Procesamiento embebido de señales cerebrales relacionadas con la imaginación de movimientos para aplicaciones de BCI

Arturo Castañeda Gonzalez
Jaime Paternina Roa

Tesis presentada a la Universidad Javeriana
como requisito parcial para optar por el título de
Ingeniero Electrónico

Noviembre 2013

Procesamiento embebido de señales cerebrales relacionadas con la imaginación de movimientos para aplicaciones de BCI

Arturo Castañeda Gonzalez
Jaime Paternina Roa

Director: Ing. Diego Mendez Chaves

Tesis presentada a la Universidad Javeriana
como requisito parcial para optar por el título de
Ingeniero Electrónico

Noviembre 2013

PONTIFICIA UNIVERSIDAD JAVERIANA

Facultad de Ingeniería
Carrera de ingeniería electrónica

Rector Magnífico: Padre Joaquín Emilio Sánchez García S.J.
Decano Académico Facultad de Ingenierá: Ing. Jorge Luis Sánchez Téllez, M. Sc.
Decano del Medio Universitario: Padre Antonio José Sarmiento Nova S.J.
Director del Departamente: Ing. Francisco Viveros Moreno
Director de Carrera: Ing. Jairo Alberto Hurtado Londoño, Ph. D.
Director del Proyecto: Ing. Diego Mendez Chaves

Noviembre 2013

NOTA DE ADVERTENCIA

La Universidad no se hace responsable de los conceptos emitidos por algunos de sus alumnos en los proyectos de grado. Solo velará porque no se publique nada contrario al dogma y la moral católica y porque no contengan ataques o polémicas puramente personales. Antes bien, que se vea en ello el anhelo de buscar la verdad y la justicia.

Artículo 23 de la Resolución No. 13, del 6 de julio de 1946,
por la cual se reglamenta lo concerniente a Tesis y Exámenes
de Grado en la Pontificia Universidad Javeriana.

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

Hoy por hoy, los avances en cuanto al estudio del comportamiento del cerebro humano no han sido lo suficientemente efectivos para describirlo, es tan complejo su accionar que a razón de esto durante años el ser humano ha luchado día a día por entender el funcionamiento del mismo, tanto así, que el presidente de Estados Unidos, Barack Obama, ha presentado en los últimos meses un ambicioso proyecto para estudiar el cerebro humano con una inversión inicial de cien millones de dólares. La iniciativa quiere acelerar el desarrollo y la aplicación de nuevas tecnologías para permitir a los investigadores obtener un mapa dinámico del cerebro que permita conocer cómo interactúan los complejos circuitos neuronales. [1]

Para todas estas investigaciones es necesario contar con una interfaz hombre-máquina que procese las señales provenientes del cerebro. A esto se le llama BCI (Brain-Computer Interface) el cual es un tipo de comunicación que tiene como objetivo crear sistemas que procesen señales cerebrales de un individuo convirtiéndolas en comandos específicos para un sistema de control particular; expresado de otra forma es transformar los pensamientos en acciones reales alrededor de nuestro entorno, gracias a la interacción de las diferentes disciplinas del conocimiento como son las neurociencias, la ingeniería biomédica y las ciencias de la computación.

El alcance de las aplicaciones para BCI va desde la diversión y el ocio hasta la predicción de epilepsias, la rehabilitación de la movilidad, las comunicaciones militares y la facilidad de desplazamiento para personas discapacitadas. Es en esta última donde el desarrollo de la tesis tiene cierta implicación, puesto que el resultado esperado a futuro es poder ayudar a personas enfermas de cuadriplejia a recobrar parcial o totalmente su autonomía.

La cuadriplejia es un trastorno que produce parálisis de los brazos, las piernas y el tronco. La causa más frecuente puede ser un traumatismo, que generalmente se debe a lesiones de la médula espinal las cuales afectan las vertebras cervicales [2]. Las personas con cuadriplejia tienen una calidad de vida limitada como consecuencia de su discapacidad motriz, además, los avances tecnológicos existentes no permiten suplantar todas las actuaciones del ser humano.

El proyecto implementará un clasificador de señales cerebrales relacionadas con la imaginación de movimientos basados en el desarrollo realizado en el proyecto de grado *Clasificación de Señales Cerebrales Relacionadas con la Imaginación de Movimientos para Aplicaciones de BCI* [3], llevándose a cabo en un sistema embebido mejorando su portabilidad, ya que la mayoría de los prototipos que se han planteado hasta el momento son desarrollados para sistemas de grandes capacidades de cómputo. Esta característica los hace voluminosos, restringiendo la integración con otros sistemas electrónicos y la creación de aplicaciones

portables.[4][5][6]

Teniendo en cuenta la arquitectura del sistema embebido, se produce una disminución notable del consumo de energía en comparación con los computadores de alto rendimiento. En consecuencia, se presentaría una disminución de las dimensiones de la fuente, dándole mayor autonomía y sostenibilidad.

Para realizar las pruebas es necesario obtener muestras de señales cerebrales, para esto se tomó una base de datos en Alemania llamada "Brain-Computer Interfacerealizada para la competencia BCI 2003 (Data ser III) [7]. Para la toma de dichas señales la comunidad participante debió imaginar el movimiento de una de sus manos catalogando cada ensayo con la etiqueta correspondiente (derecha o izquierda). Ahora bien, para almacenar las señales y desarrollar el algoritmo del programa se utilizará una computadora de board unica (SBC) PandaBoard la cual se seleccionó teniendo en cuenta aspectos como la documentación encontrada, los sistemas operativos soportados, los módulos de conectividad y las características técnicas.

En el desarrollo e investigación realizado por los ingenieros Salazar, Zaccaro y Cardenas [3], se analizan diferentes técnicas de extracción de características y clasificadores, encontrando que la combinación que mejor porcentaje de clasificación entregaba era la transformada discreta de wavelets (DWT) junto a análisis de componentes principales (PCA) y máquinas de vectores de soporte (SVM). Para implementar esto en el sistema embebido se decide emplear el lenguaje de programación Python utilizando las librerías PyWavelets para DWT y PyMI para SVM.

Como producto final se obtendrá un porcentaje de clasificación dependiendo de la cantidad de muestras ingresadas como prueba y sus respectivas etiquetas. Además, se enviaran señales cerebrales continuas en tiempo por medio de un socket TCP/IP emisor de conexión alámbrica o inalámbrica se envían las señales cerebrales de la base de datos, creando una ventana de tiempo en el receptor donde una vez que se ingresen datos se realizará el proceso de extracción de características y clasificación simultáneamente, teniendo así más de un proceso en ejecución al tiempo.

Cabe resaltar que el proyecto no se enfoca en el desarrollo de elementos netamente científicos sino que a su vez proporciona soluciones a futuro de necesidades insatisfechas del ser humano, todo con el fin de lograr mejorar la calidad de vida de las personas con limitaciones físicas. Sin embargo es importante resaltar que la interdisciplinariedad es un factor fundamental en el desarrollo de este proyecto, así mismo que el conocimiento desarrollado a lo largo de este documento pueda servir como base para la creación de nuevos proyectos de grado con un enfoque similar para lograr un trabajo conjunto y generar un producto final.

A continuación, se presentarán detalladamente los conceptos teóricos mencionados anteriormente, también se mostrará el diagrama de bloques y de flujo del algoritmo explicando el porqué de la utilización del hardware y software seleccionados. Por último se analizarán los resultados obtenidos en el proyecto, teniendo en cuenta factores como rendimiento de máquina, porcentajes de clasificación y análisis temporal.

Capítulo 2

Marco teórico

2.1. BCI

BCI (Brain-Computer Interface) es un tipo de comunicación hombre-máquina, que tiene como objetivo crear sistemas que procesen las señales cerebrales de un individuo y las convierta en comandos específicos para un sistema virtual o físico. Esta interfaz posee dos características que la diferencian de otros medios de comunicación hombre-máquina. La primera permite una conexión natural entre el humano y diferentes sistemas tanto físicos como virtuales; esta característica da la sensación de estar en un medio ajeno a su cuerpo, y la segunda consiste en extraer la información tanto cognitiva como emocional del usuario[8].

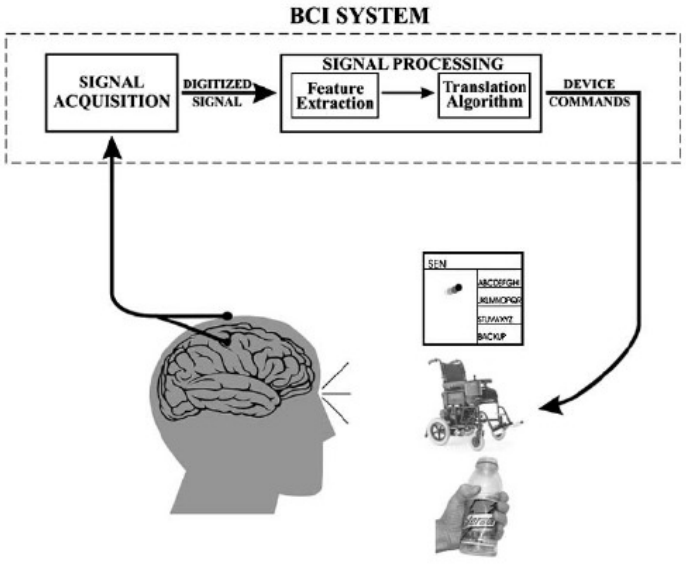

Figura 2.1: Operación básica de un sistema BCI. Tomado de [9].

Un sistema BCI está compuesto básicamente por tres partes como se puede ver en la figura 2.1;las cuales son:

Adquisición: Se encarga de extraer la actividad cerebral y digitalizarla.

Procesamiento de las señales: A éste módulo llega la señal digitalizada aplicándose diferentes decodificadores o técnicas para generar una serie de comandos predeterminados.

Aplicación: Su función es recibir los comandos y mostrarlos de una forma entendible para el usuario.

Para la adquisición de las señales se utilizan diferentes tipos de sensores. Si bien se sabe que el cerebro emite suaves impulsos eléctricos que pueden ser medidos en micro voltios y mediante un electroencefalograma(EEG), estas señales pueden ser registradas como una serie de tiempo estocástica no estacionaria, en la que la información asociada a los procesos mentales no se puede observar de forma directa. Existen modos de captura clínicos como los utilizados para hacer un electroencefalografía o productos comerciales como Emotiv[10] o MindWave[11].

El procesamiento de las señales se puede dividir en tres diferentes estados: preprocesamiento, extracción de características y detección y clasificación. El preprocesamiento consiste en mejorar las cualidades de las señales como la relación señal a ruido (SNR por sus siglas en inglés Signal to Noise Ratio). Con un bajo SNR será mas difícil detectar las características de las ondas cerebrales ya que estas pueden confundirse con el ruido del medio ambiente (Ruido Gaussiano Blanco). Sin embargo, un alto nivel de SNR hará mas fácil determinar estas características porque sobrepasaran el ruido. La siguiente fase es la extracción de características, la cual consiste en cambiar el dominio de la señal para resaltar la diferencia entre las distintas señales, de esta manera se hace mas fácil y confiable la etapa de clasificación. Además, se suele minimizar el número de muestras para reducir el tiempo a la hora de clasificar. Finalmente, en la etapa de detección y clasificación las señales se dividen en diferentes clases dependiendo de sus características. Cabe aclarar que se podrían presentar dos problemas; Primero, las ondas no son estacionarias, es decir que a medida que pasa el tiempo se comportan de forma diferente y segundo que las ondas cerebrales varían de persona a persona. Es por esta razón que no se puede utilizar un clasificador simple, por ejemplo de nivel o de frecuencia, se deben utilizar métodos de clasificación supervisada o no supervisada[12].

2.2. Clasificación de ondas cerebrales

En la conciencia del ser humano se han logrado distinguir cinco tipos de ondas dependiendo de su voltaje y frecuencia.

Ondas delta: su rango de frecuencias es de 0,2-3,5 Hz con voltajes entre 10 y 50 micro voltios. Normalmente están asociadas con etapas de sueño profundo. En la actividad cerebral estas ondas se presentan en sueño profundo sin soñar, estado hipnótico, hemisferio cerebral derecho en plena actividad.

Ondas theta: se encuentran en el rango de frecuencias de 3.5 y 7.5 Hz con voltajes entre 50 y 100 micro voltios. Estas ondas se le asignan a estados mentales como relajación, tranquilidad, creatividad, inicio de actividad plena del hemisferio izquierdo y desconexión del hemisferio derecho.

Ondas alpha: Son oscilaciones electromagnéticas que se encuentran entre los 8 y 12 Hz con picos de voltaje entre 100 y 150 micro voltios. Estas ondas se presentan durante periodos de relajación, con los ojos cerrados, pero todavía despierto, se aten úan al abrirse los ojos, con la somnolencia y el sueño. Además en la etapa de creatividad, inicio de actividad plena del hemisferio izquierdo y desconexión del hemisferio derecho.

Ondas beta: están asociadas con etapas de sueño nulo, alerta máxima, vigilante y miedo. Es la situación normal cuando se esta despierto, conduciendo, o en estado de alerta. Sus frecuencias varían entre 13 y 28 Hz con voltajes de 150 a 200 micro voltios.

Ondas gamma: oscilan entre los 25 y los 100 Hz, aunque su presentación más habitual es a 40 Hz.

Adquisición de ondas cerebrales

Con el fin de estudiar y calcular la actividad cerebral, la interfaz hombre-máquina (BCI) cuenta con sensores colocados en diversos puntos de la cabeza que perciben en tiempos distintos y sobre cualquier área del cerebro cada uno de los diferentes cambios en la actividad eléctrica y magnética producidos por las ondas cerebrales. El electroencefalograma (EEG) es un sistema utilizado para recoger la señal eléctrica del cerebro con electrodos desde el cuero cabelludo. Por lo general, dicha señal es ampliada y representada en forma lineal, de esta manera es posible interpretar la activad de las distintas zonas cerebrales en un tiempo determinado.

Dado que el equipo necesario para el EEG es económico, liviano y sí se compara con las otras formas de medición cerebral, es sencillo de utilizar; este método de medición es el más reconocido y ha sido utilizado en investigaciones por décadas. Su resolución temporal, es decir, la capacidad para detectar los cambios en un intervalo de tiempo es confiable. Sin embargo, la resolución espacial y el rango de frecuencia son limitados. El EEG es susceptible a las interferencias causadas por otras actividades eléctricas, tanto internas como externas. Para alcanzar grabaciones consistentes de áreas especificas del cerebro, es indispensable hacer uso del Sistema Internacional 10-20, el cual es un sistema estándar de posicionamiento de electrodos de acuerdo a la distancia total nasion-inion, repartidos en dimensiones equivalentes al 10 %, 20 %, 20 %, 20 %, 20 % y 10 %, como se puede observar en la figura 2.2.

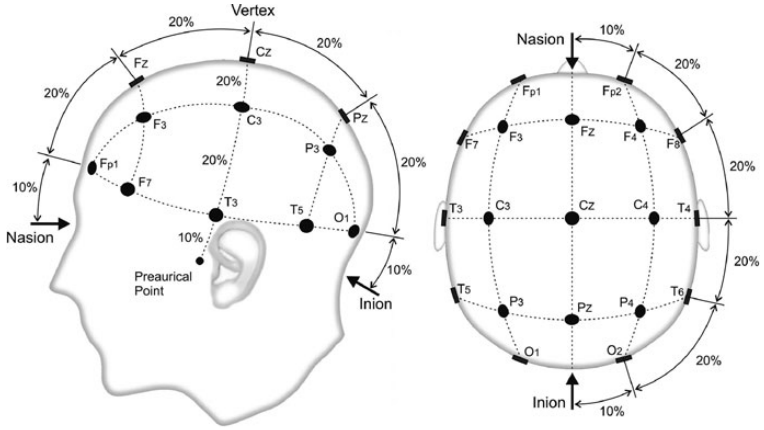

Figura 2.2: Sistema internacional de posicionamiento. Tomado de [12].

2.3. Wavelet

Una función wavelet es una onda de corta duración donde su energía se encuentra concentrada en el tiempo, es decir energía finita. Una de las características de este método es que permite analizar fenómenos transitorios (señales de muy corta duración o comienzos o finales abruptos[14]) y no estacionarios (sus componentes de frecuencia cambian en el tiempo [14]). Por otra parte, también tiene la capacidad de realizar un análisis tiempo-frecuencia el cual se caracteriza porque la información de frecuencia se encuentra ligada de forma explícita con la información temporal[15], esta característica permite analizar fenómenos tanto estacionarios como no estacionarios. Al igual que Fourier, wavelets se basa en la aproximación de señales usando superposición. La diferencia entre estos dos métodos radica en que en Wavelet varían tanto la escala como la frecuencia. Una wavelet ($\Psi(t)$) debe cumplir los siguientes postulados matemáticos.

1. Tener energía finita:
$$E = \int_{\infty}^{\infty} |\psi(t)|^2 \partial t < \infty \tag{2.1}$$

2. Cumplir con el postulado de la constante de admisibilidad:
$$E = \int_{0}^{\infty} \frac{|\hat{\Psi}(f)|^2}{f} \partial f < \infty \tag{2.2}$$
Lo que da a entender que la función no debe tener componente de frecuencia en cero, es decir $\hat{\Psi}(f) = 0$

3. Para funciones complejas ($\Psi(t)$) cuya trasformada de Fourier ($\Psi(f)$) tiene que ser real y anulada para frecuencias negativas.

2.3.1. Transformada de Wavelet

Al aplicar la trasformada de Wavelet a una señal, esta produce bloques de información en escala y tiempo. Estos bloques son producidos por una única función llamada Wavelet madre $\Psi(t)$, la cual está definida por la ecuación 2.3 donde **a** tiene como función dilatar o contraer la señal y **b** traslada la señal en tiempo. En la figura 2.3 se pueden observar algunas de dichas señales. También es de suma importancia saber que el proceso de la trasformada de Wavelet se conoce como análisis y el inverso como síntesis, pudiendo producirse diferentes niveles de descompresión de la señal en el análisis.

$$\Psi_{a,b} = \frac{W(\frac{x-b}{a})}{\sqrt{|a|}}; a, b \in \Re, a \neq 0 \tag{2.3}$$

2.3.2. Trasformada Wavelet discreta

El análisis para señales discretas usa una familia de Wavelet orto-normales [17], la cual está dada por la ecuación 2.4, donde **j** tiene como función dilatar o contraer la Wavelet madre Ψ mientras que **k** determina la posición.

$$\Psi_{j,k} = 2^{\frac{-j}{2}} \Psi(2^{-j}t - k) \tag{2.4}$$

Otro modo de apreciar la división de la señal es mediante la descompresión del árbol de Mallat el cual está compuesto por una serie de filtros pasa bajos y pasa altos como se puede observar en la figura 2.4, donde

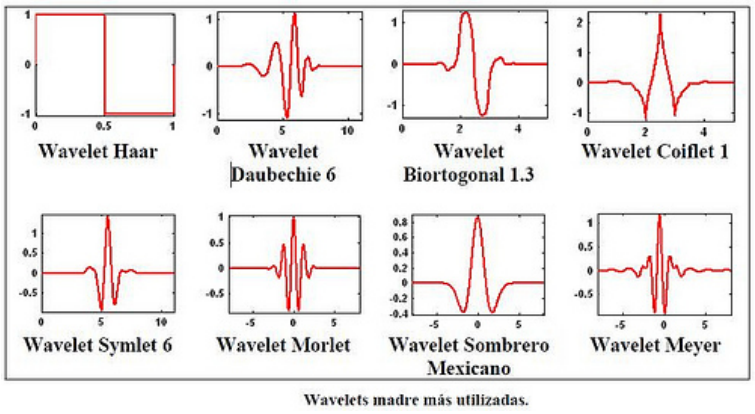

Figura 2.3: Wavelet madres más comunes. Tomada de [16].

x(n) es la señal discreta, **a(n)** aproximaciones , **d(n)** nivel de detalle, **h(n)** filtro pasa altos, **g(n)** filtro pasa bajos, ↓ 2 significa el proceso de decimación [17].

