circuito de programación USB-JTAG
	Puente USB-UART
	USB OTG PHY (solo admite host)

Push-buttons y LEDs	Dos Push-buttons
	Dos LEDs RGB
Conectores de expansión	Dos conectores Pmod: 16 E/S totales de FPGA
	Conector Shield Arduino / chipKIT: Hasta 49 E/S digitales totales de FPGA 6 entradas analógicas de 0-3,3 V de un solo extremo a XADC 8 entradas analógicas diferenciales de 0-1,0 V a XADC
	Encabezado de expansión: 12 E/S digitales de FPGA adicionales

Las características técnicas de sistema en chip y la conexión a Internet vía Ethernet vuelve factible implementar este dispositivo como un sistema de computación en el borde tradicional como la UP Board descrita en la sección 4.1.

4.2.2 Diseño e implementación de la RNA para dispositivos reconfigurables

Para implementar una RNA en un FPGA se necesitan seleccionar correctamente las herramientas de desarrollo ya que de esto depende la reconfiguración exitosa del dispositivo. Para el FPGA de Xilinx fue seleccionado el entorno de Vivado Design Suite (producido por Xilinx) en su versión 2020.1 para análisis y síntesis. También, como lenguaje descriptivo de la circuitería (HDL) que dio pie al diseño de la RNA se consideraron VHDL y Verilog, pero, por simple practicidad y experiencia se trabajó con Verilog.

Para tener la mejor comparación de desempeño de la RNA en hardware contra la implementada en el dispositivo UP Board, se usan los mismos datos obtenidos del entrenamiento de la red (pesos y sesgos), lo que implica conservar la arquitectura de red propuesta desde el comienzo.

La idea de diseñar una red neuronal en un FPGA no es nueva, de hecho, en la actualidad existen trabajos que utilizan frameworks facilitadores en esta tarea [58], [59]. Sin embargo, se recurre a lenguajes de síntesis de alto nivel (HLS) lo que aleja del objetivo respecto a la intención

de experimentar con lenguajes descriptivos de hardware (síntesis de bajo nivel), que teóricamente deben mejorar el rendimiento de la RNA. Por lo que se siguieron muy de cerca investigaciones donde se trabajara con HLDs relacionados a redes neuronales artificiales e inclusive sistemas de computación en el borde [60], [61].

La primera fase de adaptación de la red a hardware requiere de un manejo de datos en binario o hexadecimal, esto causa la necesidad de una reinterpretación de los datos que se tienen de señales EEG, pesos y sesgos de la red. Para conseguir esto fueron considerados los siguientes factores:

- Bus de datos de 32 bits.
- Bus de direcciones de 5 bits.
- Datos en punto fijo (requiere menor hardware, pero tiene menor precisión).
- Reinterpretación de los datos en 16 bits.

Para que los pesos y sesgos de la red puedan ser contenidos en 16 bits, se usan los métodos de conversión de complemento a dos en decimales (conservando el signo) [62], se codifica los datos de la siguiente manera (leyendo de izquierda a derecha):

- El primer bit representa el signo.
- Los próximos 3 bits representan el valor entero.
- Los últimos 12 bits representan los valores fraccionarios (con la intención de conservar la mayor precisión posible).

La Figura 4.7 interpreta el número -0.4072834 utilizando esta estrategia. En el caso de la conversión de datos de entrada a la red (señales EEG en el dominio de la frecuencia) la asignación de signos cambia a: el primer bit representa el signo, los siguientes 10 bits representan el número entero, y los últimos 5 dígitos corresponden al valor fraccionario, la Figura 4.8 representa el número 104.85 en binario.

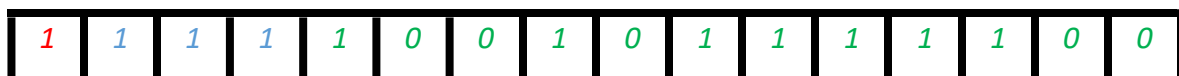


Figura 4.7. Representación del número decimal -0.4072834 en binario.



Figura 4.8. Representación del número decimal 104.85 en binario.

Entre mayor complejidad en la RNA, la cantidad de datos a convertir crece proporcionalmente, por lo que para generar escalabilidad y viabilidad para futuros diseños de redes neuronales se creó un script de preprocesamiento en Python encargado de la lectura de archivos contenedores de los valores decimales a convertir. Este script contiene un algoritmo que consta de los siguientes pasos:

1. Binarización para permitir modificar parámetros de longitud del número binario,
2. Conservación del punto decimal:
 - a. Asignación de espacios (bits) para la parte entera del valor.
 - b. Asignación de espacios (bits) para la parte fraccionaria del valor.
3. Almacenamiento de valores binarios en archivos de inicialización de memoria (MIF, por sus siglas en inglés).

Los archivos de extensión MIF son la alternativa indicada para manejar grandes cantidades de datos que requiera conocer el FPGA, estos son cargados en las direcciones de memoria ROM del dispositivo y agilizan el procesamiento de operaciones matemáticas requeridas por la RNA.

Una RNA en hardware tiene varias restricciones al momento de plantear el diseño, una de las más importantes es considerar el manejo de los datos de entrada, debido a que entre mayor sea el número de neuronas, habrá mayor número de entradas. Esta simple relación puede consumir gran parte de los recursos de hardware del dispositivo si no se maneja adecuadamente; por lo que los datos de entrada son enviados dato por dato secuencialmente a la frecuencia de operación del FPGA, mientras que los pesos y sesgos de la red son distribuidos en un bus de datos compartidos, de igual forma todas las neuronas instanciadas están ancladas al mismo bus de datos para no recurrir a repetir el ingreso de datos a distintos módulos. Las instanciaciones de las neuronas son la pieza clave de esta estrategia, permite crear las capas con el número de neuronas deseadas desde la fuente principal del proyecto en Verilog, 85 entradas, 35 neuronas

en la capa oculta y 1 neurona de salida fueron las declaraciones realizadas para la arquitectura de una RNA totalmente conectada.

La Figura 4.9 ilustra el comportamiento de la neurona descrita en hardware (el comportamiento será repetido tantas veces sea declarada), contemplando el ingreso de datos $X(l)$, la memoria de pesos W donde se encuentran almacenados en su estado binario, las lecturas de las binarizaciones a 16 bits, las operaciones matemáticas de una neurona artificial, para finalmente pasar al proceso de evaluación de una función de activación sigmoideal (Ecuación 3.3 y 3.5) descrita en hardware y extraer el resultado de la neurona u_2 para pasar a la siguiente (en caso de ser necesario).

