Proyecto Final

Procesamiento de Bioseñales en Tiempo Real en Universos Interactivos

Instituto Tecnológico de Buenos Aires

Federico Tedin
Javier Fraire

Febrero 2017

Abstract

Biosignals are defined as signals produced by live tissues. Biosignals have been used in the fields of medicine and accessibility. These signals allow us to acquire information about the patient's status which can be used to run diagnostics, to assist those with accesibilty difficulties, among others. On the other hand, interactive worlds are visual simulations that respond to the user's input, and there has been a boom in the last decades due to the advancements of video games, virtual reality and augmented reality. Traditionally, the input mechanisms have been the mouse, keyboard and more recently, gamepads and accelerometers present in virtual reality headsets. This project attempts to answer wether it's possible to engineer an interaction mechanism based on biosignals for the development of interactive worlds. Specifically, we focused our efforts in biosignals that offer a balance between practicality, price and availability, processing time and the potential uses in interactive worlds. The techniques used to obtain biosignals in this project were: EEG, which measures brain bioelectrical activity, EMG, which measures electrical activity generated by the muscles, and SpO_2, which measures oxygen levels in the blood.

The data obtained was used as an input in various interactive worlds, in an effort to give the users a more immersive experience. To achieve this goal, an analysis of the signals' features, processing mechanisms (filtering, noise reduction), feature extraction and classification was carried out to obtain information about how they could be used in interactive worlds. It was concluded that the use of biosignals in these interactive worlds is not only possible, but could significantly improve the user's experience. Even then, there's still room for improvement, like increasing the classification process' precision, or finding different uses for the signals.

Resumen

Las bioseñales consisten en señales producidas por los tejidos vivos. Las bioseñales se han utilizado en la medicina y en el campo de la accesibilidad. Su uso permite adquirir informacion sobre el estado del paciente que puede ser utilizada para diagnósticos, para asistir a pacientes con dificultades de accesibilidad, entre otros. Por otro lado los universos interactivos son simulaciones visuales que reaccionan ante entradas del usuario, y han sufrido un auge en las últimas décadas debido al avance de los videojuegos, la realidad virtual y la realidad aumentada. Tradicionalmente los mecanismos de interacción han sido el *mouse*, el teclado y más recientemente *gamepads* y acelerómetros existentes en cascos de realidad virtual. Este trabajo intenta responder si es posible implementar un mecanismo de interacción basado en bioseñales para el desarrollo de universos interactivos. Particularmente para esta ponencia, nos hemos concentrado en las bioseñales que permiten un balance entre practicidad, precio y disponibilidad, tiempo de procesamiento, y el uso que se le puede dar dentro de los universos interactivos. Las técnicas utilizadas para obtener bioseñales en este trabajo han sido: Electroencefalografía (EEG), la cual consiste en la actividad bioeléctrica cerebral, Electromiografía (EMG), la cual contempla la actividad eléctrica generada por los músculos del cuerpo, y Pulsioximetría (SpO_2), la cual permite determinar la oxigenación en sangre.

La información obtenida de las bioseñales, fue utilizada como entrada alternativa en diversos universos interactivos, dándole posiblemente una mayor inmersión a los usuarios. Para lograr este propósito, se realizó un análisis de las características de las señales y de cuáles eran los mecanismos de preprocesamiento (filtrado, reducción del ruido), extracción de características y clasificación, para obtener indicadores que permitiesen interactuar con los universos interactivos. Finalmente, se concluyó que el uso de bioseñales en tiempo real en estos universos no solo es posible, si no que posiblemente mejoraron significativamente la experiencia del usuario. Aún así, existe lugar para mejoras, como aumentar la fidelidad de la clasificación de las señales, o encontrar distintas aplicaciones para las mismas.

Índice general

Capítulo 1

Introducción

Este trabajo tuvo como objetivo el procesamiento de bioseñales en tiempo real en universos interactivos. Una bioseñal puede ser definida como una descripción de un fenómeno fisiológico [12]. El término "tiempo real", aquí, se refiere a computación gráfica en tiempo real, es decir, la generación de imágenes lo suficientemente rápido como para crear la ilusión de movimiento. Los universos interactivos son simulaciones con componentes visuales que obtienen entradas del usuario y reaccionan antes las mismas. Las entradas más comunes son el *mouse* y el teclado. En este trabajo se investigó la utilización de bioseñales como entradas adicionales para posiblemente aumentar la inmersión.

Se han encontrado muy pocas aplicaciones de uso comercial de bioseñales en universos interactivos y su alcance es muy reducido [11]. Se buscó determinar si las señales obtenidas mediante EEG, EMG y SpO_2 eran aptas para la utilización en tiempo real en universos interactivos y cómo su utilización podría mejorar la experiencia de usuario, contando sólo con dispositivos que puedan ser adquiridos en el mercado, y no dispositivos profesionales de alto costo. Los dispositivos y señales fueron elegidos a partir de cuatro criterios que se detallan más adelante. El objetivo de este proyecto no fue ofrecer un análisis cualitativo y cuantitativo de los diferentes métodos de procesamiento de señales, sino cómo utilizar los existentes para posiblemente mejorar la inmersión en universos interactivos teniendo en cuenta los desafíos que esto implica. Este trabajo tampoco buscó determinar si mejoraba o no la inmersión, sino que se limitó a verificar si era posible encontrar un esquema donde se utilizaran bioseñales en tiempo real en universos interactivos.

Capítulo 2

Distintas Aplicaciones de Bioseñales

En este capítulo se presentará el estado del arte en lo que respecta a los distintos usos de las bioseñales. Se brindarán algunos ejemplos del uso de las bioseñales en distintos campos.

2.1. ACAT

La computadora utilizada por Stephen Hawking es tal vez el caso más conocido de la utilización de bioseñales en accesibilidad. Stephen Hawking cuenta con esclerosis lateral amiotrófica, por lo que se encuentra paralizado y no puede hablar. Para poder comunicarse, Intel desarrolló un sistema compuesto por una tableta y un sensor infrarojo montado sobre sus anteojos. El sensor infrarojo detecta el movimiento en su mejilla izquierda. La tableta cuenta con una plataforma de código abierto llamada ACAT. ACAT provee un teclado virtual en la pantalla. Utilizando el movimiento de su mejilla, Hawking, puede detener el cursor donde desea y así, escribir. Es decir, es una entrada binaria. Este también utiliza un procesador de texto con predicción de palabras que permite acelerar el proceso. Luego, el sistema utiliza un sintetizador de voz para comunicar lo que escribió. Esta es tan solo una de las aplicaciones de ACAT. ACAT también le permite controlar el ratón en *Windows*, y así, controlar completamente la computadora para poder utilizar su correo electrónico, navegar por internet, entre otras cosas. ACAT puede utilizar como entrada cualquier bioseñal. [7].

2.2. Gestos Como Dispositivos de Entrada

La NASA desarrolló un sistema para controlar un avión en una simulación utilizando los movimientos musculares medidos con sensores EMG. Colocaron diversos sensores sobre una manga de tela. Con ellos, adquirieron la señal y la filtraron y eliminaron el ruido. Luego extrajeron las características y reconocieron patrones en una fase de entrenamiento. Con esta información, se aplicaron patrones de reconocimiento en una simulación interactiva. Lograron controlar un avión de guerra sin utilizar una palanca de mando. Es decir, el usuario co-

locaba la mano como si estuviese utilizando una palanca de mando y realizaba movimientos para controlar el avión (ver figura 2.1) [13].

Figura 2.1: Un usuario utilizando el dispotivo EMG para controlar un avión en una simulación [13].

2.3. *Muscleman*

Dos académicos de la Universidad Nacional de Seúl, utilizaron un dispositivo EMG y un acelerómetro para controlar un vídeojuego. Utilizando el acelerómetro, el juego era capaz de determinar si el usuario estaba dando un simple puñetazo hacia adelante, un puñetazo de abajo hacia arriba o si estaba lanzando una bola de fuego (ver figura 2.2). Usando el sensor EMG, el juego medía la fuerza realizada por el usuario y la aplicaba proporcionalmente en el juego. Es decir, si el usuario realizaba poca fuerza, el ataque era débil. En cambio, si era fuerte, el ataque era fuerte. De esta forma, se utilizó como dispositivo de entrada las propias señales del cuerpo en lugar de usar un control de mando físico o el teclado [5].

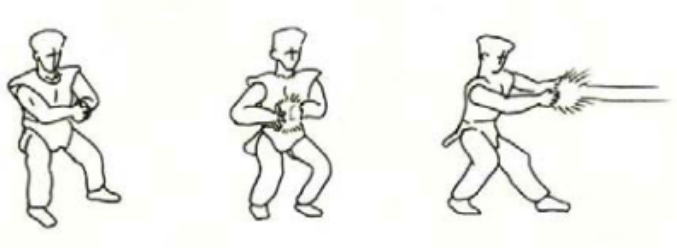

Figura 2.2: Movimiento que debe realiza un usuario para lanzar una bola de fuego en el videojuego *Muscleman* [5].

2.4. *Muse*

La empresa *Muse* desarrolló un dispositivo EEG con siete electrodos. El mismo viene acompañado con una aplicación móvil que ayuda a los usuarios a meditar. Cuando el usuario tiene la mente tranquila, se escucha un clima calmo, pero cuando el usuario está alterado se escucha un clima tormentoso. Muse utiliza distintas ondas cerebrales para detectar si el usuario se encuentra relajado o no.

2.5. *MindFlix*

Netflix desarrolló *MindFlix*. *Mindflix* utiliza un dispositivo EEG para controlar su popular servicio con la mente. Utiliza el giroscopio y el acelerómetro del dispositivo para permitirle al usuario desplazarse horizontalmente y verticalmente por la interfaz. Además, utiliza distintas ondas cerebrales para detectar cuando el usuario piensa en la palabra *play*. En caso de que el usuario piense en esa palabra, la aplicación comienza a reproducir el contenido seleccionado. Se intentó averiguar qué ondas cerebrales se utilizaban y de que forma, pero no se encontró en ningún lugar [9].

2.6. *Neurogaming*

Neurogaming es la utilización de BCI en videojuegos para mejorar la experiencia. El concepto surgió hace algunos años pero aún no se ha explorado mucho. Existe muy pocas aplicaciones comerciales de este tipo. Un ejemplo es *Throw Trucks With Your Mind* el cuál utiliza las ondas *Beta* del cerebro para lanzar camiones [11].

Capítulo 3

Marco Teórico

Dado que este proyecto se centrará en las bioseñales, resulta fundamental explicar los conceptos necesarios para su entendimiento. Primero se hablará sobre las bioseñales. Luego se detallará sobre el procesamiento de las señales (específicamente el procesamiento de bioseñales), desde su obtención hasta la extracción de características y clasificación, y su salida. Finalmente, se explicará como operan algunos sensores y se introducirá información teórica sobre la señal que se obtiene de los mismos.

3.1. Bioseñales

Como hay un gran número de mecanismos fisiológicos de interés, la cantidad de bioseñales es muy grande. Como se puede observar en la figura 3.1 las bioseñales puede ser clasificadas por lo menos por tres criterios distintos [12]:

- **Existencia:**
  - Permanentes: Existen sin necesidad de un impacto artificial, disparador o excitación externa al cuerpo y están disponibles todo el tiempo. Por ejemplo, la señal electrocardiográfica es inducida por la excitación eléctrica del corazón.
  - Inducidas: Son disparadas artificialmente, excitadas o inducidas externamente. Su duración está ligada a la duración de la excitación. Cuando el impacto artificial termina, la bioseñal se atenúa. Por ejemplo, la SpO_2 utiliza luz para detectar el oxígeno en sangre y la señal dura tanto tiempo como dicha luz se encuentre encendida.
- **Dinamismo:**
  - Casi estáticas: Exhiben pocos cambios a lo largo del tiempo. Un ejemplo de esta señal es la temperatura del cuerpo.
  - Dinámicas: Varían mucho a través del tiempo. Un ejemplo de está señal es la EKG ya que el latido del corazón cambia constantemente.
- **Origen:**

- Eléctricas: Estas incluyen señales como Electrocardiograma (EKG), que refleja la actividad eléctrica del corazón, EEG, que refleja la actividad eléctrica de las neuronas, EMG, que refleja la actividad eléctrica de los músculos, o *Galvanic Skin Response* (GSR), que refleja las variaciones de energía eléctrica en la piel.
- Magnéticas: Reflejan un campo magnético inducido. Por ejemplo un magnetocardiograma utiliza los campos magnéticos emitidos por las corrientes eléctricas generadas por la excitación del corazón.
- Mecánicas: Incluyen deformaciones del cuerpo o vibraciones en la piel. Por ejemplo un mecanorespirograma utiliza los movimientos del abdomen para medir el ritmo cardíaco.
- Ópticas: Utilizan la absorción de la luz. Se utiliza una luz artificial. Un ejemplo es la SpO_2.
- Acústicas: Utiliza diversos sonidos del cuerpo como por ejemplo sonidos cardíacos.
- Químicas: Reflejan la composición química y los cambios temporales en los elementos sólidos, líquidos y gaseosos del cuerpo.
- Térmicas: Utilizan la absorción y pérdida de calor en el cuerpo.
- Otras.

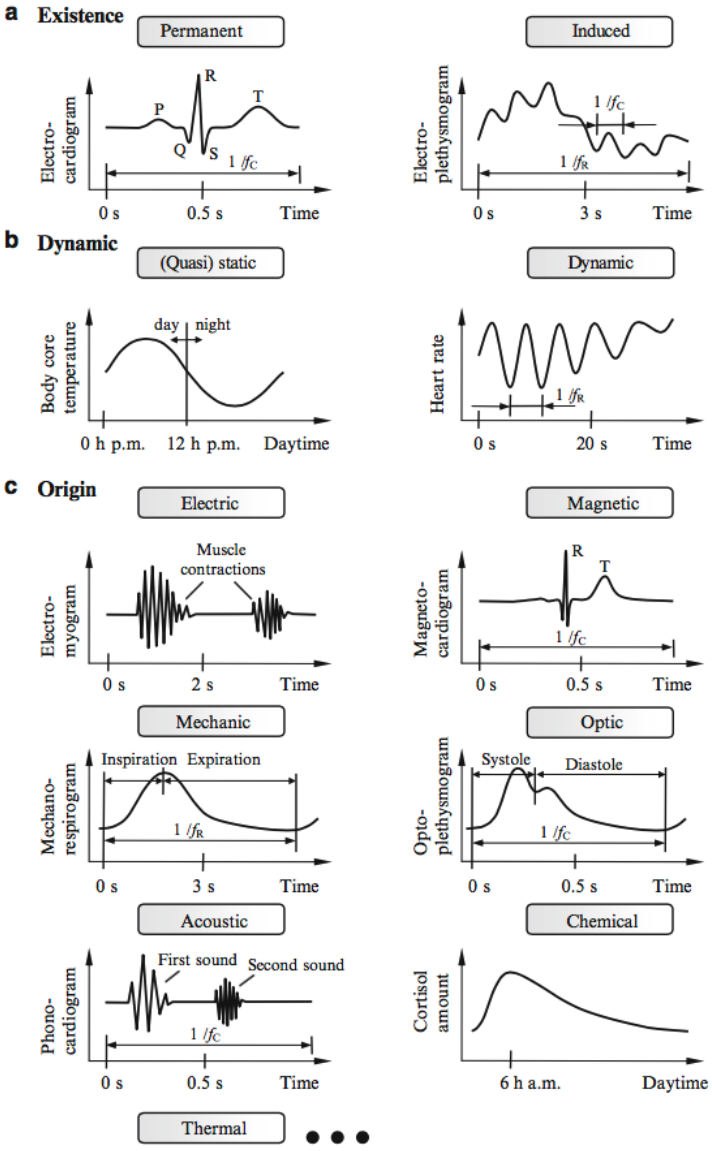

Figura 3.1: Las posibles clasificaciones de las bioseñales de acuerdo a su **(a)** existencia, **(b)** dinamismo y **(c)** origen [12].

