SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

Tecnológico Nacional de México

Centro Nacional de Investigación y Desarrollo Tecnológico

Tesis de Maestría

Método para integrar y sincronizar datos EEG y multimedia para su aplicación en el proceso de evaluación de la experiencia del usuario

presentada por

Ing. David Eduardo Fouilloux Quiroz

como requisito para la obtención del grado de

Maestro en Ciencias de la Computación

Director de tesis

Dr. Juan Gabriel González Serna

Codirector de tesis

Dr. Máximo López Sánchez

Cuernavaca, Morelos, México. Julio de 2018.

cenidet
Centro Nacional de Investigación y Desarrollo Tecnológico

SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLOGICO NACIONAL DE MÉXICO

Centro Nacional de Investigación y Desarrollo Tecnológico

Cuernavaca, Morelos a 21 de junio del 2018
OFICIO No. DCC/196/2018

Asunto: Aceptación de documento de tesis

DR. GERARDO V. GUERRERO RAMÍREZ
SUBDIRECTOR ACADÉMICO
PRESENTE

Por este conducto, los integrantes de Comité Tutorial del **Ing. David Eduardo Fouilloux Quiroz**, con número de control M16CE004, de la Maestría en Ciencias de la Computación, le informamos que hemos revisado el trabajo de tesis profesional titulado "**Método para integrar y sincronizar datos EEG y multimedia para su aplicación en el proceso de evaluación de la experiencia del usuario**" y hemos encontrado que se han realizado todas las correcciones y observaciones que se le indicaron, por lo que hemos acordado aceptar el documento de tesis y le solicitamos la autorización de impresión definitiva.

DIRECTOR DE TESIS

Dr. Juan Gabriel González Serna
Doctor en Ciencias de la Computación
7820329

CO-DIRECTOR DE TESIS

Dr. Máximo López Sánchez
Doctor en Ciencias de la Computación
7498547

REVISOR 1

Dr. Noé Alejandro Castro Sánchez
Doctor en Ciencias de la Computación
08701806

REVISOR 2

Dra. Andrea Magadán Salazar
Doctorado en Ciencias Computacionales
10654097

C.p. M.T.I. María Elena Gómez Torres - Jefa del Departamento de Servicios Escolares.
Estudiante
Expediente

NACS/lmz

SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLOGICO NACIONAL DE MEXICO
Centro Nacional de Investigación y Desarrollo Tecnológico

Cuernavaca, Mor., 25 de junio de 2018
OFICIO No. SAC/289/2018

Asunto: Autorización de impresión de tesis

ING. DAVID EDUARDO FOUILOUX QUIROZ
CANDIDATO AL GRADO DE MAESTRO EN CIENCIAS DE LA COMPUTACIÓN
P R E S E N T E

Por este conducto, tengo el agrado de comunicarle que el Comité Tutorial asignado a su trabajo de tesis titulado **"Método para integrar y sincronizar datos EEG y multimedia para su aplicación en el proceso de evaluación de la experiencia del usuario"**, ha informado a esta Subdirección Académica, que están de acuerdo con el trabajo presentado. Por lo anterior, se le autoriza a que proceda con la impresión definitiva de su trabajo de tesis.

Esperando que el logro del mismo sea acorde con sus aspiraciones profesionales, reciba un cordial saludo.

A T E N T A M E N T E
EXCELENCIA EN EDUCACIÓN TECNOLÓGICA®
"CONOCIMIENTO Y TECNOLOGÍA AL SERVICIO DE MÉXICO"

SEP TecNM
CENTRO NACIONAL DE INVESTIGACIÓN Y DESARROLLO TECNOLÓGICO
SUBDIRECCIÓN ACADÉMICA

DR. GERARDO VICENTE GUERRERO RAMÍREZ
SUBDIRECTOR ACADÉMICO

C.p. M.T.I. María Elena Gómez Torres .- Jefa del Departamento de Servicios Escolares.
Expediente

GVGR/mcr

DEDICATORIAS

A Coquis, Adriana, JC y Sophie que siempre me han apoyado en el transcurso de mi vida.

A toda mi familia que siempre me han impulsando con palabras de aliento y experiencias maravillosas.

A todos mis amigos y compañeros de CENIDET que siempre hemos estado juntos en esta experiencia, en las buenas y en las malas. A la generación 2016-2018 del departamento de Ciencias de la Computación, de las tres líneas de investigación Sistemas Distribuidos, Ingenieria de Software e Inteligencia Artificial.

AGRADECIMIENTOS

En esta sección quiero agradecer a todos los que me han apoyado en el transcurso del desarrollo de esta tesis.

Agradezco al Consejo Nacional de Ciencia y Tecnología (CONACYT) por su apoyo económico, ya que con él fue posible el desarrollo de esta tesis.

Agradezco al Centro Nacional de Investigación y Desarrollo Tecnológico (CENIDET) y a todo su personal por la oportunidad que realizar mis estudios de maestría en Ciencias de la Computación.

Agradezco también a mi Director de Tesis el Dr. Juan Gabriel González Serna por el tiempo empleado para guiarme y apoyarme durante todo el periodo de mi estancia en CENIDET. A mi Codirector, el Dr. Máximo López Sánchez, y mis revisores, el Dr. Noé Alejandro Castro Sánchez y la Dra. Andrea Magadán Salazar, quienes me regalaron su tiempo de revisiones, correcciones, consejos y sugerencias acerca del trabajo desarrollado.

Resumen

La evaluación de la experiencia del usuario (UX) permite conocer las verdaderas necesidades del usuario en base a su experiencia en la prueba de un prototipo de software. Esta evaluación permite detectar características faltantes o problemas de usabilidad de la herramienta de software.

En el transcurso de la UX, se recomienda utilizar dispositivos que registran datos biométricos para recuperar información relacionada con el subconsciente de un usuario, permitiendo enriquecer sus observaciones finales.

Con el fin de aprovechar esta tecnología, la presente investigación se desarrolló con el objetivo de crear una alternativa de software más accesible, a comparación de los productos comerciales actuales que radican en esta área de oportunidad. Herramientas actuales de código abierto o de acceso gratuito cuentan con restricciones con relación al tipo de dispositivo soportado, al ámbito hacia el que se encuentran dirigidos, y la arquitectura poco extensible con la que se desarrollaron.

Para ello se diseñó una arquitectura modular que cubre el proceso de captura y almacenamiento de información proveniente de múltiples dispositivos. Utilizando la librería LabStreamingLayer proporcionada gratuitamente por el Centro Swartz de Neurociencias Computacionales de la Universidad de San Diego, California, se desarrolló un conjunto de módulos independientes que llevan a cabo el proceso de captura de información de los dispositivos, detectan y almacenan cada flujo de información, según su tipo de procesamiento, y llevan a cabo un procesamiento adicional en datos que así lo requieran.

Los resultados obtenidos utilizando una técnica de sincronización de estampas de tiempo de alta precisión agregadas a las muestras de cada flujo de datos, proporcionados por la librería antes mencionada, demuestran que una máquina moderna con suficientes recursos tiene la capacidad de capturar información de múltiples dispositivos sin generar un desfase significativo.

Abstract

User experience (UX) evaluation allows to learn the true necessities of a user based on his experience during a testing of a software prototype. This evaluation allows to detect missing features or usability problems of the software tool.

During UX, the use of biometric devices retrieves information related to the user psyche, thus enriching their final observations.

In order to take advantage of this technology, this research was developed with the aim of creating a more accessible software alternative, compared to the current commercial products that reside in this area of opportunity. Current open source or free access tools have restrictions regarding the type of device supported, the scope to which they are directed, and the small extensible architecture in which they were developed.

To do this, a modular architecture was designed to cover the process of capturing and storing information from multiple devices. Using the LabStreamingLayer library provided freely by the Swartz Center of Computational Neurosciences of the University of San Diego, California, a set of independent modules were developed to carry out the process of capturing information from the devices, detecting and storing each information flow according to their type of processing, and carry out additional processing on data that requires it.

The results obtained using a synchronization technique of high precision timestamps added to the samples of each data flow, provided by the aforementioned library, demonstrate that any modern machine with sufficient resources has the capacity to capture information from multiple devices without generating significant lag.

Tabla de contenidos

Lista de figuras

Lista de tablas

Capítulo 1. Introducción

En esta sección se describen los antecedentes, objetivos y características del proyecto de tesis.

La computación se ha convertido en algo que se puede encontrar en la vida común, desde estaciones de cómputo dedicados hasta teléfonos celulares inteligentes. Este tipo de tecnología se encuentra fuertemente ligada a las personas hasta un punto en el que dependen de ella.

Debido a esto, el mercado de desarrollo de software ha incrementado drásticamente, lo cual resulta en la creación acelerada de aplicaciones de computadora que abarcan gran variedad de ámbitos.

Una desventaja de este crecimiento es la negligencia de evaluación de aspectos en la calidad del software causada por la competencia continua en la creación de aplicaciones innovadoras. Al involucrar fechas límite cortas para obtener una ventaja con otras empresas, se resta importancia al proceso de evaluación para acelerar el lanzamiento de las aplicaciones. Esto resulta en errores de distintos tipos al ser utilizado por los clientes, quienes pueden llegar a generar mala reputación al dejar en descubierto la falta de importancia provista en esta etapa.

La evaluación de la experiencia del usuario permite conocer las verdaderas necesidades del usuario en base a la prueba de un prototipo de una herramienta de software. El uso de este prototipo permite detectar requisitos o características faltantes o sobrantes, además de indicar en qué área es necesario enfocarse y qué futuros problemas se le presentarán al usuario final.

El principal aspecto a evaluar de la experiencia del usuario con una herramienta de software es su usabilidad, la cual se relaciona con la calidad de la interfaz de usuario y sus componentes, por ejemplo: ¿qué tan fácil es realizar una tarea específica para un nuevo usuario? o ¿qué tan fácil es que un usuario cometa un error y se le permita recuperarse?

Dispositivos provenientes del campo de la neurociencia y la pupilometría permiten obtener información adicional que revelan detalles inconscientes del usuario. Esta información es posteriormente analizada por especialistas expertos, señalando puntos clave durante la evaluación donde se destaca el evento o patrón y su significado.

El uso paralelo de distintos dispositivos dedicados a obtener información de una persona y su ambiente necesita de sistemas especializados en la captura simultánea de dicha información (Figura 1-1).

Figura 1-1. Ejemplo de ambiente de evaluación de software con dispositivos EEG y multimedia.

1.1 Antecedentes

La tesis de maestría "Metodología para Evaluación de SRSC Centrada en el Usuario, Basada en Características de Efectividad, Confianza y Satisfacción Mediante Interfaces Multimodales sobre Dispositivos Móviles Multisensoriales" (Arana Llanes, 2014), tiene como objetivo establecer una metodología que permita utilizar un conjunto de técnicas de evaluación de la experiencia centrada en el usuario para depurar aspectos negativos de una persona evaluada para validar su nivel de efectividad, confianza y satisfacción al utilizar el sistema objetivo.

La tesis de doctorado "Evaluación centrada en el usuario de sistemas de recomendación sensibles al contexto: efecto de interfaces multimodales interactivas y esquemas de explicación en la experiencia del usuario" (Alejandres Sanchez, 2016), realiza una investigación y análisis del conjunto de componentes de la experiencia del usuario propuesto por Bart Knijnenburg, enfocado en sistemas de recomendación

consciente del contexto, con el fin de evaluar medidas poco exploradas en dichas evaluaciones.

1.2 Planteamiento del problema

Para analizar información proveniente de distintas fuentes de datos de un mismo contexto es necesario identificar su posición en el tiempo de grabación, el cual es necesario capturar con un alto grado de sincronización. Un flujo de información sincronizado requiere de una generación de muestras constantes y consistentes, donde cada muestra se genera a un mismo ritmo con un mismo tipo de contenido.

Una arquitectura física que captura información de múltiples dispositivos multimedia, sensores biométricos y ambientales se enfrenta con múltiples problemas. Por ejemplo:

- Cada dispositivo debe de contar con una vía de comunicación independiente al sistema.
- Dependiendo del dispositivo, existen fuentes de información que necesitan de un procesamiento para convertir señales complejas a un formato que la computadora entienda. Un ejemplo es el flujo de video de una cámara.
- Utilizar medidas de tiempo de alta precisión.
- Capturar la información simultáneamente de distintos dispositivos, e identificar su posición temporal.
- Mantener un control en el flujo de información de cada dispositivo.

Además de esto, es necesario tomar en cuenta la cantidad de recursos necesarios para procesar la información recuperada de los dispositivos, la problemática del cuello de botella, y el software necesario para acceder a la información generada por cada dispositivo (Figura 1-2).

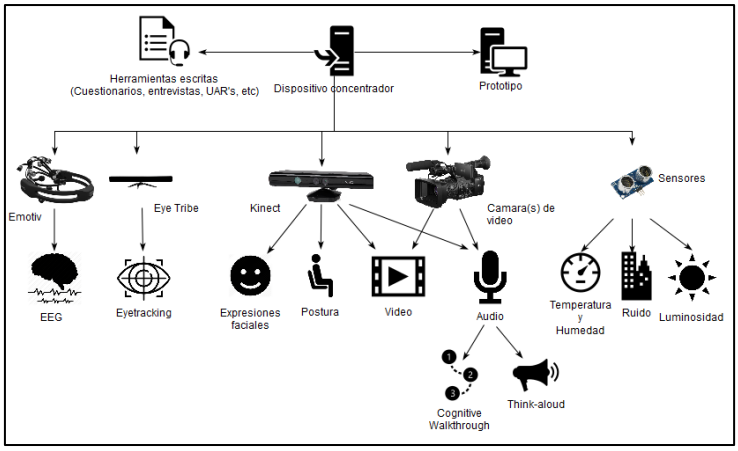


Figura 1-2. Diagrama conceptual de un ejemplo de elementos de un ambiente de pruebas con dispositivos de EEG, de seguimiento ocular y de grabación multimedia. También se muestran los distintos tipos de información que se puede recuperar de cada dispositivo (Arana Llanes, 2014).

Herramientas de software con esta funcionalidad se encuentran con distintas limitaciones, entre ellas:

- Aplicaciones de código abierto soportan un número limitado de dispositivos, ya sea por falta de actualización, dificultad para integrar nuevas tecnologías o por enfoque a ámbitos específicos.
- Aplicaciones comerciales restringen su uso por medio de la cantidad de dispositivos, duración de la grabación, o en casos más particulares, el uso de arquitecturas poco convencionales o más especializados.

Dentro de este contexto, el presente trabajo de investigación se realiza con el fin de diseñar un método de grabación sincronizada de múltiples fuentes de datos paralelas provenientes de diferentes dispositivos de grabación, con las cuales se pueda obtener información para realizar la evaluación de la experiencia del usuario de una herramienta de software.