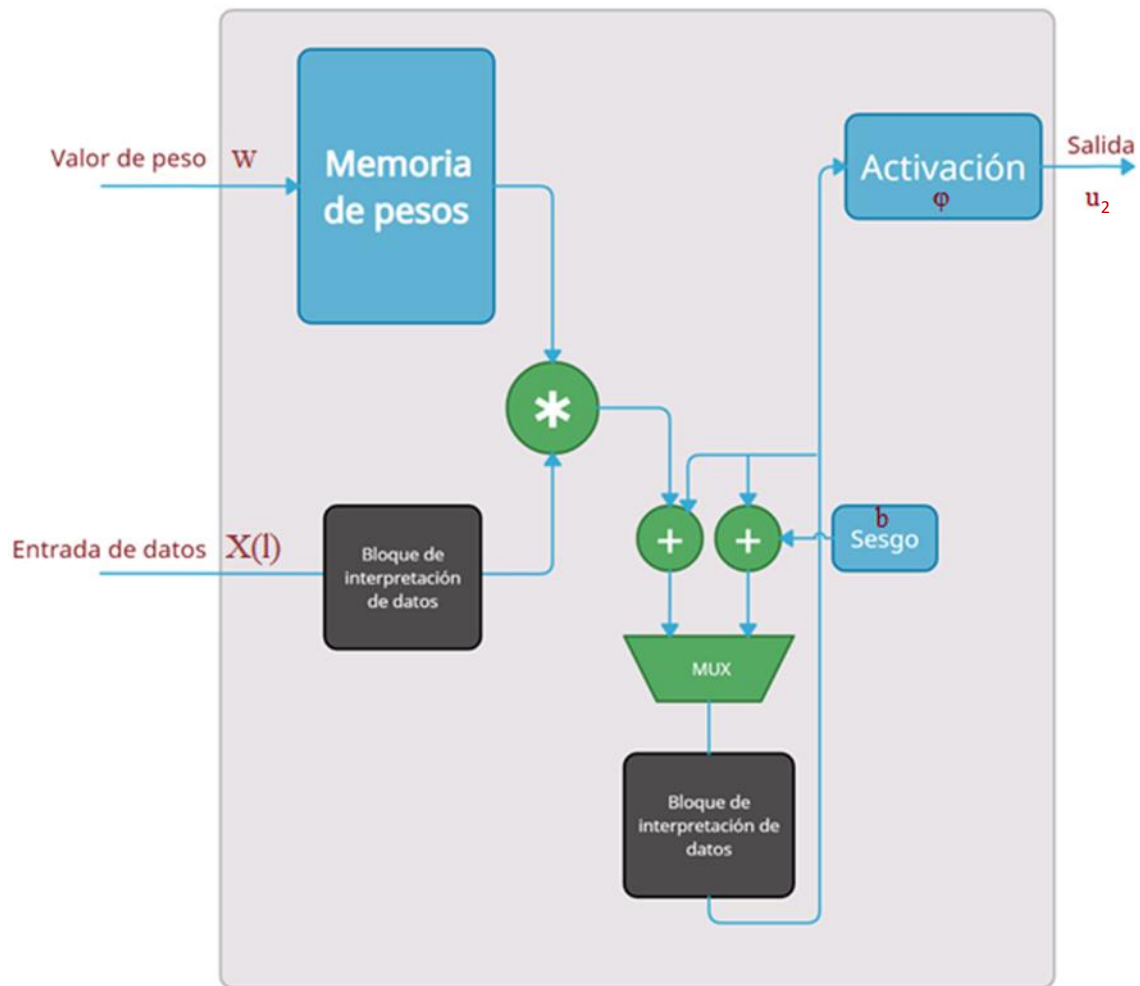


Figura 4.9. Diagrama ilustrativo del comportamiento de la neurona en hardware.

CAPÍTULO V

RESULTADOS

Con el sistema de computación en el borde y la experimentación de aceleramiento por hardware de la RNA finalizadas, se realizaron una serie de experimentos cuyos resultados se presentan en esta sección. Por ello, este capítulo se divide de la siguiente manera. La sección 5.1 se enfoca en los resultados del prototipo completo de computación en el borde, las métricas y comparaciones con otros productos encontrados en la literatura, en la sección 5.2 se detallan los efectos y consecuencias de tratar de implementar una RNA en un dispositivo de hardware reconfigurable de gama baja.

5.1 Sistema de computación en el borde

Cuando se empezó a trabajar en este proyecto, en la literatura no existía algún antecedente de prototipo similar a este. Un prototipo de esta índole genera portabilidad, reducción de complejidad para trabajar con señales EEG, permite al usuario visualizar los resultados del análisis en cualquier parte con la que cuente con una conexión a Internet y se vuelve una opción accesible debido a su costo de producción. Por ello, lo primero a destacar es la obtención de una BCI de computación en el borde, diseñado para un dispositivo especializado en aplicaciones embebidas y de IoT (UP Board). Dentro de este dispositivo se alojan los 6 módulos diseñados en Python para hacer el procesamiento de los datos, estos son nombrados y cumplen con las siguientes tareas:

- `parameters.py`: Se encarga de recibir los parámetros que se ingresan desde la consola de comandos, nombre de archivos, tiempo de adquisición, identificador OpenBCI de operación, puertos de lectura, y direcciones IP.
- `eeg_data.py`: Configura la forma de operar del dispositivo Cyton con los parámetros recibidos, habilitando los canales necesarios para tomar las muestras el tiempo seleccionado, para luego ser filtradas por el filtro Bessel (Ecuaciones (2.1) – (2.5)) implementado.
- `cutout_data.py`: Toma las matrices de datos $X_C(k)$ generadas por el anterior programa, se hace un recorte de los datos de interés (canales habilitados) y elimina el primer

segundo de lectura (explicado en la sección 2.3.1) que causa interferencia con el resto del proceso por ser únicamente ruido.

- `nn_application.py`: Los datos recibidos son pasados al dominio de Fourier $X(l)$, acto seguido son ingresados a la RNA de detección de artefactos pasando por las Ecuaciones (3.2) – (3.5). Después se almacena la cantidad artefactos encontrados y se guarda una ilustración donde son resaltadas las detecciones de la red en el dominio del tiempo.
- `upload_data.py`: Utiliza el protocolo FTP para cargar en el servidor los archivos creados por el proceso de detección de la red.
- `main.py`: Llama en orden los módulos que componen el proceso, permitiendo llegar hasta la parte de almacenamiento en el dispositivo, o habilitando comenzar desde la lectura de documentos para la evaluación de la red (en caso de tener bases de datos compatibles para ser evaluadas).