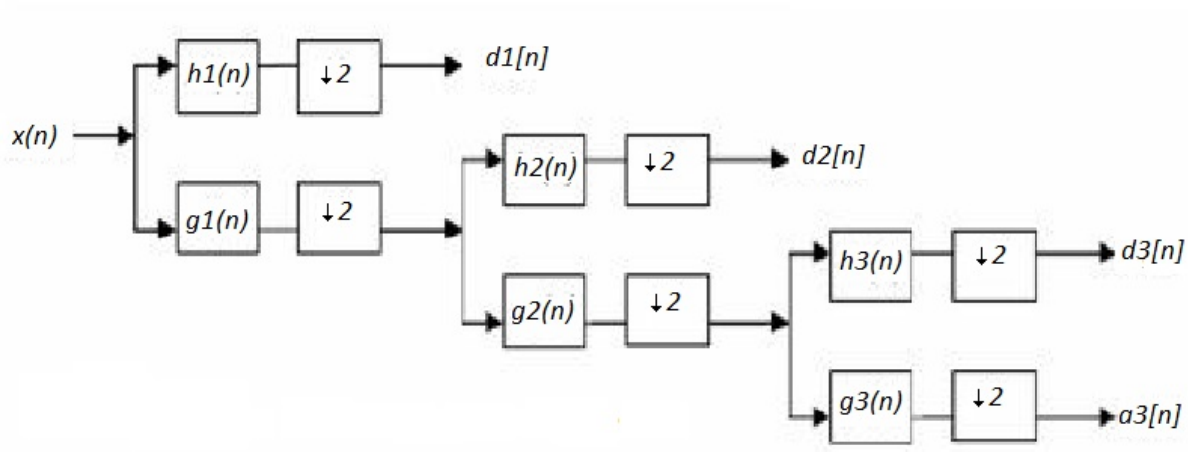

Figura 2.4: Árbol de mallat nivel 3. Tomada de [17].

2.3.3. Wavelet y ondas cerebrales

En la Figura 2.5 se aprecia una onda cerebral adquirida de la base de datos, se puede observar que esta señal posee las siguientes características: no periódica, no estacionaria (sus componentes de frecuencia cambian en el tiempo) y efectos transitorios, por consiguiente el análisis mediante wavelets es óptimo ya que cuenta con la particularidad de analizar señales con características similares a las ondas cerebrales. Otro beneficio que otorga es el análisis tiempo-frecuencia en un tiempo especifico.

2.4. Máquinas de soporte vectorial (SVM)

La base teórica en la cual están soportadas las SVM es la idea de la minimización del riesgo estructural (SRM Structural Risk Minimization) la cual consiste en encontrar el balance correcto entre precisión y

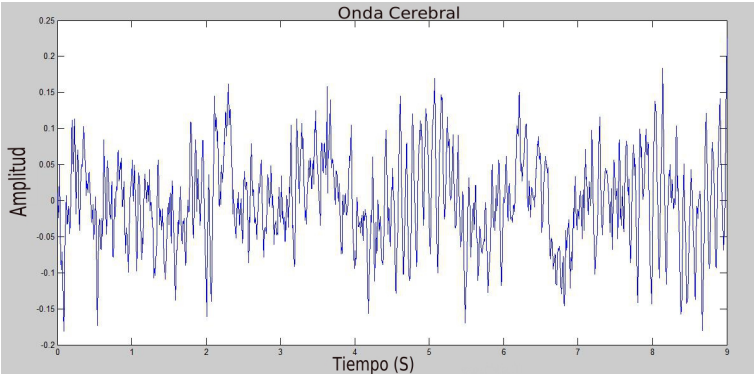

Figura 2.5: Ejemplo de onda cerebral pensando en la imaginación de movimientos graficada en MATLAB

capacidad de generalización. En otras palabras, lo que se busca con la SRM es tener un amoldamiento de muestras adecuadas evitando el ajuste bajo o el sobreajuste. Como un ejemplo de aplicación, al clasificar un árbol lo podríamos hacer por el color o por la cantidad de hojas, con la primera clasificación se cuenta con muy poca capacidad lo cual genera un ajuste bajo, mientras que con la segunda tenemos un sobre ajuste puesto que es muy difícil contar e igualar la cantidad específica de hojas que un árbol tiene.

La teoría de las SVM fue desarrollada inicialmente por V. Vapnik [18] y se basa en buscar para una tarea de aprendizaje dada con cantidad finita de datos una adecuada función que permita llevar a cabo una buena generalización, es decir la capacidad de una función para explicar el comportamiento de los datos dentro un nuevo dominio mas alto. [19] Una SVM mapea los datos de entrada y los preprocesa para representar los patrones en un espacio de mayor dimensión con respecto al original. El objetivo principal es encontrar un hiperplano de separación que divida el espacio de entrada en dos regiones y maximice el margen *m* como se puede observar en la figura 2.6.

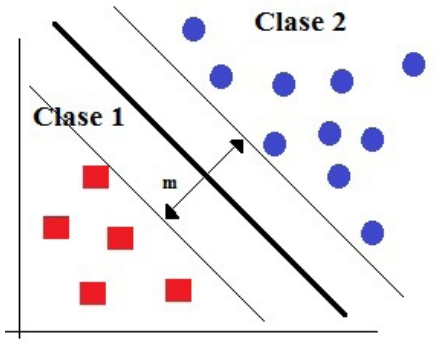

Figura 2.6: Hiperplano de separación entre dos clases de muestras

Maximizar el margen *m* es un problema de programación cuadrática, es decir que contiene una función objetivo y restricciones lineales, lo cual se resuelve por multiplicadores de Lagrange, utilizando el producto punto con funciones en el espacio. El resultado del hyperplano óptimo es una combinación de algunos puntos de entrada conocidos como vectores de soporte [20].

Dependiendo del tipo de muestras se puede dividir las SVM en dos casos, el linealmente separable y el no separable.

Caso linealmente separable

En la figura 2.7 se tiene un conjunto de datos etiquetados de la forma $(s_1, y_1), ..., (s_i, y_i))$ los cuales se pueden separar por medio de un hyperplano donde las clases no se intersectan entre ellas. A cada muestra se le da una etiqueta $y_i \in \{-1, 1\}$. Caber resaltar que es difícil encontrar un hyperplano óptimo en el espacio de entrada, por esto la solución es mapear el espacio de entrada en un espacio de características de una dimensión mayor y en este hallar el hyperplano óptimo $(x = \varphi(x))$ donde la notación del correspondiente vector es el espacio de características con un mapeo φ de $\Re^N$ a un espacio de características X, con la idea es hallar un hyperplano de la forma $w \cdot X + b = 0$ definidos por el par (w, b) donde si es linealmente separable debe cumplir con las inecuaciones 2.5. Para este caso se puede encontrar un único hyperplano optimo para el cual el margen entre diferentes clases está maximizado como se observa en la Figura 2.7.

$$\begin{cases} (w \cdot X + b) \geq 1, & y_i = 1 \\ (w \cdot X + b) \leq -1 & y_i = -1 \end{cases} i = 1, ..., n \tag{2.5}$$

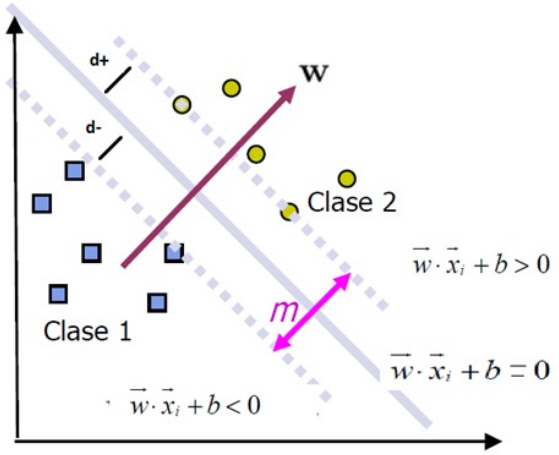

Figura 2.7: Conjunto de muestras linealmente separable por SVM. Tomado de [20]

Caso no linealmente separable

El caso no linealmente separable se presenta en la figura 2.8, en el cual las muestras de la clase 1 traslapan las de la clase 2. Para estos casos se separan utilizando una función kernel.

Utilizando función kernel: En la mayoría de los casos se debe utilizar funciones kernel para la clasificación, la razón es que las muestras en el mundo real en la mayoría de los casos no son linealmente separables y el porcentaje de error para utilizar margen suave es demasiado alto. Por lo tanto, la solución es convertir el espacio de entrada a un espacio de características como se puede observar en la figura 2.9, la cual por medio de una función calcula el producto punto a las muestras de entrada en el espacio de características Z, obteniendo así un espacio más manejable.

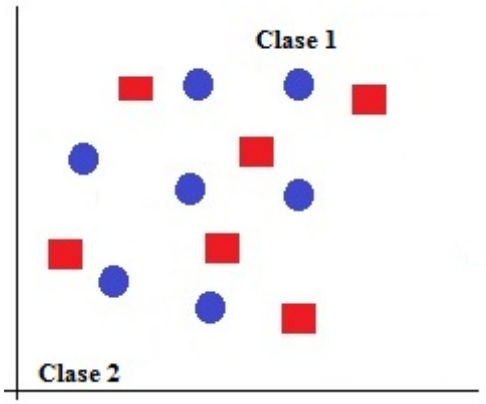

Figura 2.8: Conjunto de muestras no linealmente separable por SVM

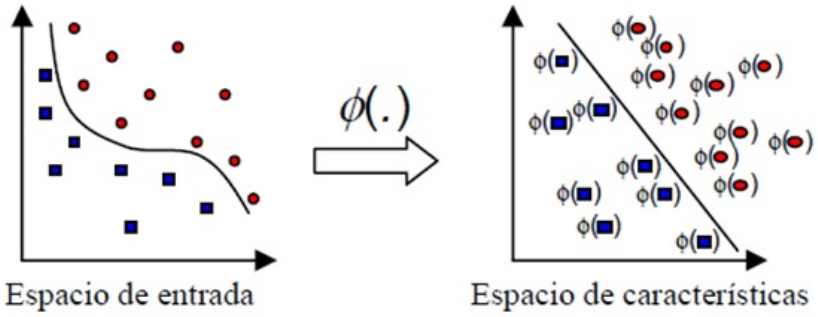

Figura 2.9: Transformación ideal de un conjunto de muestras utilizando funcion kernel. Tomado de [20]

2.5. Sistemas embebidos

Entre los componentes principales de un sistema embebido se puede distinguir el hardware, el software primario y el sistema operativo que entrega los mecanismos para ejecución de procesos. Ahora bien, el software que se utiliza la mayoría de las veces requiere pequeñas cantidades de memoria, posee capacidades limitadas de procesamiento y además mantiene un bajo consumo de energía en todo momento.

Características: Es de suma importancia mencionar las características que diferencian a los sistemas embebidos de otros sistemas de computo.

Aplicación específica: Un sistema embebido usualmente genera una sola acción (programa) y la hace de manera repetitiva, debido a que son equipos para tareas específicas.

Interacción con el entorno: La mayoría de los sistemas embebidos deben reaccionar ante cambios en el ambiente para el cual están diseñados generando diferentes acciones en tiempo real, ya sean cálculos o mediciones dentro de un límite corto de tiempo.

Confiabilidad y disponibilidad: Estos sistemas al estar enfocados en tareas específicas deben ser muy confiables y a prueba de errores, puesto que alguna mínima equivocación en el cálculo

podría generar errores graves, un ejemplo podría ser el sistema de control de frenado de un automóvil.

Limitaciones: Los sistemas embebidos deben ser poco costosos, poco voluminosos, tener buen desempeño en tiempo real y consumir poca energía.

Facilidad de mantenimiento y actualización: Estos equipos no deben ser complejos y deben ser fáciles de actualizar.

2.5.1. Componentes principales de un sistema embebido

En la figura 2.10 se observa el diagrama de bloques general de un sistema embebido que se divide en cuatro partes fundamentales: el procesador, las memorias, los dispositivos de entrada y los de salida. El primer bloque es la unidad central de procesamiento la cual puede estar basada en un microprocesador, microcontrolador, o DSP dependiendo del tipo de sistema. El segundo bloque son los dispositivos de entrada que la mayoría de veces son los encargados de realimentar el sistema, existe un módulo de entradas/salidas analógicas y digitales el cual se usa para digitalizar señales físicas del exterior. El tercer bloque son los dispositivos de salida los cuales se encargan de mostrar el proceso de la tarea realizada por el sistema, algunos tienen entradas HDMI, Serial, etc. Por último esta el bloque de memoria donde se almacenan todos los datos de manera estática o dinámica.

La comunicación de los sistemas embebidos es trascendental, por lo tanto la mayoría cuentan con entradas de red cableada o inalámbricas basadas en protocolos internacionales como el RS-232, RS-485, USB, Wi-fi, etc. Otros módulos importantes internos son el módulo de reloj el cual es el encargado de generar las diferentes señales de control del procesador y el de energía que genera las diferentes tensiones y corrientes necesarias para alimentar los distintos circuitos del SE. [25]

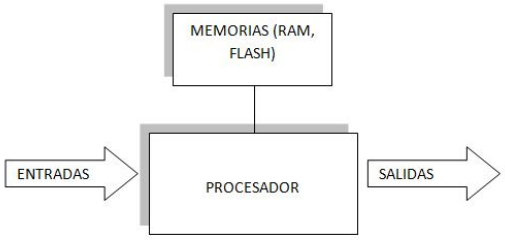

Figura 2.10: Diagrama en bloques de un sistema embebido

2.6. Sistema operativo GNU/linux

El Sistema Operativo (SO) es un conjunto de rutinas de software que permite una conexión entre la máquina y el usuario, proporcionándole un acceso a los recursos de la máquina de una forma fácil y segura, en otras palabras el SO se encarga de codificar todas las peticiones del usuario en una forma que el hardware las pueda ejecutar y de igual manera reflejarle el resultado al usuario de una forma amigable y entendible. GNU/Linux es un término usado para referirse a la combinación entre el núcleo Linux creado en 1991 por

Linus Torvalds con el proyecto GNU fundado por Richard Stallman en 1983. Este software esta cubierto bajo la licencia GPL [28] la cual permite usar, copiar, modificar y redistribuirlo pero únicamente bajo la misma licencia. Existen varias distribuciones que utilizan o que se basan en el núcleo Linux, como por ejemplo Ubuntu [29], Debian[30] y Gentoo[31]. Así mismo el núcleo de Linux tiene diferentes versiones las cuales pueden clasificarse de distintas maneras teniendo en cuenta una codificación basada en cuatro números, en esta codificación el primer número hace referencia a la versión del núcleo en sí, el segundo número comprende la revisión del kernel, el tercero por su parte corresponde a cambios menores de drivers de hardware (este solo se cambia cuando se incluyen nuevas características o nuevos drivers) y por último el cuarto número se encarga de los BUG-FIXES y los parches de seguridad. Las características de este SO son:

Multitarea: el núcleo esta implementado para ejecutar a la vez múltiples procesos, estos pueden ser ejecutados en un único procesador o en varios si la arquitectura lo permite.

Multiusuario: esta característica permite que varios usuarios compartan los mismos recursos de una forma organizada y segura. Además, cada usuario tiene diferentes niveles de acceso para garantizar la seguridad del sistema y su privacidad.

Multiplataforma: Linux soporta una gran cantidad de arquitecturas, entre las mas destacadas se encuentran: la serie i386 para procesadores intel de 32 bits, **arm y armel** para procesadores ARM utilizada por un gran número de microcontroladores [32], AMD64 procesadores de 64 bits.

Administración de memoria: Linux posee un espacio de intercambio, el cual tiene como función apoyar a la memoria física cuando esté sobrecargada, ese espacio de intercambio se encuentra alojado en el disco duro de la máquina. Este tipo de memoria se conoce como memoria virtual, la cual tiene como ventaja proporcionarle una memoria adicional al SO; sin embargo, ésta es más lenta que la memoria física. Ese espacio de intercambio se suele llamar como SWAP.[33].

Redes: Linux posee una capa de red muy fiable y rápida, soporta un gran número de protocolos como TCP/IP tanto la versión 4 como la versión 6.[33]

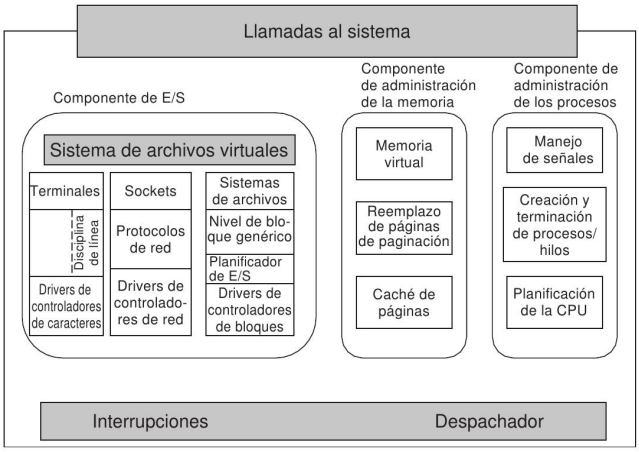

Figura 2.11: Estructura del kernel de un SO. Tomado de [26]

El kernel se comunica directamente con el hardware, pero este a su vez esta compuesto por diferentes componentes como se puede ver en la figura 2.11. En el primer nivel se encuentran los manejadores de interrupciones, los cuales son la forma primordial de comunicación con los diferentes dispositivos de hardware. El componente E/S contiene todos los drivers del núcleo que permiten intercambiar información entre el sistema y los elementos de hardware. En Linux existen dos tipos distintos de drivers: drivers de dispositivo de caracteres y drivers de dispositivos del bloques. La diferencia radica en que los segundos permiten los accesos aleatorios y búsquedas [26].
El componente de la administración de la memoria es el que permite la comunicación entre la memoria física y la memoria virtual (Swap); además, mantiene en caché los elementos ejecutados recientemente.
El componente de administración de procesos tiene como función crear o finalizar hilos. Por ultimo, el bloque llamadas al sistema es el encargado de atender todas las llamadas provenientes del mismo.

2.7. Procesos computacionales

A medida que pasa el tiempo, las computadoras realizan mayor cantidad de tareas simultáneamente, toda esta actividad hace conmutar la CPU y este cambio debe efectuarse rápidamente, cuando se tiene mas de un procesador se hace mas difícil manejar las actividades en paralelo. Por lo cual se ha diseñado un modelo teórico llamado procesos secuenciales, que facilita la comprensión de los procesos que se ejecutan en paralelo. “Un proceso es simplemente una instancia de un programa que se está ejecutando”[26].

El modelo conceptual le asigna una CPU virtual a cada proceso activo, donde cada uno de estos CPU tiene su propio contador lógico pero desde luego solo existe un contador físico. Cuando uno de estos procesos inicia se copia el contador lógico en el físico y cuando estos terminan el contador físico se guarda en el contador lógico, pero sin duda solo un proceso se ejecuta en un instante de tiempo determinado. Todo proceso esta ligado a: una entrada, una salida y un estado. Existen principalmente cuatro formas de iniciar un proceso: “arranque del sistema, la ejecución desde un proceso, una petición hecha por el usuario y el inicio de un trabajo por lotes”[26], de igual forma se puede terminar un proceso debido a: salida normal, salida por error, error fatal y eliminado por otro proceso. Cada uno de estos procesos tienen tres estados los cuales son: ejecución, listo y bloqueado.

Cuando se utiliza la multiprogramación la eficiencia de la CPU se puede mejorar. Para analizar el uso de la CPU se utiliza un método probabilístico, un proceso tarda una cierta parte (p) de su tiempo esperando una interrupción E/S, con n procesos en memoria, entonces el uso del CPU está dado por la formula 2.6.

$$usodelCPU = 1 - p^n \tag{2.6}$$

Capítulo 3

Especificaciones

Para la selección del hardware y el software es necesario tener en cuenta la frecuencia promedio a la que se muestrean los datos en un electroencefalograma. En el caso de la base de datos utilizada para la tesis y el dispositivo comercial EMOTIV la frecuencia es de 128 Hz (7,81 ms por muestra). Teniendo en cuenta que el objetivo a futuro de este proyecto es la integración del algoritmo realizado con un electroencefalograma el sistema seleccionado debe cumplir las siguientes especificaciones:

1. La velocidad de respuesta del sistema debe ser menor a 0.33 s siendo este el tiempo de clasificación promedio obtenido en el desarrollo realizado por los ingenieros Salazar, Zaccaro y Cardenas [3].

2. Se debe contar con una conexión fiable tipo TCP/IP para así evitar la pérdida de datos al momento de la recepción de señales cerebrales continuas en tiempo.

3. Es importante que la tarjeta de desarrollo cuente con un módulo bluetooth para que en próximas investigaciones se tomen las señales de un electroencefalograma EMOTIV el cual tiene este tipo de comunicación.