1.3 Objetivos

1.3.1 Objetivo general

Diseñar un método para procesar, integrar y sincronizar la información generada a partir de distintos dispositivos de captura audio-visual, ambiental y cognitiva, para facilitar el análisis del comportamiento y el contexto introspectivo, en las evidencias obtenidas dentro de las pruebas de evaluación de la experiencia del usuario de una herramienta de software.

1.3.2 Objetivos específicos

- Integrar e implementar un ambiente automatizado para la grabación de la experiencia del usuario, por medio de múltiples fuentes de datos generados por distintos dispositivos en un mismo lapso de tiempo.
- Diseñar y evaluar la técnica de integración y sincronización de datos.

1.4 Justificación

Los dispositivos para registrar señales neurofisiológicas (EEG) y los dispositivos de grabación multimedia, así como tecnologías para la captura de movimiento ocular, permiten obtener información complementaria para analizar el comportamiento y experiencia que experimenta un usuario cuando utiliza un sistema de software.

Por ejemplo, dispositivos que capturan la actividad cerebral permiten analizar la manera en la que un evento afecta a una persona, sin que sea necesario mirar sus anotaciones. También, permiten identificar si dicha persona compartió su experiencia, o si omitió detalles importantes que pueden ayudar a un investigador a obtener mejores conclusiones (Arana Llanes, 2014).

Cada dispositivo de grabación cuenta con una especialización en cierto tipo de datos, ya sea de video, audio u otro tipo de señales. Sin embargo, al grabar información de múltiples dispositivos es necesario sincronizarla con una misma referencia de tiempo.

1.5 Alcances y limitaciones

1.5.1 Alcances del proyecto

- La metodología se centra en los métodos y técnicas de sincronización e integración de flujos de datos.
- La visualización paralela de los archivos multimedia sincronizados se despliega en una interfaz gráfica de usuario.
- Se implementaron módulos de liga de información a un mínimo de 5 dispositivos.

1.5.2 Limitaciones del proyecto

- No se implementaron métodos de refinación y filtrado de la información obtenida, a través de dispositivos de señales EEG, para la eliminación de artefactos o ruido externo. Sólo se captura y procesa la información en "crudo".
- No se asegura que los archivos resultantes son compatibles con herramientas de reproducción de archivos multimedia.
- No se desarrolla una metodología para evaluar la experiencia del usuario, ya que se utiliza la metodología desarrollada en (Arana Llanes, 2014).

1.6 Estructura del documento

El presente documento se organiza con los siguientes capítulos:

- En el segundo capítulo, "Marco teórico", se abordan los conceptos y términos más comunes o relevantes del proyecto.
- En el tercer capítulo, "Estado del arte", se presenta la información de técnicas de métodos de captura y sincronización de datos multimedia, biométrico y/o ambiental, así como la información de sistemas, productos y patentes que apliquen técnicas de métodos de captura y sincronización de dato multimedia, biométrico y/o ambiental.
- El cuarto capítulo, "Análisis de técnicas de sincronización", se presenta un análisis de las técnicas principales de sincronización, y experimentos relacionados con ellas.

- En el quinto capítulo, “Análisis de dispositivos multimedia”, se describen las características de los dispositivos multimedia considerados en el ambiente de evaluación de la experiencia del usuario.
- En el sexto capítulo, “Diseño de la técnica de sincronización”, presenta el diseño del prototipo que sirvió para probar la técnica de sincronización elegida y su papel en un sistema de captura de información paralela.
- En el séptimo capítulo, “Implementación de la técnica”, se describe la manera en la que se implementó la técnica de sincronización para el sistema de captura de información paralela propuesta en este documento.
- En el octavo capítulo, “Pruebas y resultados”, se muestran los experimentos realizados y los resultados obtenidos a partir de los mismos en el sistema de grabación paralela desarrollado.
- En el noveno capítulo “Conclusiones”, se presentan las conclusiones del proyecto de tesis y los trabajos futuros del mismo.
- En el décimo capítulo “Referencias”, se presenta la información bibliográfica de las referencias utilizadas en este documento de tesis.

Capítulo 2. Marco teórico

En este capítulo se presentan los conceptos fundamentales en los cuales se basa el contenido del documento.

- **Interacción Humano computadora (HCI)**

El termino Interacción Humano Computadora es utilizado en el estudio del diseño y el uso de tecnología computacional enfocado en las interfaces aplicadas en la comunicación entre una persona y una computadora. Este enfoque implica cualquier herramienta que tenga como objetivo el dialogo abierto entre las dos entidades (Jacko, 2012).

- **Experiencia del usuario (UX)**

El estándar internacional de las ergonomías de la Interacción Humano Computadora, o ISO 9241-210, define la experiencia del usuario como "las percepciones y respuestas de una persona en respuesta al uso normal o anticipado de un producto, sistema o servicio". Esto implica las emociones, creencias, preferencias, percepciones, respuestas físicas y psicológicas, conductas y logros de un usuario antes, mientras y después de utilizar el elemento que evalúa (International Organization for Standardization, 2010).

Como se menciona brevemente en el documento del estándar ISO 9241-210, el término de usabilidad aplica en algunos aspectos de la experiencia del usuario, ya que al tener un criterio de usabilidad es posible inspeccionar elementos de la experiencia del usuario desde otro punto de vista (International Organization for Standardization, 2010).

- **Electroencefalografía (EEG)**

La encefalografía es un método de monitorización de registros electrofisiológicos, utilizado para grabar actividad eléctrica del cerebro.

Un dispositivo para lectura de señales EEG no invasivo se compone generalmente de electrodos colocados alrededor del cuero cabelludo de una persona, los cuales cuentan con un medidor de oscilaciones neuronales, que detectan la actividad del potencial eléctrico de las neuronas cercanas al electrodo (Niedermeyer E., 2004).

- **Sincronización de flujos de datos**

La sincronización de flujos de datos es el proceso de generación de consistencia y coherencia a través de distintos flujos de datos, manteniendo una armonización e integridad de dichos datos sobre un lapso de tiempo (Agarwal, Starobinski, & Trachtenberg, 2002).

Aplicando esto, a un sistema multimedia, se refiere a la generación de relaciones temporales entre los medios que maneja.

- **Integración de datos EEG**

Se han desarrollado distintas formas para integrar y sincronizar información de este tipo con flujos de datos multimedia; sin embargo, sólo se han utilizado combinando los datos de diversas fuentes de información mientras se graban en tiempo real o utilizando algoritmos robustos que intentan sincronizar varias fuentes de información en base de coincidencias, como los potenciales relacionados a eventos (ERP) que son utilizados para inducir actividad en el cerebro por medio de acciones no invasivas (Lopez-Calderon & Luck, 2014).

Gran cantidad de algoritmos se encuentran desarrollados en lenguajes de programación especializados para manejar operaciones matemáticas complejas, como MATLAB. La desventaja de esta forma de implementación es la falta de portabilidad o disponibilidad para comunicarse con otro tipo de arquitecturas que no se encuentran soportadas por el lenguaje, así como el uso de dispositivos cuyos recursos deben ser utilizados óptimamente en servicios de la nube (Makeig, Delorme, & Brunner, 2013).

- **Contador de alta precisión**

Un contador de alta precisión es una medida de tiempo independiente y no sincronizada con referencias de tiempo externas al de un equipo de cómputo. Se utilizan principalmente en cálculos de rendimiento de la computadora o de la red, las cuales incluyen el tiempo de respuesta computacional, el transporte y la latencia, así como la ejecución del perfilamiento de código (Microsoft, 2017).

Un ejemplo de uso es su aplicación en la sincronización de tiempo de una red local con el protocolo NTP o Network Time Protocol.

- **Estampa de tiempo**

Una estampa de tiempo es una secuencia de caracteres o información codificada que identifican el momento en el que ocurrió un evento específico, el cual puede ser tan preciso como una fracción de segundo.

En la computación, comúnmente, se le conoce como estampa de tiempo a los metadatos incluidos en un archivo para indicar momentos específicos como la fecha y hora de creación, de modificación o de último acceso. El estándar principal de las estampas de tiempo es el Reloj Universal Coordinado (UTC), aunque pueden utilizarse otros.

- **Reloj Universal Coordinado (UTC)**

El Reloj Universal Coordinado es el estándar de tiempo más importante utilizado como base para el tiempo civil, ya que es la combinación de la escala Internacional de Tiempo Atómico (TAI) y el Tiempo Universal UT1, los cuales mantienen unidades escalares de segundos que permiten identificar una división temporal de segundos, minutos, horas y días.

Una desventaja del uso de este estándar en medidas temporales es el problema del segundo intercalar o adicional, el cual ocurre debido a la dificultad de localizar la cantidad de segundos exactos contenidos en un día solar. La escala UT1 toma como medida de día solar un total de 86400 segundos, lo cual entra en conflicto con el objetivo del estándar UTC para representar el tiempo de un día.

El estándar UTC se encuentra sincronizado con la escala UT1 con una precisión de 1 segundo, mientras que su diferencia con la escala TAI, la cual representa la unidad de segundos en el geoide, es de alrededor de 35 segundos (Madore, 2010).

- **Metadatos**

Los metadatos son datos que describen a otro tipo de datos, comúnmente utilizados para proveer de información importante acerca de un flujo de información en especifico, por ejemplo, datos de autor de un libro, la estructura de un documento o datos necesarios para la administración de recursos, como las estampas de tiempo.

Los metadatos permiten facilitar la búsqueda y recuperación de información en base a un conjunto de datos específicos, acelera la administracion de archivos en base a sus características estructurales y documentales, además de permitir la interoperabilididad por el uso de estandares ya definidos (Comité Coordinador Permanente de la Infraestructura de Datos Espaciales del Perú, 2017).

- **Medio de comunicación**

Un medio de comunicación u objeto medio es la forma en la que se transmite un flujo de información de un punto a otro, y cómo se debe procesar para realizar para que el receptor la pueda utilizar.

Si este medio es un flujo de datos que contiene relaciones temporales entre sus unidades consecutivas de información, se dice que es dependiente del tiempo, y si las duraciones de dichas unidades son equitativas se le da el nombre de medios continuos. Un ejemplo de medios continuos son el video o el audio.

Ejemplos de medios independientes del tiempo son los medios representados por objetos estáticos como un texto o una gráfica (Blakowski & Steinmetz, 1996).

- **Sistema multimedia**

Un sistema multimedia es un sistema o aplicación que soporta el procesamiento integrado de varios tipos de medios, con la condición de integrar al menos un medio dependiente del tiempo (Boronat, Lloret, & García, 2009).

Capítulo 3. Estado del arte

En esta sección se presentan los artículos que investigan las principales técnicas de sincronización e integración de flujos de datos EEG con datos multimedia para generar un paquete o archivo virtual independiente.

3.1 Estado del arte

Entre las técnicas de sincronización se observa que no existen muchos artículos publicados, ya que esta área de conocimiento es principalmente empleada en el área de psicología y ámbitos relacionados con la salud mental.

3.1.1 Synchronizing physiological data and video in a telemedicine application: A multimedia approach (Lambert, Hachicha, Ahmed, Pinna, & Garda, 2015)

Se propone utilizar una implementación basada en el formato multimedia Matroska (MKV) para almacenar y sincronizar los flujos de datos multimedia por medio del uso de etiquetas de tiempo.

Dicho formato es un estándar abierto que ofrece una forma de contenedor que almacena distintos flujos de datos multimedia comprimidos. El flujo de datos obtenido por medio del dispositivo de señales EEG es empaquetado en varias partes y almacenado con una estampa de tiempo.

Esta técnica propone utilizar un sistema de compresión de dichos flujos en un solo paquete que permita almacenar flujos de datos etiquetados con estampas de tiempo. Este proceso se realiza después de obtener los datos necesarios para hacer el empaquetamiento, por lo que es necesario implementar una función para generar las etiquetas síncronamente con una librería aparte.

3.1.2 A Data Hiding Technique to Synchronously Embed Physiological Signals in H.264/AVC Encoded Video for Medicine Healthcare (Peña, Ávila, Muñoz, & Lavariega, 2015)

Se implementa una técnica en la que se oculta la lectura de datos electroencefalográficos de un bebé dentro de la grabación de video en tiempo real (Figura 3-1).

Para realizarlo se oculta la información al colocarla en posiciones específicas dentro de la grabación de video en forma de píxeles de color gris, donde su posición es interpretada como un valor en específico. Cada píxel se coloca en una sección del video no visible para una persona.

Esta técnica permite incluir los datos EEG dentro de las imágenes de manera imperceptible para un observador, lo cual se logra utilizando el concepto de los videos para esconder los datos por medio de los píxeles de cada fotograma del video.

Para explicar esta técnica, se define un video como una secuencia ordenada de imágenes que representan un momento o animación. A su vez, una imagen de computadora está conformado por un conjunto de píxeles de colores que representan una imagen cualquiera, cuyo número se ha incrementado exponencialmente con el paso del tiempo. En la actualidad cualquier video cuya fuente no ha sido modificada deja un área de píxeles "vacío" o con un color nulo, el cual se relaciona con el "aspecto" o el área de grabación del dispositivo.

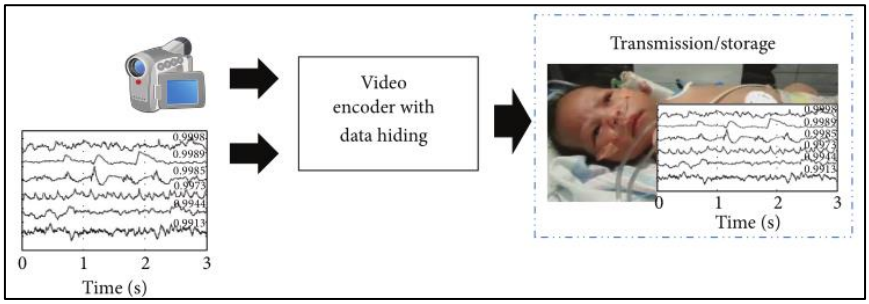


Figura 3-1. Representación de la técnica de ocultación de datos, diseñada por los autores del artículo (Peña, Ávila, Muñoz, & Lavariega, 2015).

3.1.3 EVS: An EEG-Video Synchronization System Using High Performance Counter (Zhou, Zheng, H., & Yang, 2013)

Se implementa un sistema que sincroniza grabaciones de video y un flujo de datos EEG con un contador de alta precisión. Este contador se genera por medio de la aplicación del método de calibración y el contador de alta precisión contenido por cualquier computadora moderna para obtener una precisión satisfactoria.

Para ello se recurrió a enviar una señal de código corto a todas las computadoras de la red para sincronizar las distintas computadoras que reciben cada flujo de datos.

Esta técnica utiliza estampas de conteo con el número especificado por un algoritmo para etiquetar los flujos de datos. Los números de estas estampas son generados a partir del número de pulsaciones que emite un contador de computadora, el cual se asemeja

vagamente a un reloj atómico, ya que no es tan exacto, pero es confiable en sus mediciones.