El proceso completo del sistema parte de la extracción de datos EEG $X_c(k)$ del sujeto, estos datos son leídos por el dispositivo OpenBCI Cyton, y son transmitidos a la red con la ayuda del módulo OpenBCI Wi-Fi Shield. La tarjeta UP Board es la encargada de recibir dichos datos para ser procesados dentro de los programas previamente explicados, para finalmente, llegar al servidor cPanel, de donde son tomados los documentos que se ponen a disposición del usuario en una página web escrita en PHP. En la Figura 5.1 se plasma el proceso que ejecutado por el sistema. La Figura 5.2 contiene un extracto de la página web que almacena los resultados generados por la RNA de detección.

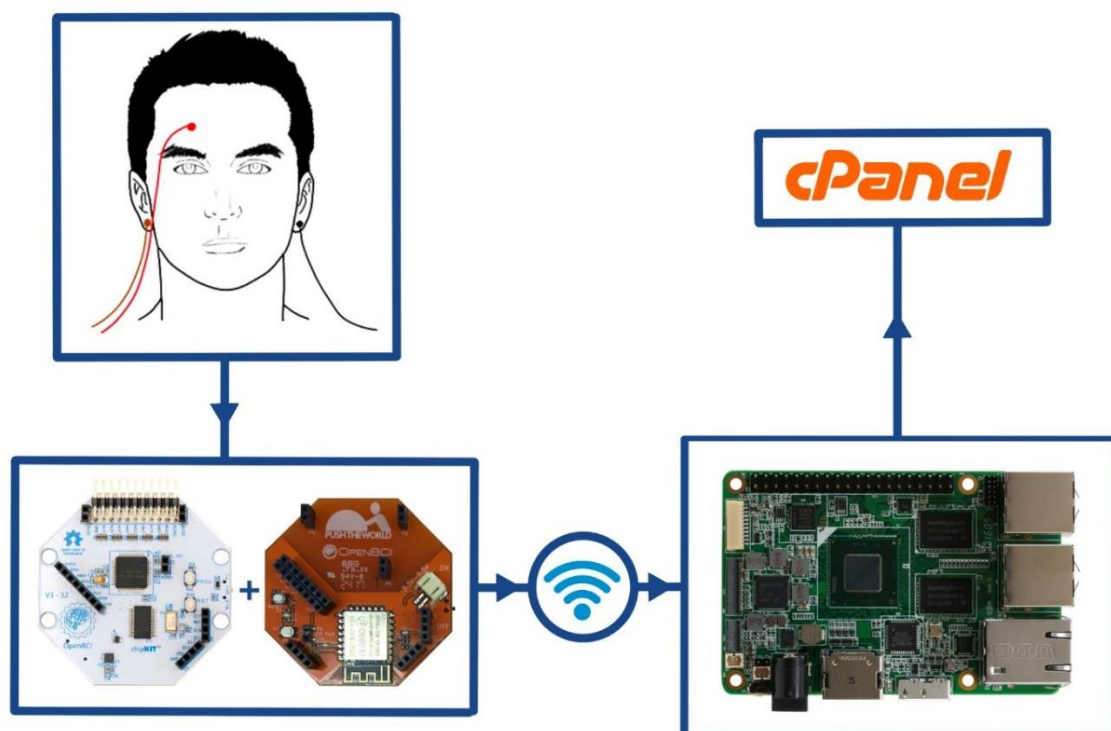


Figura 5.1. Proceso del sistema completo BCI de computación en el borde.

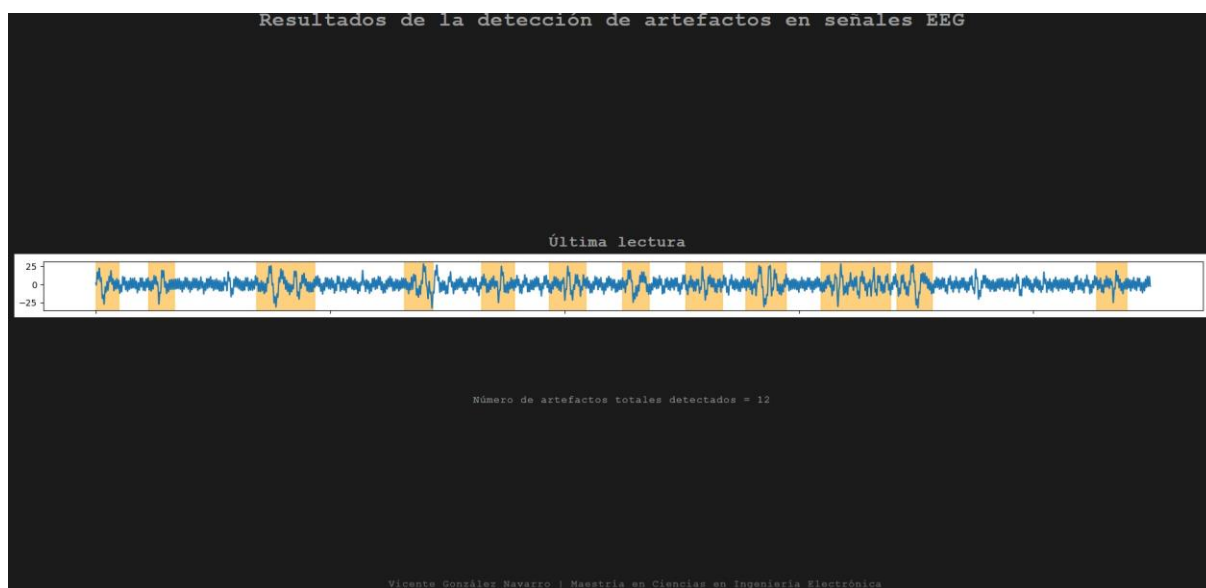


Figura 5.2. Despliegue de información en la página web.

5.1.1 Métricas del sistema de computación en el borde

El rendimiento medido del sistema es abordado en dos fases, el de la RNA de detección de artefactos de señales EEG y el del sistema de computación en el borde. La Tabla 5.1 muestra la matriz de confusión de la red y la Tabla 5.2 contiene varias de las métricas de visualización de desempeño de la RNA.

Tabla 5.1. Matriz de confusión.

	Verdadero Positivo	Verdadero Negativo
Positivo Predicho	VP = 58	FP = 0
Negativo Predicho	FN = 2	VN = 60

Tabla 5.2. Métricas de desempeño.

Medida	Resultado
Sensibilidad	0.9667
Precisión	1.0000
Exactitud	0.9833
Especificidad	1.0000
Valor-F	0.9831

Como ya fue mencionado, no existe un sistema similar con el que hacer comparativas, ya que el trabajo desarrollado permite que sistemas BCI escalen a una aplicación de Internet de las Cosas. Por ejemplo, en [63] se propuso un sistema de comunicación basado en técnicas BCI haciendo detección de artefactos de parpadeos en señales EEG. Este sistema se diseñó utilizando Python para un dispositivo Raspberry Pi 3 modelo B con una arquitectura de perceptrón multicapa y el dispositivo de adquisición de señales EEG es el NeuroSky Mindware Mobile, el cual solo dispone de un electrodo fijo para la lectura de datos EEG. No obstante, este proyecto no tiene las bases necesarias para beneficiarse de la computación en el borde debido a que opera en un sistema de cómputo genérico.