4. El periodo al cual se muestrean las señales (0.00781 s) es menor que el tiempo de clasificación (0.33 s), por esta razón y como no se desea perder datos del EEG es necesario seleccionar un sistema embebido que tenga un procesador de doble núcleo para así realizar dos procesos independientes los cuales serían la clasificación y la captura de datos.

3.1. Diagramas de bloques

Descripción breve del proyecto

La base de datos "Brain-Computer interface en Alemania"[7] está compuesta por ensayos de tres canales que serán divididos en dos partes: los datos que se destinarán para entrenamiento y los que se destinarán para prueba, dicha división se hará mediante correlación cruzada.

A los datos de entrenamiento se les aplicará la transformada discreta de wavelet (DWT) obteniendo un vector de características con el cual se entrenará la máquina de soporte vectorial (SVM). Con el clasificador entrenado se proporcionan los datos de prueba aplicando DWT y SVM obteniendo así un resultado de clasificación entre los dos tipos de señales (Derecha e Izquierda), con este resultado se iguala el algoritmo

propuesto en el trabajo de grado “Clasificación de Señales Cerebrales Relacionadas con la Imaginación de Movimientos para Aplicaciones de BCI"[7] y se cumple con la primera parte del proyecto.

En la segunda parte se realizará una simulación de envío de señales cerebrales continuas en tiempo como lo haría un electroencefalograma EMOTIV tomando como referencia la base de datos mencionada anteriormente [7]. Para la simulación se aplicará un algoritmo que concatene las señales de prueba generando una continua en tiempo como se presentaría en el cerebro humano, posteriormente se envían por un socket a la tarjeta de desarrollo a la frecuencia real que fueron tomadas, al llegar los datos se almacenarán las señales en una ventana de tiempo desplazable a la cual se le aplicará el proceso de clasificación descrito anteriormente.

En la figura 3.1 se muestra el diagrama en bloques del proyecto inicial en el cual se clasifican las señales cerebrales guardadas en la base de datos y en la figura 3.2 se muestra el proceso de envío de señales cerebrales continuas en tiempo.

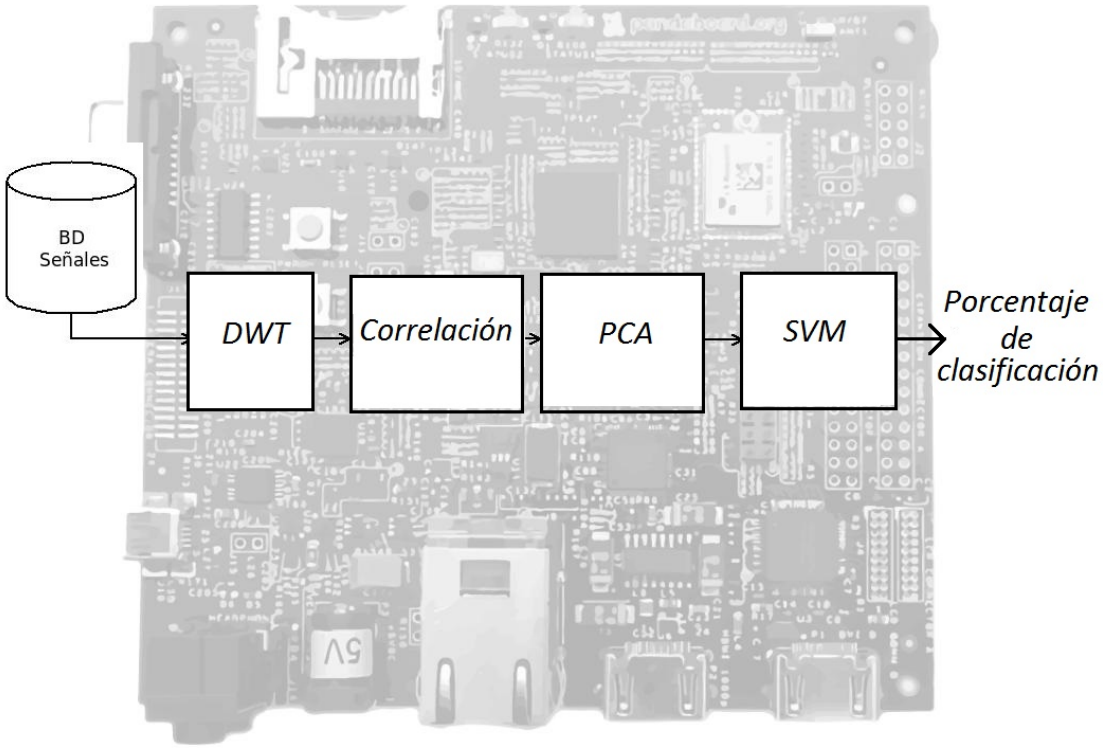

Figura 3.1: Diagrama de bloques proyecto inicial

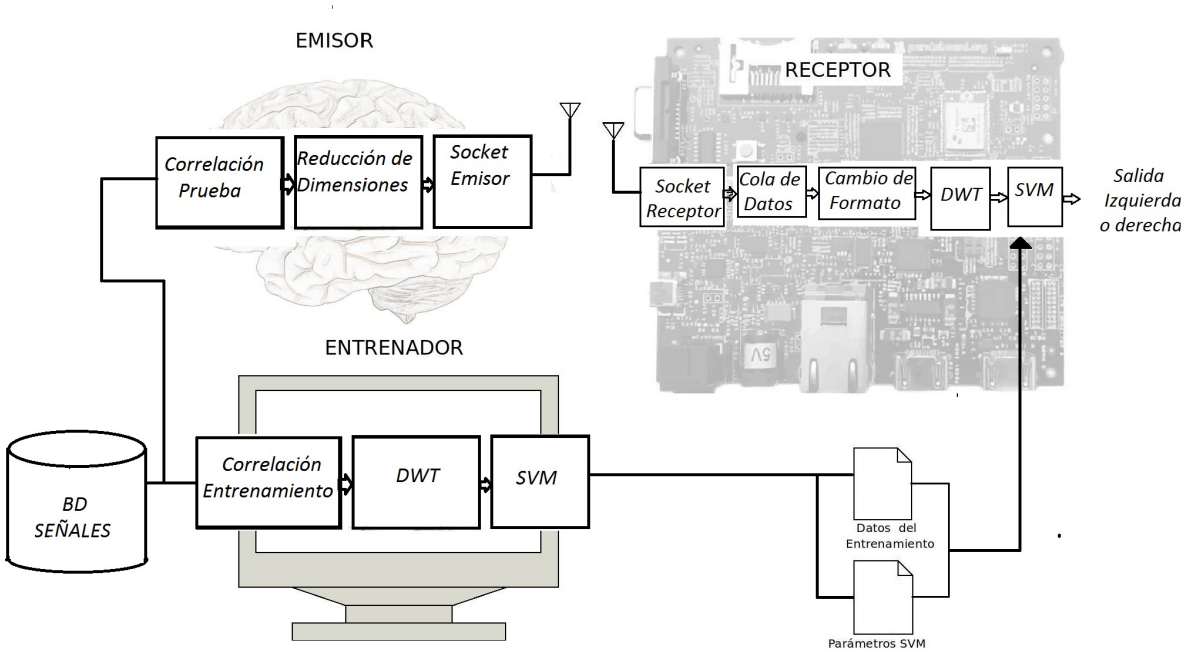

Figura 3.2: Diagrama de bloques del envío de señales cerebrales continuas en tiempo

3.2. Hardware

Sistema embebido

Es de suma importancia escoger el sistema embebido adecuado para el proyecto, por esto se citan en el cuadro 3.1 las siguientes computadoras de board única (SBC por su nombre en ingles Single Board Computer) con sus respectivas características técnicas.

	Panda Board	**Beagle Board**	**Raspberry Pi**	**SIEBOT2**
Procesador	Dual core ARM9 1Ghz	ARM8 720 Mhz	AR11 700Mhz	ARM9 454 Mhz
Mem RAM	1 Gb	256 Mb	256-512 Mb	64 Mb
Documentación	Alta	Media	Regular	Poca

Cuadro 3.1: Comparativo entre diferentes sistemas embebidos

Una SBC es una computadora constituida por una tarjeta madre donde se encuentra todo integrado, el procesador o microprocesador, las memorias y los módulos de entrada/salidas. Estas tarjetas son utilizadas en aplicaciones que no requieren demasiado procesamiento y que cumplen tareas específicas, en el caso de este trabajo la clasificación de señales cerebrales. Otra característica importante son las pequeñas dimensiones que estas poseen.

Después de analizar diferentes aspectos y comparar las SBC propuestas se decidió seleccionar la PandaBoard [35]. La primera razón por la que se escogió esta tarjeta fue por disponibilidad puesto que se encontraba en el laboratorio de la facultad y se podría trabajar con ella de inmediato; otra razón no menos importante que la anterior fue la cantidad considerable de documentación existente en Internet como proyectos, hojas de datos, manuales de instalación y una gran comunidad de desarrolladores dispuestos a colaborar con cual-

quier inquietud. Paralelamente a la documentación, la tarjeta soportaba sistemas operativos Linux el cual fue seleccionado previamente por las razones que se expondrán más adelante. También fue importante analizar los tipos de conectividad del sistema, la pandaBoard cuenta con modulos wi-fi, WLAN/Bluetooth, ethernet y USB que servirán al momento de interactuar con otros dispositivos de forma remota. La conexión bluetooth podría servir para futuros desarrolladores puesto que el modelo de electroencefalografía (EMOTIV [10]) proporciona las señales por este medio de comunicación. Por último también fue criterio de selección el tipo de procesador (doble núcleo) dado que serviría para realizar programación multiproceso de ser necesario. En la figura 3.3 se muestra la tarjeta seleccionada.

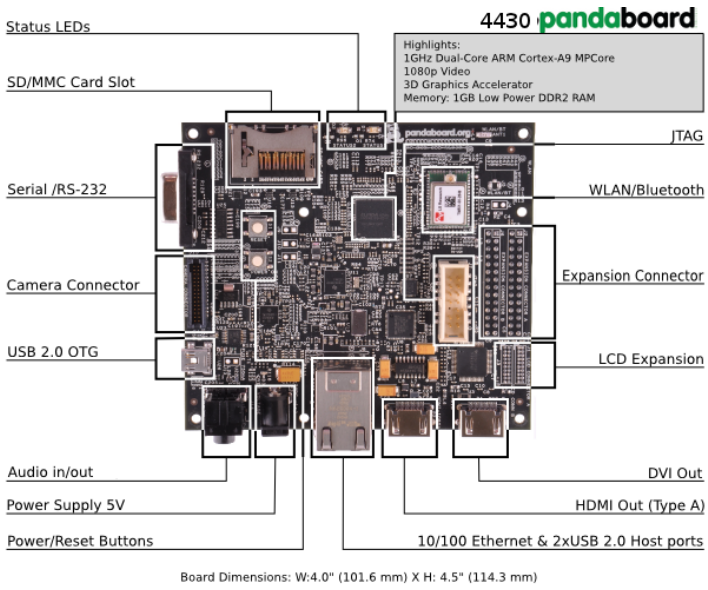

Figura 3.3: Puertos pandaboard. Tomado de [35].

3.3. Software

3.3.1. Lenguaje de programación

El lenguaje de programación es el encargado de expresar procesos por medio de un conjunto de símbolos y expresiones de fácil entendimiento para el usuario las cuales se transforman en una secuencia lógica para resolver un problema en particular. Entre los procesos que se llevan a cabo al momento de generar un programa informático está la prueba, la depuración y la compilación (si aplica) ya que existen dos formas de ejecutar un programa, por medio de un interprete conociéndose este como lenguaje interpretado, o lenguaje compilado. El tipo de ejecución es independiente del lenguaje que se esté utilizando, pero cabe resaltar que algunos son diseñados para ser interpretativos, como es el caso de Java y Python. El objetivo final de la compilación es generar un código de máquina en sistema binario y así ser entendido por los sistemas de computo.

Entre los lenguajes de programación propuestos para el proyecto se encontraban C++, Java y Python. El lenguaje seleccionado fue Python, siendo este de alto nivel creado por Guido Van Rossum, un científico de la computación de nacionalidad holandesa a finales de los años 80. Este fue diseñado con el fin de obtener

códigos de fácil acceso y lectura, multiplataforma y soporte de objetos. [36]

Con relación a la popularidad de Python se puede notar que en los últimos años se ha incrementado la cantidad de usuarios debido a la fácil utilización y la rapidez de aprendizaje que éste tiene, tanto así, que en el 2011 linux journal le otorgo por tercer año consecutivo el reconocimiento como el mejor lenguaje de programación escogido por votación de desarrolladores y usuarios [37]. Esta es la principal razón por la cual se escoge a python como lenguaje de programación ya que será muy fácil para los proyectos futuros entender los algoritmos realizados.

Otra de las ventajas de Python es que se ejecuta con bytecode haciéndolo así muy rápido. Además se cuenta con una cantidad considerable de librerías bien documentadas y explicadas como el caso de PyWavelets y pyml que se utilizarán en el desarrollo del objetivo del documento.

Al ser este un lenguaje interpretado de scripts puede ser ejecutado en diferentes arquitecturas, haciéndolo así multiplataforma evitando que se tenga que escribir el código desde cero en futuros avances. La ultima razón para escoger python es la existencia de un driver para el electroencefalograma EMOTIV [10] con código libre el cual se podrá unir al desarrollo de esta tesis.

3.3.2. Sistema operativo

Al desarrollar una aplicación sobre un sistema embebido es de utilidad un sistema operativo (SO), éste presenta una serie de herramientas que facilitan y optimizan el uso del microcontrolador como lo son los diversos drivers para cada uno de los dispositivos hardware que éste contiene y el soporte a distintos lenguajes de programación. Entre los SO para circuitos embebidos se tienen: QNX[38], VxWorks[39] y SO con núcleo Linux.

Desarrollar aplicaciones para sistemas embebidos con núcleo Linux implica beneficios como la reutilización de algunos componentes realizados por terceros los cuales se encuentran bajo la licencia GPL [28], la cual le otorga al usuario el poder de usar y editar el software sin costo alguno. Otro beneficio apreciable es que Linux posee una considerable cantidad de drivers o programas como se hace referencia en [40]; esta serie de componentes permite realizar un desarrollo más específico y permite crear proyectos o programas cada vez más complejos.

El SO con núcleo linux tiene implementaciones comerciales como: Montavista[41] y Timesys[42] e implementaciones libres: Embedded Debian[43], Ubuntu[29], Poky[44]. Las comerciales mencionadas proveen herramientas y entornos que facilitan la construcción del kernel (especializado en tiempo real) y la adaptación para el sistema embebido, pero tienen el problema de ser incompatibles con todo el hardware comercial, mientras que las implementaciones libres soportan mayor cantidad.

Otra de las ventajas del SO linux es que se han implementado plataformas como Launchpad[45] en las cuales mediante desarrollo colaborativo de software se agiliza el proceso de fallas. Algunos de sus principales componentes son: bugs para hacer seguimiento de los errores en programas, blueprint para nuevas especificaciones y translations, un portal para traducciones.

El núcleo Linux en su origen no soportaba tiempo-real pero se han creado parches que permiten su implementación como PREEMPT-RT[46], con el cual es posible cambiar la latencia del kernel, es decir se reduce el tiempo en que éste puede presentar interrupciones[47]. Además de estos beneficios el kernel Linux está implementado para una gran cantidad de arquitecturas [48]; las cuales soportan multitareas, lo que permite que varios programas se ejecuten a la vez por medio de procesos o hilos[49] permitiendo SMP

(Symmetric Multi-Processing)[50].

Por las razones mencionadas anteriormente se implementará un SO con nucleo linux; para definir la distribución a usar se crea el cuadro comparativo 3.2 en el cual la mejor opcion es Ubunto Core porque tiene gran cantidad de binarios portados a la arquitectura ARM, contiene todos los drivers de la PandaBoard y Pip, el instalador de librerías de Python.

	Ubuntu core	**Ubuntu desktop**	**Poky**	**Emdebian**
Pandaboard Soportada	✓	✓	no totalmente	no totalmente
Repositorios Disponibles	Si	Si	No	No
Python y Librerías	Si	Si	No	Si
Pip	Si	Si	No	Si
Documentación Disponible	Alta	Alta	Regular	Baja
Requisitos mininos de hardware	Pocos MB de RAM 20MB de Flash	RAM 256 MB Disco Duro 5GB	Pocos MB de RAM Pocos MB de flash	pocos MB de RAM Pocos MB de Flash

Cuadro 3.2: Comparativa entre diferentes distribuciones

Capítulo 4

Desarrollo

4.1. Instalación ubuntu core en pandaboard

Antes de comenzar con el desarrollo de la aplicación es necesario instalar el sistema operativo en la PandaBoard. Ubuntu core es una distribución compuesta por un sistema de archivos reducidos especialmente diseñado para máquinas de pocos recursos fundamentalmente en arquitectura ARM; esta distribución se puede descargar de la página oficial de Ubuntu [51] y se guió del manual .ºMAP Ubuntu Core"[52] documentado por OMAPpedia.

A continuación se describirá brevemente la instalación del SO y del software necesario para desarrollar la aplicación:

1. Preparar una SD con dos particiones `rootfs` y `boot`:

   boot: Con sistema de archivos FAT32.

   rootfs: Con sistema de archivos EXT3.

   Cabe resaltar que la partición de `rootfs` debe ser mayor que `boot`. Se recomienda que la SD sea al menos de 5GB. El tamaño de `boot` debe ser por lo menos de 15MB y el resto de la memoria se destina a `rootfs`.

2. Descargar los archivos de arranque y el sistema de directorios.

   MLO: Es un x-loader. Es el encargado de iniciar los relojes del sistema y la memoria después ejecuta el u-boot.

   U-boot: Inicia diferentes componentes del sistema y le pasa el control al kernel.

   UImage: Kernel Linux.

   Filesystem: Sistema de archivos.

3. Copiar MLO, U-boot y UImage en `boot`. Descomprimir el sistema de archivos en `rootfs`.

4. Modificar y crear algunos archivos para que Ubuntu-core 12.04 inicie por la consola serial.

5. Configurar las instancias de red.

6. Configurar los repositorios de Ubuntu; estos repositorios son una base de datos la cual contiene todos los binarios oficiales para la arquitectura del sistema, en este caso armhf.

7. Actualizar el sistema.

8. Instalar los drivers correspondientes para la PandaBoard.

9. Instalación de Python 2.7, pip (instalador de librerías de python), Scipy (conjunto de paquetes especializados para Ingenierías, ciencias y matemáticas). Todos estos paquetes se encuentran en los repositorios de Ubuntu.

10. Instalar PyWavelets y PyMl mediante pip.

Esta guía se explicará a profundidad y claridad en el anexo Ïnstalación de Ubuntu Core".1.

4.2. Proyecto inicial

En la figura 3.1 se muestra el diagrama en bloques general de la migración del código realizado para matlab por los ingenieros Salazar, Zaccaro y Cardenas [3]. El código se divide en 5 partes fundamentales: la base de datos, DWT (se explicará en la sección 4.3.1),PCA, correlación cruzada y SVM.

Base de datos: La adquisición de las señales cerebrales se tomará del banco de datos de Brain-Computer Interface en Alemania [7], destinada a la comunidad participante en la competencia BCI 2003 (Data ser III) . En este se dispone de 140 ensayos de tres canales (C3,CZ,C4), cada una de estas muestras fue registrada durante 9 segundos a una frecuencia de muestreo de 128 Hz. (1152 muestras). Ahora bien, en la figura 4.1 se muestra la configuración en tiempo de cada ensayo, luego de dos segundos se presenta un estímulo acústico para indicar el inicio de la prueba y posteriormente se muestra una cruz durante un segundo, finalmente entre 3 y 9 segundos se le presenta al usuario una flecha hacia la derecha o izquierda, mientras éste debe imaginar el movimiento de la mano correspondiente a la dirección indicada (clase 1 - clase 2).