3.2. Obtención y Procesamiento de Señales

En la figura 3.2 se pueden observar las distintas etapas por las que pasa una bioseñal al ser procesada: obtención de la señal fisiológica de un sensor, utilización de un transductor, procesamiento analógico, digitalización, transmisión, procesamiento de la señal digital, extracción de características y clasificación, y salida del clasificador.

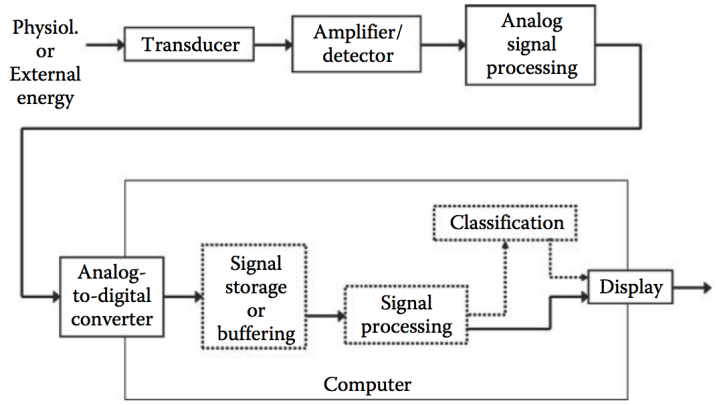

Figura 3.2: Representación esquemática de las etapas del procesamiento de las bioseñales [22].

3.2.1. Obtención de la Señal Fisiológica

Esta es la primer etapa en el procesamiento de bioseñales. Se debe obtener la señal de algún sensor. Puede ser un electrodo, un fotodetector, u otros. La señal viene en forma de energía. Dicha energía puede ser generada por el proceso en si (por ejemplo la actividad eléctrica producida por los músuclos), puede ser energía indirectamente relacionada al proceso o puede ser producida por una fuente externa [22].

3.2.2. Utilización de un Transductor

Un transductor es un dispositivo que convierte energía de un tipo a otro. El propósito aquí es transferir información, no transformar energía. Se convierte la energía a energía eléctrica. Por lo general, la salida de un transductor de un biosensor es voltaje. En la figura 3.3 se puede observar una representación de un transductor. El trandsuctor es el elemento más crítico en el sistema ya que es la interfaz entre el sujeto y el resto del sistema. Este determina que tan invasivo es un dispositivo. Luego de pasar por el transductor, se utiliza un amplificador o detector [22].

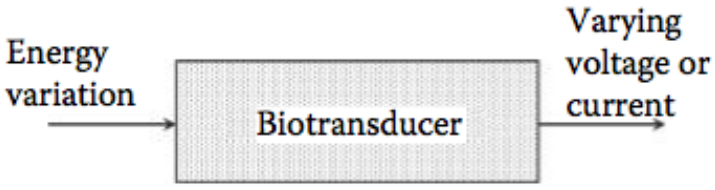

Figura 3.3: Representación de un transductor utilizado en biosensores [22].

3.2.3. Procesamiento de la Señal Analógica

La mayoría de los sensores procesan la señal analógica obtenida del transductor. El procesamiento más común es restringir el rango de frecuencias o ancho de banda utilizando filtros analógicos. Los filtros analógicos son dispositivos electrónicos que remueven un determinado rango de frecuencias. El filtrado es usualmente donde se establece el ancho de banda de todo el sistema. La ganancia de un filtro es la relación entre la amplitud de la señal de salida y la amplitud de la señal de entrada y se define como:

$$\text{Ganancia}(f) = \frac{\text{Valores de salida}(f)}{\text{Valores de entrada}(f)}$$

Los filtros más comunes pueden observarse en al figura 3.4 y son:

- **Pasa bajos:** Atenúa las frecuencias superiores a un determinado valor.
- **Pasa altos:** Atenúa las frecuencias inferiores a un determinado valor.
- **Pasa bandas:** Atenúa las frecuencias fuera de un determinado intervalo.
- **Supresión de bandas:** Atenúa las frecuencias dentro de un determinado intervalo [22].

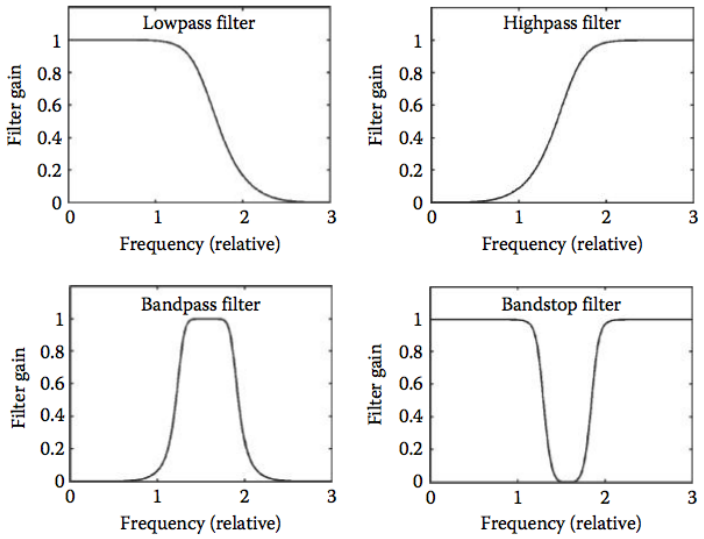

Figura 3.4: Influencia en la frecuencia de los filros básicos [22].

3.2.4. Digitalización

La mayor parte de las bioseñales se presentan en forma analógica. Por este motivo, deben ser digitalizadas para poder ser utilizadas por una computadora. Este proceso generalmente ocurre en un dispositivo de captura. Un componente electrónico convierte un voltaje analógico en una representación digital equivalente. Una onda continua ($x(t)$) se convierte en una onda discreta ($x[t]$), una

función de números reales que son definidos como enteros en puntos discretos en el tiempo. Estos números se conocen como muestras y los puntos discretos en el tiempo son usualmente tomados en intervalos regulares (T), también conocida como frecuencia de muestreo:

$$f_s = \frac{1}{T_s} Hz$$

Entonces, la señal analógica es segmentada en varias partes para que pueda caber en la memoria de una computadora. Este concepto se conoce como ventana [22]. En la figura 3.5 se puede observar el resultado de digitalizar una señal continua.

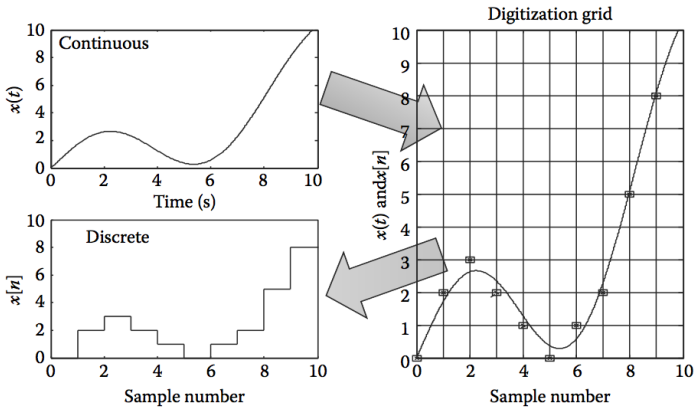

Figura 3.5: Digitalizar una señal continua, esquina superior izquierda, requiere partir la señal en tiempo y amplitud, lado derecho. El resultado es una serie de números disgretos que se aproximan a la señal, esquina inferior izquierda. [22].

Digitalizar una señal implica segmentar la amplitud y el tiempo. Si bien resulta evidente que la señal digitalizada obtenida no es igual a la analógica, antes ciertas restricciones matemáticas la información contenida digitalmente permitirían regenerar con tanta aproximación como se desee la señal original, balanceando la información en el campo frecuencial y temporal. El grado de aproximación de la señal digital a la analógica depende de la segmentación realizada. Como se mencionó, se segmentan las siguientes dos componentes:

- **Amplitud:** También se conoce como cuantificación. El grado de aproximación en este caso depende de la cantidad de valores distintos que son utilizados para representar la señal, es decir, la cantidad de bits utilizados. El nivel de cuantificación (q) se define como el tamaño del segmento de amplitud . El efecto de la cuantificación es la adición de ruido y este es determinado por q. En la figura 3.6 se puede observar el ruido añadido por la cuantificación. Por lo general los conversores usan 8, 12 y 16 bits de salida. Pues la mayoría de las señales, no cuentan con una relación señal-ruido lo suficientemente elevada como para utilizar una resolución mayor.

- **Tiempo:** Se refiere al muestreo. El criterio de *Nyquist* establece que la frecuencia máxima (frecuencia de *Nyquist*) obtenida es la mitad que la frecuencia de muestreo. El teorema de *Shannon-Nyquist* establece que para poder recuperar la señal analógica a partir de la señal digital, se debe muestrear al doble que la máxima frecuencia contenida en la señal original, es decir, $f_s > 2\,f_{max}$. Utilizar frecuencias de muestro más altas es innecesario e implica tener que guardar más información y tener que procesarla a una velocidad mayor . Aún así, en la práctica es muy común que se utilicen frecuencias de muestreo de tres hasta cinco veces f_{max}. De esta forma se incrementa el espacio entre frecuencias y se facilita la tarea de eliminar frecuencias indeseadas [22]. Esto se debe a que se cuenta con una mayor granularidad al tratar con las frecuencias. Por ejemplo si se cuenta con una frecuencia de muestro de $128\,Hz$, se obtienen 64 valores al realizar la Transformada de *Fourier* (en realidad son 128 pero por simetría, solo se utiliza la mitad). Entonces, se cuenta con escalones de $2\,Hz$, es decir, si se quiere obtener la frecuencia $8Hz$, el valor en realidad representa $8\,Hz - 9\,Hz$. En cambio, si se utilizan $256\,Hz$, el escalón es de $1\,Hz$ lo que simplifica la obtención de determinadas frecuencias.

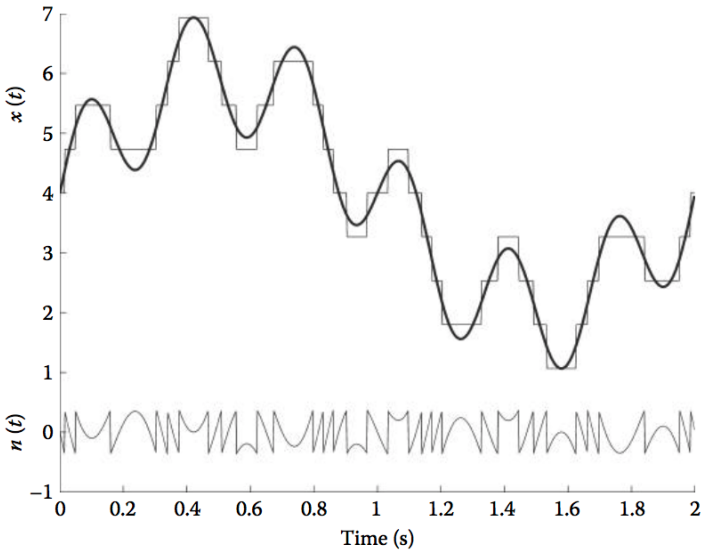

Figura 3.6: El efecto de la cuantificación de la señal original puede verse como ruido añadido [22].

3.2.5. Transmisión de la Señal

Luego de digitalizar la señal, el dispositvo puede transmitirla hacia alguna computadora. Algunos dispositivos la utilizan directamente y no necesitan transmitirla. La transmisión puede tomar muchas formas. La señal puede ser transmitida por puerto serie, por USB, por *Bluetooth*, por *Wi-Fi*, entre otros. El protocolo que se utilice para transmitir la señal depende del medio y es deci-

sión del fabricante cuál utilizar. Por ejemplo si es un puerto serie, se transfiere bit a bit. Luego, en la computadora debe haber un controlador que pueda recibir esta información transmitida.

3.2.6. Procesamiento de la Señal Digital

Como se mencionó anteriormente, la señal digital puede presentar ruido que se genera al digitalizarla. En algunos casos, hay que aplicar filtros para reducir el ruido. Existen diversos filtros. Uno de ellos es el filtro *Gaussiano*. Dicho filtro suaviza la señal por lo que elimina los picos que se pudieran haber introducido. Además, el filtro *Gaussiano*, a diferencia de otros filtros, no elimina las altas frecuencias completamente. El filtro *Gaussiano* se aplica haciendo una convolución de la señal con la siguiente función:

$$g(x) = \frac{1}{\sqrt{2 * \pi} * \sigma} * e^{-\frac{x^2}{2*\sigma^2}}$$

Muchas veces las señales pueden tener una cierta tendencia (trending). Para compensar esta tendencia, se utiliza una técnica conocida como *detrending*. Esta consiste en eliminar dicha tendencia. La forma más simple de realizar *detrending* es calcular la media del sensor y restarla a cada valor. Existen otros algoritmos más complejos de *detrending* y todo un amplio abanico de estudio de estas técnicas en relación al procesamiento de señales y bioseñales. Adicionalmente, las señales pueden encontrarse desfasadas. Las variaciones en la fase, el período y los ciclos de trabajo causan lo que se conoce como *jitter*. Se deben utilizar técnicas de corrección de fase y eliminación de *jitter* en caso de que esto ocurra [4].

Una vez que se redujo el ruido, se pueden aplicar otros filtros o continuar. Como se mencionó anteriormente muchos sensores tienen como salida el nivel de potencial eléctrico medidas en V, mV y μV (voltios, milivoltios y microvoltios). Esta información sin ningún tipo de procesamiento no es útil. Dependiendo de que se quiera detectar se pueden realizar distintas operaciones. Una de ellas, es la búsqueda de picos. La primera derivada de un pico tiene un cruce descendente igual a cero en su máximo. Por ello, lo que se hace es primero suavizar la señal para eliminar ruido y luego se calculan las derivadas cruzadas. Luego, si la pendiente excede un umbral, significa que se ha encontrado un cero [18]. Estos picos encontrados representan distintas cosas dependiendo el sensor utilizado. Por ejemplo, al utilizar un EMG, puede significar un impulso de fuerza. En un EEG, puede significar un pestañeo.

Otro procesamiento que se le puede aplicar es la transformada discreta de *Fourier*. La transformada de *Fourier* transforma una función que se encuentra en el dominio del tiempo a una función que se encuentra en el dominio de la frecuencia. La transformada de *Fourier* se define de la siguiente manera:

$$x_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i}{N} kn} \qquad k = 0, \ldots, N-1$$

Computar la Transformada de *Fourier* tiene un costo computacional alto. Por este motivo, generalmente, se calcula la FFT. Esta es una aproximación a la Transformada de *Fourier* y tiene un costo computacional menor. Para aplicar la FFT se acumulan valores durante un período de tiempo. Esto se conoce como ventana. A esta ventana, se le puede aplicar una función de ventana. Una función

de ventana es una función matemática en la cual el resultado es 0 si los valores se encuentran fuera de un determinado intervalo y distinto de 0 si se encuentran dentro de él.

Una vez que la función se encuentra en el dominio de la frecuencia, se puede proceder con el procesamiento. Se selecciona el rango de frecuencias de interés y se le puede aplicar un filtro pasa banda, que deja pasar un determinado rango de frecuencias de una señal y atenúa el resto. Luego de aplicar el filtro pasa banda, se cuenta con las frecuencias de interés y se continúa con el procesamiento. Una alternativa es calcular la PSD. Esta se define como:

$$P = \int_{-\infty}^{+\infty} S_{xx}(f)df \qquad \text{donde,}$$

$$S_{xx} = |X(f)|^2 \qquad \textbf{y} \qquad X(f) \text{ es la Transformada de } \textit{Fourier}$$

Esta potencia puede ser utilizada luego como una característica de interés pero esto discutirá más adelante. Otra alternativa es, por ejemplo, calcular el promedio de las frecuencias. Las posibilidades aquí son muchas y dependen de lo que se esté buscando.