Algo similar pasa en [64] donde se desarrolló un sistema BCI que adquiere señales EEG para detectar y reconocer patrones de parpadeo y movimientos oculares. Se usó Matlab para crear una red neuronal modular y con el apoyo de Emotiv Epoc+ y el dispositivo Cyton OpenBCI recolectaron las señales necesarias. A diferencia de la mención anterior, este proyecto no opera en un sistema de cómputo embebido. De esta manera, este BCI es poco portable e impráctico para su transporte, además se requiere de licencia para uso de Matlab, por lo que se vuelve en un producto de nicho comercial, punto que se solventaría fácilmente al integrarlo a un sistema en Python como el de esta tesis.

En la segunda fase se tomaron los de los resultados de las métricas del sistema de computación en el borde, Resaltando que se operó siempre en una conexión Wifi a 5GHz, con una velocidad de bajada de 30Mbps y 2Mbps de subida. La Tabla 5.3 contiene los datos correspondientes al comportamiento del sistema completo.

Tabla 5.3. Métricas del sistema.

Parámetro	Resultado
Potencia consumida	13.5W
Latencia de red	15ms – 20ms
Tiempo de arranque del sistema	800ms – 1000ms
Tiempo total de adquisición	14s
Tiempo de procesamiento total del sistema	15.5s – 17s

5.1.2 Computación en el borde contra computación en la nube

Para caracterizar una BCI funcionando como un sistema de computación en el borde o en la nube, se compararon ambos paradigmas utilizando como referencia el tiempo de latencia, procesamiento de las señales, costos computacionales y de potencia, arranque del sistema y el tiempo de procesamiento total del sistema. Para poner el BCI en línea, se aprovechó el modelo de servicios especializados en la nube de Google (Google Cloud Platform). Para ello, se creó una máquina virtual con base en el sistema operativo Ubuntu 16.04 LTS y dentro de este entorno sin interfaz gráfica fueron transferidos los archivos diseñados para el dispositivo UP

Board encargados del recorte de datos, la red neuronal artificial, y el protocolo que permite transferir los resultados al servidor web.

Debido a la naturaleza de los dispositivos OpenBCI, en específico, el módulo Wi-Fi Shield requiere que el transmisor y el receptor se encuentren en la misma red. Esto implica que la transferencia de las lecturas obtenidas por el dispositivo Cyton (señales EEG) necesita seguir ocurriendo de manera local, por lo que se modificó ligeramente el script de lectura de archivos para que al momento de obtener los documentos de las señales EEG, sean subidos a la red haciéndolos accesibles para la máquina virtual. El proceso de detección de artefactos comienza una vez el programa de reconocimiento en la nube detecta un archivo nuevo, a diferencia del sistema de computación perimetral, los resultados son almacenados fuera del sistema de cómputo de origen, así que para tenerlos disponibles de manera local son descargados al sistema una vez la detección haya terminado, de igual forma quedan a disposición del servidor para ser mostrados en la página de resultados.

Como es de esperarse, cada uno de los métodos tiene sus ventajas y desventajas. La computación perimetral permite que la transmisión de datos se haga a mayor velocidad debido a que el proceso ocurre de manera local, por lo que el tiempo de respuesta en el borde es menor a un sistema en la nube (operando en la misma red). El tiempo de arranque del sistema es de un 16.67% a 27.27% más rápido que uno en la nube. Además, el procesamiento total del sistema es aproximadamente un 38.5% más veloz. Sin embargo, el desempeño de la red puede verse beneficiado en caso de ser puesto en una nube con mejores recursos computacionales suponiendo que se puede hacer una inversión mayor en la máquina virtual, ya que la mayoría de los servicios de computación en la nube con buena infraestructura tienen un costo por hora relacionado al hardware contratado. Esto significa que a largo plazo se convierte en un gasto grande proporcional a los recursos contratados. La Tabla 5.4 muestra las métricas de comparación.

Tabla 5.4. Métricas de comparación de computación en el borde y computación en la nube.

Parámetro	Resultado en el borde	Resultado en la nube
Costo por máquina virtual	-	\$622.48/mes

Potencia consumida	13.5W	-
Latencia de red	15ms – 16ms	16ms – 16.5ms
Tiempo de arranque del sistema	800ms – 1000ms	1100ms – 1200ms
Tiempo total de adquisición	14s	14s
Tiempo de procesamiento total del sistema	15.5s – 17s	25s – 28s

5.2 Experimentación con FPGA

La intención de experimentar con el dispositivo Digilent Cora Z7 fue comprobar la viabilidad de diseñar un sistema de computación en el borde dentro de un dispositivo de desarrollo especializado en hardware de baja gama. Esto hizo que se consideraran varias posibilidades y se hicieron varias pruebas. El primer experimento de diseño partió de la idea de utilizar alguna IP (propiedad intelectual) de Xilinx. El problema, es que esto limita la flexibilidad del diseño de manera inmediata, consumiendo gran parte de los recursos de hardware que se disponían. El segundo experimento comenzó con la instalación de un sistema operativo (Petalinux) con propósito de sacar provecho al sistema en chip con el que se estaba trabajando para realizar internamente el entrenamiento de la red y la conversión al domino de la frecuencia de las señales adquiridas. Lamentablemente, el dispositivo tuvo varias fallas debido a problemas en ejecución de procesos o lectura de localidades de memoria y por cuestiones de tiempo no se llevaron a cabo estos experimentos. El intento que se explica en esta sección es la obtención de una RNA descrita puramente en circuitería con respaldo literario en procesadores Zynq para ahorrar el mayor porcentaje de recursos disponibles conservando la arquitectura de la red que se implementó en el sistema de computación en el borde.

5.2.1 RNA implementada en hardware

El proyecto diseñado en Vivado terminó con 4 documentos de formato Verilog referentes al comportamiento de la RNA y 125 archivos de inicialización de memoria, donde se encuentran los pesos, sesgos y datos a evaluar en forma binaria. Los módulos de descripción de circuitería que componen a la red y su función son:

- sigmoid.v: Contiene el diseño sigmoidal (Ecuación 3.3 y 3.5) para recibir los datos f_{qp} de las neuronas (en cadenas binarias) para ser evaluados por la función de activación φ .
- neuron.v: Este es el módulo que recibe los parámetros de arquitectura deseados, crea las capas, opera matemáticamente con los datos asignados $X(l)$, opera con las Ecuaciones 3.2 y 3.4 para sacar el resultado posterior a la función de activación sigmoidal u_2 (Ecuación 3.5).
- axi_wrapper.v: Se crea las interconexiones necesarias para la utilización de puertos y comunicaciones de datos dentro del dispositivo.
- MFinder.v: En dado caso de tener más de una neurona de salida f identifica la salida de la neurona con mayor coincidencia (no sé aplicó en el diseño final).
- main.v: Se inicializa la arquitectura de la RNA, variables, procesos y módulos que solicita el sistema.