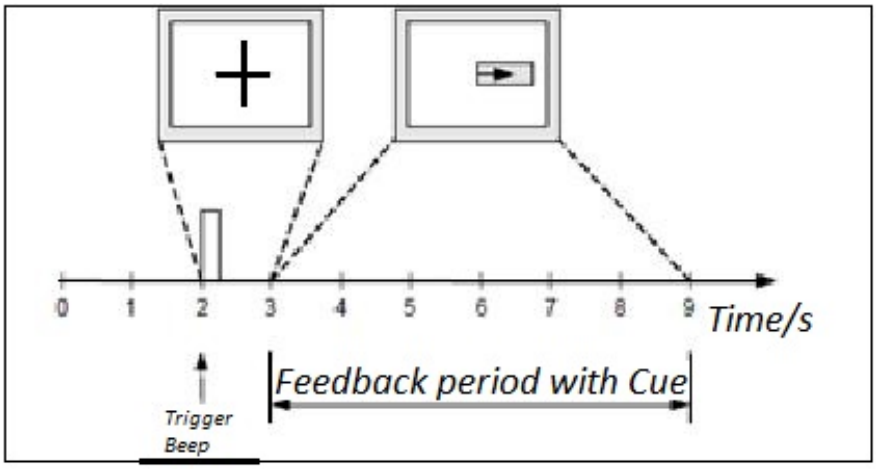

Figura 4.1: Captura de datos. Tomada de [7]

Debido a que son 1152 muestras, 3 canales por ensayo, la dimensión de la base de datos es de <1152x3x140>, de las 140 muestras totales, se tomarán 112 para entrenamiento y 28 para prueba.

El tipo de archivos en el cual se encontraban los datos en la base de datos es .mat, razón por la cual es necesario introducir la librería scipy.io que lee este tipo de archivos, los datos de la base se encuentran divididos en dos bloques, las etiquetas y las muestras por ensayo. Las etiquetas deben ser tratadas de forma especial, lo primero es convertirlas a tipo arreglo y luego de eso cambiar el valor de referencia que se encontraba en 2 para la mano derecha por -1, teniendo como resultado final 1,-1 (izquierda, derecha). Ya teniendo los datos para trabajar, conociendo el número de muestras, canales y ensayos, se procede a realizar la transformada discreta de wavelets, la cual se explicará en el numeral 4.3.1.

Correlación cruzada: La correlación cruzada es una técnica utilizada para dividir un espacio de características, en el caso del desarrollo del proyecto se empleará dicha técnica para dividir el espacio en datos de entrenamiento y datos de prueba. Como se dijo anteriormente la división de datos sera 112-28, teniendo dos grupos, uno de 112 X nc para entrenamiento y otro 28 X nc para prueba, donde nc es la cantidad de características que entrega DWT por ensayo. Para entender fácilmente la correlación se crea la figura 4.2, donde se muestra el proceso de división para cada uno de los 5 casos de w (variable que modifica la correlación). Al cambiar el w se desplaza la ubicación de los datos que se utilizarán para prueba, dejando el restante para entrenamiento.

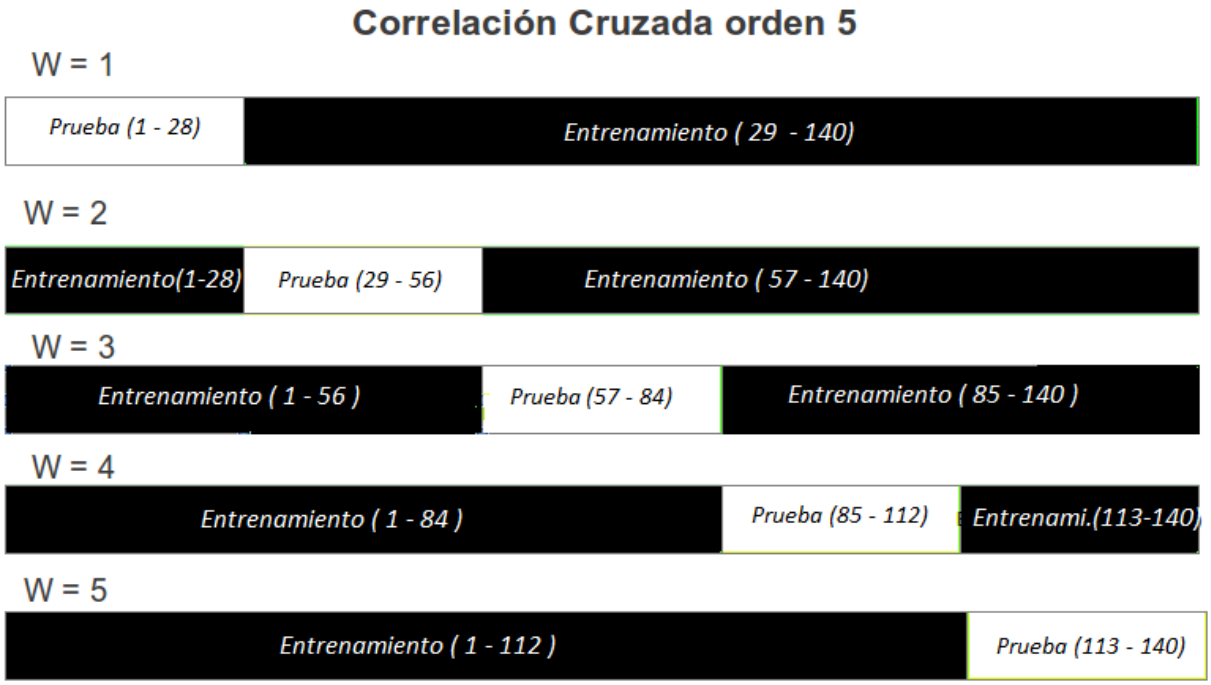

Figura 4.2: Correlación cruzada orden 5

SVM: Luego del algoritmo haber realizado los pasos anteriormente mencionados se llega a la clasificación. Ésta se divide en dos partes: entrenamiento y pruebas. Para el entrenamiento, la libreria "pyML- machine learning in Pyhton"necesita crear un documento tipo CSV (comma-separated values) el cual es

un formato abierto para representar datos en forma de tabla. El algoritmo de creación de archivo lo realiza con el siguiente formato: "Label,Feature1,Feature2,Feature3,Feature4,Feature5,Feature6", haciendo línea a línea teniendo como resultado un archivo de entrenamiento de 112 renglones, uno por cada ensayo.

Al tener creado el archivo para entrenamiento se realiza el mismo procedimiento pero en este caso para pruebas, a continuación se crea la SVM agregando el parámetro C de entrada que compensa los errores de entrenamiento y los márgenes rígidos. En este orden de ideas el parámetro C ideal sería infinito, pero cuanto mayor es éste, mayor es el tiempo de entrenamiento; para el presente desarrollo se toma 450 puesto que fue el valor óptimo encontrado. Por último se entrena la máquina y se prueba con los respectivos datos. Vale recalcar que al archivo de pruebas se les pasan las etiquetas con el objetivo de generar un informe estadístico como el ejemplo de la figura 4.3 en el que se encuentran cinco datos. El primero es una matriz de confusión donde se exponen la cantidad de muestras que se clasificaron correctamente y las que no dependiendo de la etiqueta ingresada, por ejemplo, en la figura 4.3 de las 28 muestras ingresadas, 14 eran -1 y acertó en 13, mientra que del resto que eran 1 acertó en 12 y se equivocó en 2. El siguiente dato (sucess rate) es la taza de exito, es decir el porcentaje de muestras clasificadas correctamente. Por otra parte, El tercer dato (balanced success rate) es la tasa de éxito equilibrada la cual es similar al anterior con la diferencia que éste tiene en cuenta el tamaño de cada clase, siendo esto útil para conjuntos de datos no balanceados, en el caso del ejemplo es muy similar a el valor anterior debido a que el numero de muestras eran iguales. El cuarto y el quinto dato tiene que ver con el área bajo la curva ROC, lo cual es la representación de la razón o ratio de verdaderos positivos frente a la razón o ratio de falsos positivos.

```
Confusion Matrix:
      Given labels:
        -1   1
   -1   13   1
    1   2    12
success rate: 0.892857
balanced success rate: 0.894872
area under ROC curve: 0.979487
area under ROC 50 curve: 0.979487
done.
```

Figura 4.3: Resultado de la clasificación

Algoritmo 4.1: Seudocodigo SVM

```
100 def SWM_FUNCTION(ensayos,etiquetas,caracteristicasDWT):
101     CrearDocumentoCVS de entrenamiento.
102     CrearDocumentoCVS de prueba.
103     Creacion SVM con parametro C = 450
104     Entramiento de maquina
105     Prueba de maquina
106     Almacenamiento de datos
107     Impresión de resultados
```

4.3. Envío de señales cerebrales continuas en tiempo

Como se puede observar en la figura 3.2 el envío de señales está compuesta por tres elementos: emisor, receptor y entrenador. A continuación se explicaran cada uno de ellos.

4.3.1. Entrenador

El bloque de entrenamiento a su vez está compuesto por un conjunto de bloques como se ve en la Figura4.4. Este conjunto de bloques tiene como función principal separar los datos de entrenamiento y aplicarles distintas técnicas para generar archivos con las métricas de clasificación.

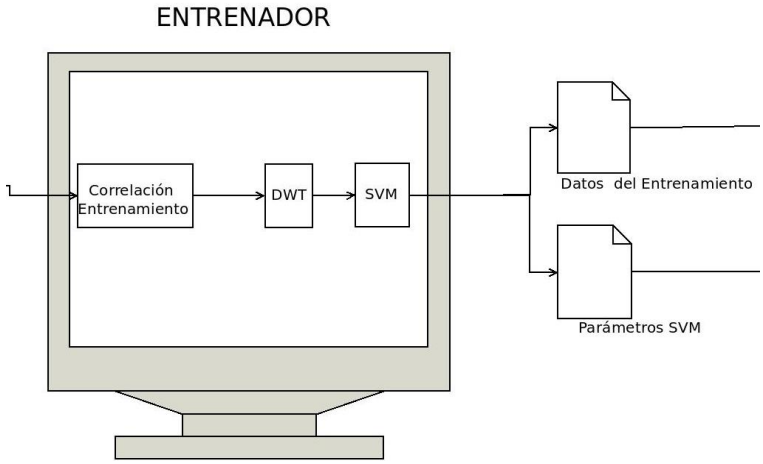

Figura 4.4: Bloque de entrenamiento

El proceso se realiza en un computador ya que posee mejores recursos que el sistema embebido mejorando el tiempo requerido para generar dichas métricas. Dicho proceso se ejecuta solo una vez teniendo como resultado la máquina de vectores de soporte lista para clasificar. A continuación se explicará cada uno de los bloques.

Correlación entrenamiento: Su comportamiento es similar a la correlación del proyecto inicial 4.2, se diferencia en que solo retorna los datos de entrenamiento; sin embargo, en este bloque se implementó una función adicional. Debido a que las señales almacenadas en la base de datos contienen en sus primeros tres segundos un estímulo acústico explicado en el bloque BD Señales 4.2 y como el objetivo es simular que la persona está imaginando el movimiento de una de sus manos por un rango de tiempo mayor es necesario eliminar dicha señalización. Por esta razón, se implementó una función que cortara las señales en un rango de tiempo determinado, la cual se puede ver en el Algoritmo 4.2

Algoritmo 4.2: cortarVector

```
1 def cortarVector(tiempoI,tiempoF,eeg):
2     muestraI=128*tiempoI
3     muestraF=128*tiempoF
4     Retorna eeg de muestraI hasta muestraf
5     con todos los canales y ensayos
```

La función recibe como parámetros: `tiempoI` tiempo inicial del rango de tiempo, `tiempof` tiempo limite de la señal, `eeg` matriz con muestras, canales y ensayos. En las líneas **2** y **3** se trasforman los tiempos al número de muestra, donde 128 es la frecuencia de muestreo de la señal. La función retorna la matriz reducida en número de muestras recortando el tiempo de la señal para todos sus canales y todos sus ensayos.

DWT: Realiza una descomposición de la señal por cada canal aplicando diferentes filtros, la descomposición utilizada es de orden 3, con wavelet madre "daubechie10"tal como en la tesis “Clasificación de Señales Cerebrales Relacionadas con la Imaginación de Movimientos para Aplicaciones de BCI"[3]; esta descomposición genera una señal de aproximación **a3** en el rango de frecuencias de 0-8 Hz, y tres señales de detalle **d1** 32-64 Hz, **d2** 16-32 Hz, **d3** 8-16 Hz. Como el rango de frecuencias de interés es de 8-13 Hz por las ondas **ALPHA** y 18-25 Hz por las ondas **BETA** como lo determinaron en el proyecto de grado anterior [3], solo se tendrán en cuenta las señales **d3** y **d2** a las cuales se les encontrará su potencia promedio agregandolos al vector de características. Del mismo modo, se agrega el valor porcentual de la energía de la señal de aproximación, quedando tres características por cada canal. Para este desarrollo se implementaron dos funciones: dwt como se puede ver en el Algoritmo 4.3 y energiaWT que se puede observar en el Algoritmo4.4 que calcula el porcentaje de energía de la señal de aproximación.

Algoritmo 4.3: Función DWT

```
1  def dwt(senal,canalesSelecionados,canales):
2      Para i Hasta canales:
3          Si es un canal seleccionado:
4              Realizar la descomposición DWT de senal
5              cálculo de la potencia promedio de d2
6              cálculo de la potencia promedio de d3
7              cálculo de porcentual de energía de a3
8
9      Retorna el valor de potencia promedio de d2 ,d3
10      y la energía  de a3
```

En el Algoritmo 4.3 se muestra la función dwt que recibe como parámetros la matriz con la señales `senal`, el vector de canales seleccionados `canalesSelecionados` y el número de canales `canales`. La función retorna los valores de la potencia promedio y el valor porcentual de la energía de **a3**. Se descompone la señal mediante la función pywt.wavedec donde se le indica el nivel de descomposición y la wavelet madre, en este caso se usó nivel 3 y wavelet db10(daubechie10).Posteriormente se calcula la potencia promedio para los coeficientes 1 y 2, los cuales representan la señales **d3** y **d2** mediante la formula 4.1, tomada de [3]. Después se calcula el valor porcentual de la energía de la señal con la función energiaWT, estas acciones se repiten para cada uno de los canales seleccionados. Finalmente se retornan la lista con el vector de características.

$$P_x = \frac{1}{L}\sum_{i=1}^{l}(x_i)^2 = \frac{1}{L}(norm(x))^2 \tag{4.1}$$

La función energiaWT recibe como parámetros todos los coeficientes de la señales descompuestas

Algoritmo 4.4: Funcion energiaWT

```
1 def energiaWT(cof,longitud):
2     cálculo de la energía total
3     cálculo de la energía de a3
4     valorPocentual=energía de a3/energía total
5     Retorna valorPocentual
```

cof y el tamaño de la señal de aproximación **a3**. En primera instancia se calcula la energía en todo el intervalo de la señal mediante la formula 4.2, donde N_1 es la muestra inicial y N_2 es la muestra final. Posteriormente se calcula la energía de la señal de aproximación para calcular el valor porcentual entre la energía de la señal **a3** y la energía total. Por último, se retorna este valor porcentual.

$$E[x[n]]_{N_1 \to N_2} = \sum_{n=N_1}^{N_2} |x[n]|^2 \tag{4.2}$$

Entrenador SVM: El entrenador SVM se crea con la librería explicada en la sección 4.2, con la diferencia que en el envío de señales cerebrales continuas en tiempo, éste se encuentra dividido en el entrenamiento y las pruebas, para el entrenamiento se crea el archivo cvs de forma similar como se realizó en el proyecto inicial con una variación, la cual consiste en que luego que se ha entrenado la máquina por medio del comando `s.train()` el resultado se almacenará en memoria por medio del comando `s.save()` lo cual generará un archivo que junto a el .cvs de los datos de entrenamiento será enviado por única vez a la pandaboard para realizar todas las pruebas.

4.3.2. Emisor

El emisor está dividido por una serie de bloques como se puede ver en la Figura 4.5. Estos bloques en conjunto tienen la función de extraer los datos de prueba de la base de datos y con estos formar una señal continua en tiempo de tres canales, dichas señales serán enviadas monotónicamente por un socket TCP/IP junto con su etiqueta correspondiente (izquierda o derecha) transmitiéndose aproximadamente cada 7,813 ms, tiempo de trasmisión usado por algunos EEG.

Correlación de prueba: El funcionamiento es similar al bloque correlación entrenamiento 4.3.1, la diferencia radica en que ésta retorna solo los datos de prueba.

Reducción de dimensiones: Tiene la función de concatenar cada uno de los ensayos y así formar una señal continua para cada canal reduciendo las dimensiones de la matriz de entrada como se muestra en la Figura 4.6. Esto se realizó con el fin de emular una situación con un usuario real donde el EEG continuamente está enviando información y no se encuentra segmentada como lo está en base de datos original.

En la Figura 4.6 **C3**, **CZ** y **C4** son los diferentes canales que tiene la base de datos, **NE** número de ensayos, **NM** número de muestras. Después de ser organizada, la señal tiene una longitud de $NM * NE$, donde el número de muestras y el número de ensayos es determinado por el bloque de correlación prueba. Lo anterior se implementó en el algoritmo mediante el método append de las listas, el cual permite agregar un nuevo elemento a la lista.

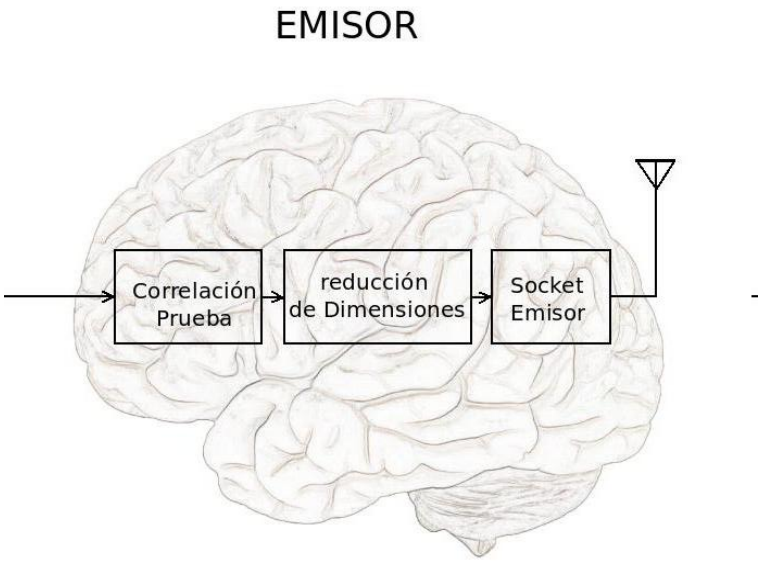

Figura 4.5: Bloque de emisor

Socket emisor: Un socket permite la comunicación alámbrica o inalámbrica entre dos máquinas o dos procesos. Se utilizó un socket TCP/IP para trasmitir la señales de los diferentes canales de una forma monotónica, transmitiendo cuatro números cada 7,813 ms. Estos 4 números están compuestos por las respectivas muestras de los tres canales y el número restante es la etiqueta del ensayo que a su vez es enviado al receptor para generar el índice porcentual de clasificación. El código implementado se puede ver en el Algoritmo 4.5.