3.2.7. Extracción de Características de Interés y Clasificación

Cuando se cuenta con la señal pre-procesada, se utilizan métodos de procesamiento de señales tradicionales que buscan características distinguibles. Se puede utilizar una regla simple como un umbral. Por ejemplo al buscar picos, si la pendiente excede un umbral se le asocia un estado (pestañeo, impulso de fuerza, etc) y sino, otro. Si se buscan patrones más complejos, hay que utilizar métodos de clasificación más robustos que utilizan reconocimiento de patrones. El reconocimiento de patrones consiste en dos pasos principales:

- **Extracción de características:** Esta etapa consiste en elegir características de la señal que sean relevantes al estado que se quiere clasificar. Por lo general estas características se colocan en un vector conocido como vector de características. El extractor utiliza como entrada una o más señales y las transforma en características útiles. Las características pueden ser por ejemplo ciertas bandas de frecuencia, calcular la PSD sobre una determinada banda de frecuencias, el valor de las derivadas en un instante del tiempo, entre otros.

- **Clasificación:** En este paso se le asigna una clase a un conjunto de características (vector de características) extraído de la señal. Aquí se utilizan algoritmos de aprendizaje automático. Existen diversos algoritmos de clasificación, por ejemplo *Linear Discriminant Analisis* (LDA), árbol de decisión, *naive Bayes*, entre otros [14]. También existen clasificadores más complejos como la utilización de redes neuronales. La elección del clasificador depende de la complejidad del problema.

Utilizar aprendizaje automático consiste en dos etapas:

- **Calibración / Aprendizaje:** Consiste en adquirir la señal de entrenamiento, optimizar las características y entrenar el clasificador. Es decir, consiste en la acumulación de datos de entrenamiento que luego son enviados al clasificador.
- **Uso:** Consiste en usar el modelo (características y clasificador) obtenido durante la calibración para poder reconocer el estado del usuario [14].

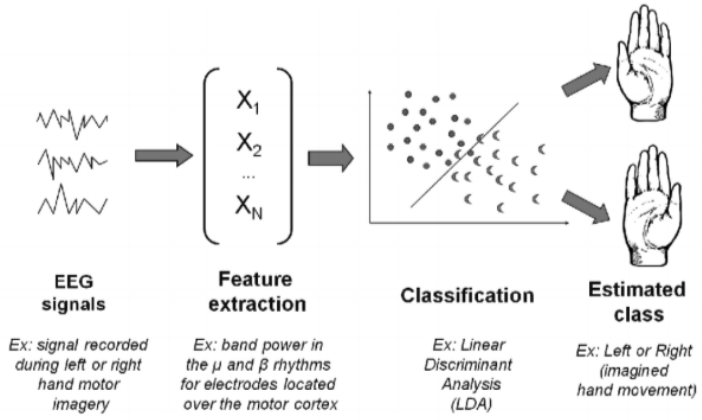

Figura 3.7: Ejemplo de las etapas por la que pasaría el procesamiento de una señal EEG [14].

En la figura 3.7 se puede observar lo descripto anteriormente. Primero, se reciben las señales. Luego, se extraen las características. En este caso se trata de frecuencias asociadas a la corteza motora. Una vez que el vector de características fue creado, se envía al clasificador (en este caso LDA). En la etapa de calibración, se entrena al clasificador indicándole en que estado se encuentra el usuario para cada vector de características. Finalmente, el clasificador determina a que clase pertenece cada vector. En este ejemplo, se intento determinar si el usuario imaginó mover la mano derecha o la izquierda.

Para medir la precisión del clasificador se utiliza una matriz de confusión. Ésta, es una matriz de dos dimensiones en la cual una dimensión representa los valores actuales y otra los valores predichos. Por ejemplo, si se quisiera detectar sin el usuario se encuentra con los ojos cerrados o abiertos, la matriz se armaría de la siguiente manera:

$$\text{Mat} = \begin{array}{c} \\ A \\ C \end{array} \begin{array}{c} \begin{array}{cc} A & C \end{array} \\ \begin{pmatrix} x & y \\ w & z \end{pmatrix} \end{array}$$

, donde A representa el estado de ojos abiertos y C, el de ojos cerrados. La primera dimensión representa el valor obtenido y la segunda el valor predicho. Por lo tanto, $Mat_{0,0}$ representa los positivos verdaderos (TP) (las veces que acertó el clasificador cuando el usuario tenía los ojos abiertos), $Mat_{0,1}$ representa los falsos negativos (FN) (valores que el clasificador clasificó como C pero en

realidad eran A), $Mat_{1,0}$ representa los falsos positivos (FP) (valores que el clasificador clasificó como A pero en realidad eran C), y $Mat_{1,1}$ representa los verdaderos negativos (TN) (valores que el clasificador clasificó como C y realmente eran C). Utilizando esta matriz se puede obtener la precisión, la cuál se define de la siguiente manera:

$$ACC = \frac{TP + TN}{P + N}$$

, donde P = total de casos positivos y N = total de casos negativos. O simplemente:

$$ACC = \frac{Mat_{0,0} + Mat_{1,1}}{Mat_{0,0} + Mat_{0,1} + Mat_{1,0} + Mat_{1,1}}$$

Los valores cercanos a 0,5 implican que el clasificador es malo y es azar. Cuantos más cercano 1, mejor el clasificador.

Para determinar que parámetros de ajuste son los mejores para un modelo se puede utilizar el método de validación cruzada. Este consiste en subdividir un conjunto de datos en dos subconjuntos, utilizar uno de ellos como datos de predicción y otro como de entrenamiento. Luego se valida el subconjunto de entrenamiento contra el subconjunto de predicción y se toma la media. La validación cruzada más simple es el método de regresión. Aquí simplemente se dividen los datos de la muestra en dos particiones: una de entrenamiento y una de validación. Este método es rápido de computar pero el problema es que la evaluación puede depender en gran medida de como es la división entre datos de prueba y de entrenamiento. La varianza puede ser muy elevada. Una mejora a este método de validación es la validación de k iteraciones cruzadas. Este consiste en dividir los datos en k subconjuntos y utilizar k veces regresión lineal. Cada iteración se elige un subconjunto como conjunto de validación y el resto son usados como conjuntos de entrenamiento. Es decir, se elige un único subconjunto de datos como conjunto de validación y los $k - 1$ restantes, se utilizan como conjunto de entrenamiento. Luego se toma el promedio de todas las iteraciones [21].

3.2.8. Salida del Clasificador

Una vez que la señal fue procesada, la información obtenida está lista para ser utilizada. La salida puede ser, por ejemplo, una API de una librería que le permite a un usuario acceder a la información de forma amigable o una visualización en la computadora.

3.3. Señales de Interés

En la sección 3.1 se introdujo el concepto de bioseñales y se mencionaron algunas de ellas. Este proyecto tiene foco en tres de ellas: EEG, EMG y SpO_2. En las siguientes secciones se brindará más información sobre las mismas.

3.3.1. EEG

La EEG detecta la actividad eléctrica de las neuronas, es decir, del cerebro. Con respecto a la clasificación, se trata de una señal permanente, ya que no debe

ser inducida externamente, dinámica, ya que varía mucho en el tiempo, y eléctrica. Los dispositivos utilizados para realizar un eletctroencefalograma cuentan con una determinada cantidad de electrodos que miden el nivel de potencial eléctrico en μV. Al utilizar uno de estos dispositivos, la señal digital que se obtiene es el nivel de potencial eléctrico en cada electrodo. El electroencefalograma ha sido una herramienta de diagnóstico y estudio del cerebro durante casi todo el siglo XX y XXI. Recientemente, se lo ha comenzado a utilizar para inferir y detectar algún patrón cognitivo reflejado en las señales que permita transferir información. Así aparece el concepto de *Brain-computer Interface* (BCI). "BCI es un método de comunicación basado en la actividad neuronal generada por el cerebro... El objetivo de BCI no es determinar el pensamiento de una persona observando su actividad neuronal, sino que es proveer un nuevo canal de salida para el cerebro que requiere una adaptación de control voluntaria por parte del usuario" [8].

Al aplicar la Transformada de *Fourier* a la señal EEG, se obtienen distintas frecuencias. En EEG, dichas frecuencias se dividen en cinco bandas:

- ***Alfa*** **(α):** $8\,Hz \leq f \leq 13\,Hz$. Las ondas *Alfa* se suprimen cuando los ojos se encuentran abiertos y hay estimulación visual presente. Cuando los ojos se encuentran cerrados, estas aumentan.
- ***Beta*** **(β):** $13\,Hz \leq f \leq 30\,Hz$. Las ondas *Beta* se asocian la concentración y aparecen como actividad de fondo en sujetos ansiosos.
- ***Delta*** **(δ):** $0{,}5\,Hz \leq f \leq 4\,Hz$. Estas ondas aparecen en etapas de sueño profundo.
- ***Theta*** **(θ):** $4\,Hz \leq f \leq 8\,Hz$. Las ondas *Theta* aparecen en las primeras etapas de sueño.
- ***Gamma*** **(γ):** $30\,Hz \leq f \leq 60\,Hz$. Algunos expertos consideran los valores de $\beta > 30\,Hz$ como una quinta banda. [8].

Como las frecuencias de interés abarcan hasta por los menos los $60\,Hz$, se debe muestrear al menos a $120\,Hz$. Como se menciono anteriormente, muchas veces se utiliza una frecuencia de muestreo mayor y por lo general en EEG, se utiliza una frecuencia de muestreo superior a los $200\,Hz$ [8].

3.3.2. EMG

La EMG consiste en registrar la actividad eléctrica producida por los músculos del cuerpo. La actividad eléctrica surge cuando una unidad motora es activada por el sistema nervioso del cuerpo: una unidad motora consiste en una motor neurona, y todas las fibras musculares a la que esta conectada. Cuando se activa la unidad motora, se genera un voltaje (llamado MUAP) y se contraen los músculos. Entonces, la señal EMG consiste en medir la suma de varios MUAPs de uno o mas músculos específicos. Con respecto a la clasificación, se trata de una señal permanente, ya que no debe ser inducida externamente, dinámica, ya que varía mucho en el tiempo, y eléctrica.

El rango de la amplitud de una señal EMG es de aproximadamente $0\ mV$ a $10\ mV$, y su frecuencia de $0\,Hz$ a $500\,Hz$. Sin embargo, el rango dominante de energía de la señal es de los $50\,Hz$ a $150\,Hz$, el cual es el que se intenta medir en este proyecto. Por lo tanto, se debe muestrear al menos a $300\,Hz$.

3.3.3. SpO_2

La pulsioximetría permite determinar el porcentaje de sangre arterial saturado con oxígeno. Esta técnica no es invasiva y se basa en medir la absorción de luz roja e infraroja que pasa por el oído o dedo de una persona [6]. Se basa medir una señal llamada *Fotopletismografía*, la cual es una medición óptica del cambio de volumen de sangre en las arterias. Estas son obtenidas irradiando dos longitudes de onda de luz distintas a través del tejido. Existen dos formas de leer la oxigenación en sangre: transmisión y reflexión de la luz. En transmisión, se envían haces de luz utilizando una luz LED y se utiliza un fotodetector del otro lado que detecta estos haces. En cambio, en reflexión, se coloca un fotodetector del mismo lado que la luz led y se detecta la reflexión de la luz [10].

El corazón bombea sangre a través de las arterias de forma continua. Este es conocido como el ciclo cardíaco. Este ciclo puede ser observado con un pulsioxímetro ya que se generan variaciones de volumen en las arterias [10]. En la figura 3.8 se pueden observar los picos generados por el ciclo cardíaco. Dichos picos son utilizados para estimar el ritmo cardíaco. Esta señal es inducida, ya que se requiere de la utilización de una luz, dinámica, ya que varía de forma considerable en el tiempo, y, óptica.

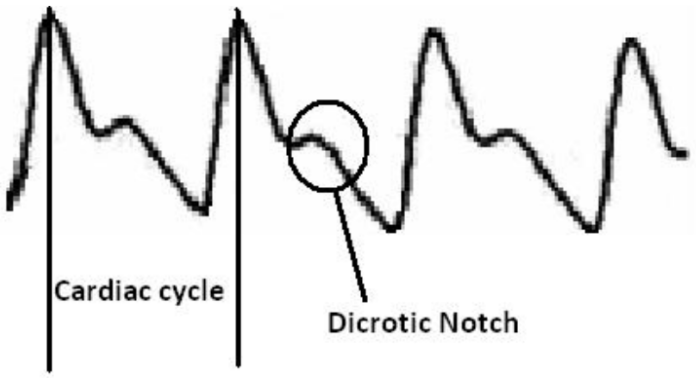

Figura 3.8: Ciclo cardíaco

.

Existen dos formas de calcular el ritmo cardíaco a partir de los picos:

- **Promedio:** Se cuentan la cantidad de picos en un determinado tiempo y luego se utiliza la regla de tres simple. El problema es que este método no permite observar cambios en el tiempo y por lo tanto no representa verdaderamente la respuesta al ejercicio, el estrés y el ambiente.
- **Latido a latido:** Se mide el tiempo (T) entre dos picos consecutivos y luego se calcula usando la fórmula $60/T$ [6].

Se pueden combinar estas dos metodologías y promediar los últimos n resultados obtenidos [6].

Capítulo 4

Implementación

En este capítulo se hablará sobre la implementación realizada. Primero, se mencionarán las señales elegidas y se dará una explicación de por qué se tomo esta decisión. Luego, se introducirá el *hardware* utilizado. Después, se explicará de que manera se procesaron las distintas señales, desde la obtención de las mismas hasta las características y clasificadores utilizados. Finalmente, se hablará de los universos 3D interactivos desarrollados. Los universos 3D fueron desarrollados en el lenguaje *C#* en *Unity*.

4.1. Elección de las señales

Primero se tuvo que elegir que señales se iban a procesar. Para llevar a cabo esta elección, se tuvieron en cuenta los siguientes criterios:

- **Practicidad:** Se buscó que el sensor requerido para obtener la señal no sea invasivo y que sea fácil de utilizar por personas no técnicas.
- **Uso en universos interactivos:** Que se le pueda dar un uso coherente en universos interactivos. Incluye el dinamismo de la señal ya que las señales casi estáticas no muestran una variabilidad suficiente para ser detectada en un período de tiempo de corta duración.
- **Tiempo de procesamiento** El tiempo de procesamiento de las señales obtenidas debía ser aceptable para uso en tiempo real.
- **Precio y disponibilidad** Se buscó que existan dispositivos disponibles en el mercado y a precios accesibles.

Utilizando estos criterios, se eligieron las siguientes señales: EEG, EMG y SpO_2. Todas ellas cumplen con la condición de ser prácticas y no invasivas, lo suficientemente dinámicas para que sean aplicables en nuestro estudio, que el tiempo de procesado sea apto para el tiempo real, y que los sensores fueron accesibles y comerciales como para poder probarlos e implementarlos.

4.1.1. EEG

Existen sensores para capturar este tipo de señal que son no invasivos. Simplemente se utilizan electrodos superficiales, que se sitúan sobre la piel, en diversas partes de la cabeza. Además, el cerebro refleja mucho el estado del usuario por lo que la información extraída de él resulta muy útil e interesante para utilizar en universos interactivos. Al investigar, se descubrió que existen diversos dispositivos (no muy costosos) que tienen una frecuencia de muestreo superior a los $200\,Hz$ y como se mencionó anteriormente, las frecuencias de interés llegan hasta $60\,Hz$, lo cual permite su uso en tiempo real. El hecho de que la frecuencia de muestreo no sea muy elevada permite que se puedan procesar los datos rápidamente. Si la frecuencia fuese muy elevada, tal vez no se podrían procesar en tiempo real y habría una latencia muy elevada. A su vez, la colocación de dicho sensor es fácil. Por lo general se coloca una banda o casco en la cabeza, por lo cuál puede ser utilizado por personas no técnicas.