3.1.4 Multiple frequency audio signal communication as a mechanism for neurophysiology and video data synchronization (C. Topper, N. Burke, & Porter Maurer, 2014)

Se implementa un sistema que permite comparar la aplicación de pulsos de señales visuales y auditivas en una grabación para sincronizar los flujos de datos provenientes de una cámara de video y un dispositivo de señales EEG. Se consideran a las señales generadas por una fuente luminosa como convencionales debido a que es una técnica comúnmente utilizada para sincronizar dispositivos sin conexión.

Para realizar la comparación se utilizan dos dispositivos: uno genera una señal luminosa de corto tiempo para ser captada por la cámara y la percepción del sujeto de prueba cada cierto tiempo, mientras que el otro dispositivo genera una señal auditiva con una frecuencia única a través del micrófono de la cámara de video para ser escuchado por el sujeto (Figura 3-2).

Como resultado se denota la importancia de considerar la frecuencia de grabación de los dispositivos, el retraso entre cada grabación, y cualquier pérdida de datos por este suceso, ya que puede afectar los resultados de sincronización por señales visuales. Por otra parte, el método de sincronización por señales auditivas difícilmente sufre de errores provocados por estos eventos ya que cada ciclo de pulsos contiene un cuadro aleatorio de frecuencia alta en conjunto de una señal "contadora" única de frecuencia baja.

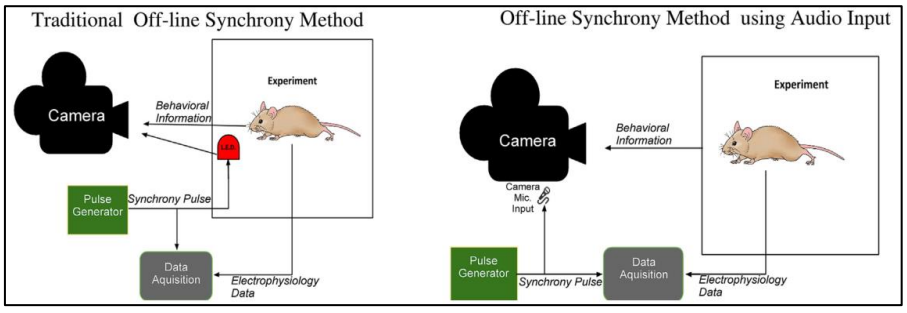


Figura 3-2. Representación de las técnicas de sincronización por señales ambientales, diseñada por los autores del artículo (C. Topper, N. Burke, & Porter Maurer, 2014).

3.1.5 Tabla comparativa de técnicas analizadas del estado del arte

A continuación se presenta una tabla comparativa que muestra las características de cada técnica analizada en el estado de la práctica (Tabla 1).

Tabla 1. Tabla comparativa de técnicas analizadas del estado del arte.

Técnicas de sincronización	Tipo de técnica	Modificación de datos multimedia	Sincronización en tiempo real	Sincronización de dispositivos sin conexión	Tipo de señales sincronizadas
Estampas de tiempo (Synchronizing physiological data and video in a telemedicine application: A multimedia approach)	Integración de metadatos	✓			Archivos de contenedor multimedia
Incrustar datos ocultos en video (A Data Hiding Technique to Synchronously Embed Physiological Signals in H.264/AVC Encoded Video for Medicine Healthcare)	Integración directa de datos	✓	✓		Video EEG
Algoritmo con contador de alta precisión (EVS: An EEG-Video Synchronization System Using High Performance Counter)	Algoritmo, Integración de metadatos	✓	✓		Video EEG
Señal de luminosidad (Multiple frequency audio signal communication as a mechanism for neurophysiology and video data synchronization)	Algoritmo, Señal física			✓	Video EEG
Señal auditiva (Multiple frequency audio signal communication as a mechanism for neurophysiology and video data synchronization)	Algoritmo, Señal física			✓	Audio Video con audio EEG

3.2 Estado de la practica

Una de las instituciones más involucradas en este tipo de investigación es el Centro Swartz de Neurociencias Computacionales de la Universidad de San Diego, California, que ha realizado investigaciones y herramientas que permiten identificar y analizar las señales cerebrales vinculadas con las actividades y las emociones que puede representar un patrón. Herramientas como el EEGLAB, MoBILAB y procesadores de señales EEG permiten manipular y visualizar la información obtenida por distintos dispositivos especializados para ser analizada rigurosamente, con evidencia simultáneamente generada (University of California San Diego, 2010).

A continuación se describen los artículos que explican la manera en la que se implementaron o utilizaron distintas herramientas de software para realizar la integración y sincronización de flujos de datos EEG.

3.2.1 A short review and primer on online processing of multiple signal sources in human computer interaction applications (Torniainen & Henelius, 2016)

En este artículo se describe el estado del arte hasta septiembre de 2016 en sistemas en línea de procesamiento en tiempo real de flujos de datos.

Existen aplicaciones llamadas "*Stream Processing Systems*" (también conocidos como SPS) que toman como prioridad el procesamiento de datos en tiempo real de los flujos de datos de gran volumen y velocidad provenientes de dispositivos cognitivos, para la extracción y evaluación de los datos crudos. Una desventaja de este tipo de sistema es que dejan el almacenamiento de dicha información como un objetivo no prioritario.

Librerías como *Lab Streaming Layer* son utilizadas para facilitar la recepción, procesamiento y sincronización de flujos de datos de dispositivos distribuidos en una red local, convirtiendo la información a un formato entendible por los *SPS*. Mientras tanto, otras herramientas permiten construir un servicio centralizado que distribuye la carga del procesamiento entre distintos nodos para sincronizarlos y manipularlos en el dispositivo principal, como el *framework MIDAS*.

Sistemas *SPS* actuales se enfocan a soportar una cantidad limitada de dispositivos especializados, además de agregar limitantes al software o a sus licencias de uso para restringir su expansión a medios emergentes.

3.2.2 EEGLAB - an Open Source Matlab Toolbox for Electrophysiological Research (Makeig, Delorme, & Brunner, 2013)

EGGLAB es una "caja" o un conjunto de herramientas de MATLAB de código abierto que se especializa en obtención, procesamiento y análisis de datos generados por un dispositivo de señales EEG (Figura 3-3).

Permite sincronizar los flujos de datos de todos los sensores del dispositivo EEG para permitir la creación de marcadores de evento, la visualización de datos y el análisis independiente de cada componente detectado.

Esta herramienta ha servido como base para construir aplicaciones de manipulación de información EEG de distintos sensores.

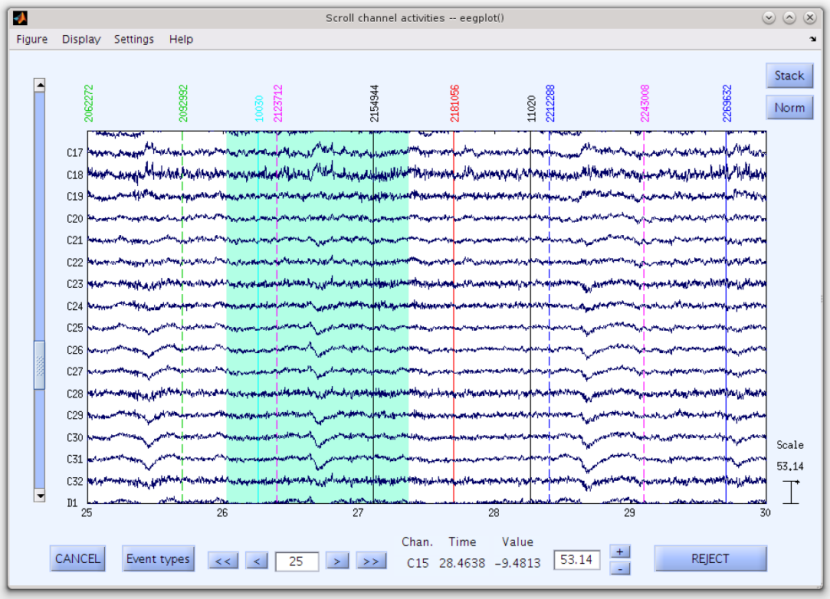


Figura 3-3. Interfaz gráfica de la herramienta EEGLAB (Makeig, Delorme, & Brunner, 2013).

3.2.3 ERPLAB: an open-source toolbox for the analysis of event-related potentials (Lopez-Calderon & Luck, 2014)

ERPLAB es una caja de herramientas de MATLAB de código abierto que se especializa en el procesamiento y análisis de datos de potenciales relacionados a eventos obtenidos por medio de

interfaces cognitivas, como los dispositivos EEG (Figura 3-4). Para recibir los datos de dicho dispositivo, integra la herramienta EEGLAB.

También permite detectar, corregir y rechazar parte de los resultados que una persona considere innecesarios, además de aplicar filtros a dicha información para afinarla.

Esta herramienta es una extensión de EEGLAB que tiene como objetivo analizar los datos ERP generados a partir de eventos planificados con anterioridad y su relación con las señales generadas.

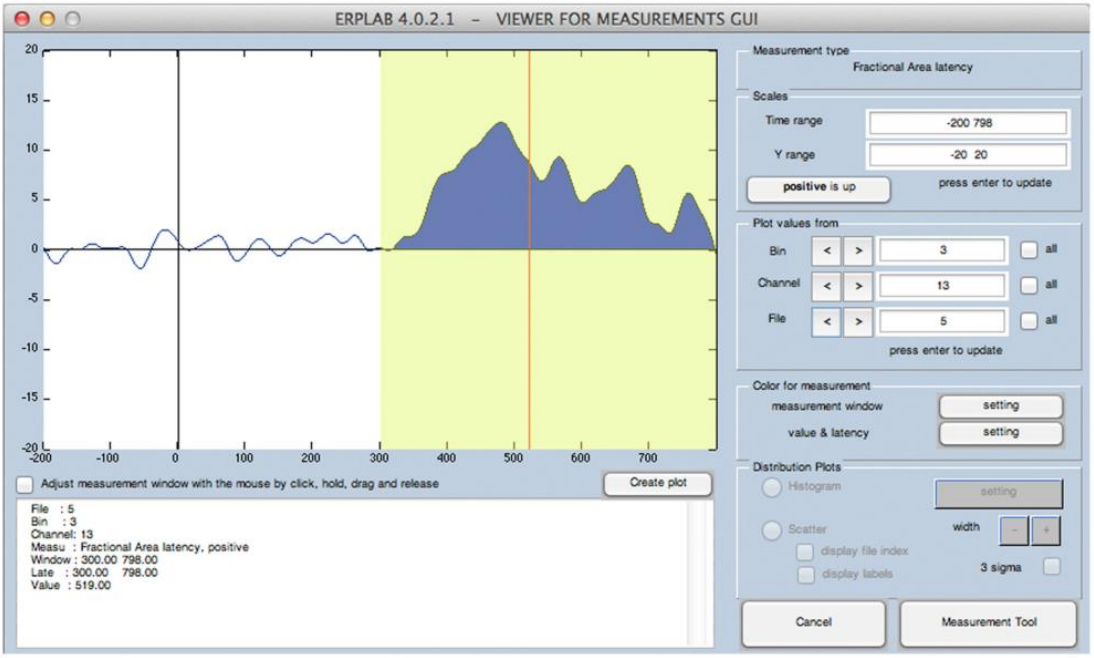


Figura 3-4. Interfaz gráfica de la herramienta ERPLAB (Makeig, Delorme, & Brunner, 2013).

3.2.4 MoBILAB: an open source toolbox for analysis and visualization of mobile brain/body imaging data (Ojeda, Bigdely-Shamlo, & Makeig, 2014)

MoBILAB es una caja de herramientas de MATLAB de código abierto que se especializa en la colección sincronizada de actividad cerebral (EEG) y la actividad conductual (captura de movimientos corporales y oculares), además de eventos de ambiente (grabaciones de video del entorno), para estudiar las dinámicas de cuerpo-cerebro que soportan el proceso cognitivo de las acciones humanas (Figura 3-5). Debido a esto, esta herramienta se enfoca más en el análisis del vínculo entre los movimientos corporales y la actividad cerebral que en otros ámbitos.

Se utiliza la librería LSL para capturar y sincronizar la información por medio de rangos de tiempo relativos, los cuales son utilizados entre una colección de computadoras conectadas a través de una red local.

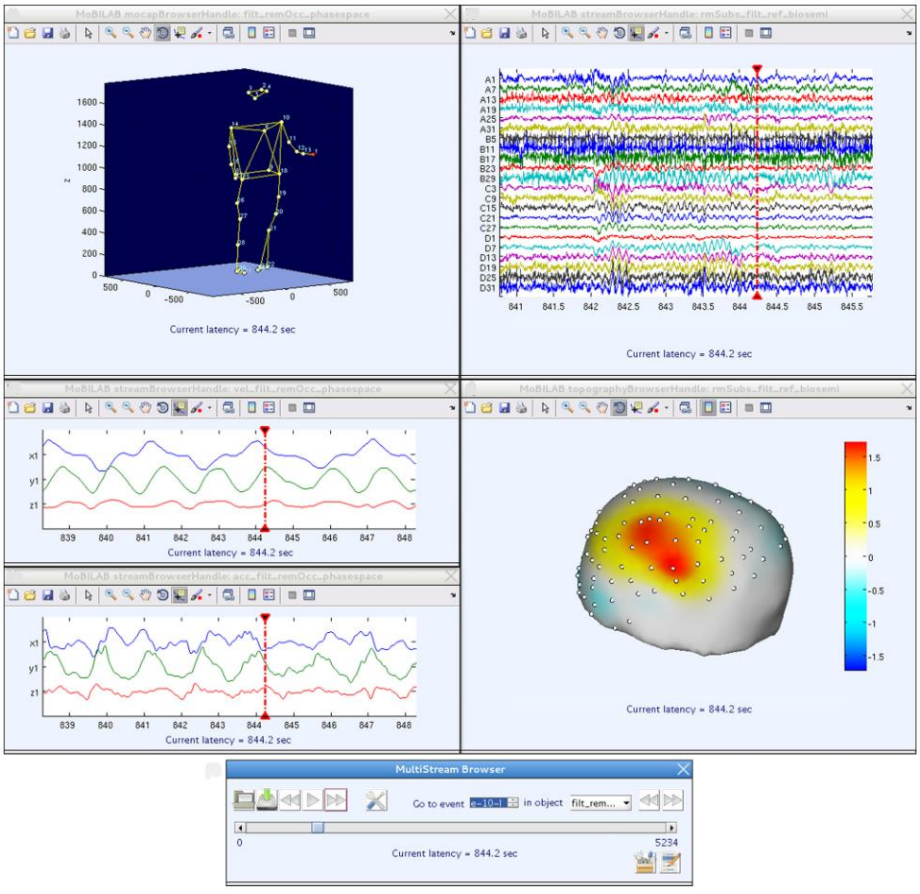


Figura 3-5. Interfaz gráfica de la herramienta MoBILAB (Ojeda, Bigdely-Shamlo, & Makeig, 2014).