El diseño es exitosamente sintetizable, pero, por los recursos de hardware del dispositivo no fue implementable. Esto sucedió porque el FPGA se quedó corto en porciones de procesadores de señales digitales (DSP) y de interconexiones de entrada y salida del sistema. La Figura 5.3 muestra la gráfica de utilización de recursos arrojada por el entorno de desarrollo. La Tabla 5.5 contiene la información respecto al hardware consumido al implementar una neurona simple e implementar la red neuronal artificial completa.

Tabla 5.5. Estadísticas de consumo de hardware.

Recurso	Total	Por neurona		RNA completa	
		Utilizado	Porcentaje	Utilizado	Porcentaje
LUT	14,400	27.36	0.19%	3320	23.06%
FF	28,800	25.92	0.09%	3029	10.52%
BRAM	50	0.145	0.29%	17.5	35%
DSP	66	0.594	0.9%	72	109.09%
IO	100	0.86	0.86%	105	105%
BUFG	32	0.007936	0.0248%	1	3.13%

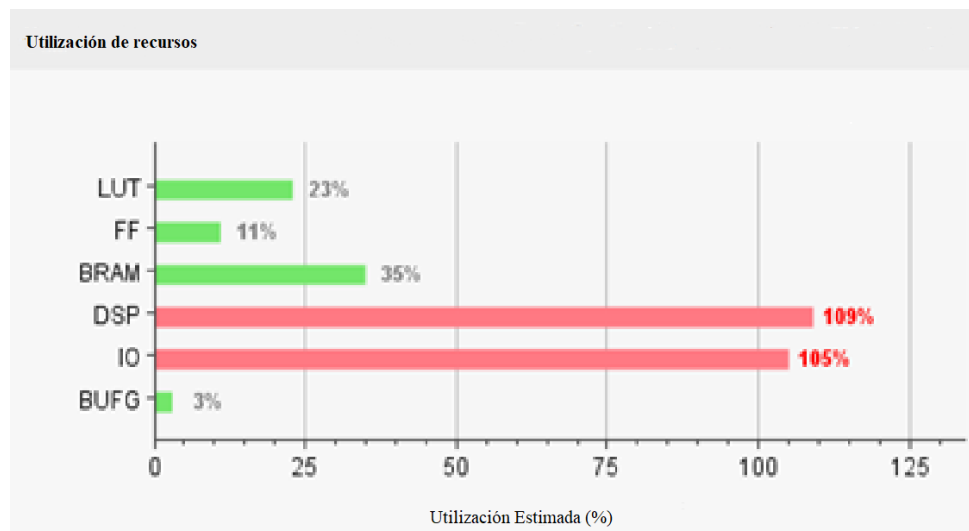


Figura 5.3 Gráfica de utilización de recursos.

Es complicado dar una opción general óptima para cualquier implementación de RNA, porque la asignación de recursos de hardware ocurre al momento de sintetizar la descripción y depende enteramente del entorno de desarrollo por lo que se escapa del control del diseñador. No obstante, con base en los experimentos y los datos obtenidos, se concluye que las características adecuadas de un FPGA para aplicaciones de aprendizaje automático (ML), haciendo enfoque en aplicaciones similares a las de este proyecto, sin importar la compañía desarrolladora de FPGAs de procedencia, son las que se muestran en la Tabla 5.6.

Tabla 5.6. Recomendaciones de recursos de FPGA para ML.

Recurso	Total
LUT	14,400
FF	28,800
BRAM	50
DSP	160
IO	200
BUFG	32

5.2.2 Desempeño de la RNA

Al no poder implementar el diseño en el dispositivo, la alternativa para obtener el desempeño de la RNA es recurrir a la simulación. Objetivamente, al igual que el desempeño que se obtiene de la RNA diseñada para la UP Board estos resultados no están ligados al hardware donde se implementaron, por lo que son datos en los que se pueden utilizar en futuras aplicaciones en dispositivos de gamas mayores. La Tabla 5.7 contiene los resultados de desempeño de la simulación de esta red.

Tabla 5.7. Métricas de desempeño en hardware.

Medida	Resultado
Sensibilidad	0.7656
Precisión	0.7833
Exactitud	0.8167
Especificidad	0.8036
Valor-F	0.7903

La razón por la que estos resultados varían considerablemente respecto a los de la RNA diseñada en Python es la perdida de datos que sufren los valores binarizados para poder ser interpretados por la descripción de la circuitería, debido a que gran parte de estos datos son valores decimales muy pequeños y requerirían un gran número de bits para aproximarse apropiadamente en forma binaria.

CAPÍTULO VI

CONCLUSIONES Y TRABAJO FUTURO

En esta tesis se desarrolló un sistema de interfaz cerebro computadora para sistemas de computación en el borde, utilizando la tarjeta OpenBCI Cyton y el módulo Wi-Fi Shield como los dispositivos que adquieren y transmiten las señales EEG, siendo procesados por una red neuronal artificial totalmente conectada para la detección de artefactos oculares. La RNA junto a los otros programas de operación diseñados en Python están contenidos en la tarjeta de desarrollo UP Board, para finalmente enviar la información de la señal EEG destacando las detecciones de artefactos en una página web. Además, fue explorada la posible integración de un FPGA como dispositivo de borde.

El desempeño que muestra la RNA del prototipo final es aceptable, pero, es importante resaltar que los datos de entrenamiento y evaluación están relativamente contenidos en grupos pequeños, por lo que ampliando el número de muestras se puede esperar un cambio perjudicial en estos resultados. Caso contrario, el aumentar el número de bits con los que se interpretan los datos y muestras puede ayudar a obtener mejoría en las métricas de evaluación de la RNA. Sin embargo, para la aplicación buscada esta RNA tiene un comportamiento aceptable y abre la posibilidad de ser utilizada a futuro para interpretar intenciones del usuario o como medio de comunicación con base en sus parpadeos consientes.

La tarjeta de desarrollo UP Board es capaz de realizar las tareas de computación en el borde de esta investigación. Su comportamiento es superior al de algunos dispositivos más popularizados en el mercado y es relativamente accesible para todos los usuarios. Si fuera necesario escalar o aumentar el objetivo del proyecto con otra RNA de mayor arquitectura el rendimiento no habría sido mermado drásticamente debido a las capacidades computacionales del dispositivo, a diferencia de varios dispositivos de desarrollo encontrados en la literatura. El software que ejecuta el dispositivo es 100% trasladable a otro dispositivo, mientras sea capaz de ejecutar programas de lenguaje Python, esto pensando en futuros cambios de sistema operativo o en mejoras de recursos de cómputo.