4.1.2. EMG

Para capturar esta señal, existen electrodos invasivos y no invasivos por lo cual cumple con el primer criterio. Las frecuencias de interés son entre $0\,Hz$ y $150Hz$ y existen diversos dispositivos con frecuencias de muestreo mayores a $300\,Hz$ por lo cual se puede utilizar en tiempo real. Además, con este tipo de dispositivos, se puede medir la fuerza realizada por los músculos. Resulta muy interesante poder medir y detectar cambios en la fuerza realizada por los músculos y utilizar esta información para controlar acciones en universos interactivos. Existen también otro tipo de electrodos de monitoreo EMG: los intramusculares, que van insertados dentro del musculo del sujeto. Este tipo de electrodos no fueron utilizados en este proyecto ya que son muy invasivos, y muy imprácticos dentro del contexto de diseñar un sistema que pueda ser utilizado fácilmente por cualquier persona sin conocimientos técnicos del área.

4.1.3. SpO_2

La SpO_2 no es una técnica invasiva. Como se menciono anteriormente, utiliza luces y fotodetectores. El ritmo cardíaco resulta de mucho interés para ser utilizado en universos interactivos ya que su estado puede reflejar que tan nervioso se encuentra el usuario y utilizar dicha información para alterar el estado de la simulación. A su vez, los picos se miden en tiempo real. Su utilización es tan simple como colocar el pulsioxímetro en un dedo.

Se decidió utilizar un pulsioxímetro en lugar de un dispostivo EKG ya que es más práctico. El *Olimex SHIELD-EKG-EMG* resultaba muy incómodo de utilizar debido a que uno de los electrodos debía colocarse en una pierna y otro en un brazo, lo que hubiese ocasionado dificultad para operar una computadora mientras se utilizaba. Si bien un dispositivo EKG es más preciso que un pulsioxímetro, estudios han demostrado que la estimación realizada por un pulsioxímetro es muy precisa cuando el ritmo cardíaco se encuentra por debajo del 89 % de su máximo [16]. Como el usuario se encuentra sentado en una computadora, jamás alcanzará dichos valores.

Además, como ya se contaba con dos señales eléctricas permanentes, se buscó una señal con otra clasificación.

4.2. Hardware

Para la elección de los dispositivos de *hardware* se tuvieron en cuenta los criterios antes mencionados buscando un compromiso entre costo y fiabilidad así como también se tuvo en cuenta su facilidad de uso.

4.2.1. EEG

En el caso de EEG, se decidió utilizar el *Muse Brainsensing Headband*. Este dispositivo cuenta cuenta con siete electrodos y diversas funcionalidades. En la figura 4.1 se puede observar la configuración del dispositivo. Cuenta con tres electrodos de referencia sobre la frente, dos electrodos frontales y dos electrodos en las orejas. Los electrodos de referencia se utilizan para medir el nivel de potencial eléctrico. Es decir, el potencial de los restantes electrodos, se mide en base a los electrodos de referencia.

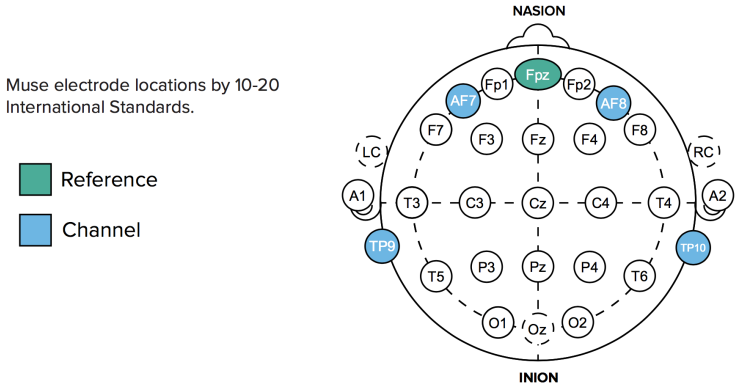

Figura 4.1: Diagrama de la ubicación de los electrodos en el *Muse Brainsensing Headband*. AF7 y AF8 son los electrodos frontales, TP9 y TP10 los electrodos en las orejas y Fpz representa los tres electrodos de referencia [2].

En la figura 4.2 se puede observar las señal EEG bruta de los cuatro electrodos del sensor.

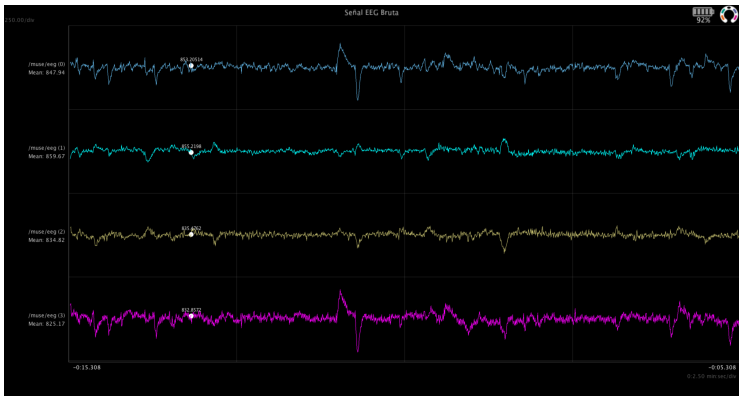

Figura 4.2: Señal EEG bruta de cada uno de los electrodos obtenido utilizando la aplicación *MuseLab*.

El dispositivo tiene una frecuencia de muestreo de $256\,Hz$. Además cuenta con acelerómetro y giroscopio lo que permite medir la orientación del usuario. También cuenta con un puerto micro USB y *bluetooth*. Permite acceder a la información bruta de cada electrodo por lo que se obtienen cinco valores. Cuatro de de los electrodos frontales y de las orejas, y el restante se obtiene de los tres electrodos de referencia. También brinda algunas facilidades como realizar la Transformada de *Fourier*, calcular la potencia de las bandas (α, β, δ, θ y γ), entre otros.

Para los desarrolladores, cuenta con SDK para *Windows* y *Unity* y cuenta con aplicaciones multiplataforma para poder acceder a los datos. Una forma de acceder a los datos es a través de *Open Sound Control System* (OSC). OSC es un protocolo que corre sobre UDP y permite el *stream* continuo de datos.

4.2.2. EMG

Para la lectura de señales EMG, se optó por utilizar la plataforma Arduino, con un modulo adicional diseñado para la captura de señales EMG y EKG. Arduino es una plataforma que consiste en varios micro controladores de diseño abierto (*Open Source Hardware*), y un software que permite programarlos de forma fácil y rápida utilizando el lenguaje de progamación *C++*. Los programas que son compilados e instalados en el *Arduino* son conocidos como *sketches*.

El modelo de Arduino elegido fue el *Arduino Mega 2560*, que cuenta con un procesador que opera a $16\,MHz$, y cuenta con $256Kb$ de memoria para programas, lo cual prácticamente permite trabajar sin restricciones de memoria en mente. A su vez, el modulo adicional elegido fue el *Olimex SHIELD-EKG-EMG*, que trae electrodos superficiales que pueden ser ubicados sobre la piel del musculo que se intenta monitorear. Los electrodos cuentan con tres terminales, marcadas con las letras L (izquierda), R (derecha) y D (referencia) (figura 4.3). Los valores leídos por los electrodos son procesados por tres filtros: dos filtros pasa altos de $16\,Hz$, y un filtro *Bessel* de $40\,Hz$, que similar a un filtro *Gaussiano*.

Figura 4.3: Electrodos superficiales para el modulo *Olimex SHIELD-EKG-EMG* [3].

Para poder leer las señales EMG, se debe preparar el sistema de monitoreo. Primero, se debe montar el modulo EMG sobre el micro controlador *Arduino*. El *Arduino* debe tener instalado un *sketch* específico, detallado en la sección 4.3.2. Luego, las terminales L y R se deben posicionar sobre la piel del musculo a monitorear, y la terminal D debe ser posicionada sobre una superficie del cuerpo a utilizar como referencia, que preferiblemente no contenga musculos, o tenga musculos no relacionados a los de las otras dos terminales (a este circuito se le llama *Driven Right Leg Circuit* en inglés). En este proyecto, se opto por medir la actividad muscular de un brazo, junto a la mano, al formar un puño cerrado apretado. Por lo tanto, los electrodos fueron colocados sobre el músculo flexor común superficial de los dedos, que se encuentra sobre la cara anterior de antebrazo [17]. Una vez realizados los pasos anteriores, se puede proceder a conectar el micro controlador a la computadora vía un cable USB 2.0.

Para comprobar el correcto funcionamiento del dispositivo, se utilizó un software llamado *ElectricGuru*, recomendado por los fabricantes del módulo EEG. Si el sistema de monitoreo fue preparado correctamente, el software debería mostrar una pantalla similar a la siguiente:

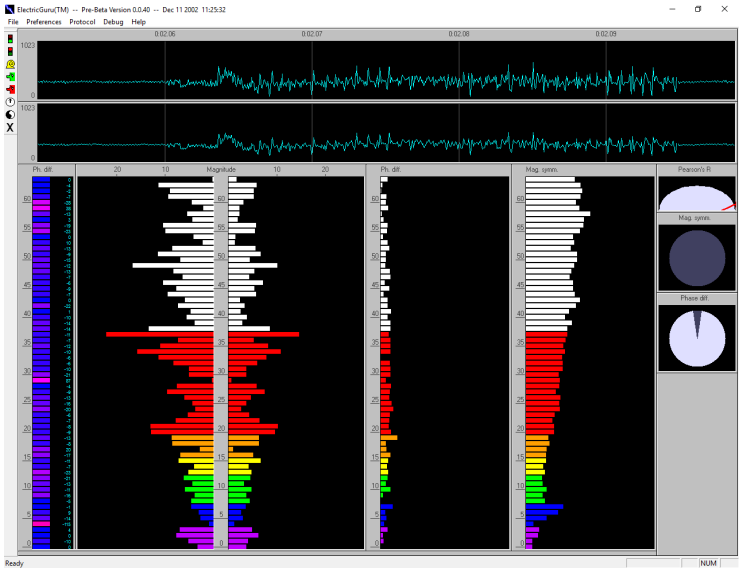

Figura 4.4: Pantalla principal de *ElectricGuru*, leyendo señales EMG.

4.2.3. SpO_2

Para medir el ritmo cardíaco se decidió utilizar el dispositivo *CMS50D+ Contec Pulse Oximeter*. Éste es un pulsioxímetro de transmisión de alta precisión, fácil de utilizar y de bajo consumo. Dicho dispositivo utiliza el puerto serie para enviar datos utilizando un formato propietario. Los datos más relevantes que envía el sensor son:

- **Picos:** Indica si hubo un pico.
- **Ritmo cardíaco:** Estimado utilizando una ventana de 30 segundos.
- **Oxígeno:** Porcentaje de oxígeno.

El dispositivo se conecta a la computadora con un cable USB 2.0 que internamente convierte de serie a USB. Se debe contar con un controlador de puerto serie en la computadora que permita su lectura. La frecuencia de muestreo es de $256\,Hz$. El margen de error del dispositivo utilizado es de $\pm 2\,\%$ para la medición de SpO_2 y ritmo cardíaco. Éste margen de error pequeño se debe a que el *CMS50D+* es un dispositivo médico, con lo que se necesita que sea lo más preciso posible.

4.3. Obtención y Procesamiento de Señales

Para cada uno de los sensores se evaluaron y aplicaron distintas alternativas sobre como procesar la señal. El objetivo al procesar las señales fue que dicho procesamiento sea lo suficientemente preciso como para poder ser utilizado en tiempo real en universos 3D.

4.3.1. EEG

Para leer la señal se decidió utilizar *OSC*. Se tomó esta decisión ya que al momento de iniciar el proyecto, el SDK para Windows se encontraba en estado beta por lo que no era confiable. Además implicaba un desarrollo en el lenguaje *C* mientras que el desarrollo se encontraba en *C#* y, como solo se encontraba disponible para *Windows*, no permitía que el proyecto sea multiplataforma. Tampoco se utilizó el SDK para *Unity*, ya que este era únicamente para dispositivos móviles.

Una gran desventaja de este dispositivo, es que no cuenta con soporte para la aplicación *MuseIO*, que permite obtener paquetes OSC a través de la computadora. Por este motivo, se terminó utilizando una aplicación móvil llamada *Muse Monitor*, que nos permitió conectar el dispositivo al teléfono celular y desde allí enviar los paquetes OSC hacia la computadora. En un principio se pensó que la latencia podría ser un problema ya que cada paquete debe realizar dos trayectos, primero desde el dispositivo hacia el teléfono celular y luego hacia la computadora, pero esto no fue un problema ya que el flujo de datos no era masivo y los paquetes no eran de gran tamaño.

Inicialmente, se utilizó un *framework* para procesar los paquetes OSC pero este no resultó ser fiable. Se buscaron alternativas pero no hubo ninguna que cumpla con los estándares del equipo. Por este motivo, se desarrollo un procesador de paquetes OSC propio basado en otras implementaciones. Esto implicó leer el estándar, comprender como funciona e implementarlo.

Los estados que se buscaron detectar fueron ojos abiertos (A) y ojos cerrados (C). Por este motivo, la información que se necesitaba eran las ondas α. Como se mencionó anteriormente, el dispositivo *Muse* permitía acceder a la potencia de las ondas α. Para obtener dichas ondas el dispositivo sigue el siguiente procedimiento:

1. Obtener una ventana de 25 muestras (pues envía 10 muestras por segundo y el sensor obtiene 256 muestras por segundo) y aplicar la función de ventana $Hann$.

2. Obtener la Transformada de *Fourier* de la ventana.

3. Aplicar un filtro pasa banda para utilizar únicamente las frecuencias de interés ($8\,Hz \leq f \leq 13\,Hz$).

4. Realizar la PSD sobre esos valores. A diferencia de solo integrar, el dispositivo aplica log_{10} a cada valor antes de realizar la integral.

El vector de características se confeccionó utilizando dos valores de *Alpha* consecutivos, es decir, de la siguiente manera:

$$\vec{v} = \left[\begin{array}{cc} \alpha_i & \alpha_{i+1} \end{array} \right]$$

En la etapa de entrenamiento se le asignaba un estado a cada uno de estos vectores. Se le indicaba a al usuario en que momento abrir y cerrar los ojos y se registraba el estado para dichos valores. La precisión obtenida con este método fue en promedio de aproximadamente un 65 %. Esto se debe a que aún cuando el usuario se encontraba con los ojos abiertos, *Alfa* alcanzaban valores tan altos como cuando el usuario se encontraba con los ojos cerrados debido a posibles

ruidos del sensor y a el hecho de que las bioseñales varían mucho. Se podía observar que el clasificador era muy preciso en detectar ojos cerrados pero muy impreciso en detectar ojos abiertos. La precisión en ojos cerrados era de un 99 % y en ojos abiertos de un 25 %.

Para intentar mejorar la precisión, se decidió aumentar el tamaño de la ventana con la expectativa de que al tener una mayor cantidad de valores, los picos anómalos que se generaban cuando el usuario estaba con los ojos abiertos, se redujeran ya que al contar con más muestras es probable que se obtenga una mayor cantidad de valores que corresponden al estado de ojos abiertos. Se podría decir que los picos anómalos se promedian al tener una mayor cantidad valores normales. Cuando se trabaja con bioseñales, cuantos más datos se tomen, más robusta será la clasificación. El problema es que para tomar más datos se necesita más tiempo. Entonces, aquí aparece un compromiso entre robustez y latencia. Como en este proyecto la utilización fue en tiempo real, hubo que considerar seriamente este compromiso. A su vez, cuanto mayor sea la frecuencia de muestreo, mayor es el espacio entre frecuencias y como consecuencia se pueden representar con más granularidad dichas frecuencias. El dispositivo utiliza una frecuencia de $10\,Hz$ para los valores de potencia de *Alfa* y se necesitaba una frecuencia mayor, por lo que se decidió calcular dicha potencia utilizando los datos brutos. El dispositivo envía un arreglo de cinco elementos. Como se mencionó anteriormente, el quinto valor es de referencia por lo que se ignora en el procesamiento. El procedimiento implementado fue el siguiente:

1. Obtener una ventana 256 muestras, es decir, un segundo.
2. Obtener la Transformada de *Fourier* de la ventana.
3. Quedarse únicamente con las frecuencias $8\,Hz - 13\,Hz$.
4. Realizar la PSD sobre esos valores.