3.2.5 iMotions: Biometric Research Platform (iMotions A/S, 2017)

iMotions es una plataforma de presentación de estímulos y recolección de datos que permite diseñar estudios, presentar estímulos de cualquier tipo, recolectar datos, crear marcadores y anotaciones en la grabación global y exportar resultados en datos crudos y visualizaciones gráficas.

Permite integrar una gran cantidad de sensores biométricos de distintos tipos (EEG, Eyetracking, ritmo cardiaco, respiración, entre otros), además de incluir una API para dispositivos no incluidos (figura 3-6).

La arquitectura física de la plataforma establece realizar el procesamiento y sincronización de información a partir de una sola computadora, con todos los dispositivos de entrada y salida de información conectadas a ella.

Al momento de su investigación, se cotiza un costo aproximado de $ 30,000 USD por la licencia de uso núcleo con el módulo de integración de un dispositivo EEG.

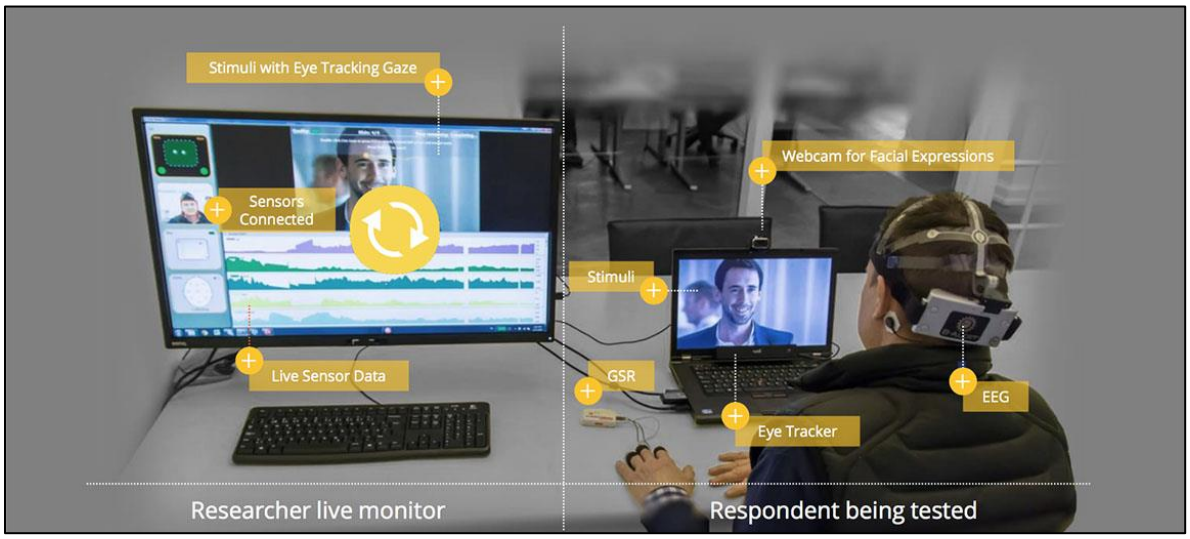


Figura 3-6. Instalación de laboratorio de la plataforma iMotions, según su página de internet y documentación (iMotions A/S, 2017).

3.2.6 Quad Server, de Eyetracking, Inc (Eyetracking, Incorporated)

Quad Server es una aplicación de la empresa Eyetracking Inc., diseñada para la integración y sincronización de múltiples fuentes de datos localizados dentro de una red local de computadoras (figura 3-7).

Fue originalmente desarrollado para cumplir un requisito impuesto por la Agencia de Proyectos Avanzados de Investigación de Defensa del gobierno de Estados Unidos, con la finalidad para simplificar el proceso de integración de virtualmente cualquier formato de datos.

La técnica utilizada para la sincronización es la aplicación de estampas de tiempo de alta precisión como marcado de cada paquete obtenido en los flujos de datos de la red.

Para integrar la información se utiliza una base de datos interna donde se almacena para consultarla posteriormente, permitiendo exportar la información a archivos comúnmente utilizado para analizar la información.

Figura 3-7. Interfaz gráfica del sistema Quad Server (Eyetracking, Incorporated).

3.2.7 Tabla comparativa de herramientas de software del estado de la técnica

A continuación se presenta una tabla comparativa que muestra las características de cada herramienta de software investigada en el estado de la práctica (Tabla 2).

Tabla 2. Tabla comparativa de herramientas de software del estado de la técnica.

Herramientas de software	Tipo de software	Tipos de datos sincronizados	Análisis automatizado de información obtenida	Manipulación de datos generados	Técnica utilizada
EEGLAB (EEGLAB - an Open Source Matlab Toolbox for Electrophysiological Research)	Caja de herramientas de MATLAB	Flujos de datos EEG	✓	✓	Estampas de tiempo
ERPLAB (ERPLAB: an open-source toolbox for the analysis of event-related potentials)	Caja de herramientas de MATLAB	Flujos de datos EEG, eventos ERP	✓	✓	Estampas de tiempo, registros de eventos
MoBILAB (MoBILAB: an open source toolbox for analysis and visualization of mobile brain/body imaging data)	Caja de herramientas de MATLAB	Flujos de datos EEG, movimiento corporal, movimiento ocular y video	✓		Estampas de tiempo
iMotions Biometric Research Platform	Plataforma de software	Eyetracking, EEG, ECG, EMG, GSR, video, audio y otros vía API		✓	No especifica
Quad Server	Plataforma de software	Eyetracking, EEG, fNIR, EKG, GSR, entre otros.		✓	Estampas de tiempo

Capítulo 4. Análisis de técnicas de sincronización

En este capítulo se presentan la investigación de las técnicas de sincronización e integración de datos multimedia.

En la investigación del estado del arte se encontró que las principales formas de sincronización se dividen en dos tipos, 1) las técnicas basadas en creación de metadatos de unidades de tiempo, y 2) las técnicas basadas en la localización de coincidencias por medio de eventos captados por dos o más flujos de información.

El primer tipo utiliza datos como las estampas de tiempo para intentar señalar con exactitud momentos específicos en la grabación, los cuales son utilizados como referencias para mantener consistencia en su reproducción. Mientras tanto, el segundo tipo de técnicas se apoya en eventos físicos que son captados por varios dispositivos para delimitar un momento de sincronización, lo cual permite crear un desfase en las grabaciones para sincronizarlo.

La aplicación de una o más técnicas de sincronización del segundo tipo necesita mantener un registro de metadatos dentro de las grabaciones, lo cual requiere del uso de una forma de integración en un medio virtual. Esto permite almacenar toda la información generada en un solo lugar, con una estructura especifica.

4.1 ¿Qué es una técnica de sincronización?

Una técnica de sincronización es aquella técnica utilizada para dar consistencia a un conjunto de datos en base a su generación a través de una línea de tiempo. En otras palabras, distribuye la información obtenida en tiempo real conforme se va detectando por los dispositivos de entrada del ambiente en el que se graba.

Existen distintas formas en las que se sincronizan los datos, desde la deducción matemática de la hora de recepción de información hasta la creación de eventos físicos que son difíciles de generar en un ambiente común. Todas estas técnicas tienen un fin en común, asociar datos obtenidos de manera simultánea a un momento en específico.

La forma en la que se realiza cada técnica de sincronización varia en base al medio y el proceso donde se realiza, el cual se divide en metadatos y en coincidencias únicas.

4.2 ¿Cómo se relaciona el proceso de sincronización en base a un sistema multimedia?

Tomando en cuenta la especificación del modelo de sincronización de sistemas multimedia que describe Gerald Blakowski y Ralf Steinmetz en (Blakowski & Steinmetz, 1996) y Edward Chow en (Chow, 2012), un sistema multimedia es aquel que soporta el procesamiento integrado de múltiples tipos de información provenientes de distintos medios. Este modelo define tres etapas de sincronización:

- ***La sincronización de muestras a nivel intra-medio (figura 4-1)***

Dentro de esta etapa se obtiene el flujo de información de un dispositivo, controlando su generación de muestras para mantener un ritmo constante y consistente de los datos producidos. El flujo resultante permite determinar en qué lapso de tiempo se generó una determinada muestra.

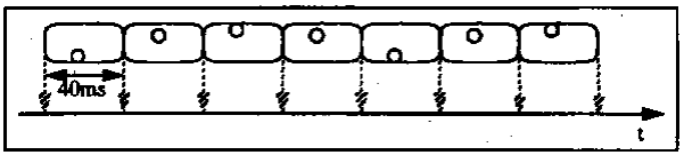


Figura 4-1. Ejemplo de generación de muestras de imagen cada 40 milisegundos (Chow, 2012).

- ***La sincronización de objetos a nivel inter-medio (figura 4-2)***

Una vez que se genera el flujo de información constante y consistente de cada dispositivo, se obtiene la secuencia de datos de cada flujo de información y se identifica su posición en base a una línea de tiempo global "inter-media".

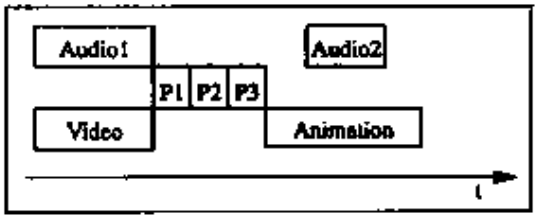


Figura 4-2. Ejemplo de línea temporal con un conjunto de objetos o flujos de información multimedia (Chow, 2012).

- ***La sincronización de información en base a eventos (figura 4-3)***

Al término de la grabación se verifica que las secuencias de datos se encuentren en su respectivo lugar, aplicando un proceso de sincronización postprocesamiento al utilizar una especificación determinada, ya sea en base a eventos, ejes de tiempo u otros metadatos.

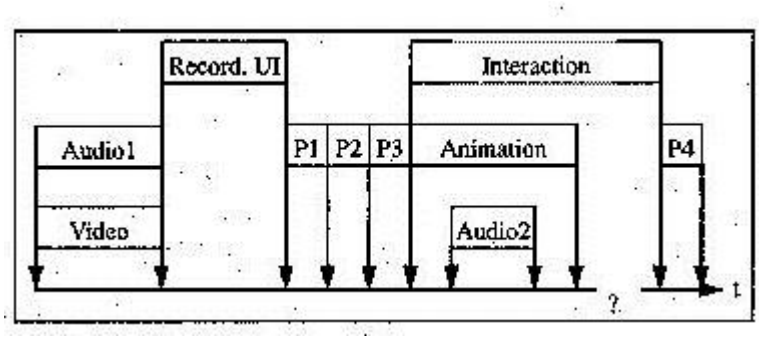


Figura 4-3. Ejemplo de afinación de objetos de línea de tiempo en base a eventos activados por ejes de tiempo (Chow, 2012).

4.3 Técnicas de sincronización basadas en metadatos

Las técnicas de sincronización basadas en metadatos se centran en la aplicación de etiquetas virtuales, los cuales al ser utilizados en conjunto de algoritmos matemáticos se obtiene una aproximación de la generación de las muestras de datos obtenidas por un dispositivo de entrada. Esto permite generar una línea de tiempo hipotética donde se asigna un momento a cada dato generado.

La estimación de retraso por medio de estampas de tiempo de dos muestras provenientes de dispositivos diferentes de grabación en una red, la predicción de tiempo de generación de muestras distintas y el uso de estampas de alta precisión son algunos ejemplos de técnicas de sincronización basadas en metadatos, cada uno con su grado de precisión basado en la forma en la que se implementa.

Una de las características más importantes del uso de estas técnicas es la manera en la que se deben manejar las muestras, ya que se debe tener acceso directo al flujo de muestras para analizar sus propiedades temporales y espaciales en su medio virtual.

Debido a esto, en esta investigación se analiza la manera en la que funciona la técnica de aplicación de estampas de alta precisión para etiquetar el tiempo de generación de cada una de las muestras obtenidas por diferentes dispositivos conectados a una computadora central (Zhou, Zheng, H., & Yang, 2013).

La técnica de aplicación de estampas de alta precisión consiste en la adquisición veloz de una unidad de tiempo local precisa que permita ubicarse en un momento específico en una línea de tiempo, la cual depende de la arquitectura del procesador de la computadora. A partir de esta estampa de tiempo se deduce el momento en el que se genera una muestra y se asocia, almacenándose en un registro continuo que permite vincularse con muestras de otros flujos de información.

El uso de estampas de tiempo de alta precisión puede aplicarse entre varios nodos localizados en una red local de computadoras. Para ello es necesario obtener el desfase entre los relojes de los nodos por medio de la ecuación 1 basada en el protocolo NTP de transferencia de datos:

$$Clock\ Offset\ (OFS) = \frac{(TB_1 - TA_1) + (TB_2 - TA_2)}{2} \qquad (Ecuación\ 1)$$

Donde:

- TA_1 es la estampa de tiempo generada por el nodo *A* cuando se envía un paquete al nodo *B*.

- TB_1 es la estampa de tiempo que genera el nodo *B* al recibir el paquete del nodo *A*.
- TB_2 es la estampa de tiempo que genera el nodo B al enviar de regreso al nodo *A* un paquete con las estampas de tiempo anteriores.
- TA_2 es la estampa de tiempo que se genera al obtener el paquete del nodo *B*.

A partir del valor de desfase se puede convertir la unidad de tiempo del nodo *B* a la unidad de tiempo del nodo *A*, eliminando inconsistencias generadas por el transporte de las muestras a través de la red local.

Una implementación de la adquisición de estampas de tiempo de alta precisión es la utilizada en la librería LabStreamingLayer (Swartz Center for Computational Neuroscience (SCCN), UCSD, 2012), diseñada para simplificar el uso de este tipo de técnicas. Se utilizan las funciones de la clase Chrono de la librería Boost de C++ (Dawes, Abrahams, & Rivera, 1999) para generar estampas de tiempo a partir del contador de procesador de una computadora, el cual inicia al encender la computadora y su medida de tiempo en microsegundos es calculada a partir en conjunto de su frecuencia de procesamiento.

Para verificar el tiempo de generación de la estampa de tiempo y el procesamiento de muestras del dispositivo se utiliza el lenguaje de programación Python en conjunto de un dispositivo de video. Para calcular el tiempo de ejecución se utiliza la función clock() de la librería de tiempo de Python (figura 4-4, línea 10, 12 y 15).

```
1  import sys, os, time
2  from libs.pylsl import local_clock
3  import numpy as np
4  import cv2
5
6  cap = cv2.VideoCapture(0)
7
8  for i in range(0, 10):
9
10     start = time.clock()
11     timestamp = local_clock()
12     mid = time.clock()
13     ret, frame = cap.read()
14     frame.tostring()
15     end = time.clock()
16
17     print "Generacion de estampa de tiempo: {} milisegundos".format((mid-start)*1000)
18     print "Captura de muestra de video: {} milisegundos\n".format((end-mid)*1000)
```

Figura 4-4. Extracto de código de generación de muestras con verificación de velocidad de rendimiento.