No hubo un punto de comparación claro en la literatura, ya que no existe un prototipo similar a este. Por lo que los resultados obtenidos pese a no ser comparables son defendibles en el sentido de que opera de manera exitosa en un corto periodo de tiempo, y que como es de esperarse en un dispositivo de este giro, no es necesario estar conectado a la red para realizar el procesamiento de detección, pero se puede complementar con ella para mostrar los resultados a agentes externos que tal vez la requieran, ya sea para temas de monitoreo, salud, interacción con un tercero, etc.

El sistema de computación en el borde presentó las ventajas esperadas contra a un sistema de cómputo en la nube, respecto a velocidad, seguridad y costos del prototipo. Hacer las pruebas abordadas en la sección 5.1.2 también ayudó a demostrar la facilidad de escalabilidad de los programas desarrollados al largo de este proyecto.

Respecto a la experimentación de aceleramiento por hardware utilizando el dispositivo Cora Z7, se demuestra que es posible diseñar una RNA que detecte artefactos oculares de señales EEG. Sin embargo, si se quiere llevar a la implementación será estrictamente necesario cambiar a un dispositivo con mejores recursos de hardware. Varios ejemplos de distintos dispositivos en distintas gamas que integran al menos un FPGA podrían ser:

- Arty Z7: Zynq-7000 SoC Development
- Nexys A7: FPGA Trainer Board
- Versal AI Edge Series VE2802 (próximo lanzamiento especializado en control por AI)
- Xilinx Zynq-7000 SoC ZC702 Evaluation Kit
- Intel Cyclone® V GX
- Intel Agilex™ F-Series FPGAs and SoC FPGAs

Esto se debe que es necesario incrementar la longitud de bits que deben representar a los datos de evaluación, el número de porciones de DSP disponibles, y la disponibilidad de interconexiones que se puedan hacer. Lo anterior vuelve algo difícil la idea de trabajar con dispositivos de desarrollo de entrada de bajas prestaciones de hardware como fue este caso, y evidentemente, si se desea escalar la aplicación los recursos tendrán que crecer proporcionalmente a ella.

Aspectos fundamentales y recomendados para prototipos de RNAs que requieran o puedan tener aceleramiento por medio del FPGA:

- La institución o centro de desarrollo debe realizar minuciosamente una investigación con base en el presupuesto y las necesidades del proyecto para definir que dispositivo de hardware ofrece las mejores características, no solo de hardware, si no del software de diseño con el que cuentan, ya que día a día nuevos programas para facilitar estas tareas son puestos a disposición de los consumidores.
- Los algoritmos no son de fácil interpretación para descripciones de hardware, por lo que hacer el entrenamiento dentro del FPGA es un desperdicio de los recursos del dispositivo (a menos que el dispositivo esté especializado en AI), pero es una gran manera de acelerar el compute de la red implementada. En dispositivos como los de este proyecto lo mejor es hacer un entrenamiento externo ya que suele ser entrenada una única vez, para luego trasladarlo a algún tipo de HDL compatible con el FPGA.
- Si el dispositivo lo permite, trabajar con software de alto nivel puede facilitar la tarea del diseño e implementación de la red, siempre recordado que esto probablemente necesitará mayor parte del hardware sobre el que se aplique debido a la optimización de reconfiguración de estas herramientas.

REFERENCIAS

- [1] B. Kaur, D. Singh and P. Roy, "Age and gender classification using brain-computer interface", *Neural Computing and Applications*, vol. 31, no. 10, pp. 5887-5900, 2018. Available: 10.1007/s00521-018-3397-1.
- [2] R. Leeb, L. Tonin, M. Rohm, L. Desideri, T. Carlson and J. Millan, "Towards Independence: A BCI Telepresence Robot for People With Severe Motor Disabilities", *Proceedings of the IEEE*, vol. 103, no. 6, pp. 969-982, 2015. Available: 10.1109/jproc.2015.2419736.
- [3] C. Ieracitano, N. Mammone, A. Hussain and F. Morabito, "A novel multi-modal machine learning based approach for automatic classification of EEG recordings in dementia", *Neural Networks*, vol. 123, pp. 176-190, 2020. Available: 10.1016/j.neunet.2019.12.006.
- [4] J. Valenzuela-Valdes, M. Lopez, P. Padilla, J. Padilla and J. Minguillon, "Human Neuro-Activity for Securing Body Area Networks: Application of Brain-Computer Interfaces to People-Centric Internet of Things", *IEEE Communications Magazine*, vol. 55, no. 2, pp. 62-67, 2017. Available: 10.1109/mcom.2017.1600633cm.
- [5] R. Chatterjee, T. Maitra, S. Hafizul Islam, M. Hassan, A. Alamri and G. Fortino, "A novel machine learning based feature selection for motor imagery EEG signal classification in Internet of medical things environment", *Future Generation Computer Systems*, vol. 98, pp. 419-434, 2019. Available: 10.1016/j.future.2019.01.048.
- [6] K. Singh and J. Malhotra, "IoT and cloud computing based automatic epileptic seizure detection using HOS features based random forest classification", *Journal of Ambient Intelligence and Humanized Computing*, 2019. Available: 10.1007/s12652-019-01613-7.
- [7] C. Coogan and B. He, "Brain-Computer Interface Control in a Virtual Reality Environment and Applications for the Internet of Things", *IEEE Access*, vol. 6, pp. 10840-10849, 2018. Available: 10.1109/access.2018.2809453.