Este procedimiento se aplica por cada uno de los cuatro electrodos. Por lo que cada segundo, se cuenta con cuatro valores. El vector de características se forma con estos cuatro valores de la siguiente manera:

$$\vec{v} = \begin{bmatrix} \alpha_{AF7} & \alpha_{AF8} & \alpha_{TP9} & \alpha_{TP10} \end{bmatrix}$$

Con este vector de características, se alcanzó una precisión de hasta un 90 %. Esto se debe a que, si bien empeoro la precisión de detectar ojos cerrados, incrementó notablemente la de ojos abiertos.

Se probó aplicar un filtro *Gaussiano* a la señal cruda pero se obtuvieron exactamente los mismos resultados. Esto significa que el sensor no cuenta con ruido sensible a un filtrado *Gaussiano*. Ya que en la documentación no se especifica que tipos de filtros se aplican, la ausencia de ruido *Gaussiano* podría ser explicada por dos motivos: o el sensor ya cuenta con filtros que reducen el ruido sensible a filtrados *Gaussianos*, o bien que la etapa de cuantificación es lo suficientemente precisa como para no introducir ruido blanco (o una combinación ambos). A su vez, se implementó un *detrending* lineal de la señal pero tampoco se notaron diferencias sustanciales. El método de *detrending* calculaba la media de todas las muestras obtenidas del sensor durante un cierto periodo de tiempo, y luego le restaba ésta a todas las muestras leídas más adelante. Al haber alcanzado el nivel de precisión que se buscaba, se decidió concluir con

el procesamiento de la señal en este punto. Es decir, no se aplicaron filtros de pasa-banda ni funciones de ventana porque no se consideró necesario.

En la etapa de entrenamiento, en lugar de tomar todos los valores, se ignoran los valores de transición. Cuando se le indica al usuario que abra o cierre los ojos, se descartan dos segundos antes y después de la transición de estados. Esto se debe a que los valores que corresponden a la transición no son significativos. Los que realmente importan son cuando se está en un estado u otro.

4.3.2. EMG

Una vez preparado el sistema de monitoreo EMG descripto en la seccion 4.2.2, se puede comenzar a leer los datos. Los micro controladores *Arduino* permiten comunicarse a una computadora mediante un puerto de serie que es establecido a traves de la conexión USB. Para poder leer del puerto de serie desde el sistema operativo, es necesario conocer los parametros de la conexión (velocidad, paridad, cantidad de bits de información, etc.), y el nombre del puerto de serie. Los valores utilizados fueron los predeterminados utilizados en el programa de ejemplo referenciado en el manual del módulo EMG [19].

En la version inicial del *sketch*, la frecuencia de muestreo toma el valor de $256\,Hz$. Por el teorema de muestreo de Nyquist-Shannon, esto quiere decir que dadas estas muestras, solo es posible reconstruir señales con frecuencias iguales o inferiores a $128\,Hz$. Teniendo esto en mente, el procedimiento llevado a cabo para leer los datos de EMG fueron los siguientes:

1. Obtener una ventana 128 muestras, es decir, medio segundo.
2. Obtener la Transformada de *Fourier* de la ventana.
3. Quedarse únicamente con las frecuencias en el rango de $50\,Hz - 128\,Hz$. Las frecuencias en el rango de $128\,Hz - 150\,Hz$ no son capturadas.
4. Separar el rango de frecuencias resultante en dos mitades: $50\,Hz - 89\,Hz$ y $90\,Hz - 128\,Hz$.
5. Realizar la PSD sobre ambas mitades por separado.

Luego de haber realizado los pasos mencionados, el resultado son dos valores numéricos representativos de la intensidad de la señal recibida. Utilizando estos dos valores, se construyo el vector de características de la señal EMG. Con este vector, se lograron precisiones de predicción muy altas (superiores a $95\,\%$), utilizando tiempos de entrenamiento de alrededor de 5 minutos.

Se implementó la técnica de *detrending* de muestras, pero al no encontrar mejoras significativas en precision de los resultados, se decidió no utilizarlo para reducir la cantidad de código en el proyecto. También, no se aplicó un filtro de *Gauss* ya que la precisión alcanzada fue lo suficientemente alta para las necesidades de este proyecto (como se explica en la sección 5.2). Además, como se mencionó anteriormente, el módulo EMG ya utilizaba filtros, por lo que la señal no tenía cantidades significativas de ruido.

Para el entrenamiento, el procedimiento fue simple. Primero, se preparo el sistema de monitoreo EMG, y se colocaron los electrodos sobre la piel del los músculos mencionados en la sección 4.2.2. Luego, con el brazo y mano relajados, se comenzó a entrenar un clasificador con las muestras leídas y procesadas. A

continuación se procedió a tensar el brazo y cerrar la mano con fuerza, con tal de generar una diferencia de potencial significativa, en intervalos intermitentes de alrededor de 15 segundos. Durante todo el proceso, se le informa al clasificador a que categoría pertenece cada dato ingresado (musculo tensionado o relajado).

Luego de experimentar con el valor preestablecido de $256\,Hz$ para la frecuencia de muestreo, se decidió intentar con una frecuencia mas alta, $512\,Hz$. La primera ventaja que esto trae es que, si se sigue tomando una ventana de 128 muestras para la generación de un vector de características, el tiempo transcurrido entre cada valor generado pasa de ser de $0,5$ segundos a $0,25$ segundos. Desde un punto de vista de interacción de usuario, esto quiere decir que la simulación puede reaccionar mas rápido ante cambios de tension de musculo, lo cual permite que la experiencia sea mas fluida y tenga un mejor tiempo de respuesta. La segunda ventaja que trajo el cambio de frecuencia de muestreo es que, por el teorema de Nyquist-Shannon mencionado anteriormente, el rango de frecuencias de $50\,Hz - 150\,Hz$ deseado ahora pudo ser analizado en su enteridad. Al aumentar la frecuencia de muestreo, también aumentó la cantidad de datos transmitidos desde el micro controlador hasta la computadora, por lo que hubo que aumentar la velocidad de transmisión del puerto de serie para compensar.

Al igual que en EEG, al transicionar de un estado de tension de musculo a otro, se ignora cierta cantidad de muestras luego del momento de la transición, ya que solo son de interés los datos leídos al mantener un estado de tension constante.

Al realizar los experimentos, se notó que ocasionalmente los datos leídos contenían grandes cantidades de ruido, por lo que eran prácticamente inutilizables. El problema encontrado fue que el sujeto conectado al modulo EMG no estaba eléctricamente aislado al suelo, lo cual introducía ruido a los valores leídos. El problema fue resuelto asegurándose de que el sujeto utilizara calzado con suficiente aislación eléctrica, como por ejemplo botas o zapatillas deportivas.

4.3.3. SpO_2

Como se mencionó anteriormente, el dispositivo elegido utilizaba un formato propietario. Por este motivo, no se podía obtener una correcta lectura de los datos. La única forma de obtener los datos era realizando ingeniería inversa sobre el protocolo utilizado. Se investigó y se encontró cómo era la estructura interna de cada paquete recibido. Una vez que se podían leer paquetes, estos debían ser procesados. El ritmo cardíaco calculado por el sensor utiliza una ventana de 30 segundos. Esta duración era inaceptable para utilizar en tiempo real ya que reflejaba el ritmo cardíaco calculado sobre los 30 segundos anteriores, la cual es una duración de tiempo demasiado extensa en una aplicación interactiva. Por ésta razón, se decidió calcular el ritmo cardíaco utilizando los picos del ciclo cardíaco, utilizando una ventana más pequeña.

El dispositivo obtiene 256 muestras por segundo, pero el ritmo cardíaco de una persona en reposo se encuentra entre los 60 - 100 BPM. Por lo tanto, para cada latido, hay múltiples paquetes consecutivos indicando que hubo un pico. Para evitar estimar incorrectamente, se estableció un tiempo mínimo t segundos que debía transcurrir entre picos. Entonces, si se recibía un paquete indicando un pico luego de menos de t segundos del anterior, simplemente se lo considero como parte del mismo pico del ciclo cardíaco.

En una primera aproximación, se utilizó una ventana de 10 segundos y se

calculaba el ritmo cardíaco utilizando la técnica de promedio. Dicha técnica no resultó efectiva ya que el ritmo cardíaco contaba con grandes fluctuaciones lo que causaba que sea inutilizable en tiempo real. Luego, se empezó a utilizar la técnica de medir el tiempo entre latidos y realizar el promedio de los últimos nueve valores. Esta técnica resultó efectiva indicando un ritmo cardíaco estable y suficientemente preciso. Se comparó con el valor de ritmo cardíaco obtenido por el sensor para determinar si era preciso. A su vez se comparó con lo medido por un *Apple Watch*.

Con este sensor simplemente se buscaba observar si el ritmo cardíaco superaba un umbral por lo que no requirió reconocimiento de patrones.

4.4. Desarrollo de Universos 3D Interactivos

Para el desarrollo de universos 3D se decidió utilizar el motor gráfico para videojuegos *Unity 3D*, junto al lenguaje de programación *C#*. Se tomó esta decisión por diversas razones. En primer lugar, el motor *Unity* fue diseñado para ser multiplataforma: puede ser utilizado para crear juegos y/o aplicaciones para *macOS*, *Windows*, *GNU/Linux*, *iOS* y *Android*, entre otras. En segundo lugar, también se eligió por su flexibilidad y robustez: es utilizado por varios estudios profesionales de videojuegos, y también por una gran cantidad de usuarios independientes, por lo que cuenta con una gran comunidad de creadores.

El motor gráfico *Unity* puede ser utilizado/extendido con distintos lenguajes de programación, siendo *C#* uno de ellos. *C#* es un lenguaje creado por *Microsoft* como parte de la plataforma *.NET*. La plataforma *.NET* es similar a la plataforma *Java*: su diseño esta centrado en una maquina virtual llamada *Common Intermediate Language* (CIL), que ejecuta código de tipo CIL, que es obtenido al compilar código *C#*:

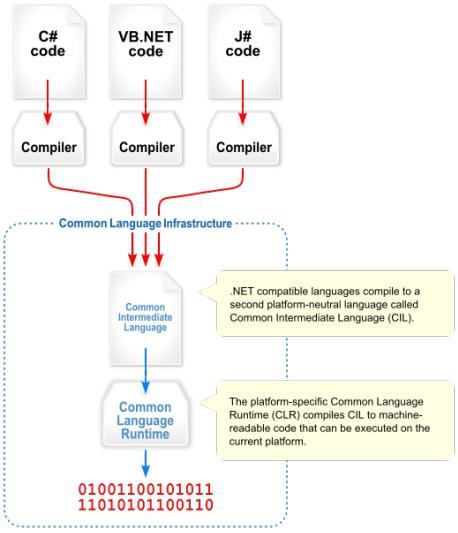

Figura 4.5: Diagrama de la arquitectura de la plataforma *.NET*.

Aunque *Unity* permite el uso de *C#* para su extensión, no utiliza la má-

quina virtual *CLR* para su funcionamiento. En cambio, utiliza una herramienta propietaria llamada IL2CPP, que traduce código CIL a código *C++*. Al ser *C++* un lenguaje extremadamente popular y difundido, éste puede ser luego compilado a cualquier código nativo que sea necesario para cada plataforma individual. De ésta forma, *Unity* permite utilizar código compilado a CIL en cualquier plataforma, sin utilizar una maquina virtual:

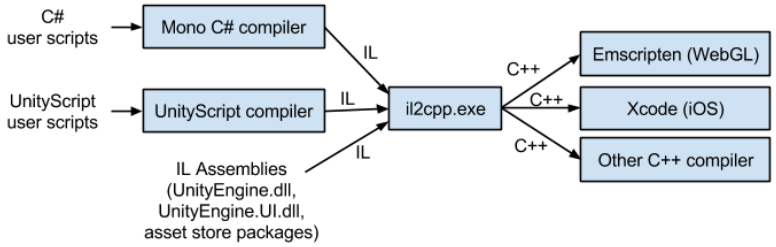

Figura 4.6: Diagrama de la arquitectura de compilación para el motor gráfico *Unity*. Las siglas *IL* equivalen a el CIL del diagrama 4.5 [20].

Como se muestra en la figura 4.6, cualquier archivo conteniendo código en formato CIL puede ser utilizado como parte del juego o aplicación. Como consecuencia de ésto, se pudieron utilizar librerías diseñadas para la plataforma *.NET* como parte del desarrollo del proyecto. Una de las librerías utilizadas fue *Accord.NET* (versión 3.3.0), que cuenta con varios módulos relacionados al aprendizaje automático y a la matemática [1]. También, se utilizó la librería estándar de *C#* para el manejo de las conexiones a los puertos de serie.

El diseño interno de *Unity* está basado en el sistema de entidades y componentes, conocido como *Entity-Component System* (ECS). En los sistemas ECS, existen entidades, que son simples contenedores de componentes. Las componentes pueden tener cualquier tipo de comportamiento o propiedades. Éstos sistemas se centran en el concepto de composición, en lugar de herencia, como en la programación orientada a objetos clásica. En *Unity*, las entidades son llamadas *GameObjects*. Estos son objetos que pueden tener múltiples componentes. Un tipo de componente muy utilizado son los *MonoBehaviour*, que derivan su comportamiento de un archivo de código especificado por el programador. Con dichos archivos se puede controlar a los objetos y componentes, y alterar sus estado. Otros ejemplos de componentes son materiales, cuerpos rígidos, etc. *Unity* tiene un ciclo de vida, el cual es seguido por los *GameObjects*. En la figura 4.7 se puede observar una versión simplificada del ciclo de vida.

Cuando comienza el programa, luego de que se hayan inicializado las variables y los sistemas internos del motor, se llama a la función *Awake()* de todas las componentes. Aquí se pueden establecer referencias entre objectos para que luego puedan intercambiar información. En este momento, el juego ya está listo para comenzar. Luego, se llama a *Start()* para terminar con la inicialización. Después comienza el ciclo principal de juego. Aquí se llama a *Update()* y *FixedUpdate()*. La diferencia entre ambos es que *Update* es llamado una vez por cuadro pero *FixedUpdate* se encuentra ligado al motor físico. Por este motivo,

puede ser llamado cero o más veces por cuadro. En cada actualización, el programador es responsable de realizar los cambios necesarios al estado del juego. Finalmente, se comienzan a liberar los recursos y se llama a *OnDestroy*, en el cual se llevan a cabo las acciones necesarias antes de que finalice el juego.

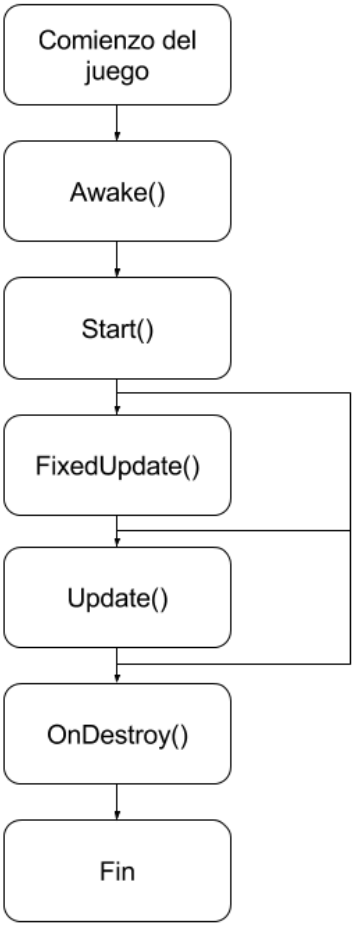

Figura 4.7: Versión simplificada del ciclo de vida de *Unity*.

El método *Update* fue elegido para leer datos de los sensores. El problema que esto traía era que dicho método no puede ser bloqueado, ya que el ciclo principal de *Unity* se ejecuta en un único *thread* de procesamiento. *Update* debe procesar información rápidamente para que el motor continue con su ciclo de actualizado y renderizado. Ante esta limitación, se decidió que los sensores operen en otros *threads*. De esta forma, distintas rutinas podían leer y procesar información de los sensores y el juego luego podía obtener dicha información en los *Update*. La arquitectura descrita se aplicó a la lectura y procesamiento de todas las señales elegidas, la cual se puede observar en la figura 4.8.