Como se puede observar, en la figura 4-5, en la información resultante, las estampas de tiempo de alta precisión necesitan un promedio de 0.03 milisegundos para generarse, mientras que las muestras de video se generan en un lapso de tiempo de entre 4 y 8 milisegundos.

```
Generacion de estampa de tiempo: 0.0480608833404 milisegundos
Captura de muestra de video: 16.6724604139 milisegundos

Generacion de estampa de tiempo: 0.0303296836614 milisegundos
Captura de muestra de video: 6.21571870852 milisegundos

Generacion de estampa de tiempo: 0.0261301890006 milisegundos
Captura de muestra de video: 6.12752932064 milisegundos

Generacion de estampa de tiempo: 0.0256635784827 milisegundos
Captura de muestra de video: 3.72448515362 milisegundos

Generacion de estampa de tiempo: 0.0335959572865 milisegundos
Captura de muestra de video: 3.91112936077 milisegundos

Generacion de estampa de tiempo: 0.0279966310721 milisegundos
Captura de muestra de video: 3.62136422917 milisegundos

Generacion de estampa de tiempo: 0.0312629046972 milisegundos
Captura de muestra de video: 3.80707521528 milisegundos

Generacion de estampa de tiempo: 0.0298630731435 milisegundos
Captura de muestra de video: 8.27907041853 milisegundos

Generacion de estampa de tiempo: 0.0303296836614 milisegundos
Captura de muestra de video: 3.84113778309 milisegundos

Generacion de estampa de tiempo: 0.0307962941793 milisegundos
Captura de muestra de video: 4.55271882283 milisegundos
```

Figura 4-5. Resultados de verificación de velocidad de procesamiento.

Este experimento se realizó con el fin de identificar la cantidad de tiempo utilizado para generar una estampa de tiempo y procesar una muestra generada por un dispositivo, permitiendo identificar qué medidas se deben aplicar para prevenir el desfase de las muestras. En este caso se identifica que la generación de estampas de tiempo de alta precisión es más veloz que la recuperación de la última muestra de video, por lo que la mejor opción es generar la estampa y después procesar la muestra del dispositivo.

4.4 ¿Qué es una técnica de integración de datos?

Una técnica de integración de datos es un método de juntar y "combinar" dos o más tipos de información en un solo contenedor virtual. Hay dos formas de realizar esta combinación de datos:

- La primera es modificar la estructura principal de la información almacenada e introducir los datos complementarios de una manera compatible a la estructura principal. En otras palabras, se elige la información con una estructura que permita incrustar otro tipo de información dentro de ésta (Peña, Ávila, Muñoz, & Lavariega, 2015). Esta forma de integración de datos no permite manipular fácilmente la información incrustada ya que es necesario extraerla por métodos manuales o apoyándose de técnicas del área de visión artificial. En la figura 4-6 se observa una manera en la que se incrustan los datos EEG por medio de su visualización gráfica por medio del software del dispositivo, integrándose al video.

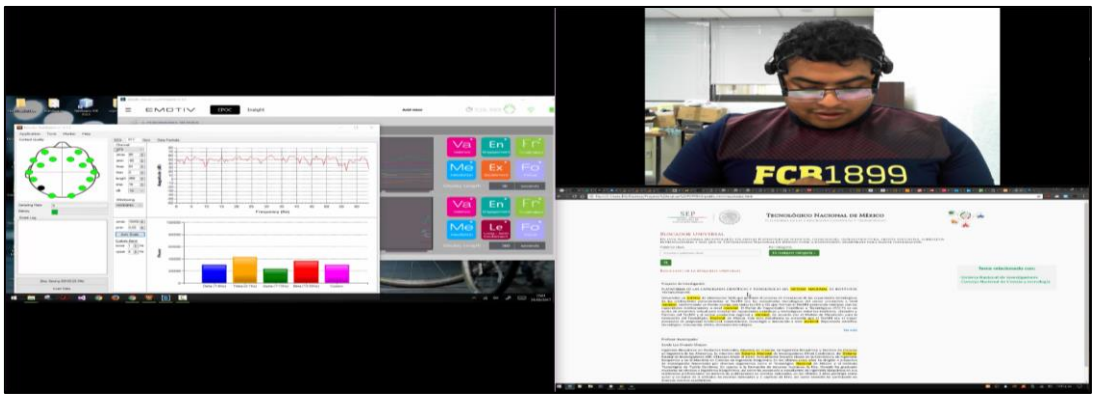


Figura 4-6. Ejemplo de incrustación de datos en video al colocar valores de datos EEG. Los datos EEG quedan almacenados como gráficas de barras e histogramas en la sección lateral izquierda, visualizadas por medio de la interfaz gráfica del software del dispositivo.

- La segunda forma de integrar la información es distribuirla en diferentes flujos, generar un contenedor virtual que permita almacenar cada uno junto con metadatos que las describan y permitan agregar información adicional, como eventos de sincronización y estampas de tiempo (Lambert, Hachicha, Ahmed, Pinna, & Garda, 2015). En la figura 4-7 se muestra la estructura en la que se guarda la información en un contenedor virtual, vinculando los flujos de información con referencias de tiempo.

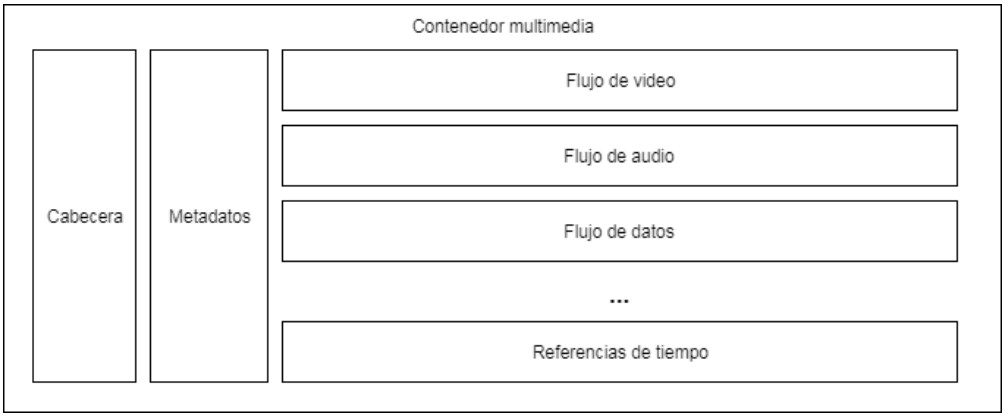


Figura 4-7. Representacion de un contenedor multimedia con capacidad de almacenar flujos de informacion mas primitivos. Cada flujo de informacion se encuentra vinculada con al menos una referencia de tiempo.

El contenedor Matroska (MKV), los videos Flash (FLV), QuickTime (MOV, QT) y el flujo de transporte BDAV MPEG-2 (m2ts) son los ejemplos más conocidos de contenedores multimedia que permiten la manipulación de datos primitivos con estampas de tiempo, los cuales contienen especificaciones de pistas o conceptos similares que definen el inicio y el fin de conjuntos de datos secuenciales de los flujos de información. Por ejemplo, el contenedor Matroska cuenta con una especificación que contempla la inclusión de flujos de información no convencionales en medios audiovisuales, como la información de flujos de datos primitivos generados por sensores o maquinaria especializada, la cual tiene la capacidad de ser vinculada por medio de la creación de bloques de datos vinculados con una determinada estampa de tiempo (Matroska, 2016).

Capítulo 5. Análisis de dispositivos multimedia

En este capítulo se detalla la investigación del uso de distintos dispositivos de captura de información multimedia dentro de un ambiente de prueba para la evaluación de la experiencia centrada en el usuario.

5.1 Captura de datos de Electroencefalografía

Método de monitorización de registros electrofisiológicos, utilizado para grabar actividad eléctrica del cerebro.

Esta señal permite detectar procesos motrices, lingüísticos, perceptivos, emocionales y cognitivos que ocurren en rangos de milisegundos, los cuales pueden ser usados para analizar las experiencias generadas a partir del uso de un producto. Dichas experiencias pueden llegar a ser difíciles de expresar en un lenguaje hablado o escrito por el sujeto que lo experimenta, por lo que esta señal es de gran utilidad para expertos en el área de las neurociencias.

Para realizar esta captura en un ambiente de prueba para la evaluación de la experiencia del usuario es necesario contar con un dispositivo de señales EEG no invasivo, el cual se compone generalmente de electrodos con medidores de oscilaciones neuronales colocados alrededor del cuero cabelludo de una persona para detectar la actividad del potencial eléctrico de las neuronas cercanas a cada electrodo.

5.1.1 EMOTIV Epoc / Epoc +

a) Descripción del dispositivo

La empresa australiana EMOTIV se dedica al constante descubrimiento de la manera en la que el cerebro humano funciona por medio del desarrollo de tecnologías relacionadas con interfaces cerebro computadora (BCI), como la encefalografía. Es considerado como el pionero y el líder de mercado de este campo de la investigación.

Figura 5-1. Dispositivo Emotiv Epoc

Este dispositivo es un producto de la empresa EMOTIV, permite capturar información EEG por medio de 14 canales de electrodos diseñados para realizar investigaciones contextuales y aplicaciones avanzadas de BCI al proveer acceso a datos crudos de la más alta calidad (figura 5-1).

El tipo de datos obtenido a partir de estos electrodos es un arreglo de valores numéricos generados entre 128 y 256 veces por segundo (como según se configure).

Entre los canales se encuentran los siguientes valores correspondientes al modelo de referencia 10-20 (W Homan, Herman, & Purdy, 1987): AF3, F7, F3, FC5, T7, P7, O1, O2, P8, T8, FC6, F4, F8 y AF42. Además, su software permite reducir el ruido en los datos, y configurar localidades de activación de técnicas P3 y P4 para potenciales evocados.

Cuenta con un Kit de Desarrollo de Software que permite obtener la información del dispositivo y procesarla con módulos propietarios de la empresa, siendo compatible con gran cantidad de dispositivos, desde computadoras personales hasta dispositivos móviles.

b) Beneficios

La aplicación de métodos para la detección de emociones y estados mentales por medio de estos dispositivos permite abordar temas de análisis psicológico con mayor facilidad al verificar la actividad bioeléctrica del cerebro en relación con sus acciones.

c) ¿Cuál es la forma en la que se obtienen sus datos?

Cada electrodo obtiene un rango de valores en microvolts en varias frecuencias que indican la actividad eléctrica generada por las neuronas localizadas debajo del área. En el transcurso de un segundo se detectan alrededor de 5 frecuencias principales: delta, theta, alpha, beta y gamma.

En base a distintas investigaciones realizadas a través de los años, cada frecuencia se asocia con distintos estados cognitivos, estados emocionales e información relacionada con la actividad cerebral.

Existen distintos métodos oficiales y no oficiales para acceder a los flujos de información generados por el dispositivo. Entre los principales se encuentran:

- **Emokit**: Es un conjunto de librerías en distintos lenguajes con la finalidad de acceder a los datos crudos (sin procesar) generados por las diademas Epoc (Brocious, Machulis, Lemaignan, Olorin, & Schumacher). (https://github.com/openyou/emokit)
- **Emotiv SDK Community Edition**: Versión de código abierto del SDK y API oficial de los dispositivos Emotiv. Permite obtener información del dispositivo y aplicar algoritmos de procesamiento y filtrado en los datos por medio de funciones de caja negra creadas por la empresa. Es multiplataforma y reemplaza la versión del SDK de paga de investigación y educación. Cuenta con funciones que necesitan una licencia especial para ser utilizadas (The Emotiv Team). (https://github.com/Emotiv/community-sdk)
- **Pyemotiv**: Librería de Python de código abierto que permite adquirir datos de la diadema Epoc usando los archivos del antiguo SDK de investigación de Emotiv (Hearn). No soporta ambientes Linux. (https://github.com/thearn/pyemotiv)
- **python-emotiv**: Librería de Python de código abierto utilizado para adquirir información del dispositivo Epoc. Es una implementación de ingeniería inversa del protocolo original generado por el dongle con el que se conecta inalámbricamente el dispositivo. Se diseñó con el fin de soportar cajas con Raspberry Pi y microprocesadores ARM. Su última actualización fue en Julio de 2015 (Çağlayan). (https://github.com/ozancaglayan/python-emotiv)
- **Mushu**: Librería de Python de código abierto diseñada para la adquisición de señales de dispositivos tipo Interfaz Cerebro Computadora, la cual soporta el dispositivo EPOC de Emotiv y el dispositivo g.USBamp de g.tecis (Venthur, 2012). (https://github.com/bbci/mushu)

d) Limitaciones

- Los electrodos del dispositivo deben de colocarse en las posiciones correctas en relación al área que representan, ya que las lecturas pueden verse afectadas al detectar información irrelevante o que no le corresponde.
- Las almohadillas de los electrodos deben mantenerse con un determinado nivel de humedad para mantener el contacto entre el circuito sensor de los electrodos y el cráneo de la persona, siendo necesario administrarles una solución salina cada determinado tiempo.
- Las librerías oficiales para obtener información del dispositivo necesitan de una licencia de pago para acceder a la información cruda obtenida por cada electrodo, de lo contrario sólo se puede obtener información procesada por el software.

5.1.2 NeuroSky Mindwave

a) Descripción del dispositivo

La empresa NeuroSky se dedica a la manufactura de tecnologías de interfaces cerebro computadora para aplicaciones en productos de consumidor. Esta compañía adapta tecnologías de encefalografía y electromiografía para abarcar áreas de aplicación como el entretenimiento, la educación, la automatización y la salud. Sus productos utilizan tecnología de bajo costo con técnicas de sensor seco en dispositivos EEG (NeuroSky, s.f.).

Figura 5-2. Dispositivo NeuroSky Mindwave

Este dispositivo se encuentra conformado por dos sensores: un canal de electrodo relacionado con el área FP1 en la parte frontal superior izquierda del rostro, mientras que el segundo sensor es un clip para la oreja que contiene un electrodo de tierra y referencia para la diadema (figura 5-2).

Su fuente de poder es una batería AAA que permite un uso de alrededor de 8 horas continuas.

b) Beneficios

Este dispositivo, al tener un costo menor a otras opciones profesionales, permite experimentar casos cuyas entradas de información EEG se encuentran reducidas debido a la limitación de la cantidad de electrodos disponibles. Esto permite verificar la capacidad de adaptación de los métodos de adquisición de datos que se utilicen dentro del ambiente de evaluación de UX.

c) ¿Cuál es la forma en la que se obtienen sus datos?