Algoritmo 4.5: Soket Emisor

```
1  crear un nuevo objeto socket
2  vincular el socket a la máquina
3  capturar los datos de el  receptor
4  Para i Hasta  número de pruebas:
5      Para j hasta número de ensayos:
6
7          dato=concatenación de los valores de los 3 canales con la etiqueta
8          enviar datos por el socket
9          tiempo de espera de 7,813 ms
10
11 enviar "fin" por  el socket
12 cerrar conexión con el receptor
```

La primera acción que realiza el algoritmo es crear un objeto tipo socket, por defecto se crea de flujo y de la familia AF_INET, enviando la información en orden y realizando conexiones con otros dispositivos correspondientemente. El siguiente paso es enlazar el socket a la máquina con el método `bind`, recibe como parámetros la dirección y el puerto del socket. Luego de esto se espera el establecimiento de la conexión con otro dispositivo, una vez se establece retorna un objeto tipo socket que hace refe-

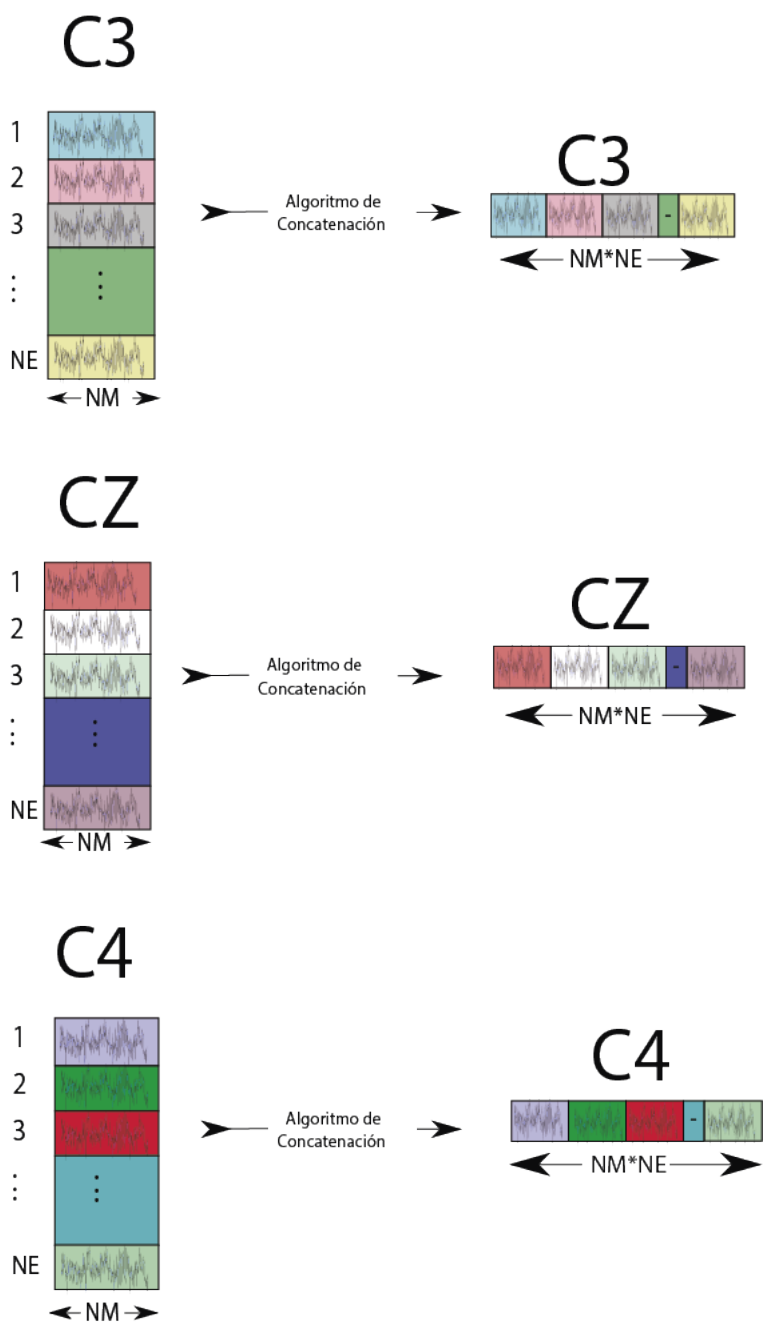

Figura 4.6: Concatenación de los ensayos

rencia al cliente y una tupla con la dirección y el puerto de la conexión. Para enviar los datos de cada canal es necesario dos ciclos anidados, uno recorriendo las muestras y otro recorriendo las pruebas,en estos ciclos se concatena la etiqueta de la respectiva prueba con los 3 canales y estos cuatro números se convierten a tipo string, porque por un socket solo se pueden enviar cadenas de texto. Esta cadena se almacena en la variable `dato` para luego ser enviada por el método `send`. La línea **9** genera un tiempo de espera de 7,813 ms para simular la taza de envío de algunos EEG. Para finalizar la recepción de datos se envía la cadena `"fin"` desde el emisor, y por último se cierra la conexión con el cliente y el

socket emisor.

4.3.3. Receptor

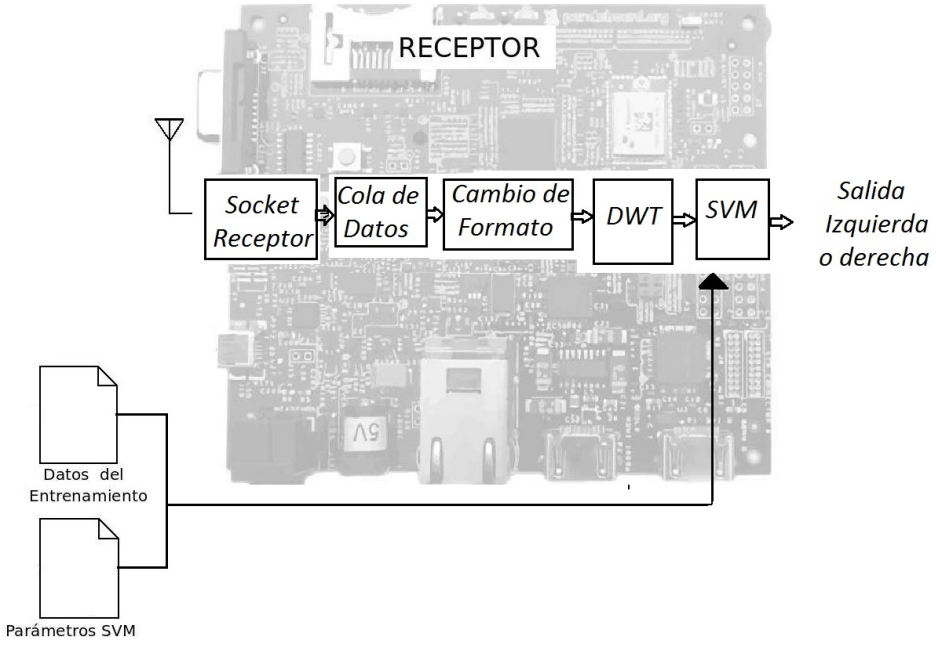

Figura 4.7: Bloque de recepción

Socket receptor: Tiene la función de recibir las cadenas de texto provenientes del emisor y guardarlas en una lista para posteriormente ser procesada por otros algoritmos. Se recibe una cadena de texto y el problema con este tipo de datos es que los algoritmos de DWT y SVM no los reconocen, por esto es necesario cambiar el tipo de datos lo cual se realiza en el bloque cambio de formato 4.3.3 que convierte las cadenas de texto en arreglos. Las instrucciones implementadas se pueden ver en el Algoritmo 4.6.

Algoritmo 4.6: Soket Receptor

```
1 crear un nuevo objeto socket
2 conexión con el Emisor
3 capturar datos
4 agregar  los datos a una lista
```

Primero se crea el objeto socket, en la siguiente línea se realiza la conexión con el emisor indicándole la dirección y el puerto correspondendientes con los datos ingresados en el enlace del bloque socket Emisor 4.5. Se reciben los datos mediante el método `recv` al cual se de debe indicar el número máximo de bytes a recibir, por último queda guardar la cadena de texto en una lista.

Cola de datos: Al ser un envío continuo en tiempo es necesario almacenar los datos en memoria para luego procesarlos, para este desarrollo se decide crear una cola de datos como se puede observar en la figura 4.8.

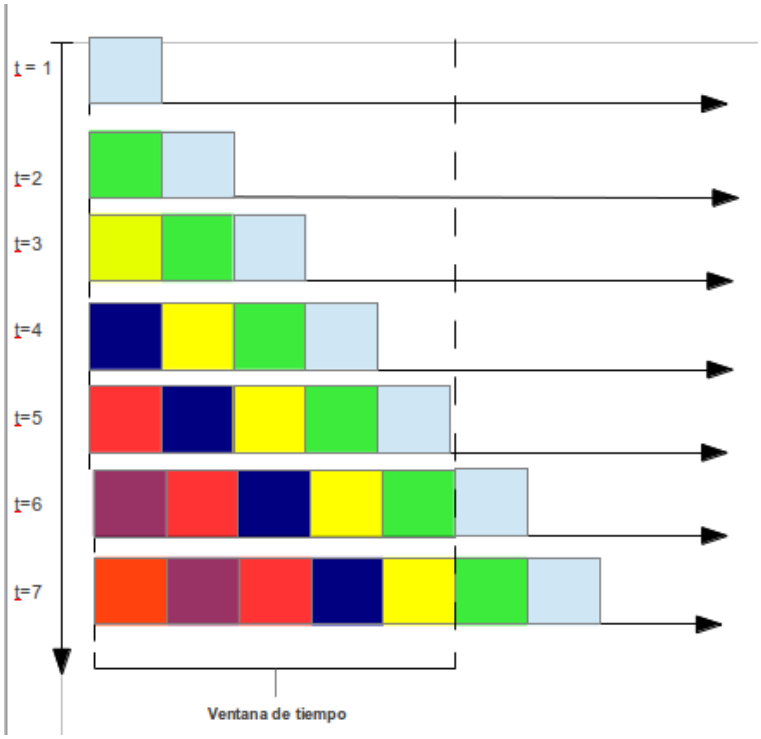

Figura 4.8: Cola de datos

La cola es un tipo de almacenamiento en el cual los primeros datos que ingresaron son los primeros en ser eliminados, al analiza la figura 4.8 se ven dos partes importantes: la primera de ella es el llenado de la ventana temporal por primera vez lo cual ocurre en los cinco segundos iniciales del ejemplo (tamaño de la ventana igual a 5). La segunda parte importante ocurre para t=6 y t=7 donde se visualiza el desplazamiento de los datos hacia la derecha, eliminando el primer dato que ingresó y agregando un dato nuevo al comienzo de la ventana.

En el caso del algoritmo de recepción de datos del proyecto se tienen estas dos fases. Para el llenado se tiene la variable `segundosVentana` en la cual se especifica el tiempo en segundos que tendrá la ventana, para el calculo de numero de muestras dicha variable se multiplica por 128.

Al estar llena la ventana por primera vez, las muestras son enviadas a un proceso computacional donde se realiza la extracción de características y clasificación. Vale recalcar que en ningún momento se deja de recibir datos, el proceso de la cola sigue trabajando debido a la implementación multiproceso que se realizó. Existe otra variable importante llamada `bufer`, la cual es la cantidad de muestras que se corre la ventana para enviar nuevamente un grupo de datos a clasificación, con lo dicho anteriormente se puede relacionar dicha variable con el tiempo mínimo de respuesta del sistema.

El protocolo por el cual se envían los datos es TCP/IP con el siguiente formato por muestra "[’Sensor1’,’Sensor2’,’Sensor3’, etiqueta]", al momento de realizar pruebas se encontró con el problema que en algunos casos se unían dos datos debido a que el protocolo no permite perdida de información y al perder sincronismo se envian los datos juntos, por tal razón se diseñó un pequeño código que analiza la cadena recibida y en caso de encontrar datos pegados los divide almacenándolos por separado, todo el desarrollo de esto se realiza con manejo de strings.

Cambio de formato: Como los datos trasmitidos por el socket son cadenas de texto es necesario convertirlas a números para poder aplicar el algoritmo de extracción de características y el de clasificación; cambiar el tipo de los datos es la función de este bloque. El diagrama de flujo del código desarrollado se puede ver la Figura 4.9.

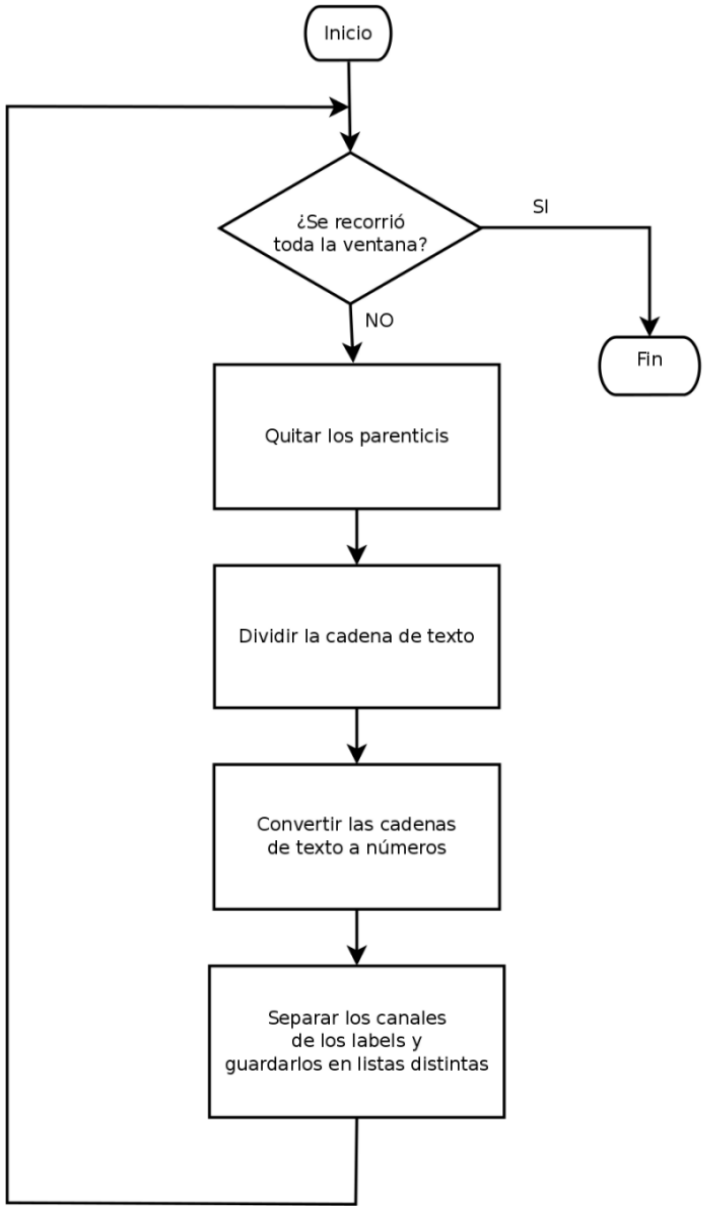

Figura 4.9: Diagrama de flujo de cambio de formato

El algoritmo diseñado cambia el formato de la cola de cadena de datos a un vector de números, para esto se debe recorrer la longitud total de la lista desarrollando un ciclo que recorra cada elemento. Buscando entender mejor el funcionamiento de este segmento es apropiado realizar un ejemplo donde los valores de la matriz `registroDatos` son '[-0.07861328 0.00488281 0.05224609 1.]' una cadena de caracteres. Las cuatro operaciones que realiza el algoritmo son:

1. Quitar los paréntesis al inicio y al final de la cadena quedando de esta forma '-0.07861328

0.00488281 0.05224609 1. ’.

2. Formar un vector de cadenas de texto [’-0.07861328’, ’0.00488281’, ’0.05224609’, ’1.’].
3. Convierte el vector de cadenas de texto en un vector de números [-0.07861328 0.00488281 0.05224609 1.]
4. separa los canales de las etiquetas y los guarda en listas distintas. Canales [-0.07861328 0.00488281 0.05224609]. Y etiquetas [1].

DWT Y SVM Prueba: El proceso de extracción de caracteristicas y SVM es muy parecido al que se explicó en la sección 4.2, con la diferencia que solo se realiza dicho proceso cuando las etiquetas de la ventana de tiempo que se va analizar son mayores al 80 % de algún tipo de las clases, es decir que en las transiciones entre derecha e izquierda no se realiza procesamiento puesto que para datos estadísticos no se podría concluir si la clasificación es errónea o acertada. Vale recordar que la máquina de vectores de soporte fue entrenada anteriormente y cargada con los datos necesarios para que en el momento de la recepción únicamente se realice el proceso de clasificación.

Multiproceso

Utilizando programación lineal es necesario clasificar una señal en un tiempo inferior a 7,813 ms lo cual es poco tiempo teniendo en cuenta que se deben trasformar los datos de cadenas de texto a arreglos de números, y después descomponer la señal en wavelets para encontrarle la energía y la potencia promedio a algunos de sus componentes, esto con el fin de encontrar el vector de características para finalmente clasificarlo en la SVM. Además que el sistema embebido no posee tantos recursos como un computador, por esta razón se hace necesario cambiar el paradigma de programación lineal a en paralelo a través de procesos.

El proceso principal va a ser el encargado de recibir y guardar las muestras trasmitidas por el emisor; por otra parte, el otro proceso se encargará de tratar la ventana de señal. Adicionalmente, con este tipo de programación se usa de una mejor forma los dos núcleos que tiene la PandaBoard.

Python posee librerías especializadas para la programación en paralelo [53]. Las rutinas implementadas se pueden ver en el Algoritmo 4.7.

Algoritmo 4.7: Multiproceso

```
1
2
3  bandera=0
4  Mientras que  bandera sea igual 0:
5      lanzar proceso de clasificación
6      agregar el proceso a una lista
7      capturar datos
8      Si datos es "fin":
9          salir
10     Sino :
11         guardar datos en lista
12     Para i Hasta numero de procesos:
13         Si  el proceso i NO esta activo:
14             terminar proceso i
```

Capítulo 5

Análisis de resultados

5.1. Resultados previos

Antes de analizar los resultados del proyecto es necesario enfocarse en el desarrollo realizado en la tesis "Clasificación de Señales Cerebrales Relacionadas con la Imaginación de Movimientos para Aplicaciones de BCI"[3]. El objetivo de dicha tesis era analizar diferentes combinaciones de técnicas de extracción y clasificadores. Los clasificadores ensayados fueron: análisis lineal discriminante (LDA), máquinas de vectores de soporte (SVM), y redes neuronales artificiales(ANN). Para cada uno de estos se prueban diferentes parámetros de entrada, en el caso de SVM se realizan pruebas con diferentes kernel (lineal, polinomial, y RBF), estas pruebas se ven reflejadas en el cuadro 5.1. El kernel que mejor resultado de clasificación obtuvo fue el lineal con un porcentaje del 92,86 % y por lo tanto se utilizará para los siguientes ensayos.

Kernel	**Parámetro**	**Porcentaje de Clasificación %**
Lineal		92,86
Polinomial	Orden=2	78,57
	Orden=3	64,29
	Orden=4	60,71
RBF	σ=0.1	46,43
	σ=1	82,14
	σ=10	85,71

Cuadro 5.1: Parametros SVM. Tomado de [3]

Luego de seleccionar los parámetros óptimos para cada clasificador se realizan diferentes combinaciones de técnicas de extracción, variando el parámetro w de la correlación cruzada, en los cuadros 5.2, 5.3, 5.4 se muestran los cinco mejores resultados para cada clasificador.

Al analizar los cuadros 5.2, 5.3, 5.4 con un porcentaje de clasificación promedio en MATLAB de 75.02 % se decide utilizar las técnicas DWT, PCA Y SVM para la migración del código a Python en el sistema embebido puesto que presentó el mejor resultado. Por último es importante realizar una comparación entre los tiempo de ejecución que consume cada algoritmo, para MATLAB [3] se realizará ingresando al código y tomando dichos tiempos para DWT, PCA, y clasificación SVM (los resultados se pueden observar en el cuadro 5.5 y se analizarán en la sección 5.2.2).

Características					Subconjuntos de Prueba %					
DWT	**CWT**	**CSP**	**Lyapunov**	**PCA**	**1**	**2**	**3**	**4**	**5**	**Promedio %**
X				X	71,43	64,29	92,86	75	71,43	75,002
X					71,43	60,71	92,86	75	71,43	74,286
X	X			X	82,14	64,29	78,57	71,43	71,43	73,572
X	X				78,57	64,29	78,57	71,43	75	73,527
X			X		67,86	60,71	92,86	71,43	71,43	72,858

Cuadro 5.2: Resultados SVM (MATLAB).Tomado de[3]

Características					Subconjuntos de Prueba %					
DWT	**CWT**	**CSP**	**Lyapunov**	**PCA**	**1**	**2**	**3**	**4**	**5**	**Promedio %**
X				X	71,43	57,14	92,86	75	67,86	72,858
X					71,43	57,14	92,86	75	67,86	72,858
X	X			X	82,14	64,29	78,57	71,43	67,86	72,858
X	X				82,14	64,29	78,57	71,43	67,86	72,858
X			X		64,29	60,71	92,86	75	67,86	72,144

Cuadro 5.3: Resultados LDA (MATLAB). Tomado de[3]

Características					Subconjuntos de Prueba %					
DWT	**CWT**	**CSP**	**Lyapunov**	**PCA**	**1**	**2**	**3**	**4**	**5**	**Promedio %**
X				X	71,43	63,39	92,68	73,04	67,86	73,68
X					70,36	63,57	92,86	73,39	67,86	73,608
X	X		X	X	73,57	62,86	77,5	77,32	70,89	72,428
X			X	X	65	58,93	92,68	74,64	68,21	71,892
X	X			X	75,54	63,04	77,5	70,71	71,43	71,644

Cuadro 5.4: Resultados ANN (MATLAB). Tomado de [3]

Tiempos	**W=1 [s]**	**W=2 [s]**	**W=3 [s]**	**w=4 [s]**	**w=5 [s]**	**Promedio [s]**
t DWT	0,31830	0,31430	0,31380	0,31040	0,31480	**0,31432**
t SVM prueba	0,00730	0,00740	0,00810	0,00730	0,00730	**0,00748**
t PCA	0,00750	0,00770	0,00790	0,00740	0,00750	**0,0076**
Total	**0,3331**	**0,3294**	**0,3298**	**0,3251**	**0,3296**	**0,3294**

Cuadro 5.5: Tiempos de ejecución en MATLAB

5.2. Clasificación SVM Python

Durante el desarrollo del proyecto se realizan diferentes pruebas, la primera de ellas está relacionada con la clasificación utilizando las técnicas que mejor resultado entregaron en MATLAB [3] (DWT,PCA Y SVM). Para comprender mejor los resultados se divide el análisis en dos partes: una parte porcentual donde se observan los resultados del clasificador y un análisis de tiempos de ejecución del algoritmo en el cual se contrastan los dos lenguajes (Matlab Vs. Python).