- [8] M. Hosseini, D. Pompili, K. Elisevich and H. Soltanian-Zadeh, "Optimized Deep Learning for EEG Big Data and Seizure Prediction BCI via Internet of Things", *IEEE Transactions on Big Data*, vol. 3, no. 4, pp. 392-404, 2017. Available: 10.1109/tbdata.2017.2769670.
- [9] K. Ericson, S. Pallickara and C. W. Anderson, "Analyzing Electroencephalograms Using Cloud Computing Techniques," 2010 IEEE Second International Conference on Cloud Computing Technology and Science, 2010, pp. 185-192, doi: 10.1109/CloudCom.2010.80.
- [10] B. He, *Neural engineering*. New York: Springer, 2013, pp. 87-100.
- [11] O. Mecarelli, *Clinical electroencephalography*. Cham: Springer, 2019.
- [12] E. Friedrich, R. Scherer and C. Neuper, "The effect of distinct mental strategies on classification performance for brain-computer interfaces", *International Journal of Psychophysiology*, vol. 84, no. 1, 2012. Available: 10.1016/j.ijpsycho.2012.01.014.
- [13] Y. Rezaei Tabar and U. Halici, "Brain Computer Interfaces for Silent Speech", *European Review*, vol. 25, no. 2, pp. 208-230, 2016. Available: 10.1017/s1062798716000569 [Accessed 11 July 2021].
- [14] O. Markand, *Clinical evoked potentials*. Cham: Springer, 2020.
- [15] G. Pfurtscheller, "Functional brain imaging based on ERD/ERS", *Vision Research*, vol. 41, no. 10-11, pp. 1257-1260, 2001. Available: 10.1016/s0042-6989(00)00235-2.
- [16] C. Nguyen, G. Karavas and P. Artemiadis, "Inferring imagined speech using EEG signals: a new approach using Riemannian manifold features", *Journal of Neural Engineering*, vol. 15, no. 1, p. 016002, 2017. Available: 10.1088/1741-2552/aa8235.
- [17] P. Saha, M. Abdul-Mageed and S. Fels, "Speak Your Mind! Towards Imagined Speech Recognition With Hierarchical Deep Learning", *Cornell University*, 2019. Available: arXiv:1904.05746.
- [18] S. Cirani, G. Ferrari, M. Picone and L. Veltri, *Internet of things: Architectures, Protocols and Standards*, 1st ed. The Atrium, Southern Gate, Chichester, West Sussex, UK: Wiley, 2019, pp. 1-7.

- [19] Z. Mahmood, Cloud Computing. Swindon, United Kingdom: Springer International Pu, 2016, pp. 4-14.
- [20] R. Buyya and S. Srirama, Fog and edge computing, 1st ed. Hoboken, NJ, USA: John Wiley & Sons, 2019, pp. 82-84.
- [21] N. Chu, "How Should the State of the Brain Be Described?: A call to standardize descriptions of brain states for data collection and research", IEEE Consumer Electronics Magazine, vol. 5, no. 3, pp. 55-59, 2016. Available: 10.1109/mce.2016.2556838.
- [22] Y. Xiao, Y. Jia, X. Cheng, J. Yu, Z. Liang and Z. Tian, "I Can See Your Brain: Investigating Home-Use Electroencephalography System Security", IEEE Internet of Things Journal, vol. 6, no. 4, pp. 6681-6691, 2019. Available: 10.1109/jiot.2019.2910115.
- [23] M. Alhussein, G. Muhammad, M. Hossain and S. Amin, "Cognitive IoT-Cloud Integration for Smart Healthcare: Case Study for Epileptic Seizure Detection and Monitoring", Mobile Networks and Applications, vol. 23, no. 6, pp. 1624-1635, 2018. Available: 10.1007/s11036-018-1113-0.
- [24] X. Zhang, L. Yao, S. Zhang, S. Kanhere, M. Sheng and Y. Liu, "Internet of Things Meets Brain-Computer Interface: A Unified Deep Learning Framework for Enabling Human-Thing Cognitive Interactivity", IEEE Internet of Things Journal, vol. 6, no. 2, pp. 2084-2092, 2019. Available: 10.1109/jiot.2018.2877786.
- [25] H. Ning, Y. Li, F. Shi and L. Yang, "Heterogeneous edge computing open platforms and tools for internet of things", Future Generation Computer Systems, vol. 106, pp. 67-76, 2020. Available: 10.1016/j.future.2019.12.036.
- [26] Y. Chen, Y. Sun, N. Lu and B. Wang, "Channel-reserved medium access control for edge computing based IoT", Journal of Network and Computer Applications, vol. 150, p. 102500, 2020. Available: 10.1016/j.jnca.2019.102500.
- [27] F. Xhafa, B. Kilic and P. Krause, "Evaluation of IoT stream processing at edge computing layer for semantic data enrichment", Future Generation Computer Systems, vol. 105, pp. 730-736, 2020. Available: 10.1016/j.future.2019.12.031.

- [28] Z. Zhao, P. Lin, L. Shen, M. Zhang and G. Huang, "IoT edge computing-enabled collaborative tracking system for manufacturing resources in industrial park", *Advanced Engineering Informatics*, vol. 43, p. 101044, 2020. Available: 10.1016/j.aei.2020.101044.
- [29] P. Zeng, B. Pan, K. Choo and H. Liu, "MMDA: Multidimensional and multidirectional data aggregation for edge computing-enhanced IoT", *Journal of Systems Architecture*, vol. 106, p. 101713, 2020. Available: 10.1016/j.sysarc.2020.101713.
- [30] L. Kuang, T. Gong, S. OuYang, H. Gao and S. Deng, "Offloading decision methods for multiple users with structured tasks in edge computing for smart cities", *Future Generation Computer Systems*, vol. 105, pp. 717-729, 2020. Available: 10.1016/j.future.2019.12.039.
- [31] M. Sazgar and M. Young, *Absolute epilepsy and EEG rotation review*. Springer, 2019.
- [32] S. Ganguly and R. Singla, "Electrode Channel Selection for Emotion Recognition based on EEG Signal," *2019 IEEE 5th International Conference for Convergence in Technology (I2CT)*, Bombay, India, 2019, pp. 1-4, doi: 10.1109/I2CT45611.2019.9033929.
- [33] S. Haggag, S. Mohamed, A. Bhatti, H. Haggag and S. Nahavandi, "Noise level classification for EEG using Hidden Markov Models," *2015 10th System of Systems Engineering Conference (SoSE)*, San Antonio, TX, USA, 2015, pp. 439-444, doi: 10.1109/SYSOSE.2015.7151974
- [34] Y. Huang, K. Wang, Y. Ho, C. He and W. Fang, "An Edge AI System-on-Chip Design with Customized Convolutional-Neural-Network Architecture for Real-time EEG-Based Affective Computing System," *2019 IEEE Biomedical Circuits and Systems Conference (BioCAS)*, Nara, Japan, 2019, pp. 1-4, doi: 10.1109/BIOCAS.2019.8919038.
- [35] "Setting up for EEG Openbci Documentation", *Docs.openbci.com*, 2021. [Online]. Available: <https://docs.openbci.com/docs/01ettingstarted/02biosensingsetups/EEGSetup>. [Accessed: 1- Feb- 2021].
- [36] "User API — brainflow documentation", *Brainflow.readthedocs.io*, 2021. [Online]. Available: https://brainflow.readthedocs.io/en/stable/userapi.html?Highlight=cyton#_cppv4n9brainflow8boardids11cyton_BOARD. [Accessed: 01- Feb- 2021].