El electrodo obtiene un rango de valores en microvolts en varias frecuencias que indican la actividad eléctrica generada por las neuronas localizadas debajo del área. En el transcurso de un segundo se detectan alrededor de 5 frecuencias principales: delta, theta, alpha, beta y gamma.

En base a distintas investigaciones realizadas a través de los años, cada frecuencia se asocia con distintos estados cognitivos, estados emocionales e información relacionada con la actividad cerebral.

A continuación se indican las principales formas de obtener información del dispositivo:

- **ThinkGear Connector (TGC)**: Programa en segundo plano que funge como servidor que administra la conexión con el dispositivo y obtiene la información que genera a través de su electrodo, permitiendo acceder a ella a través de

conexiones de socket. Funciona en Windows y Mac OSX (NeuroSky, s.f.). (http://developer.neurosky.com/docs/doku.php)

- **ThinkGear Communications Driver (TGCD)**: Librería compartida DLL o Bundle que permite establecer conexión con el dispositivo sin necesidad de generar conexiones por socket para obtener la información del dispositivo. Funciona en sistemas operativos y lenguajes que permitan la llamada a funciones de librerías compartidas .dll y .bundle (NeuroSky, s.f.). (http://developer.neurosky.com/docs/doku.php)
- **ThinkGear Communications Protocol**: Protocolo utilizado para obtener información por medio de puertos seriales, donde además de recibir la información se debe descifrar en base al documento especificado por el protocolo. Comúnmente utilizado con lenguajes y dispositivos con acceso a puertos de entrada/salida serial, como un Arduino (NeuroSky, s.f.). (http://developer.neurosky.com/docs/doku.php)

d) Limitaciones

- El dispositivo cuenta con un electrodo que se coloca en la parte izquierda de la frente, lo cual limita el área de recuperación de información y la aplicación del mismo.

5.2 Captura de datos de movimiento ocular

Método de grabación de datos utilizado para obtener la ubicación aproximada de la mirada de una persona en relación a una superficie plana.

Para obtener dicha información existen dos técnicas: 1) la primera implica utilizar un emisor de luz infrarroja de onda corta y una cámara de alta resolución o sensor óptico similar que reciba la luz reflejada, la cual no es percibida por nuestros ojos y permite obtener una imagen clara de la córnea y el iris de cada ojo; mientras tanto, 2) la segunda técnica utiliza una cámara de video para detectar la forma de los ojos por medio del reconocimiento de patrones.

El cálculo de posición de la mirada se realiza por medio de algoritmos que utilizan características de los ojos como el tamaño de la pupila y el tamaño del iris para detectar el ángulo hacia donde se está observando, la cual al combinarse con una calibración personalizada se obtienen coordenadas aproximadas referentes a la superficie que se observa.

Además de obtener la posición aproximada de la mirada, este tipo de dispositivo permite obtener información relacionada del estado de carga cognitiva y su nivel de emoción a partir de la dilatación de sus pupilas y la velocidad de su parpadeo.

5.2.1 Eye Tribe

a) Descripción del dispositivo

La empresa The Eye Tribe se dedicaba al desarrollo de dispositivos de captura del movimiento ocular, enfocándose en la construcción de componentes a precio competitivo (The EyeTribe, 2016) (figura 5-3). A finales del año 2016 fue vendida a la empresa Oculus VR, que forma parte de Facebook, cesando la producción de este dispositivo.

Figura 5-3. Dispositivo TheEyeTribe.

El dispositivo cuenta con una cámara y un módulo LED infrarrojo de alta resolución, los cuales se alimentan por medio de un puerto Micro-B y un cable USB 3.0. Incluye una entrada que permite agregar un soporte para posicionarse sobre una mesa o sobre uno de los lados de dispositivos móviles (figura 5-4).

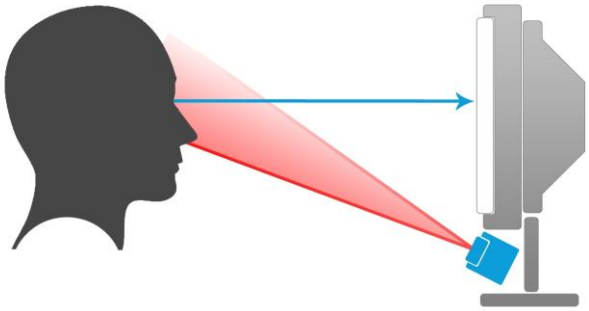


Figura 5-4. Representación de captura de movimiento ocular por medio de un sensor infrarrojo.

Cuenta con un SDK que permite obtener las coordenadas de la mirada de una persona después de ser procesado por un módulo de caja negra del dispositivo. Es compatible con los sistemas operativos Windows y MacOS, mientras que en distribuciones Linux necesita de software de terceros para funcionar (figura 5-5).

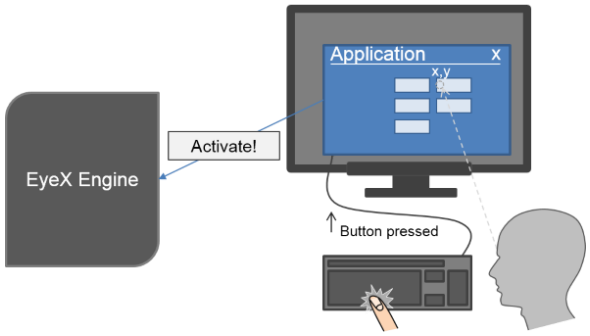


Figura 5-5. Ejemplo de software de Eyetracking de caja negra en modalidad de servidor (Tobii Technology AB, 2014).

b) Beneficios

Este dispositivo tiene un bajo costo monetario y una mayor precisión en sus lecturas que otras opciones disponibles, como empresas con tecnologías similares, o técnicas de detección de mirada por medio de reconocimiento de patrones a través de cámaras de video.

La aplicación de métodos para la localización de la mirada permite detectar puntos visuales de interés de la interfaz gráfica del producto evaluado en el ambiente, lo cual permite encontrar comportamientos específicos en base a lo que se enfoca en un determinado momento.

c) ¿Cuál es la forma en la que se obtienen sus datos?

Este dispositivo utiliza la técnica del emisor infrarrojo de corta frecuencia para obtener información del movimiento ocular. A continuación se indican las principales formas de obtener información del dispositivo:

- **Eyetribe Server**: Programa servidor que realiza la conexión con el dispositivo y pone a disposición su información por medio de conexión a través de sockets. Para hacer las peticiones es necesario utilizar mensajes específicos para recibir información y mantener la conexión activa (The EyeTribe, 2016). (https://github.com/EyeTribe/sdk-installers)
- **PyTribe / PyGaze**: Librería utilizada para simplificar la conexión y las peticiones realizadas al servidor de Eyetribe. El conjunto de librerías PyGaze además permite realizar pruebas con imágenes estáticas y simulaciones para recabar datos relacionados a ellas (Dalmaijer, Mathôt, & Van der Stigchel, 2014). (https://github.com/esdalmaijer/PyTribe)

d) Limitaciones

- La empresa fue comprada en 2016 por Oculus VR, cesando cualquier servicio de mantenimiento, soporte y reparación de las unidades vendidas.
- Se necesita de la aplicación servidor para obtener datos del dispositivo, la cual sólo se encuentra disponible en sistemas operativos Windows y Mac OS.
- La persona que lo utilice necesita tener los parpados de una manera que se permita detectar la mayor parte del iris en todo momento.
- No permite utilizar materiales transparentes para auxiliar en la visión de la persona que lo utilice, ya que interfiere con el reflejo de la luz infrarroja.

- El mínimo movimiento de cabeza del usuario tiene la posibilidad de generar un error en la calibración las lecturas del dispositivo, generando muestras con información de la mirada desviada.

5.3 Captura de datos de video

Método de grabación de imágenes secuenciales utilizadas para representar un momento específico dentro de un lugar y tiempo determinado. Cada imagen obtenida de la secuencia es tomada en cuenta para calcular la velocidad y la calidad de la reproducción del video.

El video obtenido por medio de este tipo de captura es utilizado para analizar visualmente el contexto en el que sucede la grabación, detectando eventos físicos significativos que dispositivos de otro tipo pueden no detectar. Existen técnicas de visión artificial por medio de reconocimiento de patrones que permiten automatizar este proceso al detectar objetos y eventos relacionados a los mismos.

5.3.1 Cámara web

a) ***Descripción del dispositivo***

Dispositivo de video de bajo costo que alimenta a una computadora de la imagen que va capturando en tiempo real (figura 5-6). Generalmente conectado por medio de entradas USB o integrado en el hardware de la computadora, permite el acceso a su conexión de manera continua y en sesiones de tiempo indefinidas.

Figura 5-6. Cámara web.

b) ***Beneficios***

Este dispositivo tiene un bajo costo monetario y mayor flexibilidad de uso a comparación de otros dispositivos de video más especializados.

Obtener contenido de video de este dispositivo permite guardar información del estado del ambiente de prueba, los eventos relacionados con dicho ambiente y los movimientos conscientes e inconscientes del sujeto de prueba.

c) ***¿Cuál es la forma en la que se obtienen sus datos?***

Este dispositivo utiliza la técnica del emisor infrarrojo de corta frecuencia para obtener información del movimiento ocular. A continuación se indican las principales formas de obtener información del dispositivo:

- **Software dedicado a la captura de video**: Programa dedicado a la captura y generación de un archivo de video a través de codificadores especializados para la compresión de video. Si se desea tener mayor control con el procesamiento de la información es necesario contar con acceso a su API o sólo se ejecuta de manera paralela, sin mucho control sobre su configuración.
- **OpenCV**: Librería utilizada para proveer acceso a dispositivos de video a nivel de aplicación a desarrolladores de software, permitiendo aplicar algoritmos y procesos de visión artificial al flujo de datos resultante. También cuenta con herramientas para manipular y generar contenido multimedia (OpenCV team, s.f.). (https://opencv.org/)
- **Acceso directo a los datos crudos del dispositivo**: Acceso directo al flujo de información cruda por medio de los drivers y APIs del dispositivo. No asegura que el código implementado funcione con otros dispositivos.

d) ***Limitaciones***

- La calidad del video depende de los componentes del dispositivo y su costo. Mientras más barato sea, mayor probabilidad de que su resolución de captura sea menor debido al bajo costo de sus componentes.

- Como con todos los dispositivos, para obtener un flujo con una velocidad constante es necesario que cuente con su propio bus de transmisión de datos con la computadora a la que se encuentra conectada.

5.3.2 Captura de pantalla de computadora

a) ***Descripción del dispositivo***

La captura de pantalla es una grabación digital de la salida de video de la computadora (figura 5-7). Se genera a partir de la captura de imágenes sucesivas del escritorio y las ventanas de la interfaz gráfica del sistema operativo, la cual tiene la misma calidad que con la que se percibe.

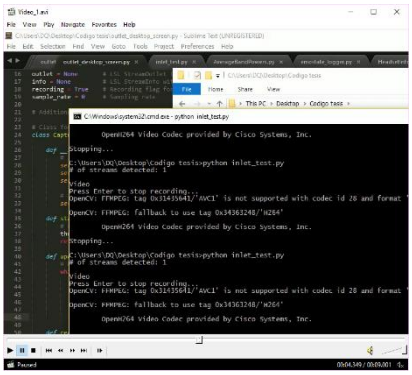


Figura 5-7. Ejemplo de captura de pantalla.

Este tipo de grabación puede llegar a ser costoso en relación a recursos computacionales mientras la interfaz gráfica cuente con más resolución, ya que el área de captura es mayor.

b) ***Beneficios***

Obtener contenido de video de la pantalla permite guardar información acerca de las acciones de una persona al utilizar una computadora en una prueba.

Esta grabación puede ser utilizada en conjunto con los datos de movimiento ocular para generar una serie de lecturas que indican la forma en la que el usuario recorre la pantalla, que secciones le llama más la atención y otros aspectos psicológicos similares.

c) ***¿Cuál es la forma en la que se obtienen sus datos?***

Este dispositivo utiliza la técnica del emisor infrarrojo de corta frecuencia para obtener información del movimiento ocular. A continuación se indican las principales formas de obtener información del dispositivo:

- **Software dedicado a captura de video**: Programa dedicado a la captura y generación de un archivo de video a través de codificadores especializados para la compresión de video. Si se desea tener mayor control con el procesamiento de la información es necesario contar con acceso a su API o ejecutar el software de manera paralela, sin mucho control sobre su configuración.
- **Librerías de capturas sucesivas de imágenes**: Librerías utilizadas por desarrolladores para sacar capturas sucesivas de la pantalla de una computadora, cuyo objetivo es la velocidad y el consumo bajo de recursos. Algunos ejemplos son Multiple ScreenShots de Python, Auto-Screenshot de Android y las funciones nativas de sistemas operativos.

d) ***Limitaciones***

- Mientras mayor sea la resolución o tamaño de la salida de video de la computadora, mayor cantidad de píxeles se deberán copiar. Para aligerar la carga resultante de este proceso es recomendable utilizar una tarjeta de video dedicada.

5.3.3 Kinect

a) ***Descripción del dispositivo***

Dispositivo de video de bajo costo que alimenta a una computadora con dos tipos de imagen, cada una con una determinada función (figura 5-8). Cuenta con dos cámaras y cuatro micrófonos: una cámara es utilizada para obtener imágenes a color, y la segunda

obtiene la profundidad de los objetos por medio de un transmisor de ondas infrarrojas de corto alcance.

Figura 5-8. Dispositivo Kinect.

Además de obtener estas imágenes, el SDK que provee Microsoft genera otros flujos de información extra, como la generación del esqueleto de las personas que detecte al mismo tiempo, y la creación de un mapa detallado de puntos del reconocimiento del rostro de una persona (Microsoft, 2012).

b) ***Beneficios***

Los flujos de información del dispositivo y de su SDK permiten obtener información detallada acerca del esqueleto y la postura de una persona en pruebas donde no existan restricciones de libertad de movimiento, además de obtener un flujo de video y audio que permiten analizar el ambiente con el que interactúa.

c) ***¿Cuál es la forma en la que se obtienen sus datos?***

Este dispositivo utiliza la técnica del emisor infrarrojo de corta frecuencia para obtener información del movimiento corporal.

La forma principal de obtener su flujo de información es utilizando el **SDK** y las librerías que proporciona Microsoft. Sin embargo, debido a los requisitos de uso del SDK, el programa sólo se desarrolla y ejecuta en ambientes Windows.