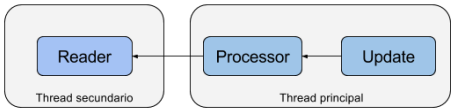

Figura 4.8: Arquitectura utilizada en los universos 3D.

Para cada sensor, se cuenta con una instancia de la clase *Reader* correspondiente, que es el encargado de obtener los datos del mismo. Esta instancia se ejecuta en otro *thread*, ya que las operaciones de lectura son bloqueantes. Dicha instancia coloca los datos obtenidos por el sensor en una cola. Luego, una instancia de la clase *Processor* puede consumir esos datos desde el *thread* principal del juego. *Processor* cuenta con un método *Update*, el cual es llamado desde el ciclo del juego, es decir, desde el método *Update* de la rutina que se ejecuta en *Unity*. Cada vez que se llama a *Update*, el procesador obtiene datos de la cola del lector (la deja vacía), los procesa y los deja disponibles para que el método *Update* de *Unity*, los pueda utilizar sin problemas. Es decir, en cada actualización se obtienen todos los datos no procesados en la actualización anterior y se procesan. Cabe destacar que *Unity* no permitía el uso de la *ConcurrentQueue* de *.NET* (cola de datos que permite operaciones concurrentes), ya que utiliza una versión desactualizada de la plataforma. Por este motivo, se realizó una implementación propia. Ésta arquitectura nos permitió a su vez, desarrollar una aplicación en *C#*, por fuera de *Unity*, con el fin de realizar pruebas y registrar sesiones de entrenamiento.

En todos los universos desarrollados se buscó que el uso de las bioseñales tenga sentido y no sea forzado. Es decir, que sea lógico y cómodo para el usuario. Se buscó que mejoren la experiencia, en lugar de empeorarla. Si la predicción fallara y el usuario se encontrara realmente en otro estado, se buscó que esta situación no perjudique al usuario. Es decir, que no le quite vida o decremente sus posibilidades de ganar. En el peor caso, que beneficie al usuario, en lugar de perjudicarlo.

4.4.1. Universo 3D Utilizando Bioseñales Cerebrales

Se desarrolló un vídeojuego de terror en el que el usuario debía escaparse de un laberinto. Dicho laberinto es oscuro, está custodiado por *Zombies* (criaturas hostiles que atacan al jugador al verlo) y hay bombas escondidas. El usuario cuenta con una pistola y los movimientos se controlan utilizando el teclado y el *mouse*. Como se mencionó anteriormente, se utilizaron ondas cerebrales para detectar si el usuario se encontraba con los ojos abiertos o cerrados. En este videojuego, cuando el usuario cierra los ojos y no realiza movimientos, puede escuchar los sonidos que generan las bombas y las puede visualizar en la pantalla luego de abrir los ojos, para poder desactivarlas. A su vez, se acumula el tiempo que el usuario se encuentra con los ojos cerrados, y luego de abrir los ojos, se puede observar a los *Zombies*, bombas, municiones y paquetes de salud a través de las paredes. Este habilidad dura tantos segundos como los acumulados con los ojos cerrados, hasta un valor máximo de 10 segundos. El personaje puede escuchar y visualizar bombas, *Zombies*, entre otros, ya que al cerrar los ojos, se concentra e incrementa la sensibilidad auditiva. Aumentar los sentidos es un recurso muy utilizado en muchos videojuegos. La diferencia es que en este

videojuego, el usuario realmente se encuentra con los ojos cerrados por lo que se utiliza el estado actual en lugar de un simple botón. Sin embargo, al estar con los ojos cerrados, corre el riesgo de ser atacado y no poder responder a tiempo. En la figura 4.9 se puede observar una captura de pantalla que muestra lo que ve el usuario al abrir los ojos luego de haberlos mantenido cerrados por unos segundos.

Figura 4.9: Captura de pantalla del universo desarrollado para el dispositivo EEG. Aquí se observa el resultado de mantener los ojos cerrados durante algunos segundos. Los elementos verdes son bombas, los cyan *Zombies*, los amarillos municiones y los rojos paquetes de salud.

La clase *EEGProcessor* se encarga de obtener lecturas del cerebro de una cola. Dicha cola se encuentra en *EEGReader* y es esta clase la que se encarga de leer los datos del sensor y publicarlos a la cola. *EEGProcessor* permite que se especifique una lista de funciones a las cuales llamar una vez que se procesaron los datos. Luego de obtener 256 lecturas de cada electrodo, se calcula la FFT para cada electrodo. Es en este instante en el que se llama a dichas funciones. De esta manera, una rutina en *Unity* es llamada y obtiene la información necesaria del procesador en ese instante. En este caso, el estado de los ojos. En otras palabras, cuando el procesador tiene información para enviar, le notifica a la rutina de *Unity* para que éste la utilice.

Como se utilizó reconocimiento de patrones, se implementó una interfaz para entrenar al clasificador. Dicho entrenamiento tiene una duración de entre 1 y 10 minutos (determinable por el usuario) , en los cuales se le indica al usuario que abra y cierre los ojos por períodos de tiempo aleatorio. Durante el entrenamiento se identifica el estado en el que está el usuario. Una vez finalizado el entrenamiento, el usuario puede comenzar a jugar.

Debido a que la precisión del clasificador era de un 75 % (como se verá en la sección 5.1), se decidió establecer un umbral para determinar en que momento el usuario se encuentra con los ojos cerrados. Cada vez que se predice que se encuentra con los ojos abiertos, se decrementa un contador, y cada vez que se predice que se encuentra con los ojos cerrados, se incrementa. El contador tiene como valor mínimo cero y el valor máximo es establecido por el usuario. Luego, cuando el contador supera un umbral, se considera que los ojos están cerrados hasta que dicho contador se encuentre nuevamente por debajo del umbral o hasta que el usuario realice movimientos con el personaje. Al procesar las ondas

cerebrales, se observó, que en ocasiones, aún cuando el usuario se encontraba con los ojos abiertos, las ondas *Alfa*, obtenían valores muy altos. Por este motivo, podían ser clasificadas como ojos cerrados. Al utilizar un umbral, se impone la condición de que las ondas *Alfa* se encuentren elevadas por un determinado periodo de tiempo. Algo que no debería suceder al encontrarse con los ojos abiertos. Este comportamiento se controla con dos variables (umbral mínimo y umbral máximo) accesibles desde el menú de pausa. Si se identifica que el usuario registra valores de *Alfa* elevados por períodos de una duración de tiempo determinada, se deben incrementar tanto el umbral mínimo, como el máximo. En caso contrario, se deben decrementar. La utilización de este sistema, puede causar retardo en la detección de estado ya que se debe esperar a tener una mínima cantidad de lecturas de datos para determinar el estado. Por este motivo, se debió tomar una decisión de compromiso en el momento de elegir los valores. No debían ser muy bajos, ya que se iba a detectar falsamente que el usuario se encontraba con los ojos cerrados. Pero tampoco, debían ser muy altos porque el retardo iba a ser elevado. Como el usuario debe esperar al menos tantos segundos como especifique el umbral (pues las ventanas de muestras son de 1 segundo), se decidió sumar el valor del umbral mínimo a la cantidad de segundos acumulados.

4.4.2. Universo 3D Utilizando Bioseñales de los Músculos

Se desarrolló un universo 3D interactivo en el que el usuario al tensar el músculo acumula energía. Luego, al liberar la tensión, envía un proyectil eléctrico que impacta contra otros objetos. Cuanto mayor sea el tiempo que se mantiene tensión, mayor es la fuerza del proyectil y mayor es la energía liberada en el impacto contra otros objetos. En el universo 3D se cuenta, por ejemplo, con barriles colocados uno encima del otro. Dichos barriles pueden ser derrumbados arrojando proyectiles. Se decidió utilizar la tensión para este tipo de ataque ya dentro de la simulación, los proyectiles se crean desde el el brazo y mano del jugador virtual. En la figura 4.10 se puede observar una captura de pantalla de dicho universo.

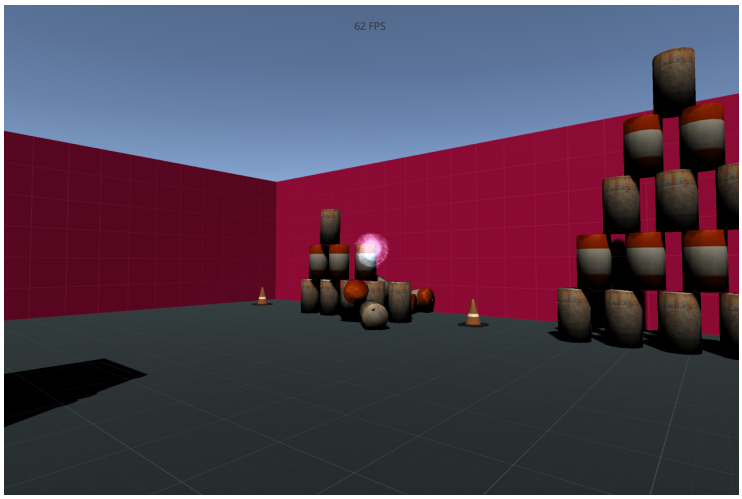

Figura 4.10: Captura de pantalla del universo desarrollado para el dispositivo EMG.

La fuerza se modeló como un simple contador. Cada actualización en la que el músculo se encuentra tenso, se incrementa el contador. Si se encuentra relajado, se decrementa el contador. Dicho contador nunca alcanza valores menores a 0 y nunca valores mayores a un máximo establecido anteriormente. Una vez que el contador supera un umbral, si se detecta una relajación, se libera el proyectil cargado. Se puede observar una pequeña latencia al liberal el proyectil. Esto se debe a que, como se utilizan 128 muestras para extraer las características y predecir un estado, se obtienen cambios de estado cada cuarto de segundo. La velocidad y tamaño del proyectil lanzado son proporcionales a el valor del contador, por lo que hay un incentivo para mantener el músculo tensado por una mayor cantidad de tiempo.

Al igual que con la simulación que utiliza información del EEG, aquí se cuenta con un lector y un procesador. El procesador también permite que otros objetos se subscriban a eventos. Cuando el procesador obtuvo la cantidad de muestras necesarias y las procesó, notifica a los objetos que se hayan suscrito. De esta manera, la rutina que se ejecuta en *Unity* puede conocer el estado en cada actualización y realizar las acciones necesarias. Aquí no hizo falta la utilización de un umbral para determinar un cambio de estado, ya que la precisión era muy alta. Este universo también cuenta con una interfaz con instrucciones de entrenamiento.

4.4.3. Universo 3D Utilizando el Ritmo Cardíaco

Se continuó con el desarrollo del videojuego en el que se utilizaron las bioseñales del cerebro. Se desarrolló un nuevo nivel que sucede luego de que el jugador escape del laberinto. Al escapar del laberinto, éste se encuentra en el centro de un campo, con una linterna y un arma, rodeado de *Zombies*. El objetivo es sobrevivir a todas las olas de ataques de *Zombies*. Cuando el jugador comienza a ponerse nervioso, el ritmo cardíaco aumenta. Se utiliza esta información para simular el temblor de las manos del jugador virtual. Cuanto más alto es el ritmo cardíaco, más movimientos aleatorios realiza el arma, dificultando

la tarea de apuntar. Este movimiento imita a la realidad, ya que cuando uno se encuentra nervioso, le resulta más difícil realizar acciones con las manos, como apuntar un arma. El jugador puede también quedarse sin municione, lo que genera mayor nerviosismo en el usuario porque no cuenta con ninguna forma de defenderse. En la figura 4.11 se puede observar una captura de pantalla del universo desarrollado.

Figura 4.11: Captura de pantalla del universo desarrollado para el dispositivo SpO_2. En la esquina superior derecha se puede observar el ritmo cardíaco del usuario.

Aquí también se cuenta con un procesador (*SPO2Processor*) y un lector (*SPO2Reader*). El procesador expone un método que permite obtener el ritmo cardíaco actual. De esta manera, la rutina en *Unity* obtiene este valor en cada actualización y en base a él, realiza movimientos sobre la cámara. Dichos movimientos se modelaron utilizando funciones trigonométricas para que sean armónicas y agradables a la vista. La magnitud de los movimientos se controla con dos variables: mínimo y máximo. Una establece el umbral mínimo que se utiliza para determinar si el usuario está nervioso y la otra el máximo. Luego, si el el ritmo cardíaco es mayor al umbral mínimo, se interpola linealmente entre el mínimo y el máximo.

4.5. Arquitectura de la librería implementada

El proyecto desarrollado sigue la estructura clásica de un proyecto *Unity*. Se cuenta con el directorio *Assets*, que contiene imágenes, archivos de audio, y archivos de código que forman parte del proyecto. Dentro de este directorio, se decidió separar los archivos relacionados a las simulaciones interactivas, y los archivos relacionados a la lectura y procesamiento de las bioseñales. El directorio *pf-core* contiene todas las clases y rutinas necesarias para poder establecer una conexión con los sensores, procesar los datos leídos, y exponer los resultados para que puedan ser utilizados en otras aplicaciones.

Como se mencionó en la sección 4.4, la plataforma *.NET* utiliza código CIL como representación intermedia, que puede ser ejecutado en una maquina virtual. El código *CIL* puede ser almacenado como un archivo ejecutable (por ejemplo, un archivo *.exe* para *Windows*), o como una librería dinámica (por ejemplo, *.dll*). En el caso de nuestro proyecto, sería fácil generar un archivo *.dll*

a partir del directorio *pf-core*, que luego podría ser utilizado en otros proyectos. Por ejemplo, si se quisiera crear una aplicación que registre el ritmo cardíaco de un paciente durante un intervalo de tiempo y luego almacene los resultados en un archivo, se podría incluir nuestra librería *.dll* para poder acceder al sensor SpO_2.

Capítulo 5

Resultados y Análisis

En este capítulo se presentan los resultados obtenidos al procesar las señales y se realiza un breve análisis. Al registrar sesiones, se creaba una única sesión y luego se dividía en múltiples partes. Debido a que el posicionamiento del sensor es importante y varía cuando se coloca y se retira, y a que el estado del cuerpo cambia de un instante al otro, las bioseñales sufren variaciones. Por este motivo, se utilizaban datos de una única sesión tanto como para entrenar como para predecir. De esta manera, se buscaba que la señal sea lo más consistente posible. Para calcular la precisión se utilizó el método de validación cruzada de k iteraciones. Se tomó esta decisión ya que, como se mencionó en la sección 3.2.7, partir las sesiones en dos partes y utilizar una para entrenamiento y otra para validación no resulta representativo debido a que la elección del lugar en donde se parten los datos puede influir mucho en el resultado. Se utilizaron valores de k iguales a 5, 10 y 15. La elección de estos valores se basó en la cantidad de muestras con la que se contaba. Luego, se promediaban los resultados arrojados por cada valor de k y ese número reflejaba la precisión de la sesión. Tanto en EEG como en EMG se analizaron los siguientes clasificadores: *naive Bayes*, LDA, *Support Vector Machines* (SVM) y árboles de decisión. En ningún caso se realizó un análisis del por qué algunos clasificadores resultaron mejores que otros ya que no fue el foco de este proyecto.

5.1. EEG

Como se mencionó en la sección 4.3.1 del capítulo anterior, primero se comenzó utilizando los valores de potencia *Alfa* que brindaba el sensor ($\alpha_{10\,Hz}$). Para intentar mejorar la precisión, se decidió realizar una implementación propia ($\alpha_{256\,Hz}$) y confeccionar el vector de características con cuatro elementos. Para determinar que clasificador utilizar, se utilizaron todos los conjuntos de datos y se promedio la precisión de cada clasificador. En la tabla 5.1 se pueden observar los resultados de cada clasificador.