- [37] "Code Samples — BrainFlow documentation", *Brainflow.readthedocs.io*, 2021. [Online]. Available: https://brainflow.readthedocs.io/en/matlab_docs/Examples.html?highlight=bandpass. [Accessed: 01- Feb- 2021].
- [38] J. Goerzen, *Foundations of Python network programming*. Berkeley (Calif.): Apress Media, LLC, 2004, pp. 275-294.
- [39] F. Rosenblatt, "The Perceptron—a perceiving and recognizing automaton", *Cornell Aeronautical Laboratory*, vol. 85-460-1, 1957.
- [40] M. Minsky and S. Papert, *Perceptrons*. (Cambridge, Mass.: MIT Pr.), 1969.
- [41] D. Livingstone, *Artificial neural networks*. Totowa, NJ, USA: Humana Press, 2009.
- [42] E. Micheli Tzanakou, *Supervised and Unsupervised Pattern Recognition Feature Extraction and Computational Intelligence*. Boca Raton, Florida, USA: CRC Press LLC, 2000.
- [43] R. Rojas, *Neural Networks A Systematic Introduction*. Berlin: Springer, 1996, pp. 23-25.
- [44] M. Gabella, "Topology of Learning in Feedforward Neural Networks," in *IEEE Transactions on Neural Networks and Learning Systems*, doi: 10.1109/TNNLS.2020.3015790.
- [45] P. Nerisafitra and T. D. Wulan, "Backpropagation Neural Network for Determination of Mount Merapi Activities," *2019 International Conference on Computer Science, Information Technology, and Electrical Engineering (ICOMITEE)*, Jember, Indonesia, 2019, pp. 90-93, doi: 10.1109/ICOMITEE.2019.8921182.
- [46] P. W. Samodro, S. W. Sihwi and Winarno, "Backpropagation Implementation To Classify Dysgraphia In Children," *2019 International Conference of Artificial Intelligence and Information Technology (ICAIIIT)*, Yogyakarta, Indonesia, 2019, pp. 437-442, doi: 10.1109/ICAIIIT.2019.8834520.
- [47] M. I. Taqyudin, B. Irawan and C. Setianingsih, "Wood Classification Based on Fiber Texture Using Backpropagation Method," *2019 International Conference on Sustainable*

- Engineering and Creative Computing (ICSECC)*, Bandung, Indonesia, 2019, pp. 245-250, doi: 10.1109/ICSECC.2019.8907197.
- [48] A. Kaur and A. Jasuja, "Health monitoring based on IoT using Raspberry PI," *2017 International Conference on Computing, Communication and Automation (ICCCA)*, 2017, pp. 1335-1340, doi: 10.1109/CCAA.2017.8230004.
- [49] A. López-Vargas, M. Fuentes and M. Vivar, "IoT Application for Real-Time Monitoring of Solar Home Systems Based on Arduino™ With 3G Connectivity," in *IEEE Sensors Journal*, vol. 19, no. 2, pp. 679-691, 15 Jan.15, 2019, doi: 10.1109/JSEN.2018.2876635.
- [50] F. L. Valiente et al., "Internet of Things (IOT)-Based Mobile Application for Monitoring of Automated Aquaponics System," *2018 IEEE 10th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management (HNICEM)*, 2018, pp. 1-6, doi: 10.1109/HNICEM.2018.8666439.
- [51] "UP Specifications", UP Bridge the Gap. [Online]. Available: <https://up-board.org/up/specifications/>. [Accessed: 28- Apr- 2021].
- [52] H. Amano, *Principles and structures of FPGAs*. Yokohama: Springer, 2018, pp. 23-30.
- [53] Y. Xu and X. Guo, "Application of FPGA and complex embedded system in sports health data monitoring system", *Microprocessors and Microsystems*, p. 103445, 2020. Available: 10.1016/j.micpro.2020.103445.
- [54] C. Hao et al., "A Hybrid GPU + FPGA System Design for Autonomous Driving Cars," *2019 IEEE International Workshop on Signal Processing Systems (SiPS)*, 2019, pp. 121-126, doi: 10.1109/SiPS47522.2019.9020540.
- [55] Y. Liu, J. Briones, R. Zhou and N. Magotra, "Study of secure boot with a FPGA-based IoT device," *2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS)*, 2017, pp. 1053-1056, doi: 10.1109/MWSCAS.2017.8053108.

- [56] P. Dondon, J. Carvalho, R. Gardere, P. Lahalle, G. Tsenov and V. Mladenov, "Implementation of a feed-forward Artificial Neural Network in VHDL on FPGA," 12th Symposium on Neural Network Applications in Electrical Engineering (NEUREL), 2014, pp. 37-40, doi: 10.1109/NEUREL.2014.7011454.
- [57] A. Brown, "Cora Z7 Reference Manual - Digilent Reference", Reference.digilentinc.com, 2021. [Online]. Available: <https://reference.digilentinc.com/programmable-logic/cora-z7/reference-manual>. [Accessed: 30- May- 2021].
- [58] Y. Guan et al., "FP-DNN: An Automated Framework for Mapping Deep Neural Networks onto FPGAs with RTL-HLS Hybrid Templates," 2017 IEEE 25th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM), 2017, pp. 152-159, doi: 10.1109/FCCM.2017.25.
- [59] J. Ngadiuba et al., "Compressing deep neural networks on FPGAs to binary and ternary precision with hls4ml", Machine Learning: Science and Technology, vol. 2, no. 1, p. 015001, 2020. Available: 10.1088/2632-2153/aba042.
- [60] K. Vipin, Y. Akhmetov, S. Myrzakhme and A. P. James, "FAPNN: An FPGA based Approximate Probabilistic Neural Network Library," 2018 International Conference on Computing and Network Communications (CoCoNet), 2018, pp. 64-68, doi: 10.1109/CoCoNet.2018.8476889.
- [61] K. Vipin, "ZyNet: Automating Deep Neural Network Implementation on Low-Cost Reconfigurable Edge Computing Platforms," 2019 International Conference on Field-Programmable Technology (ICFPT), 2019, pp. 323-326, doi: 10.1109/ICFPT47387.2019.00058.
- [62] C. Kormanyos, "Fixed-Point Mathematics, Real-Time C++", Springer, Berlin, Heidelberg, pp. 267-289, 2018. Available: 10.1007/978-3-662-56718-0.
- [63] C. Cañedo Figueroa, ANÁLISIS DE ARTEFACTOS EN SEÑALES EEG PARA SU APLICACIÓN EN UN SISTEMA DE COMUNICACIÓN POR MEDIO DE PARPADEOS. Chihuahua, México: Instituto Tecnológico de Chihuahua, 2017.

- [64] O. Rivera Calderón, RECONOCIMIENTO DE MOVIMIENTO OCULAR Y PARPADEO MEDIANTE ANÁLISIS EEG Y ARTEFACTOS. Chihuahua, México: Insituto Tecnológico de Chihuahua, 2019.