5.2.1. Análisis porcentual

Con la combinación DWT Y SVM se recrea la prueba realizada en Matlab pero ahora en el sistema embebido utilizando Python, dicha prueba es dividida en dos partes: la primera de ella utilizando la técnica PCA y la segunda sin ella. Los resultados de la clasificación se muestran en el cuadro 5.6, donde se observa que PCA no tiene ningún efecto en el resultado de cada una de las muestras, adicionalmente, el mejor porcentaje de clasificación (w = 3) igualó al entregado por el algoritmo realizado en Matlab.

PCA	**w=1 [s]**	**w=2 [s]**	**w=3 [s]**	**w=4 [s]**	**w=5 [s]**	**promedio [s]**
SI	71,43	64,29	92,86	67,86	67,86	72,86
NO	71,43	64,29	92,86	67,86	67,86	72,86

Cuadro 5.6: Resultados SVM (PYTHON)

Con el objetivo de tener una mejor visión del resultado obtenido en cada lenguaje de programación se realiza la gráfica 5.1 que contrasta Python vs. Matlab, obteniendo los datos de los cuadros 5.6,5.2. La gráfica consta de cuatro barras de colores, donde azul y verde corresponden a Python con y sin PCA respectivamente, mientras rojo y morado equivale a Matlab. En el grupo de muestras 1 y 3 los resultados fueron iguales en los dos lenguajes, en el grupo 2 con una diferencia de 3,58 % Python superó a Matlab sin utilizar PCA, mientras que en los grupos 4 y 5 Matlab superó a Python con porcentajes de 7,14 % y 3,57 %. El resultado promedio de clasificación lo supera Matlab por una diferencia mínima de 2,16 % con PCA y 1,46 % sin PCA. Debido a esta discrepancia se decide analizar paso a paso los resultados obtenidos en cada sección del código para así tener una conclusión precisa del porqué de esto, en la sección 5.3 se muestra el análisis.

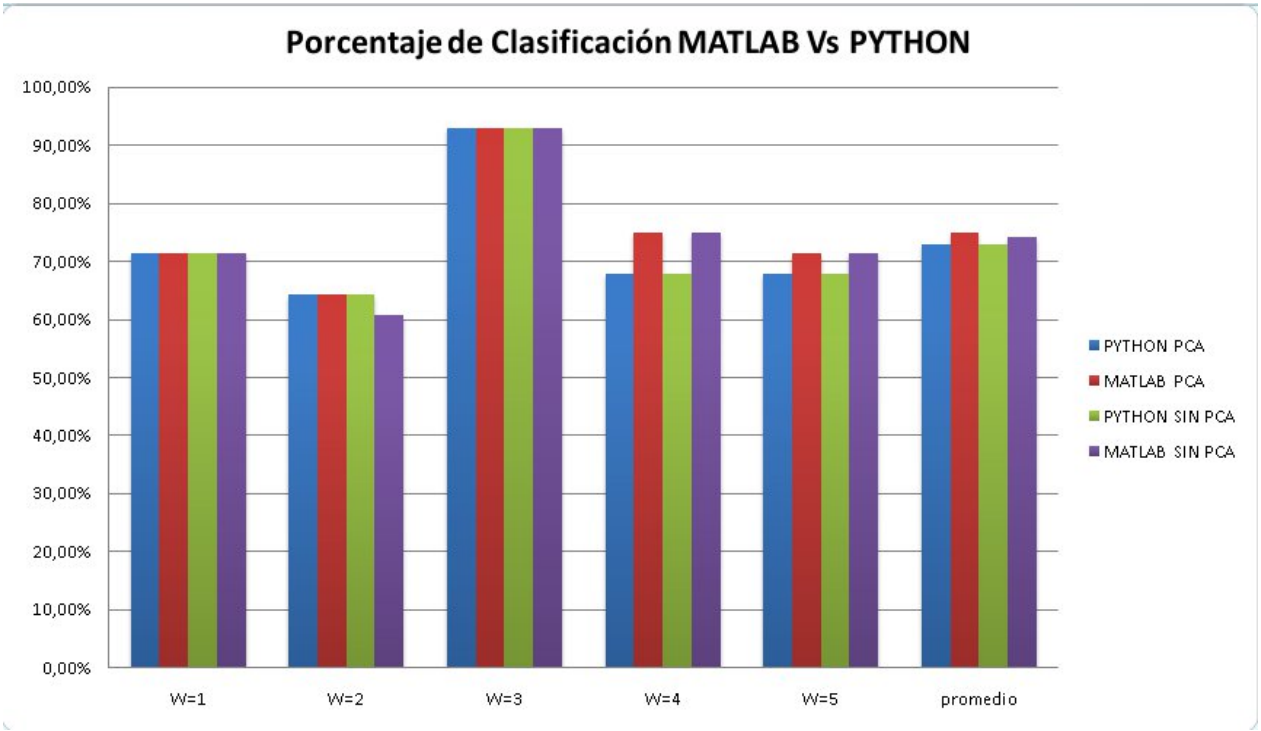

Figura 5.1: Porcentaje de Clasificacion SVM MATLAB Vs PYTHON

5.2.2. Comparación en tiempos de ejecución (Matlab Vs. Python)

Con el fin de realizar un análisis comparativo en tiempo de ejecución de un programa es necesario que las pruebas sean realizadas en una misma máquina, esto para que la velocidad de procesamiento no afecte la medición del tiempo, por tal razón se deciden realizar pruebas en Matlab y en Python en el computador de escritorio, puesto que no se tenía el desarrollo de Matlab para el sistema embebido y no era objeto de la tesis.

En el cuadro 5.7 se muestran los resultados de la medición en tiempo para DWT, PCA y prueba SVM. Las pruebas fueron realizadas para las cinco posibilidades de la correlación cruzada (w=1,2,3,4,5).

Medida	**W=1**	**W=2**	**W=3**	**w=4**	**w=5**	**Promedio**
t DWT	0,2858	0,2769	0,2757	0,2822	0,2805	**0,28022**
t SVM prueba	0,0004	0,0004	0,0004	0,0004	0,0004	**0,00037**
t PCA	0,0002	0,0002	0,0002	0,0002	0,0002	**0,00022**
Total	**0,286376**	**0,277487**	**0,276299**	**0,282774**	**0,281119**	**0,280811**

Cuadro 5.7: Tiempos de procesamiento en PYTHON

Para realizar una comparación entre el tiempo de ejecución en Matlab y Python se considera necesario realizar la gráfica 5.2 tomando los datos de los cuadros 5.5 y 5.7. En dicha gráfica se realiza un análisis porcentual de tiempo entre los dos lenguajes de programación, identificando con color azul el tiempo de ejecución en Matlab, y en rojo en Python. Para todos los procesos el tiempo de ejecución fue más corto en Python, sobresaliendo el tiempo de clasificación y el tiempo de PCA, con un porcentaje del 90 % en Matlab

contra algo menor al 5 % en Python. Aunque baja un poco la precisión, el sistema es más rápido.

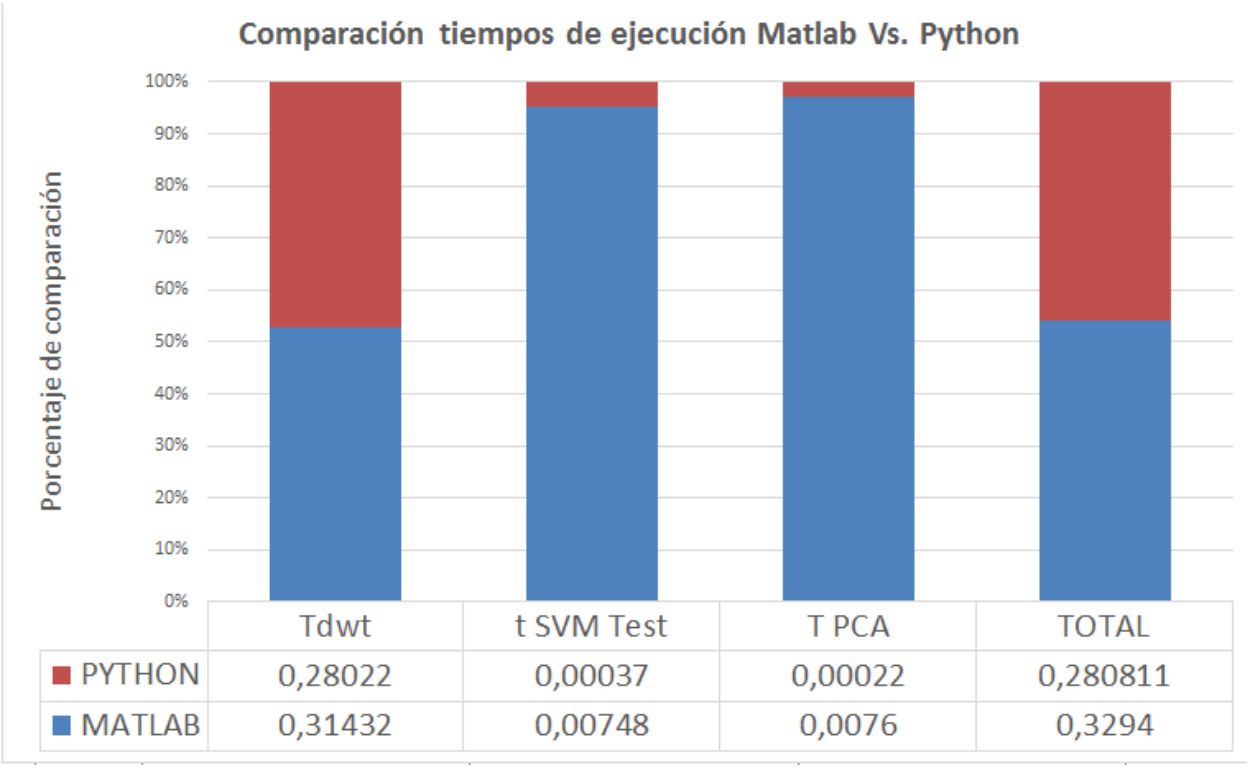

Figura 5.2: Comparación tiempos de ejecución MATLAB Vs PYTHON

5.3. Resultados paso a paso del algoritmo

Dado que en el porcentaje de clasificación se obtiene una leve diferencia entre los dos lenguajes de programación se decide realizar un análisis mas detallado del algoritmo de clasificación migrado de Matlab a Python, es decir, bloque a bloque se compararan las entradas y salidas del sistema. En la figura 5.3 se muestran las dimensiones esperadas en cada uno de los bloques que componen el algoritmo.

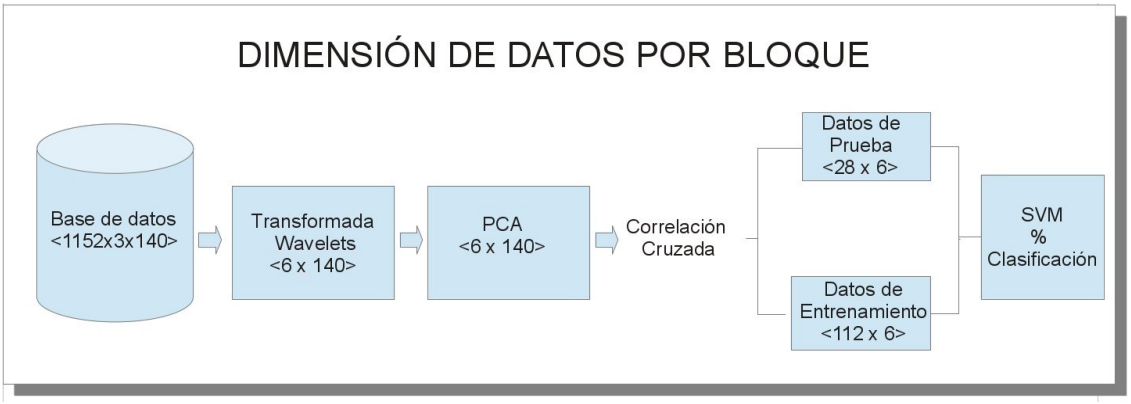

Figura 5.3: Dimensión de datos por bloque

Para empezar el análisis detallado se toma el primer ensayo de la base de datos, lo cual tendría unas dimensiones de <1152 muestras x 3 canales>, para el objetivo de la prueba se toman las 3 primeras y las 3 últimas muestras, en el cuadro 5.8 se muestran los datos.

No. Muestra	**MATLAB**			**PYTHON**		
	CH1	**CH2**	**CH3**	**CH1**	**CH2**	**CH3**
1	0,0103	0,0811	0,0625	0,0103	0,0811	0,0625
2	-0,0239	0,0762	0,0342	-0,0239	0,0762	0,0342
3	-0,0098	0,0776	0,0659	-0,0098	0,0776	0,0659
1150	0,1084	0,1313	0,0649	0,1084	0,1313	0,0649
1151	0,1733	0,1836	0,1045	0,1733	0,1836	0,1045
1152	0,2256	0,2456	0,1294	0,2256	0,2456	0,1294

Cuadro 5.8: Comparación base de datos en MATLAB Vs PYTHON

Como se observa en el cuadro 5.8, no existe diferencia alguna en los datos de entrada al sistema, el próximo bloque es la transformada discreta de wavelet donde por cada ensayo se obtienen 6 características. En el cuadro 5.9 se ven dichas características para el ensayo que se mostró en la base de datos.

DWT	**Carac.1**	**Carac.2**	**Carac.3**	**Carac.4**	**Carac.5**	**Carac.6**
MATLAB	0,01343	0,00153	55,28339	0,01041	0,00238	57,35147
PYTHON	0,01343	0,00153	55,28339	0,01041	0,00238	57,35147

Cuadro 5.9: Comparación DWT MATLAB Vs. PYTHON

Los datos a la salida del bloque DWT son iguales, el próximo paso en el algoritmo es el análisis de características principales (PCA), donde a la salida de este se obtienen 6 características por ensayo. Los resultados para el primer ensayo se analizan en el cuadro 5.10.

PCA	**Carac.1**	**Carac.2**	**Carac.3**	**Carac.4**	**Carac.5**	**Carac.6**
PYTHON	-78,9719	-10,4341	-0,1153	0,0090	0,0007	0,0025
MATLAB	-78,9719	-10,4341	-0,1153	0,0090	0,0007	0,0025

Cuadro 5.10: Comparación PCA MATLAB Vs. PYTHON

Luego de transformarse las características en el bloque PCA, se llega a la correlación cruzada donde se dividen las muestras que serán tomadas para prueba (28) y entrenamiento (112). Para realizar la comparación entre los dos lenguajes se toman las cinco primeras muestras de cada uno, los resultados obtenidos fueron exactamente los mismos.

El último paso del algoritmo es el clasificador SVM, donde se presenta la discrepancia en los datos. El resultado de clasificación para esta prueba fue: 71,43 % para Matlab, y 67,86 % en Python, la diferencia en dicho porcentaje se debe a que las librerías utilizadas para cada programa son diferentes y aunque baja un poco la precisión, el sistema es más rápido. Por otra parte se considera innecesario utilizar para el envío de señales cerebrales continuas en tiempo la técnica de extracción de características PCA, debido a que no realiza ninguna mejora en Python y contrario a esto alarga el tiempo de procesamiento por muestra.

5.4. Envío de señales cerebrales continuas en tiempo

Con el envío de señales cerebrales continuas en tiempo se tienen dos nuevas características: tiempo de ventana y tamaño del buffer. El tiempo de ventana está relacionado con la cantidad de muestras que se enviarán a la máquina de entrenamiento para ser procesadas, en el ejemplo de la figura 5.4 se muestra una ventana en tres momentos diferentes de tiempo, al pasar los segundos, la ventana se va desplazando hacia la derecha almacenando nuevos datos pero como se puede observar la cantidad de muestras se mantiene constante. También se puede apreciar que al momento de las transiciones se cuenta con un tiempo de buffer, el cual está ligado directamente a la respuesta del sistema puesto que es el tiempo que tarda el algoritmo en enviar datos al clasificador entre ventana y ventana. Teniendo en cuenta estas variables se decide realizar diferentes escenarios de pruebas como los que se explicarán a continuación.

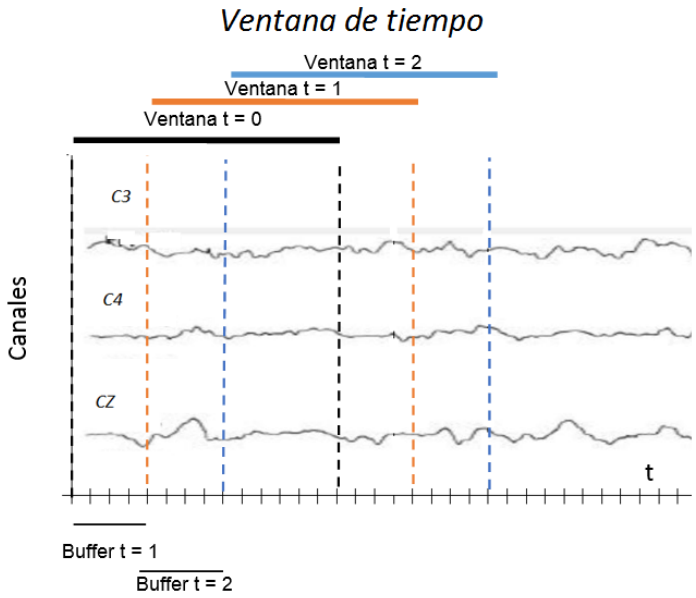

Figura 5.4: Ventana de tiempo y Tamaño del buffer

5.4.1. Tiempo de ventana por canal

El primer escenario que se creó fue dependiente de la variación de la ventana de tiempo, para todas las combinaciones posibles de los tres canales se varió dicha ventana obteniendo el porcentaje de clasificación, los tiempos de procesamiento y las muestras descartadas (muestras donde el porcentaje de etiquetas de la ventana eran menor al 80 %). En los cuadros 5.11, 5.12, 5.13, 5.14 se muestran los resultados obtenidos y al analizar los promedios se puede observar que el valor que más fluctúa es el porcentaje de éxito comparándolo con los demás. El cuadro 5.11 tiene el mayor porcentaje de éxito con un 81,06 %, por esta razón y concordando con la investigación realizada en la tesis “Clasificación de Señales Cerebrales Relacionadas con la Imaginación de Movimientos para Aplicaciones de BCI"[3] los canales que mejor resultado presentaron fueron C3 y C4 y la razón de esto es porqué están ubicados en el lóbulo central (lado derecho e izquierdo) donde estudios comprueban que se hallan los pensamientos acerca de movimientos. Analizando con más detalle los resultados de los cuadros se puede ver que al aumentar el tamaño de la ventana, se aumenta el porcentaje de

clasificación y el número de muestras descartadas, esto debido a que en el momento de las transiciones entre las dos clases es mayor el tiempo en que no se puede determinar la etiqueta correspondiente y el aumento del porcentaje es debido a la cantidad de muestras que se clasifican. A mayor cantidad, mayor precisión. Por esta razón y manteniendo una relación equivalente entre ganancia/perdida se selecciona el tamaño óptimo de ventana igual a 2 segundos, que presenta un porcentaje de éxito de 81,64 % . Como aclaración, todas las pruebas fueron realizadas con buffer igual a 40 muestras.