Clasificador	Precisión $\alpha_{10\,Hz}$	Precisión $\alpha_{256\,Hz}$
Naive Bayes	0,656	0,615
LDA	0,667	0,689
SVM	0,627	0,5374
Árbol de decisión	0,6723	0,755

Cuadro 5.1: Resultados de utilizar distintos clasificadores sobre todas las muestras.

Se puede observar que los mejores resultados se obtuvieron utilizando un árbol de decisión con un vector de cuatro características obtenido de 256 muestras. Como en ambos métodos el árbol de decisión fue el que mejores resultados arrojó, se utilizó dicho clasificador.

En la tabla 5.2 se pueden observar los resultados de todas las sesiones.

Sujeto	Precisión $\alpha_{10\,Hz}$	Precisión $\alpha_{256\,Hz}$
1	0,565	0,665
2	0,831	0,881
3	0,768	0,881
4	0,727	0,909
5	0,510	0,660
6	0,645	0,655
7	0,656	0,795
8	0,682	0,593

Cuadro 5.2: Resultados de utilizar distintas formas de calcular potencia de *Alfa* en distintos sujetos utilizando como clasificador un árbol de decisión.

La primer observación que se puede realizar es que utilizar $\alpha_{256\,Hz}$ es más preciso que utilizar $\alpha_{10\,Hz}$ en la mayoría de los casos. Como se mencionó anteriormente, esto se debe a que en el primer caso se toman 256 muestras para calcular la potencia de *Alfa* mientras que en el segundo se utilizan únicamente 25. A su vez, en el primer caso se utilizan los valores obtenidos por cada electrodo por separado mientras que en el segundo caso se promedian y se arma el vector de características con dos valores consecutivos. Utilizar las mediciones de los distintos electrodos en un mismo período de tiempo describe mejor el estado que utilizar valores de dos períodos consecutivos. Se consideró inaceptable que en algunos sujetos la precisión diera muy cercana al azar. Por este motivo y por todo lo mencionado anteriormente, se utilizó el método $\alpha_{256\,Hz}$.

En las figuras 5.1 y 5.2 se observan gráficos de dispersión utilizando los valores de potencia de *Alfa*. En ambos se observa una clara separación de estados. El estado de ojos cerrados contiene valores mayores que el estado de ojos abiertos. La figura 5.1 cuenta con una cantidad mayor de valores atípicos, particularmente, una cantidad mayor de valores de *Alfa* elevados en el estado de ojos abiertos. A su vez, en la figura 5.2 los valores para ojos abiertos se encuentran mayormente concentrados por debajo de los valores para ojos cerrados. Por estos motivos, la precisión es mayor al utilizar el vector de cuatro dimensiones. Debido a la separación que se observa en los gráficos, fue suficiente utilizar un clasificador simple.

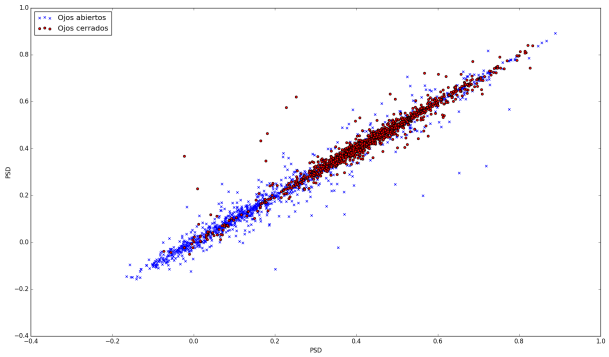

Figura 5.1: Gráfico de dispersión del vector de características de dos características del sujeto 2

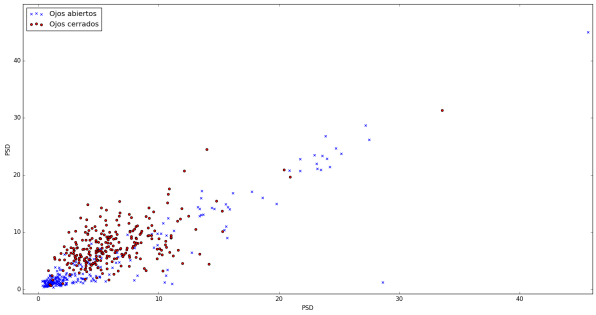

Figura 5.2: Gráfico de dispersión del vector de características de cuatro características del sujeto 2. Para transformar de cuatro dimensiones a dos, se promediaron los dos primeros valores y los dos valores finales, es decir, AF7 con TP9 y AF8 con TP10.

Al comparar las figuras 5.3 y 5.2, se puede observar, que hay una mayor separación entre estados en la figura 5.2. Por este motivo, la precisión del sujeto 2 fue un 15 % superior.

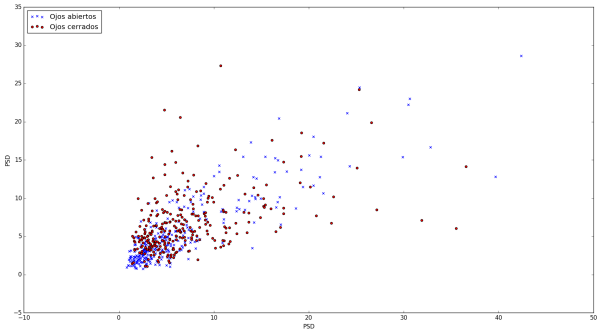

Figura 5.3: Gráfico de dispersión del vector de características de cuatro características del sujeto 1.

Cabe destacar que al contar con un vector de cuatro características, para graficar se necesitarían cuatro dimensiones. Para abordar este problema se decidió transformar el vector en un vector de dos componentes donde la primer componente es el promedio de los dos primeros valores y la segunda, el promedio de los dos últimos valores, es decir, el promedio de AF7 y TP9 por un lado, y el promedio de AF8 y TP10 por el otro.

Otro aspecto a analizar es el hecho de que la precisión varía mucho de sujeto a sujeto, la variación inter-personal intrínseca de cada individuo. Esto puede deberse a diversos motivos, pero generalmente se atribuyen a la variabilidad natural biológica. Muchas bioseñales presentan adicionalmente una variación inter-personal notoria, es decir cambios entre las señales de la misma persona en diferentes momentos. Además, pueden presentar un fenómeno reflexivo, donde al saber que está siendo analizada cada persona, se encuentra nerviosa, por ejemplo, por lo que puede afectar las ondas cerebrales. A su vez, si la persona se encuentra cansada, por ejemplo, los estímulos al campo visual tienden a ser menores, lo que genera que lo valores de *Alfa* sean mayores. Otro de ellos es que el cerbero de dos personas distintas responde formas diferentes antes los estímulos. Esta gran diferencia puede apreciarse al observar que el sujeto 8 obtuvo una precisión mucho menor que el resto de los sujetos.

Si bien las ondas cerebrales varían mucho de persona a persona y de acuerdo al instante del tiempo, se realizó la prueba de utilizar como sesión de entrenamiento una sesión del sujeto 2 y como sesión de predicción una sesión del sujeto 1. Los resultados fueron remarcablemente positivos ya que se obtuvo una precisión de 0,654. Este valor no es comparable con los obtenidos anteriormente ya que dichos valores utilizaron validación cruzada y éste simplemente utilizo un conjunto de datos para entrenamiento y otro para validación. De esta forma, se observa que el sistema de clasificación ofrece un aceptable nivel de robustez para esta aplicación.

Utilizar un umbral para determinar si el usuario se encontraba con los ojos cerrados, compensó la falta de precisión del clasificador. Se obtuvieron buenos resultados en el universo interactivo con tan solo 1 minuto de entrenamiento. Si bien agregó una pequeña latencia, mejoró la robustez del sistema.

5.2. EMG

Como se mencionó en la sección 4.3.2, los valores leídos fueron procesados para obtener un vector de dos características cada 128 muestras, y luego fueron alimentados a un clasificador. Para la elección de clasificador, se realizaron pruebas sobre cuatro clasificadores distintos. Se tomaron seis sesiones de EMG grabadas, y se aplico la técnica de validación cruzada con k iteraciones (mencionada en la sección 3.2.7) por cada clasificador. Los resultados fueron los siguientes:

Clasificador	Precisión
Naive Bayes	0,962
LDA	0,951
SVM	0,457
Árbol de decisión	0,973

Cuadro 5.3: Precisiones de distintos clasificadores sobre las sesiones de EMG.

Como se puede observar en la tabla 5.3, los niveles de precisión de todos los clasificadores, excepto el de SVM, fueron muy elevados. Se decidió proceder a utilizar el de Árboles de decisión ya que produjo el resultado más elevado. No se realizaron pruebas adicionales ya que se consideró que el nivel de precisión alcanzado era lo suficientemente alto para fines prácticos de la simulación. Como se observará más adelante, existe una clara separación entre los valores de un estado y el otro por lo que un clasificador simple es más que suficiente. Además, los clasificadores o esquemas más complejos requieren más poder de procesamiento y requieren una gran dedicación de tiempo para determinar los mejores parámetros. Luego de haber elegido el clasificador, se pudo realizar un estudio mas detallado, sesión por sesión. En la siguiente tabla se detalla: el sujeto utilizado, la precision alcanzada luego de entrenar al clasificador, y la duración total en segundos de las muestras tomadas. Para calcular la precisión obtenida en cada sesión, se utilizó nuevamente el método de validación cruzada de k iteraciones.

Sujeto	Precisión	Duración (Segundos)
1	0,970	603
2	0,955	239
3	0,994	393

Cuadro 5.4: Resultados de entrenamientos utilizando lecturas de EMG para distintos sujetos. Frecuencia de muestreo: $256\,Hz$.

Como muestran los valores de la tabla 5.4, los niveles de precisión alcanzados son muy elevados: en las tres sesiones realizadas, se logró un nivel de precisión de 95 %, o mayor. Es necesario remarcar que al medir señales de EMG, existen varias fuentes de ruido eléctrico, que pueden afectar los valores leídos y por lo tanto perjudicar la precision del entrenador. El efecto neto de éstas fuentes de ruido varía de sesión en sesión, ya que depende de factores como la cantidad de movimiento de los electrodos en relación a la piel, o la presencia de dispositivos que emitan radiación electromagnética [15].

A continuación, se muestran los gráficos de dispersión de las dos sesiones más largas, que permiten visualizar las diferencias entre los vectores de características de ambos estados de tensión de músculo.

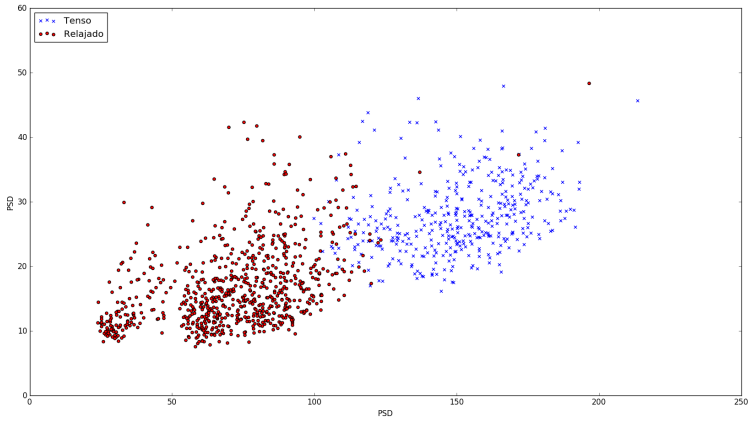

Figura 5.4: Gráfico de dispersión del vector de características EMG para la sesión del sujeto 1.

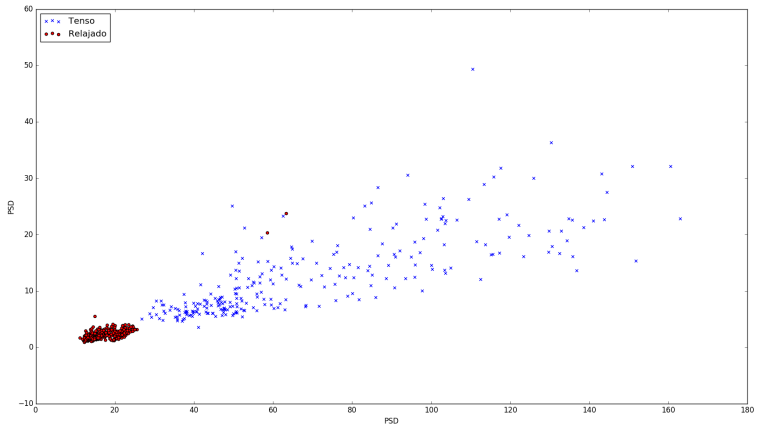

Figura 5.5: Gráfico de dispersión del vector de características EMG para la sesión del sujeto 3.

En la figura 5.4, se puede observar una clara separación entre las muestras tomadas con el musculo relajado, y las muestras tomadas con el musculo tensado. En la figura 5.5 se puede observar el mismo efecto, con una separación aun mas marcada. Esta clara separación fue la que permitió utilizar un clasificador simple. También, se puede notar como las muestras tomadas con músculo relajado se encuentran concentradas en un área pequeña, debido a la poca variación

de potencial medido de muestra en muestra. Por el otro lado, las muestras tomadas con músculo tensado se encuentran dispersas, ya que al tensar el músculo no siempre es posible mantener el mismo nivel fuerza ejercida.

Como se menciono en la sección 4.3.2, la frecuencia de muestreo del modulo EMG fue cambiada de $256\,Hz$ a $512\,Hz$. Para comprobar que la precision de predicción se mantuvo luego de este cambio, se grabaron nuevas sesiones.

Sujeto	Precisión	Duración (Segundos)
1	0,972	284
2	0,991	339
3	0,984	292
4	0,953	130
5	0,969	146

Cuadro 5.5: Resultados de entrenamientos utilizando lecturas de EMG para varios sujetos. Frecuencia de muestreo: $512\,Hz$.

Como se puede observar en la tabla 5.5, el nivel de precision se mantuvo elevado, luego de haber realizado el cambio de frecuencia de muestreo. Las dos sesiones mas largas fueron graficadas, nuevamente, para poder observar la relación entre los valores de los vectores de características y el estado del músculo:

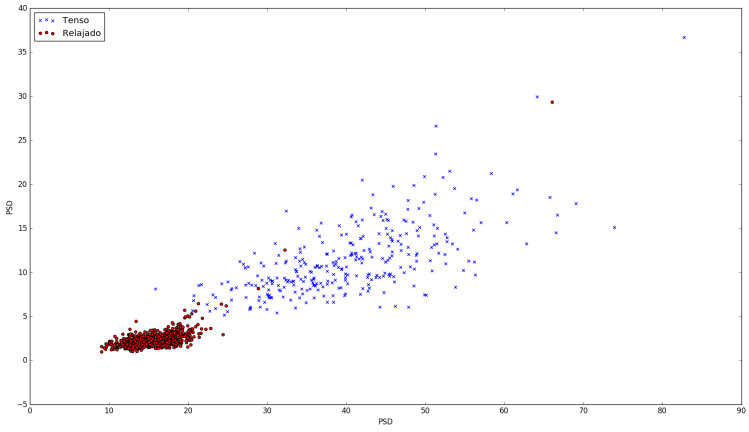

Figura 5.6: Gráfico de dispersión del vector de características EMG para la sesión del sujeto 1 ($512\,Hz$).

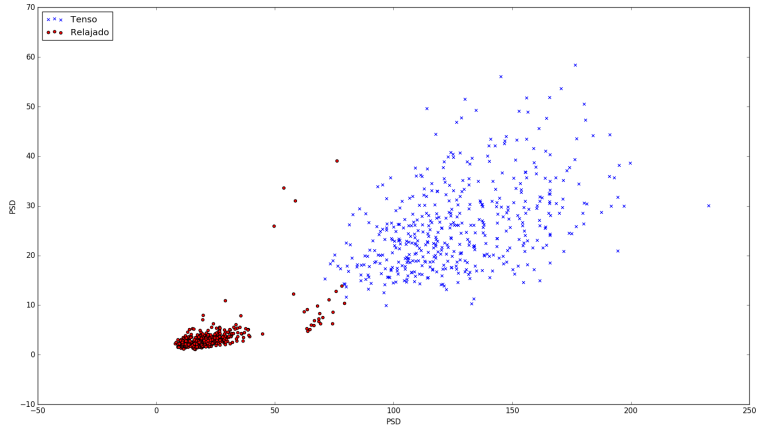

Figura 5.7: Gráfico de dispersión del vector de características EMG para la sesión del sujeto 3 ($512\,Hz$).