Para sistemas operativos Linux y Mac OS es necesario utilizar APIs que utilicen ingeniería inversa para obtener los datos crudos del dispositivo (The OpenKinect community, 2000). Por ejemplo, librerías como **OpenNi** (http://openni.org), **OpenKinect** (https://openkinect.org/wiki/Main_Page) y **OpenCV** (https://opencv.org/).

d) ***Limitaciones***

- Las librerías del SDK de Microsoft tienen un consumo de recursos de procesamiento pesado por el procesamiento del esqueleto de las personas que detecta, lo cual es crítico en equipos que se encuentren trabajando simultáneamente con múltiples dispositivos de entrada de datos.
- El acceso limitado a datos crudos de librerías no oficiales evita su implementación en sistemas y soluciones multiplataforma.

5.4 Captura de datos de audio

Método de grabación de datos utilizado para obtener información auditiva acerca del ambiente y los participantes de un momento dado.

A partir de esta información se detectan mensajes, eventos y anomalías generadas a partir de cualquier individuo u objeto visible o no visible localizado en el ambiente donde se encuentra el sensor.

5.4.1 Micrófono

a) ***Descripción del dispositivo***

Dispositivo transductor que convierte sonido a señales eléctricas, ya sea por medio de un cable rodeado por un campo magnético, un diafragma vibrante con un plato capacitor, o un cristal de material piezoeléctrico que reacciona con la presión (Audio-Technica U.S., Inc., 2005) (figura 5-9).

Figura 5-9. Micrófono.

En la actualidad este tipo de dispositivo se encuentra comúnmente integrado con cámaras de video como opciones de grabación de audio básico, mientras que alternativas individuales permiten obtener frecuencias auditivas más precisas.

b) ***Beneficios***

Debido a que permite obtener información relevante acerca del ambiente, mensajes de los individuos y elementos no visibles del mismo, además de incluirse en conjunto de cámaras web, se incluye como elemento esencial del laboratorio de pruebas.

c) ***¿Cuál es la forma en la que se obtienen sus datos?***

Dependiendo del mecanismo, este dispositivo recibe las vibraciones o los cambios de precio generados a partir de la generación de sonidos en el ambiente.

La forma principal de obtener su flujo de información es utilizando librerías de acceso a los datos crudos de audio generados por el dispositivo de entrada de audio predeterminado. Un ejemplo de dichas librerías es la librería **PyAudio** (https://people.csail.mit.edu/hubert/pyaudio/), la cual permite configurar el buffer de datos que se recibe en lenguaje Python (Pham, 2006).

d) ***Limitaciones***

- El costo del dispositivo se relaciona con la calidad y el rango de frecuencia sonora que puede captar el mecanismo del micrófono.
- El módulo de software que tiene acceso directo con el dispositivo o los *drivers* del mismo pueden limitar el rango de frecuencias obtenidas por el dispositivo, con el fin de reducir la necesidad de procesar cualquier sonido detectado.

5.5 Captura de datos de sensores ambientales

Método de grabación de datos obtenidos por medio de sensores especializados en detectar determinadas variables del ambiente, como la temperatura o el nivel de humedad.

5.5.1 Sensor de humedad y temperatura DHT11

a) ***Descripción del dispositivo***

Sensor utilizado para obtener una lectura relacionada con la temperatura y la humedad del ambiente (figura 5-10).

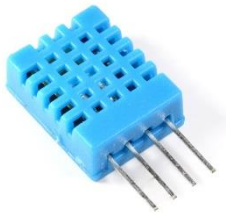


Figura 5-10. Sensor de temperatura DHT11 (Seeedstudio)

El componente que mide la temperatura es una resistencia variable que cambia en relación con la temperatura, basándose en el Coeficiente de Temperatura Negativa, mientras que el componente que mide la humedad del ambiente se conforma de dos electrodos y una superficie entre ambos electrodos que atrapa dicha humedad para generar resistencia (Seeedstudio).

b) ***Beneficios***

A partir de estudios realizados a lo largo del siglo XX, entre ellos el de la Sociedad de Psicología Británica (Howarth & Hoffman, 1984) donde se realizan experimentos para demostrar que estos elementos afectan a los estados cognitivos de una persona, principalmente la relación entre la humedad y la concentración. En base a esto, capturar la información relacionada con la temperatura y la humedad ambiental es de gran utilidad para verificar que la prueba se realizó en un ambiente adecuado.

Este componente de bajo costo permite integrar un sensor de temperatura y de humedad a un ambiente de prueba controlado, detectando cualquier anomalía generada durante la aplicación de un experimento.

c) ***¿Cuál es la forma en la que se obtienen sus datos?***

Este dispositivo obtiene la información por medio de las reacciones físicas y químicas de sus componentes, trasmitiéndolas a través de un medio digital para ser recibidas por un microcontrolador.

Existen múltiples formas de recibir la información, sin embargo, todas se centran en la comunicación de un dispositivo intermediario que traduzca la información en valores numéricos, transmitiéndola por medio de una conexión serial o similar.

d) ***Limitaciones***

1. El sensor de temperatura cuenta con una precisión de ± 2 °C y funciona en un rango de 0 °C a 50 °C.
2. El sensor de humedad cuenta con una precisión de ± 5 % y funciona en un rango de 20 % a 80 %.

5.5.2 Sensor de luminosidad TSL2561

a) ***Descripción del dispositivo***

Sensor utilizado para obtener una lectura relacionada con la intensidad de la luz del ambiente (figura 5-11).

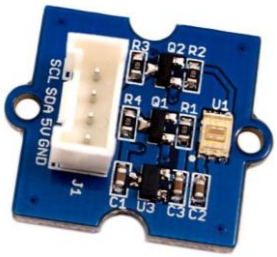


Figura 5-11. Sensor de luminosidad TSL2561
(Seeedstudio, 2012)

El componente que mide la intensidad de la luz son dos diodos sensibles al espectro infrarrojo y al espectro completo (MikeGrusin).

b) ***Beneficios***

Este componente de bajo costo permite integrar un sensor de intensidad de luz a un ambiente de prueba controlado, detectando cualquier anomalía generada durante la aplicación de un experimento.

c) ***¿Cuál es la forma en la que se obtienen sus datos?***

Este dispositivo obtiene la información por medio de las reacciones físicas de sus componentes, trasmitiéndolas a través de un medio digital para ser recibidas por un microcontrolador (MikeGrusin).

Existen múltiples formas de recibir la información, sin embargo, todas se centran en la comunicación de un dispositivo intermediario que traduzca la información en valores numéricos, transmitiéndola por medio de una conexión serial o similar.

d) ***Limitaciones***

3. El sensor puede detectar luz de entre 0.1 y 40,000 lux.
4. El sensor trabaja en temperaturas entre -40°C y 85°C.

5.5.3 Sensor de ruido LM386

a) ***Descripción del dispositivo***

Sensor utilizado para obtener una lectura relacionada con la intensidad del sonido del ambiente (figura 5-12).

Figura 5-12. Sensor de ruido
LM386 (Seeedstudio, 2017)

El componente que mide la intensidad del sonido es un amplificador que actúa como un micrófono sencillo. Este micrófono tiene el objetivo de medir, no de realizar grabaciones de audio (Seeedstudio, 2017).

b) ***Beneficios***

Este componente de bajo costo permite integrar un sensor de intensidad de sonido a un ambiente de prueba controlado, detectando cualquier anomalía generada durante la aplicación de un experimento.

c) ***¿Cuál es la forma en la que se obtienen sus datos?***

Este dispositivo obtiene la información por medio de las reacciones físicas de sus componentes, trasmitiéndolas a través de un medio digital para ser recibidas por un microcontrolador (Seeedstudio, 2017).

Existen múltiples formas de recibir la información, sin embargo, todas se centran en la comunicación de un dispositivo intermediario que traduzca la información en valores numéricos, transmitiéndola por medio de una conexión serial o similar.

d) ***Limitaciones***

El sensor sólo mide la intensidad del sonido ambiental, no hace diferencia entre frecuencias para realizar el papel de un micrófono en una grabación de audio.

Capítulo 6. Diseño de la metodología de solución

En este capítulo se describe la manera en la que se diseñó la metodología de solución para resolver la problemática identificada en esta investigación.

Para analizar la información obtenida de un determinado momento a partir de múltiples dispositivos de entrada de datos localizados en un mismo ambiente de prueba es necesario distribuir los datos resultantes dentro de una línea de tiempo, la cual debe contener elementos sincronizados que representan estados o eventos importantes o significativos. Para ello se hace referencia al primer artículo del estado del arte (Lambert, Hachicha, Ahmed, Pinna, & Garda, 2015), el cual describe la aplicación de estampas de tiempo dentro de un contenedor con múltiples tipos de flujo de información, generando una línea de tiempo abstracta, donde cada muestra vinculada por una estampa define un momento u evento especifico.

Las modalidades en las que se pueden distribuir los dispositivos de entrada de un ambiente de prueba de evaluación de la experiencia del usuario en base a su información son: visual, auditiva, personal y ambiental.

Por ejemplo, un dispositivo de video puede ser utilizado para obtener información visual de las acciones de una persona, así como detectar sus expresiones faciales y señales inconscientes, mientras que la información personal obtenida por un dispositivo EEG complementa dicha información con la forma en la que se van activando sus neuronas y reconociendo los patrones emocionales y cognitivos de las áreas donde se activan.

Los elementos con los que se cuentan para armar un ambiente de pruebas para la evaluación centrada en el usuario (Figura 6-1) son:

- Dispositivo de video tipo cámara web para grabar la información visual dirigida hacia el rostro del usuario de prueba. Se obtiene información de sus expresiones faciales y eventos visuales del ambiente durante el transcurso de la prueba.
- Dispositivo de audio integrado en la cámara web para grabar la información auditiva proporcionada por el usuario de prueba, sus acciones y su ambiente. Se obtienen pruebas auditivas que contienen información de técnicas Think-Aloud (pensar en voz alta) y Cognitive Walkthrough (explicar procedimiento en voz alta).

- Flujo de información visual generado a partir de la salida de video de la pantalla de la computadora del experimento. Muestra la interacción del usuario y la computadora.
- Dispositivo de detección de movimiento ocular que obtiene información visual de los ojos del usuario para calcular internamente las coordenadas de su mirada y atributos físicos de cada ojo en un determinado momento.
- Dispositivo EEG que obtiene información personal de la actividad eléctrica cerebral cercana a los electrodos del dispositivo. Se obtiene información que describe la actividad eléctrica dividida en frecuencias, las cuales son utilizadas en conjunto de algoritmos de reconocimientos de patrones para detectar estados cognitivos y emocionales.
- Sensores ambientales, como el de humedad y temperatura, ruido y luminosidad, que obtienen información ambiental por medio de fenómenos físicos y químicos de materiales sensibles.

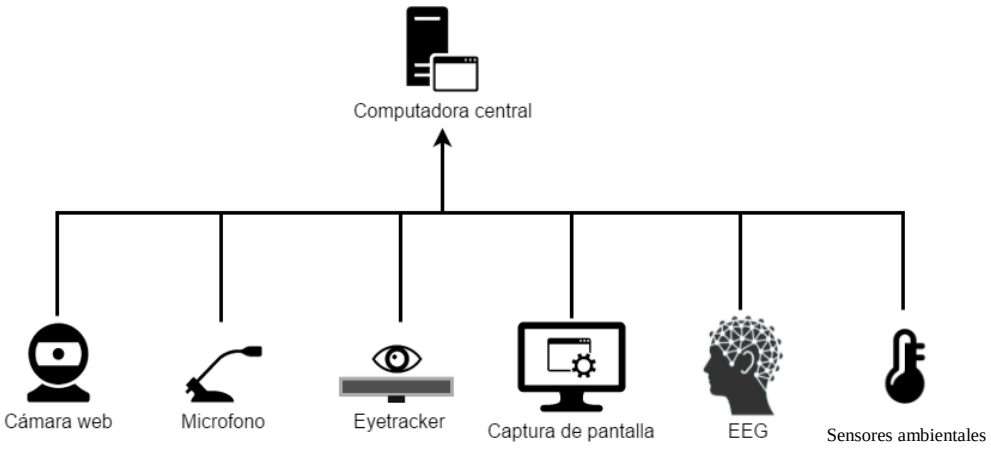


Figura 6-1. Diagrama visual de dispositivos de entrada que se comunican con la computadora central del ambiente de pruebas (Arana Llanes, 2014).

Para realizar la grabación simultánea y sincronizada de los dispositivos de entrada del ambiente de prueba fue necesario diseñar un sistema que se comunique simultáneamente con todos los dispositivos, obtenga información de ellas y las almacene en memoria con su respectivo procesamiento.

En la figura 6-2 se muestra la manera en la que se diseñaron los módulos de grabación, sincronización e integración de la información de los dispositivos.

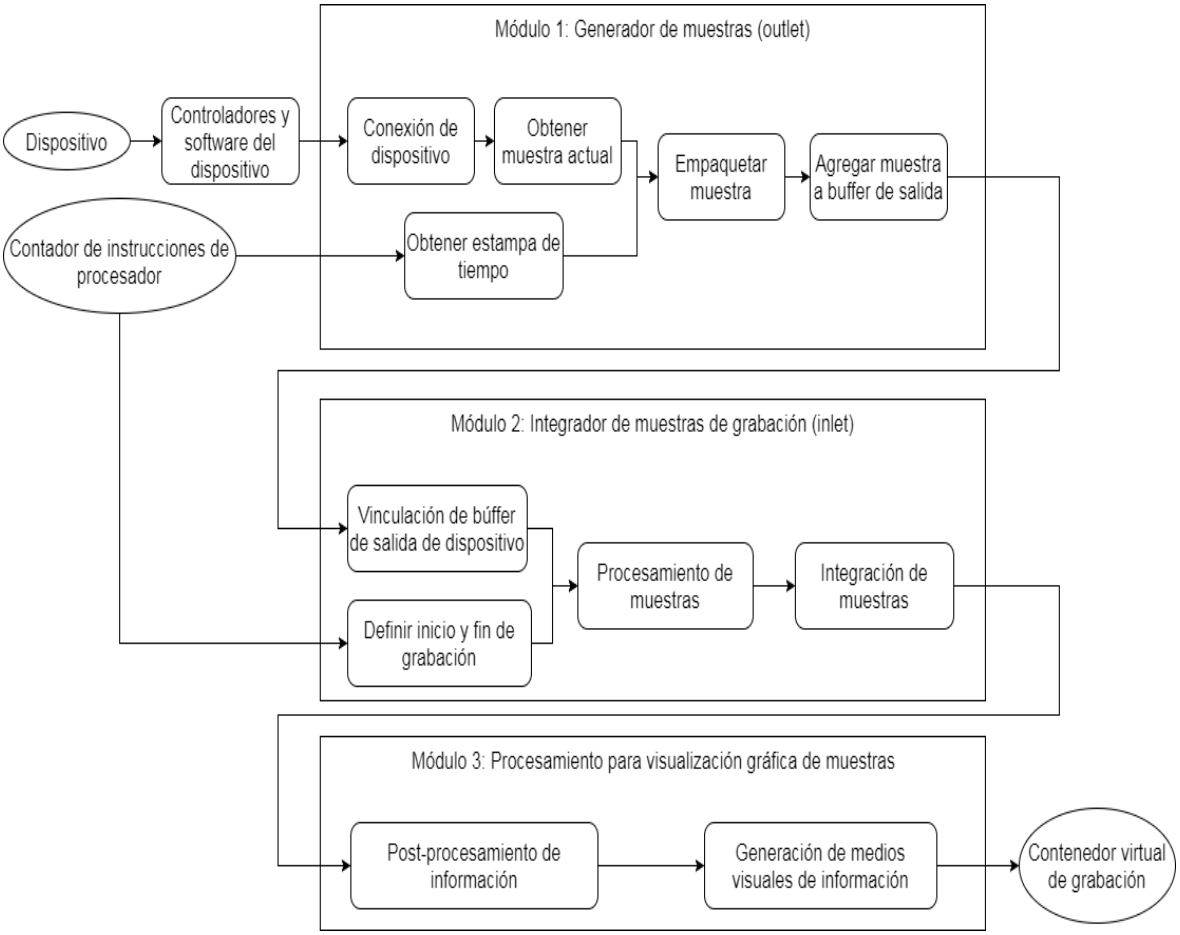


Figura 6-2. Diagrama de los módulos diseñados para la metodología de solución.