Tamaño de la ventana	Porcentaje de Éxito	Tiempo de clasificación [ms]	Tiempo de Captura[ms]	Total de Muestras	muestras Descartadas(80 %)
6 s	87,53 %	232,71	331,82	518	205
3s	82,97 %	242,39	327,72	528	105
2s	81,64 %	196,06	328,4	532	68
1s	78,28 %	148,81	328,61	535	32
0.75	74,90 %	134,71	328,56	535	25
Promedio	**81,06 %**	**190,936**	**329,022**	**529,6**	**87**

Cuadro 5.11: Tamaño de ventana para los canales C3 y C4

Tamaño de la ventana	Porcentaje de Éxito	Tiempo de clasificación [ms]	Tiempo de Captura[ms]	Total de Muestras	muestras Descartadas(80 %)
6 s	86,26 %	224,59	324,82	518	205
3s	82,03 %	250,89	320,46	528	105
2s	80,73 %	201,29	321,32	532	68
1s	77,09 %	154	321,01	534	32
0.75	75,29 %	142,18	321,07	536	25
Promedio	**80,28 %**	**194,59**	**321,736**	**529,6**	**87**

Cuadro 5.12: Tamaño de ventana para los canales C3, C4 y Cz

Tamaño de la ventana	Porcentaje de Éxito	Tiempo de clasificación [ms]	Tiempo de Captura[ms]	Total de Muestras	muestras Descartadas(80 %)
6 s	76,99 %	227,09	324,42	518	205
3s	77,30 %	250,02	320,22	528	105
2s	74,73 %	197,64	320,59	532	68
1s	70,51 %	151,86	321,01	534	32
0.75	72,88 %	138,22	320,89	535	25
Promedio	**74,48 %**	**192,966**	**321,426**	**529,4**	**87**

Cuadro 5.13: Tamaño de ventana para los canales C4 y Cz

Tamaño de la ventana	Porcentaje de Éxito	Tiempo de clasificación [ms]	Tiempo de Captura[ms]	Total de Muestras	muestras Descartadas(80 %)
6 s	82,31 %	229,26	322,12	518	205
3s	76,12 %	245,09	321,3	528	105
2s	73,00 %	197,5	320,62	532	68
1s	71,71 %	149,93	320,72	534	32
0.75	70,78 %	138,99	320,69	535	25
Promedio	**74,78 %**	**192,154**	**321,09**	**529,4**	**87**

Cuadro 5.14: Tamaño de ventana para los canales C3 y Cz

5.4.2. Tiempo de respuesta

En todo sistema, es importante conocer el tiempo de respuesta del mismo. Con un tiempo pequeño el sistema clasificará rápidamente las muestras pero en consecuencia se perderán algunas de estas debido a que el tiempo de captura es menor que el tamaño de procesamiento. Mientras que un tiempo de respuesta grande hará el sistema mas lento. Por esta razón, se realizó el Cuadro 5.15.

Tamaño Buffer	Tiempo de Respuesta[s]	Porcentaje de Éxito	Total de Muestras	Muestras Capturadas	Porcentaje de Muestras Perdidas
10	0,0781	81,51 %	2122	1607	24,27 %
20	0,1563	81,82 %	1063	1004	5,55 %
25	0,1953	80,81 %	850	846	0,47 %
30	0,2344	81,20 %	709	706	0,42 %
35	0,2734	80,34 %	608	608	0,00 %
40	0,3125	81,64 %	532	532	0,00 %
Promedio	**0,2083**	**81,22 %**	**980,667**	**883,833**	**5,12 %**

Cuadro 5.15: Tiempos de respuesta

Como se puede observar el porcentaje de éxito no varía en gran medida al tamaño del buffer. Se descarta el tamaño de 35 y de 40 muestras porque el tiempo de respuesta es alto comparándolo con el tiempo de reacción humano que es de alrededor de 215 ms [54]. Se ignoran los tamaños de 10 y 20 muestras porque se pierde un porcentaje mayor al 3 % de las mismas. La mejor opción entre los tamaños de 25 y 30 es 30 porque presenta un porcentaje de éxito mayor.

5.4.3. Estadísticas sobre los recursos

En esta sección se analizarán cómo son utilizados los recursos de la PandaBoard por el algoritmo desarrollado. Cabe recordar que el algoritmo lanza dos procesos, uno para la captura de datos y otro para la clasificación de estos. Se capturó el consumo con el comando ps con algunos parámetros, este retornó: uso porcentual del CPU, **`PID`** identificador único del proceso, **`PPID`** identificador del proceso padre, comando y **`PSR`** núcleo asignado al proceso. Como se puede ver en la Figura 5.5 el proceso con **`PID`** **10188** es el proceso padre (captura).

Graficando estos datos se obtuvieron las Figuras 5.6, 5.7, 5.8, 5.9, donde el eje de las abscisas es el tiempo en segundos en que se trasmiten todas las pruebas que aproximadamente es 178s.

```
%CPU   PID  PPID COMMAND                          %MEM PSR
10.3 10188  1537 python Receptor.py                1.7   0
4.0  11713 10188 python Receptor.py                1.3   1
```

Figura 5.5: Salida del comando ps

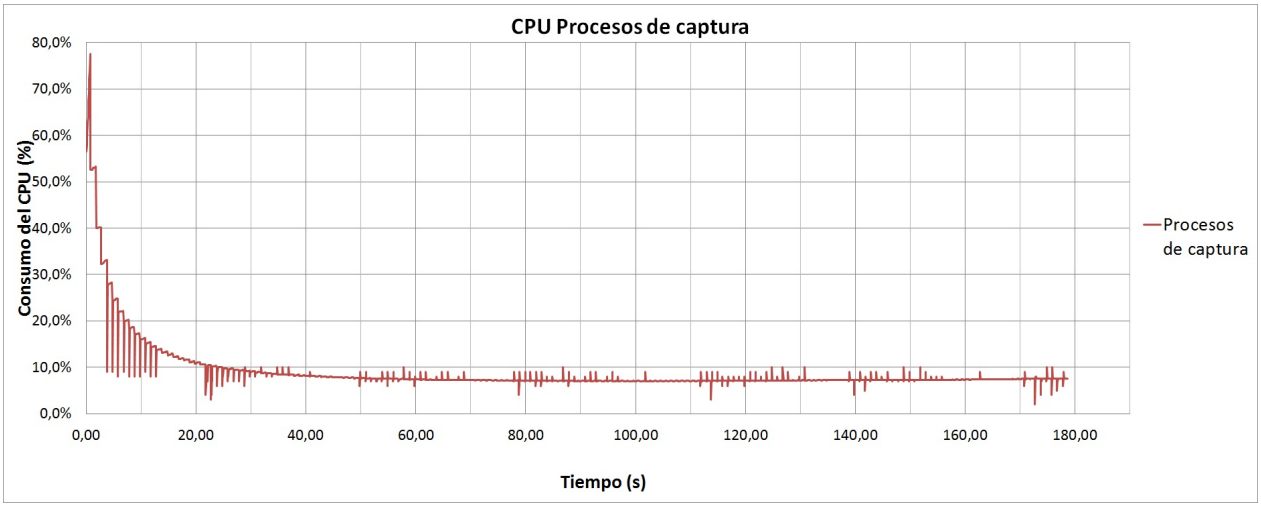

Figura 5.6: Consumo del CPU para el proceso de captura

Se puede ver en la figura 5.6 que la mayor carga para el CPU es en los primeros 20 segundos, esto debido a que en ese tiempo se instancia la SVM ya entrenada. Durante el resto de tiempo el proceso en promedio consume el 7.5 % del CPU.

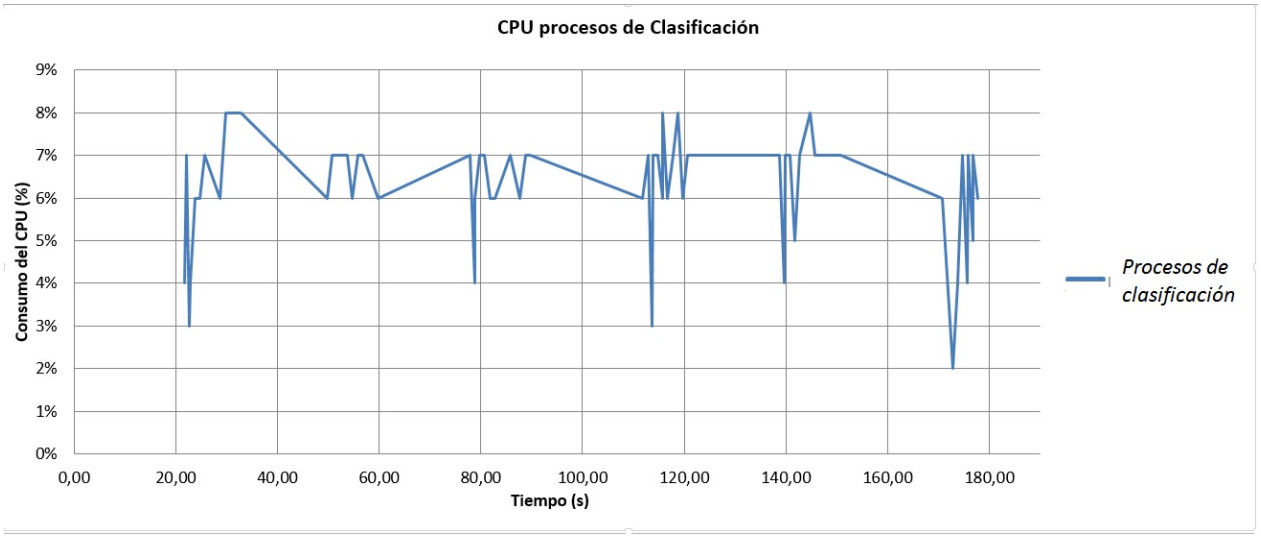

Figura 5.7: Consumo del CPU para los proceso de Clasificación

En la Figura 5.7 se muestra el consumo del proceso de clasificación. El proceso aparece aproximada-

mente a los 21 segundos de iniciarse el algoritmo debido a que primero el proceso padre tiene que inicializar la SVM y el socket del receptor. Éste no demanda mucho CPU, su uso máximo es de 8 % y en promedio utiliza el 6,3 % del procesador.

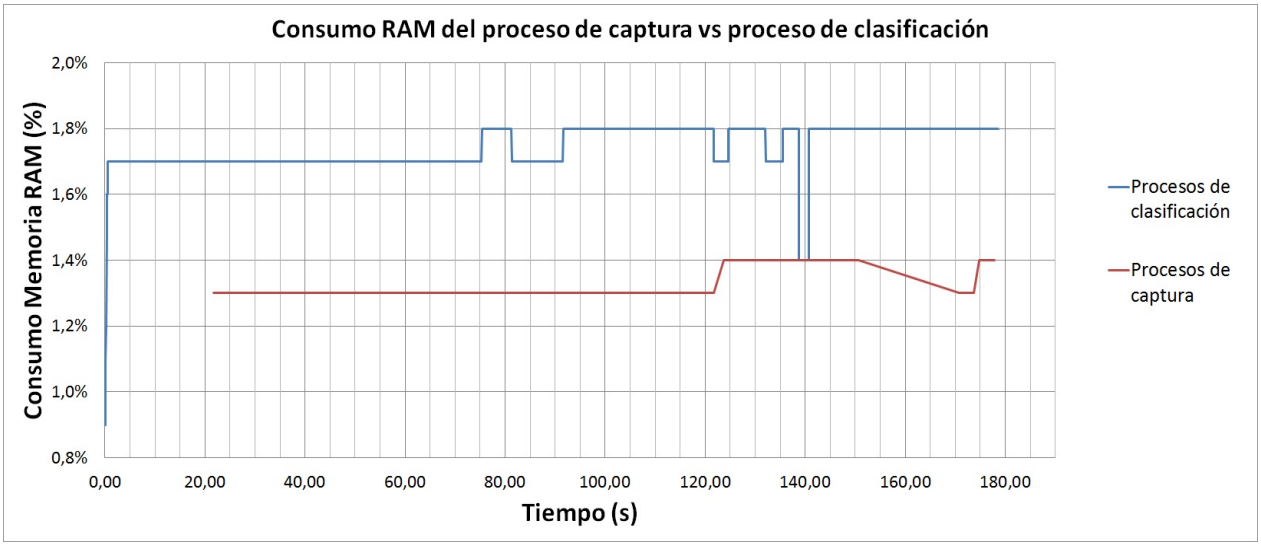

Figura 5.8: Consumo de memoria por cada uno de los procesos

Se puede observar en la Figura 5.8 el uso histórico de la memoria RAM de los dos procesos los cuales no demandan mucha memoria. Sumando los dos puntos críticos de los procesos se usa un 3,2 % de la memoria total que es de 1GB, es decir que los dos procesos consumen aproximadamente 32MB de memoria.

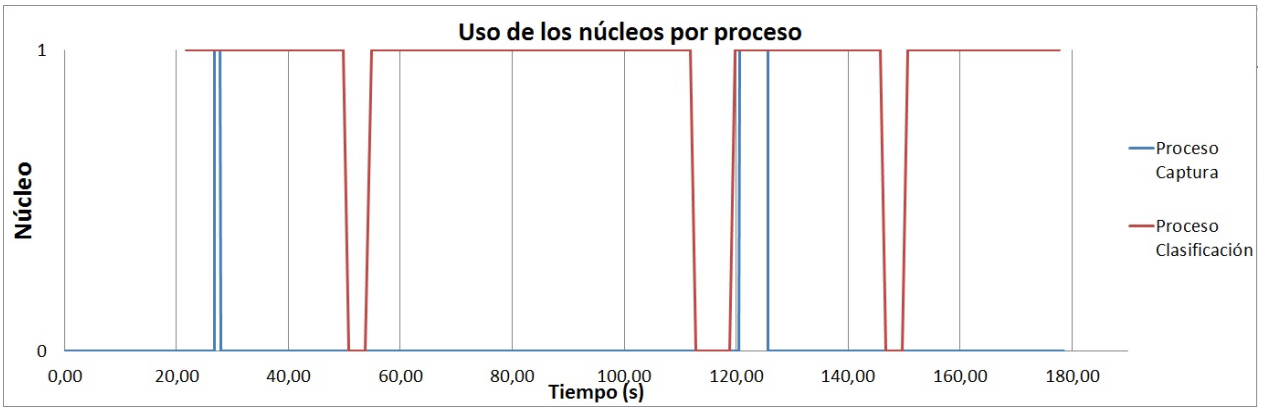

Figura 5.9: Núcleo asignado a cada proceso

En la Figura 5.9 se muestra la asignación de los procesos a cada núcleo durante el tiempo de ejecución del programa. Cada proceso actúa sobre un núcleo distinto la mayoría del tiempo; esta acción reduce en gran medida la sobrecarga del sistema.

Capítulo 6

Conclusiones

Al realizar una interfaz hombre-máquina (BCI) diseñada para procesar señales cerebrales relacionadas con la imaginación de movimientos provenientes de una base de datos, no se tiene control sobre variables como el ruido durante la adquisición, la precisión de los dispositivos de captura, las condiciones del experimento y la disposición del sujeto de prueba, razón por la cual los resultados obtenidos estuvieron sujetos a este tipo de inconveniente. Adicionalmente se contaba con un conjunto de datos limitados a 140 ensayos de los cuales 28 se destinaron para pruebas teniendo así una mínima tasa de error de 3.57 %, siendo esto una limitante al momento de clasificar las señales.

En concreto, la migración del clasificador obtuvo una mejora considerable de tiempo de ejecución con respeto al algoritmo desarrollado en Matlab, puesto que las librería de SVM en Python incorpora algoritmos escritos en c++, haciéndolos más rápidos, ejecutándose directamente en lenguaje de máquina. Sin embargo, este cambio de librería implica una reducción del 2.14

Por otra parte, el algoritmo de PCA no mejora el índice de clasificación y en cambio aumenta el tiempo de ejecución, esto debido al reducido tamaño del vector de características el cual se hace imposible optimizar. Por lo tanto, en futuras mejoras del clasificador no se debe utilizar esta técnica. Del mismo modo, se realizan pruebas modificando la dimensión de la ventana, con lo que se pudo concluir que existe una relación directamente proporcional entre tamaño y porcentaje de clasificación. A mayor cantidad, mayor precisión.

Es necesario realizar programación multiprocesos y contar con un procesador de doble núcleo para evitar perder datos al momento de la recepción, ya que el tiempo de clasificación (0.33 s) es mayor que el tiempo de muestreo de la señal cerebral (0.00781 s).

Haber seleccionado una plataforma de desarrollo de tipo single board computer (SBC) PandaBoard junto al sistema operativo GNU/linux permitió realizar un desarrollo ágil y eficiente, puesto que se cuenta con grandes recursos de drivers, módulos y librerías.

Entre los pasos a seguir para próximos desarrolladores, se tiene la incorporación del algoritmo realizado en este trabajo de grado con un dispositivo de captura EEG, modificando únicamente el tipo de conexión en la recepción de datos. También sería importante clasificar nuevos pensamientos cerebrales con el fin de poder darle mayor movilidad al producto final que se desea llegar con este avance, el cual es una silla de ruedas controlada por la mente en un sistema embebido.

Por ultimo, para la continuación del proyecto se debe optimizar el hardware seleccionado, puesto que al revisar el análisis de memoria RAM y de procesador, se llegó a la conclusión que podría reducirse a uno del 10 % de memoria y 20 % del procesador, como podría ser una Raspberry Pi.

Bibliografía

[1] La investigación arrancará con un presupuesto inicial de cien millones de dólares. www.abc.es/ciencia/20130402/abci-obama-anuncia-mayor-investigacion-201304021627.html. [Accesado 10-Junio-2013].

[2] Cuadriplejia. www.todo-en-salud.com/2010/06/cuadriplejia. [Accesado 16-Noviembre-2012].

[3] L. Salazar J. Zaccaro, J. Cardenas. Clasificación de señales cerebrales relacionadas con la imaginación de movimientos para aplicaciones de bci. tesis de pregrado, Pontificia Universidad Javeriana, 2011.

[4] K P. Aschenbrenner G. Herrnstadt C. Menon J. Webb, Z G. Xiao. Toaward a portable assistive arm exoskeleton for stroke patient rehabilitation controlled though a brain computer iterface. *The Fourth IEEE RAS/EMBS International Conference on Biomedical Robotics and Biomechatronics*, 2012.

[5] A. Vourvopoulos F. Liarokapis. Athanasios vourvopoulos fotis liarokapis. *IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications,Coventry University Coventry*, 2012.

[6] F. Liarokapis A. Vourvopoulos. Brain-controlled nxt robot: Tele-operating a robot through brain electrical activity. *Third International Conference on Games and Virtual Worlds for Serious Applications, Coventry*, 2011.

[7] Bci competition 2003. www.bbci.de/competition/ii. [Accesado 16-Noviembre-2012].

[8] Javier Minguez. Tecnología de interfaz cerebro-computador. Grupo de Robótica Universidad de Zaragoza España.

[9] G. Schalk, D.J. McFarland, T. Hinterberger, N. Birbaumer, and J.R. Wolpaw. Bci2000: a general-purpose brain-computer interface (bci) system. *Biomedical Engineering, IEEE Transactions on*, 51(6):1034–1043, 2004.

[10] Emotiv. http://www.emotiv.com/. [Accesado 26-Julio-2013].

[11] Mindwave. http://www.neurosky.com/Products/MindWave.aspx/. [Accesado 15-junio-2013].

[12] Bernhard Graimann. *BRAIN-COMPUTER ITERFACES*. 2011.

[13] El sistema cortical del lenguaje en zurdos y diestros. http://escuelaconcerebro.wordpress.com/2012/05/15/el-sistema-cortical-del-lenguaje-en-zurdos-y-diestros/. [Accesado 27-Junio-2013].

[14] Juan E. San martín. Transitorios. http://www.astormastering.com.ar/Clase_6_Transitorios.pdf. [Accesado 15-junio-2013].

[15] Julián Quiroga Sepúlveda. *Fundamentos de Señales y Sistemas*. 2011.

[16] Wavelets mas utilizadas. http://farm8.staticflickr.com/7127/7762208172_402b70438a.jpg. [Accesado 9-Feb-2012].

[17] Rafael Alberto González González. Algoritmo basado en wavelets aplicado a la detección de incendios forestales. http://catarina.udlap.mx/u_dl_a/tales/documentos/mel/gonzalez_g_ra/capitulo3.pdf, 2010. [Accesado 9-Feb-2012].

[18] V.N. Vapnik. *Static Learning Theory*. 1998.

[19] L. González Abril. Modelo de clasificación basados en máquinas de vectores de soporte. In *Asociación Científica Europea de Economía Aplicada*, 2003.