Las figuras 5.6 y 5.7 muestran, nuevamente, una clara separación entre las muestras tomadas durante el tiempo que se mantuvo el músculo relajado y tensado.

En definitiva, los niveles de precisión alcanzados utilizando las técnicas detalladas en este informe fueron muy elevados (95 % o más en todos los casos). Estos valores resultaron ser más que suficientes para el desarrollo de las simulaciones interactivas, que en sí también fueron diseñadas teniendo en cuenta que la precision de predicción prácticamente nunca podría ser del 100 %.

La precisión fue tan alta, que en el universo interactivo, nunca se lanzó un proyectil prematuramente. Es decir, los proyectiles fueron siempre lanzados cuando el usuario lo deseó y no por accidente.

5.3. SpO_2

Para medir la precision del método explicado en la sección 4.3.3, se grabó una sesión de aproximadamente un minuto para cuatro sujetos, y se comparó el promedio de BPM obtenido durante la duración del intervalo.

Sujeto	Promedio de BPM (Sensor)	Promedio de BPM (Calculado)
1	78,380	77,055
2	84,300	80,775
3	74,953	73,848
4	80,412	77,004

Cuadro 5.6: Promedios de BPM para las sesiones de SpO_2 medidas.

Como se puede ver en la tabla 5.6, los resultados obtenidos a través de nuestro método de calculo de BPM son similares a los calculados internamente por el sensor. La diferencia mas grande entre ambos resultados, para todas las sesiones, fue de aproximadamente 4 %. Ambos métodos para calcular BPM tienen

sus ventajas y desventajas: el que utiliza el sensor produce valores que fluctúan menos en el tiempo, pero representan un valor promediado sobre un intervalo de tiempo extenso, por lo que no responden rápidamente a cambios en el ritmo cardíaco. Por el otro lado, nuestro método responde mas rápidamente a los cambios, pero tiende a producir resultados que fluctúan con mayores magnitudes. Se graficaron los resultados de las sesiones de los sujetos 2 y 3 para poder comparar visualmente los resultados:

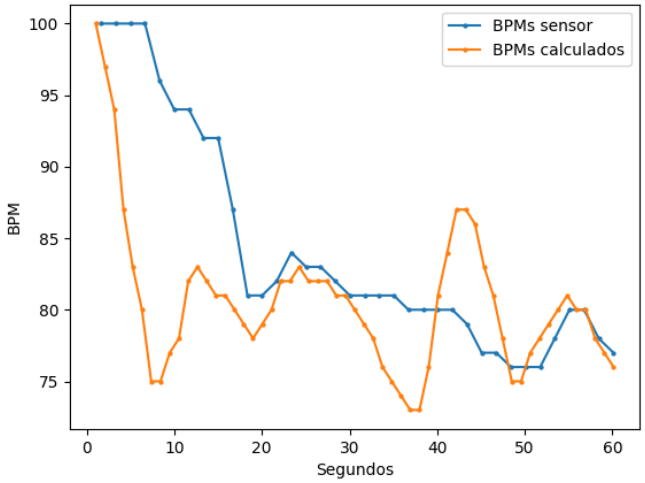

Figura 5.8: Gráfico de BPM a través del tiempo para la sesión del sujeto 2.

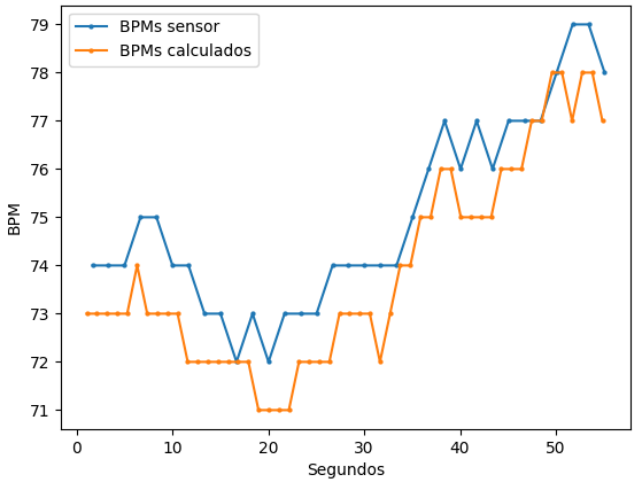

Figura 5.9: Gráfico de BPM a través del tiempo para la sesión del sujeto 3.

El gráfico 5.8 muestra como ambos métodos de medición tienden a producir

los mismos resultados, pero con algunas diferencias. El método del sensor tiende a producir los mismos valores producidos por nuestro método, pero cierto atraso. A la vez, nuestro método tiende a producir resultados que varían mas rápidamente con el tiempo. El gráfico 5.9 muestra una sesión donde ambos métodos produjeron resultados mucho mas similares. En el caso de éste proyecto, se prefirió nuestro método, ya que la simulación interactiva requiere de un tiempo de respuesta lo mas corto posible a los cambios de ritmo cardíaco del usuario. Es necesario remarcar que aunque las mediciones de BPM son discretas, se utilizaron gráficos de línea ya que éstos permitían interpretar la información más fácilmente. Los valores leídos están representados como puntos sobre las líneas.

Capítulo 6

Conclusiones

En este capítulo, se presentaran breves conclusiones sobre el proyecto realizado y también se discutirán posibles mejoras y pasos a seguir.

6.1. Conclusiones Generales

En el caso de EEG se logró darle un uso que mejora la interacción del usuario con el universo. El problema yace en que como las ondas *Alfa* varían en grandes cantidades de persona a persona, hay en personas que no funciona tan bien. Además, la precisión alcanzada, si bien es suficiente para el uso que se le dio, si se le quieren dar otros usos se debería mejorar. A su vez, el usuario debe quedarse muy quieto ya que los movimientos pueden causar que los electrodos generen lecturas incorrectas. Es posible que existan casos de personas en los que no funcionará correctamente ya que el cerebro es muy complejo. Por otro lado, si bien el procesamiento de señales es crucial, también lo es uso que se le da. Para compensar por la falta de precisión en el procesamiento de las señales, se debe escoger cuidadosamente que uso se les dará dentro del universo interactivo. Este es el caso de utilizar un umbral en el universo interactivo que usa EEG. Dicho umbral, aunque aumentó la latencia, mejoró la experiencia reduciendo los falsos positivos. Por estos motivos, se concluye que la utilización de señales EEG en un universo interactivo es realizable, pero dependerá del uso que se les de dentro del mismo. Se deberán tomar decisiones de compromiso para equilibrar la fidelidad del procesamiento, y su impacto en la simulación.

El uso del dispositivo EMG fue sin dudas el que mejores resultados arrojó. Es muy fácil de colocar y su precisión fue muy alta. Además, con tan solo unos minutos de entrenamiento ya es suficiente para obtener buenas predicciones. Se podrían utilizar dispositivos más ergonómicos que el utilizado para que el usuario se encuentre más confortable.

En el caso del dispositivo SpO_2, es totalmente utilizable en tiempo real ya que la latencia es mínima y no requiere entrenamiento. El usuario simplemente se lo coloca y puede interactuar con el universo. En este caso la dificultad más grande es crear universos lo suficientemente reales e inmersivos para lograr grandes variaciones en el ritmo cardíaco. El problema que se encontró en el dispositivo utilizado fue, que al utilizarse en un dedo, resultaba incómodo el uso del *mouse*. Se podría utilizar un dispositivo más ergonómico en la muñeca por ejemplo.

Por lo tanto, se concluye que es posible procesar bioseñales en tiempo real y utilizar la información en universos interactivos. La pregunta difícil aquí es: ¿Cómo hacerlo? Se debe hacer un análisis exhaustivo del estado que se quiera detectar para determinar que nivel mínimo de precisión se requiere y cuál sería el mejor uso dentro del universo. Además, el procesamiento debe ser lo suficientemente rápido para reaccionar rápidamente ante los cambios y introducir la menor cantidad de latencia en el universo interactivo.

En cuanto a la implementación del proyecto, se puede concluir que fue una buena decisión haber modularizado el código relacionado a la lectura y procesamiento de las bioseñales, ya que esto abre las posibilidades de utilizar las técnicas desarrolladas en otros ámbitos, con tan sólo incluir una librería.

6.2. Mejoras Posibles y Próximos Pasos

Se podría, por ejemplo, medir el nivel de fuerza ejercido utilizando un dispositivo EMG en lugar de que sea una medición binaria. El uso de bioseñales resulta mucho más interesante en el campo de la realidad virtual debido a que el usuario se encuentra totalmente inmerso en un universo. Aquí, cuanta mayor información del usuario se pueda obtener, mejor será la experiencia. Aparecerían otras limitaciones como el poder de procesamiento, ya que la realidad virtual demanda demasiado del mismo, o, el hecho de que el usuario se encuentra con un casco en la cabeza. Yendo aún más lejos, usando un dispositivo EEG, se podrían utilizar las ondas relacionadas a la corteza motora para detectar la voluntad de mover un brazo y simular dicho movimiento en un universo en realidad virtual. De esta forma, una persona que por alguna incapacidad no puede mover un brazo, lo pueda utilizar.

Este trabajo podría utilizarse como base para trabajos futuros relacionados. Las librerías generadas para procesar las señales se encuentran disponibles y podrían mejorarse para reconocer aún más estados del usuario. También se podría adaptar el trabajo para que funcione con un dispositivo de realidad virtual. Las aplicaciones del uso de bioseñales son muchas y creemos que existe mucho espacio de investigación e implementación. A medida que vayan surgiendo más y más universos interactivos que utilicen bioseñales, más tracción cobrará el campo y más se podrá avanzar. Creemos que este trabajo es un pequeño aporte en esta línea que tiene un apasionante futuro.

Apéndice A

Apéndice

A.1. Lista de Acrónimos

Se preservaron los acrónimo en inglés para facilitar la búsqueda de los mismos y para mantener consistencia con las publicaciones ya existentes. A continuación se presenta una lista de los acrónimos utilizados y su definición.

ACAT *Assistive context-aware toolkit*. 3, 6

BCI *Brain-computer Interface*. 8, 20

BPM *Beats per Minute*. 31, 48–50

CIL *Common Intermediate Language*. 32, 33, 39

ECS *Entity-Component System*. 33

EEG Electroencefalografía. 1–5, 8, 10, 16, 19, 20, 22–24, 26, 28, 31, 36, 38, 41, 51, 52

EKG Electrocardiograma. 9, 10, 23, 25

EMG Electromiografía. 1–7, 10, 16, 19, 20, 22, 23, 25, 27, 30, 31, 38, 45–47, 51, 52

FFT *Fourier Fast Transform*. 16, 36

GSR *Galvanic Skin Response*. 10

IL2CPP *Intermediate Language to C++*. 33

LDA *Linear Discriminant Analisis*. 17, 18, 41, 42, 45

MUAP *Motor Unit Action Potential*. 20

NASA *National Aeronautics and Space Administration*. 6

OSC *Open Sound Control System*. 25, 28

PSD *Power Spectral Density*. 17, 28–30

SDK *Software Development Kit*. 25, 28

SpO_2 Pulsioximetría. 1–5, 9, 10, 19, 21–23, 27, 31, 39, 40, 48, 51

SVM *Support Vector Machines*. 41, 42, 45

UDP *User Data Protocol*. 25

Bibliografía

[1] *Accord.NET Framework*. http://accord-framework.net/intro.html. feb 2017.

[2] *Hardware Specifications*. http://developer.choosemuse.com/hardware-firmware/hardware-specifications. feb 2016.

[3] *SHIELD-EKG-EMG-PA*. https://www.olimex.com/Products/Duino/Shields/SHIELD-EKG-EMG-PA/open-source-hardware. feb 2016.

[4] Chen, Chi Tsong: *Digital Signal Processing: Spectral Computation and Filter Design*. Oxford University Press, 2000, ISBN 978-0-195-13638-8.

[5] Duck Gun Park, Hee Chan Kim: *Muscleman: Wireless input device for a fighting action game based on the EMG signal and acceleration of the human forearm*. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.714.4700&rep=rep1&type=pdf.

[6] EsratJahan, TilottomaBarua, UmmeSalma: *An Overview On Heart Rate Monitoring And Pulse Oximeter System*. Department of EEE, Chittagong University of Engineering and Technology, páginas 148–149, 2014.

[7] Hawking, Stephen: *My Computer*. http://www.hawking.org.uk/the-computer.html. feb 2016.

[8] He, Bin: *Neural Engineering*. Kluwer Academic/Plenum Publishers, 2005, ISBN 0-306-48609-1.

[9] Jacobson, Daniel, Ruslan Meshenberg, Leslie Posada y Tom Richards: *Netflix Hack Day - Winter 2017*. http://techblog.netflix.com/2017/01/netflix-hack-day-winter-2017.html. feb 2016.

[10] Jawahar, Yousuf: *Design of an Infrared based Blood Oxygen Saturation and Heart Rate Monitoring Device*. Department of Electrical and Computer Engineering, McMaster University, páginas 3–4, 2009.

[11] Johnson, Steve: *Neurogaming: Interest growing in technology that picks players' brains*. http://www.mercurynews.com/2014/06/13/neurogaming-interest-growing-in-technology-that-picks-players-brains/. feb 2017.

[12] Kaniusas, Eugenijus: *Biomedical Signals and Sensors I: Linking Physiological Phenomena and Biosignals*. Springer, 2012, ISBN 978-3-642-24843-6.

[13] Kevin R. Wheeler, Charles C. Jorgensen: *Gestures as Input: Neuroelectric Joysticks and Keyboards.* Pervasive Computing, páginas 56–61, 2003. https://www.yumpu.com/en/document/view/33237239/gestures-as-input-neuroelectric-joysticks-and-keyboards.

[14] Lotte, Fabien: *A Tutorial on EEG Signal Processing Techniques for Mental State Recognition in Brain-Computer Interfaces.* páginas 2–3, 2014. https://hal.inria.fr/hal-01055103/document.

[15] Luca, Carlo J. De: *SURFACE ELECTROMYOGRAPHY: DETECTION AND RECORDING.* DelSys Incorporated, 2002. https://www.delsys.com/Attachments_pdf/WP_SEMGintro.pdf.

[16] MD, Y. Iyriboz, S. Powers EdD, J. Morrow MS, D. Ayers MS y G. Landry MS: *Accuracy of pulse oximetersin estimatingheartrate atrestand during exercise.* Exercise Physiology Laboratory, Louisiana State University, páginas 163–164, 1991.

[17] Naim Sidek, Ahmad Jazlan: *Measurement system to study the relationship between forearm EMG signals and wrist position at varied hand grip force.* https://www.researchgate.net/publication/261430545_Measurement_system_to_study_the_relationship_between_forearm_EMG_signals_and_wrist_position_at_varied_hand_grip_force.

[18] O'Haver, Tom: *Peak Finding and Measurement.* http://terpconnect.umd.edu/~toh/spectrum/PeakFindingandMeasurement.htm. feb 2016.

[19] OLIMEX Ltd.: *SHIELD-EKG-EMG bio-feedback shield USER'S MANUAL*, 2014. https://www.olimex.com/Products/Duino/Shields/SHIELD-EKG-EMG/resources/SHIELD-EKG-EMG.pdf.

[20] Peterson, Josh: *AN INTRODUCTION TO IL2CPP INTERNALS.* https://blogs.unity3d.com/2015/05/06/an-introduction-to-ilcpp-internals. feb 2016.

[21] Schneider, Jeff: *Cross Validation.* https://www.cs.cmu.edu/~schneide/tut5/node42.html. feb 2016.

[22] Semmlow, John L. y Benjamin Griffel: *Biosignal and Medical Image Processing.* CRC Press, 3ª edición, 2014, ISBN 978-1-4665-6737-5.