[20] Gustavo A. Betancourt. Las máquinas de soporte vectorial. *Scientia et technica Año IX*, Abril 2005.

[21] Drae- embeber. http://lema.rae.es/drae/?val=embeber. [Accesado 20-Julio-2013].

[22] Steve Heath. *Embedded Systems Design*. 2003.

[23] T. Wilmshurst. *An Introduction to the Design of Small Scale Embedded Systems with examples from PIC, 80C51 and 68HC05/08 Microcontrollers*. 2003.

[24] Simposio argentino de sistemas embebidos - sase2012. http://www.inti.gob.ar/noticias/slide_simposio.htm. [Accesado 20-Julio-2013].

[25] D. A. Perez. *Sistemas Embebidos y Sistemas Operativos Embebidos*. 2009.

[26] Andrew S.Tanenbaum. *Sistemas Operativos Modernos*. 2009.

[27] Pablo Ruiz Múzquiz. *Sistemas Operativos*. 2004.

[28] Gpl. http://www.gnu.org/licenses/gpl.html. [Accesado 9-Julio-2013].

[29] Ubuntu. http://www.ubuntu.com/. [Accesado 16-Julio-2013].

[30] Debian. http://www.debian.org/. [Accesado 16-Julio-2013].

[31] Gentoo. http://www.gentoo.org/. [Accesado 16-Julio-2013].

[32] Susana Marta Canel. Microcontroladores arm de 32 bits. 2002.

[33] Nicolas PONS. *LINUX Principios básicos del uso del sistema*. 2005.

[34] Christopher Yeoh Rusty Russell, Daniel Quinlan. Filesystem hierarchy standard. http://www.pathname.com/fhs/. [Accesado 9-Julio-2013].

[35] Panda board. http://pandaboard.org/. [Accesado 26-Julio-2013].

[36] Historia del software: el lenguaje python. http://bitelia.com/2011/12/lenguaje-python. [Accesado 26-Julio-2013].

[37] Python premiado como el mejor lenguaje de programación. http://www.gacetatecnologica.com/empresas/novedades/2044-phyton-premiado-como-el-mejor-lenguaje-de-programacion-.html. [Accesado 26-Julio-2013].

[38] Qnx. http://www.qnx.com. [Accesado 20-Julio-2013].

[39] Wind river vxworks. http://www.windriver.com/products/vxworks.html. [Accesado 20-Julio-2013].

[40] Linux-drivers. http://www.linux-drivers.org/. [Accesado 20-Julio-2013].

[41] Montavista. http://www.mvista.com/index.php. [Accesado 20-Julio-2013].

[42] Timesys. http://timesys.com/. [Accesado 20-Julio-2013].

[43] Embedded debian project. http://www.emdebian.org/. [Accesado 20-Julio-2013].

[44] Poky. http://www.pokylinux.org/. [Accesado 20-Julio-2013].

[45] Launchpad. https://qastaging.launchpad.net/. [Accesado 20-Julio-2013].

[46] Linux-real-time. https://rt.wiki.kernel.org/index.php/Main_Page. [Accesado 20-Julio-2013].

[47] Tatsuo Nakajima Ki-Duk Kwon, Midori Sugaya. Ktas: Analysis of timer latency for embedded linux kernel. *International International Journal of Advanced Science and Technology*, 2010.

[48] Adaptaciones a otras arquitecturas. http://www.debian.org/ports/. [Accesado 20-Julio-2013].

[49] Jeffrey Oldham Mark Mitchell and Alex Samuel. *Advanced Linux Programming*. 2001.

[50] *Multiprocessors and Linux*, lichota@mimuw.edu.pl.

[51] Ubuntu core. http://cdimage.ubuntu.com/ubuntu-core/releases/12.04/release/ubuntu-core-12.04.2-core-armhf.tar.gz. [Accesado 30-Julio-2013].

[52] David Bercovitz. *OMAP Ubuntu Core*. OMAPpedia, http://www.omappedia.com/wiki/OMAP_Ubuntu_Core. installacion de Ubuntu core para pandaboard.

[53] Multiprocessing. http://docs.python.org/2/library/multiprocessing.html#multiprocessing. [Accesado 26-Julio-2013].

[54] Reaction time statistics. http://www.humanbenchmark.com/tests/reactiontime/stats.php. [Accesado 26-Julio-2013].

[55] Linuxcounter. http://linuxcounter.net/. [Accesado 20-Julio-2013].

[56] OMAPpedia, http://omappedia.org/wiki/SD_Configuration. *SD Configuration*. configuracion de la SD.

Anexos

.1. Instalación de Ubuntu Core paso

.1.1. Requisitos

- Preparar una SD con dos particiones `rootfs` y `boot`:

  boot: Con sistema de archivos FAT32.

  rootfs: Con sistema de archivos EXT3.

  Cabe resaltar que la partición de `rootfs` debe ser mayor que `boot`. Se recomienda que la SD sea de al menos de 5GB. El tamaño de `boot` debe ser por lo menos de 15MB y el resto de la memoria se destina a `rootfs`.`boot` debe tener las opciones boot y lba activas. Para mayor información consultar en OMAPpedia [56]

- Una máquina local con SO GNU/Linux.
- Conexión serial ente la máquina local y la PandaBoard

.1.2. Descarga y configuración de la Distribución

Esta guía se realizó bajo una distribución basada en Debian [30]. Este proceso se realizara en su totalidad en la máquina local

1. Abrir una terminal.
2. Instalar algunos paquetes necesarios (Máquina local)

```
sudo apt-get install uboot-mkimage minicom
```

3. Crear una carpeta temporal para guardar todos los archivos necesarios. (Máquina local)

```
TMPDIR=$(mktemp -d)
```

4. Se entra a esta carpeta temporal.

```
cd $TMPDIR
```

5. Se crea un archivo llamado `boot.script` con el siguiente contenido

```
fatload mmc 0:1 0x80000000 uImage
setenv bootargs rw vram=32M fixrtc mem=1G@0x80000000 root=/dev/mmcblk0p2 console=ttyO2,115200n8 rootwait
bootm 0x80000000
```

6. Generar el archivo `boot.scr`.

```
mkimage -A arm -T script -C none -n "Boot Image" -d boot.script boot.scr
```

7. Se crean dos variables.

```
UBUNTU=http://cdimage.ubuntu.com/ubuntu-core/releases/12.04/release
BOOT=http://ports.ubuntu.com/ubuntu-ports/dists/precise/main/installer-armhf/current
```

8. Configuración de la partición `boot` y `rootfs`. Este procedimiento utilizara unos archivos, los cuales se pueden descargar desde Internet o utilizar los que se encuentran disponibles en el CD del proyecto.

   a) Internet

   1) Descargar las fuentes de Ubuntu 12.04.

   ```
   wget $UBUNTU/ubuntu-core-12.04.3-core-armhf.tar.gz
   ```

   NOTA:lLas fuentes están incluidos en el CD del proyecto.

   2) Bajar los archivos `MLO`, `u-boot`, `uImage`.

   ```
   wget -O MLO $BOOT/images/omap4/netboot/MLO
   wget -O u-boot.bin $BOOT/images/omap4/netboot/u-boot.bin
   wget -O uImage     $BOOT/images/omap4/netboot/uImage
   ```

   3) Copiar los archivos de boot en la partición `boot`.

   ```
   cp MLO u-boot.bin uImage boot.scr /media/boot
   ```

   4) Se descomprimen las fuentes de Ubuntu en la partición `rootfs`.

   ```
   cd /media/rootfs
   sudo tar --numeric-owner -xzvf $TMPDIR/ubuntu-core-12.04.3-core-armhf.tar.gz
   ```

   b) CD

   1) Se entra al la carpeta `UbuntuCore/boot` del CD.

   ```
   cd /media/cdrom/UbuntuCore/boot
   ```

   2) Copiar los archivos a la partición `boot`.

   ```
   cp MLO u-boot.bin uImage boot.scr /media/boot
   ```

   3) Se descomprimen las fuentes de Ubuntu en la partición `rootfs`.

   ```
   cd ../rootfs
   sudo tar --numeric-owner -xzvf ubuntu-core-12.04.3-core-armhf.tar.gz /media/rootfs
   ```

Es necesario crear algunos archivos de configuración para enlazar la consola de Ubuntu core con la consola-serial

9. Abrir la carpeta int.

```
cd /media/rootfs/etc/init
```

10. Crear un archivo llamado `serial-auto-detect-console.conf` con el siguiente contenido.

```
# serial-auto-detect-console - starts getty on serial console
#
# This service starts a getty on the serial port given in the 'console' kernel argument.
#

start on runlevel [23]
stop on runlevel [!23]

respawn

exec /bin/sh /bin/serialConsole
```

11. Entrar a la carpeta bin

```
cd ../../bin
```

crear un archivo llamado `serialConsole` con el posterior texto.

```
for arg in $(cat /proc/cmdline)
do
    case $arg in
        console=*)
            tty=${arg#console=}
            tty=${tty#/dev/}

            case $tty in
                tty[a-zA-Z]* )
                    PORT=${tty%%,*}

                    # check for service which do something on this port
                    if [ -f /etc/init/$PORT.conf ];then continue;fi

                    tmp=${tty##$PORT,}
                    SPEED=${tmp%%n*}
                    BITS=${tmp##${SPEED}n}

                    # 8bit serial is default
                    [ -z $BITS ] && BITS=8
                    [ 8 -eq $BITS ] && GETTY_ARGS="$GETTY_ARGS -8 "

                    [ -z $SPEED ] && SPEED='115200,57600,38400,19200,9600'

                    GETTY_ARGS="$GETTY_ARGS $SPEED $PORT"
                    exec /sbin/getty $GETTY_ARGS
            esac
    esac
done
```

Cambiar los permisos

```
sudo chmod +x+r serialConsole
```

12. Editar el archivo. `/media/rootfs/etc/network/interfaces` agregándole estas lineas.

```
auto eth0
iface eth0 inet dhcp
```

13. Editar el archivo `/media/rootfs/etc/resolv.conf` añadiéndole las siguientes lineas.

```
nameserver 8.8.8.8
nameserver 8.8.4.4
```

14. Eliminar el carácter‘*‘ que se encuentra entre dos ‘:‘ del usuario root en en archivo `/media/rootfs/etc/shadow`,qu de esta forma:

```
root::15937:0:99999:7:::
```

Cabe notar que no importa el valor de los números.

.1.3. Estableciendo la comunicación serial

La mayoría de computadoras actuales no poseen un puerto serial. Por tal motivo, es necesario tener una adaptador USB-serial. Para establecer la comunicación entre la máquina local y la Pandaborad se deben seguir los siguientes pasos en la máquina local:

1. Conectar el adaptador a la máquina local.
2. Buscar la TTY asociada al adaptador.

```
dmesg | grep tty
```

Este comando nos retornará una salida como se puede ver en la Figura 1
En este caso la TTY asociada es ttyUSB0.

Figura 1: Salida dmesg

3. Ejecutar minicom

```
sudo minicom -s
```

Aparecerá una pantalla como la de la figura 2.
Escoger la opción “configuración de la puerta serial ". Se desplegará una pantalla como la de la Figura 3 Alterar la configuración “A” con `/dev/<TTY asociada al adaptador>` (en este caso ttyUSB0). Finalmente, marcar salir en el menú principal de minicom.

4. Insertar la SD en la PandaBoard y conectar el adaptador USB-serial, suministro de energía, cable de red. En la consola de minicom podemos apreciar el arranque de la tarjeta.

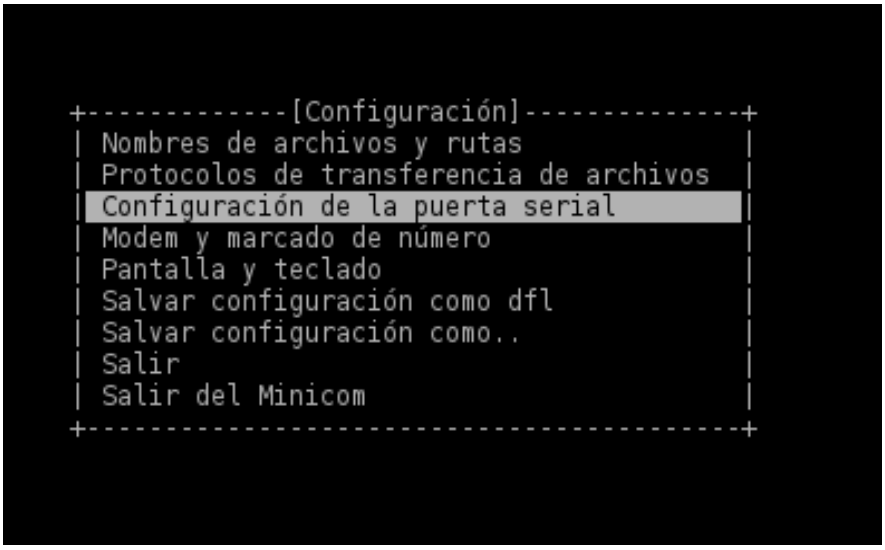

Figura 2: Pantalla principal de minicom

Figura 3: Pantalla de configuración de la puerta serial

.1.4. Instalación de librerías y herramientas necesarias

Una vez arranque el sistema sin ningún problema , se accede al sistema con el usuario root al cual le deshabilitamos la contraseña anteriormente. Para este proceso se accederá a la PandaBoard mediante la consola minicom.

1. Ejecutar el cliente DHCP con la orden

```
/usr/lib/klibc/bin/ipconfig eth0
```

2. Recargar la fuentes e instalar el editor nano.

```
apt-get update
apt-get install nano
```

3. Modificar los repositorios que se encuentran en `/etc/apt/sources.list`, cambiar su contenido por el que se muestra a continuación.

```
deb http://ports.ubuntu.com/ubuntu-ports/ precise main universe multiverse restricted
deb-src http://ports.ubuntu.com/ubuntu-ports/ precise main universe multiverse restricted
deb http://ports.ubuntu.com/ubuntu-ports/ precise-security main universe multiverse restricted
deb-src http://ports.ubuntu.com/ubuntu-ports/ precise-security main universe multiverse restricted
deb http://ports.ubuntu.com/ubuntu-ports/ precise-updates main universe multiverse restricted
deb-src http://ports.ubuntu.com/ubuntu-ports/ precise-updates main universe multiverse restricted


deb http://ppa.launchpad.net/tiomap-dev/release/ubuntu precise main
deb-src http://ppa.launchpad.net/tiomap-dev/release/ubuntu precise main
```

4. Recargar las fuentes,actualizar el sistema y instalar los controladores de la PandaBoard

```
apt-get update
apt-get dist-upgrade
apt-get install ubuntu-omap4-extras u-boot-tools --yes --force-yes
```

5. Instalar algunas herramientas que facilitaran el uso y la implementación de la aplicación. Servidor ssh para conectarnos por lan a la tarjeta y algunas utilidades de red.

```
apt-get install openssh-server net-tools wmifinfo moreutils isc-dhcp-client
```

6. Instalar algunas librerías y programas necesarios para la aplicación.

```
apt-get install python python-numpy python-pip python-scipy
apt-get install python-scitools python-symeig python-scientific
```

7. Mediante el gestor de paquetes de Python Instalamos la librería de wavelets (PyWavelets).

```
pip install PyWavelets
```

8. Descargar la librería de SVM para Python(PyML).

```
wget http://downloads.sourceforge.net/project/pyml/PyML-0.7.13.2.tar.gz
```

9. Descomprimir la librería PyML.

```
tar xvf PyML-0.7.13.2.tar.gz
```

10. Instalar PyML.

```
cd PyML-0.7.13.2
python setup.py build
python setup.py install
```

11. Con el gestor de paquetes pip instalamos la librería psutil.

```
pip install psutil
```

.2. Importar Ubuntu Core con todos los controladores y librerías

En esta sección se explica como importar todo el desarrollado realizado en el proyecto (sistema operativo, drivers, librerías y algoritmos) a una tarjeta SD vacía utilizando el archivo `UbuntuCore_PandaBoard.gz` incluido en el CD.

1. Ejecutar en una terminal el siguiente cogido.

```
gunzip -c /media/cdrom/UbuntuCore_PandaBoard.gz | dd of=/dev/sdx
```

donde `/dev/sdx` es la ruta de la SD

.3. Comando de ejecución del clasificador en python

El algoritmo `TesisBCI.py` se encuentra ubicado en la carpeta `codigoFuente/ProyectoInicial` del CD.

1. Ejecutar en una terminal el siguiente cogido.

```
python TesisBCI.py <PCA> <Canales seleccionados> <w>
```

Argumentos:

PCA: Es una argumento que puede tomar el valor de 0 para deshabilitar el algoritmo de PCA o 1 para habilitarlo.

Canales seleccionados: Este argumento determina los canales a usar(1 habilitado ,0 deshabilitado). Se debe indica el estado de los tres canales es decir que este argumento debe estar compuesto por tres dígitos.

W: Indica el grupo de correlación el cual debe estar entre 1 y 5.

En la figura 4 se puede ver un ejemplo de la ejecución de este algoritmo.

```
python TesisBCI.py 1 101 3
```

Figura 4: Ejemplo de ejecución clasificador Python

.4. Envío de señales cerebrales continuas en tiempo

La simulación consta de tres partes. El entrenador, que se debe ejecutar en la máquina local y luego enviar los archivos `maquina` y `train.csv` a a PandaBoard. El siguiente paso es ejecutar el Emisor en el PC y por ultimo el Receptor en la tarjeta.

.4.1. Entrenador

El algoritmo `Entrenador.py` se encuentra ubicado en la carpeta `codigoFuente/SimulacionTiempo/Entrenador` del CD.

1. Ejecutar en una terminal el siguiente cogido.

```
python Entrenador.py  <Canales seleccionados> <Tiempo de ventana>
```

Argumentos:

Canales seleccionados: Determina los canales a usar (1 habilitado ,0 deshabilitado). Se debe indicar el estado de los tres canales, es decir que el argumento debe estar compuesto por tres dígitos.

Tiempo de ventana: Tiempo en segundos de la ventana, se pueden utilizar los siguientes valores: 0.75, 1, 2, 3, 6. Debido a que la base de datos se divide en porciones, para calcular otro valor de ventana se debe confirmar que el residuo de la división $\frac{6}{TiempoVentana}$ sea nulo y además ser menor de 6.

En la figura 5 se puede ver un ejemplo de la ejecución de este algoritmo

```
python entrenador.py 101 1
```

Figura 5: Ejemplo de ejecución del entrenador

.4.2. Emisor

El algoritmo `Emisor.py` se encuentra ubicado en la carpeta `codigoFuente/SimulacionTiempo/Emisor` del CD.

1. Ejecutar en una terminal el siguiente cogido.

```
python Emisor.py  <Ip de la máquina> <puerto>
```

Argumentos:

Ip de la máquina: Suministrar la dirección IP de la máquina donde se ejecuta éste algoritmo.

Puerto: Puerto donde se van a transmitir los datos.

En la figura 6 se puede ver un ejemplo de la ejecución de este algoritmo

```
python Emisor.py 192.168.1.40 5014
```

Figura 6: Ejemplo de ejecución del emisor

.4.3. Receptor

El algoritmo `Receptor.py` se encuentra ubicado en la carpeta `codigoFuente/SimulacionTiempo/Receptor` del CD.

1. Ejecutar en una terminal el siguiente cogido.

```
python Receptor.py <Ip de la máquina> <Puerto> <TiempoVentana> <Buffer> <Canales seleccionados>
```

Argumentos:

Ip de la máquina: Suministrar la dirección IP de la máquina donde se ejecuta el emisor.

Puerto: Puerto donde el servidor está transmitiendo los datos.

Tiempo de ventana: Tiempo en segundos de la ventana, se pueden utilizar los siguientes valores: 0.75, 1, 2, 3, 6. Debido a que la base de datos se divide en porciones, para calcular otro valor de ventana se debe confirmar que el residuo de la división $\frac{6}{TiempoVentana}$ sea nulo y además ser menor de 6.

Buffer: Cantidad de muestras entre ventana y ventana para procesar los datos.

Canales seleccionados: Este argumento determina los canales a usar(1 habilitado ,0 deshabilitado). Se debe indica el estado de los tres canales, es decir que este argumento debe estar compuesto por tres dígitos.

En la figura 7 se puede ver un ejemplo de la ejecución de este algoritmo

```
python Receptor.py 192.168.1.10 4500 2 30 101
```

Figura 7: Ejemplo de ejecución del receptor

Nota: Este comando debe ser ejecutado con permisos de administrador.