6.1 Módulo 1: Generación de muestras (outlet)

En este módulo (figura 6-3) se obtienen las muestras más actuales de un dispositivo, la cual se obtiene por medio de la conexión a los controladores y/o el software del mismo. Se etiqueta cada muestra con una estampa de tiempo generada a partir de la cantidad de instrucciones ejecutadas por el procesador hasta el momento, agregándolo finalmente a un búfer de salida de datos para el siguiente módulo.

Se debe de mantener actualizado la muestra más actual del dispositivo, independientemente de la velocidad de etiquetación de muestras.

La estampa de tiempo se calcula por medio de la técnica definida en (Zhou, Zheng, H., & Yang, 2013), la cual describe el uso de los contadores de instrucciones ejecutadas en procesadores modernos, los cuales tienen la capacidad de ejecutar constantemente millones de instrucciones por segundo. Debido a esto, el cálculo de tiempo a partir de esta referencia llega a tener una precisión de microsegundos.

Finalmente, la muestra etiquetada con la estampa de tiempo se agrega a un búfer o cola de muestras de "salida", la cual cuenta con la capacidad de ser compartida con los siguientes módulos y permite identificarse con metadatos del dispositivo y el tipo de información que maneja.

Por ejemplo, un dispositivo de video obtiene una secuencia de imágenes, por lo que este módulo debe conseguir continuamente la última imagen capturada por el dispositivo y guardarla en una variable de fácil acceso, para así tener un hilo que esté empujando imágenes al búfer a un ritmo constante y consistente.

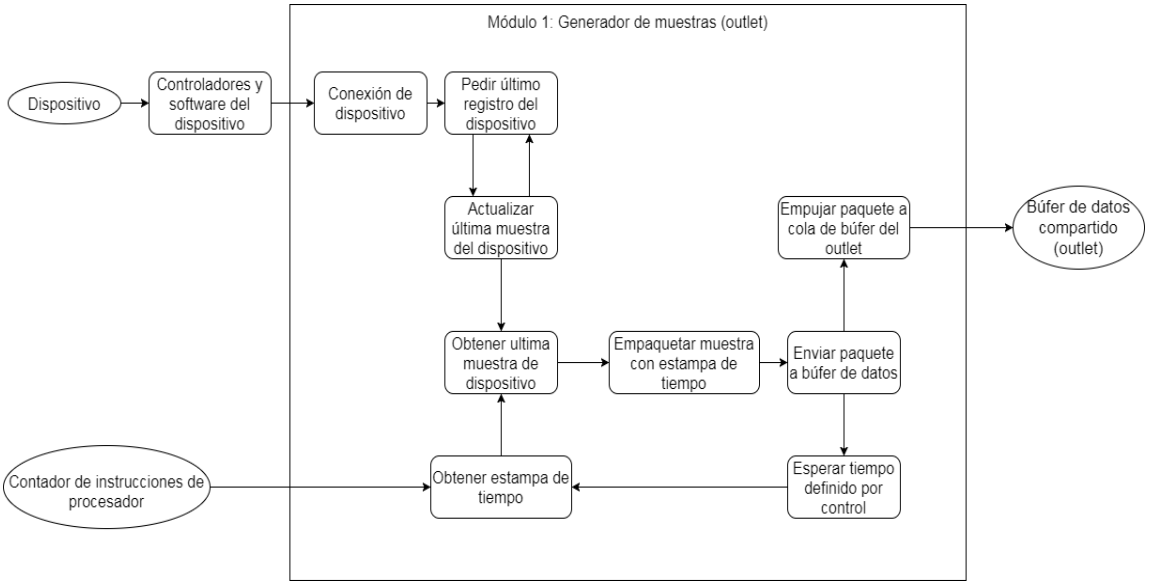


Figura 6-3. Diagrama detallado del módulo 1.

6.2 Módulo 2: Integrador de muestras de grabación (inlet)

En este módulo (figura 6-4) se obtienen los búferes de salida de los dispositivos del módulo anterior, de los cuales se obtienen los metadatos del dispositivo al que representan y a qué tipo de procesamiento deberá someterse su información. Finalmente se almacenan las muestras en un formato correspondiente al tipo de información que representan.

Para identificar qué muestras se deben de tomar en cuenta para la grabación, se utiliza el contador de instrucciones del procesador para indicar desde qué momento se inicia y en qué momento se detiene, ignorando cualquier otra muestra que no corresponda a ese rango de tiempo.

Por cada dispositivo se debe de identificar un buffer de salida, el cual contiene metadatos acerca del tipo de procesamiento al que se someten las muestras. Por ejemplo, un dispositivo de video genera un buffer de imágenes, las cuales deben de ser procesadas por un codificador de video, resultando en un contenedor multimedia de video como un MP4, MKV o AVI.

Por ejemplo, este módulo detecta el búfer del módulo anterior del dispositivo de video. El búfer trae metadatos que permiten identificar el tipo de procesamiento al que se someten las muestras, así como las características de las imágenes. Sólo se toman en cuenta las muestras que entren dentro del rango de tiempo que seleccione el usuario, tomando la apariencia de botones de inicio y fin de grabación. Finalmente, las muestras procesadas se integran con alguna herramienta que permita generar un archivo multimedia MP4 o similar.

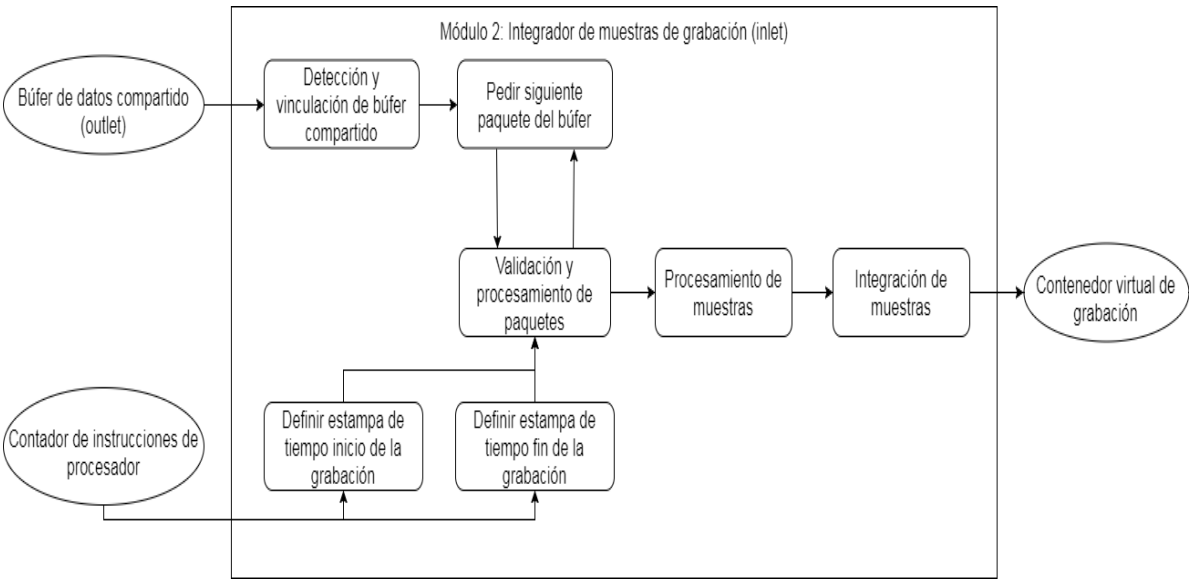


Figura 6-4. Diagrama detallado del módulo 2.

6.3 Módulo 3: Procesamiento para visualización gráfica de muestras

En este módulo (figura 6-5) se realiza el postprocesamiento y generación de medios visuales para su posterior consulta en un reproductor multimedia de consulta paralela.

Primero se realiza la limpieza y generación de datos cualitativos a partir de la información obtenida por un dispositivo, como la detección de patrones o posibles pronósticos.

A partir de la información resultante, se generan medios visuales por los cuales se aprecia más fácilmente la información, como los mapas de trazado o de calor para el movimiento ocular, además de la combinación de dos o más fuentes de información para su fácil manipulación.

Por ejemplo, aquí se detectan los archivos de video de la captura de pantalla y la información de movimiento ocular. Se genera una animación MP4 de mapas de calor con la información del segundo archivo. Para combinarlos se utiliza una herramienta de software de modificación de video que reciba ambos archivos, se les asigna una

configuración para darles un efecto de transparencia y sobreponer la animación de mapa de calor con transparencia sobre la captura de pantalla sin transparencia.

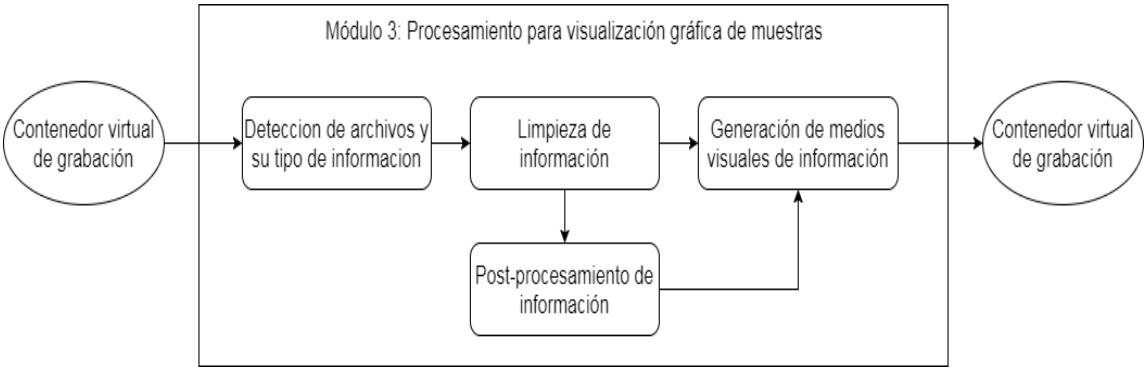


Figura 6-5. Diagrama detallado del módulo 3.

6.4 Diseño del paquete de muestra de un dispositivo y la generación de estampas de tiempo

Tomando en cuenta que cada muestra generada se deberá de etiquetar con estampas de tiempo, el proceso de generación de estampas de tiempo (figura 6-6) se describe de la siguiente manera:

1. Primero se obtiene el número de pasos recorridos por el contador del procesador, el cual indica cuántas instrucciones se han ejecutado desde que se encendió la computadora.
2. Se obtiene el número de instrucciones por segundo que se ejecutan en el procesador, calculado con el contador el tiempo desde que se encendió el CPU.
3. Se obtiene la muestra con la librería que realiza la conexión al dispositivo y se combinan en un mismo arreglo, representando un mensaje al ser introducido al búfer de datos del outlet.

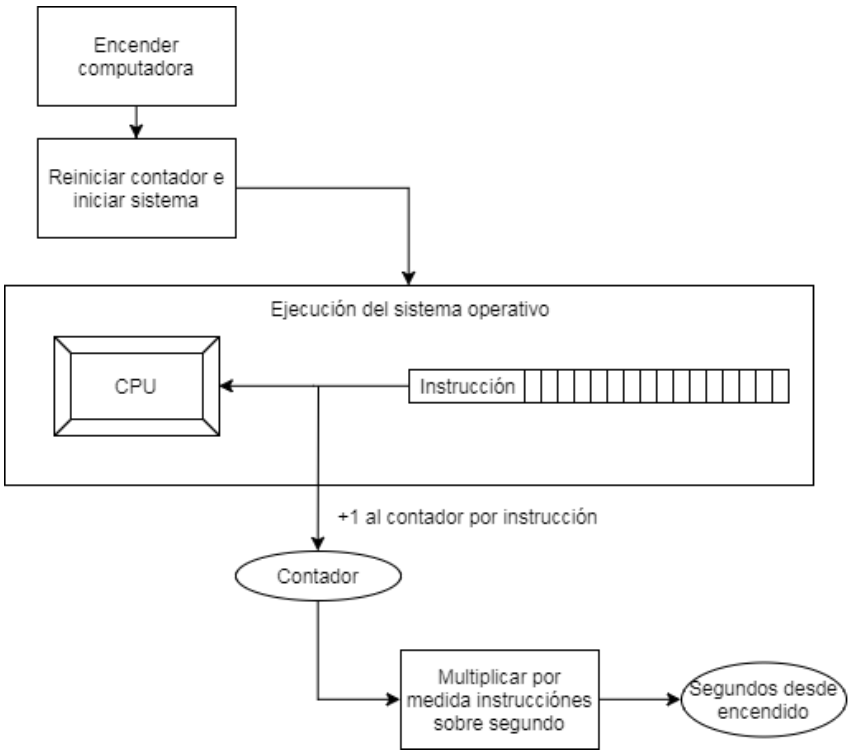


Figura 6-6. Representación del uso de contador de procesador para calcular estampas de tiempo de alta precisión, las cuales se representan en segundos, milisegundos o nanosegundos.

Para visualizar la manera en la que se estructura de cada muestra es necesario mostrar la manera en la que el primer y el segundo módulo se comunican el búfer compartido (figura 6-7), la cual consiste del uso de dos entidades socket, una de salida (generación) y una de entrada (integración). Para comunicarse entre ellos, los sockets se envían metadatos que describen a qué dispositivo representan, qué tipo y tamaño de datos manejan, así como datos más específicos para el procesamiento. Una vez que se identifica el tipo de datos que manejaran, el socket de salida envía las muestras de su búfer al socket de entrada, donde encapsula cada estampa de tiempo etiquetada con el valor de la muestra dentro de un mensaje.

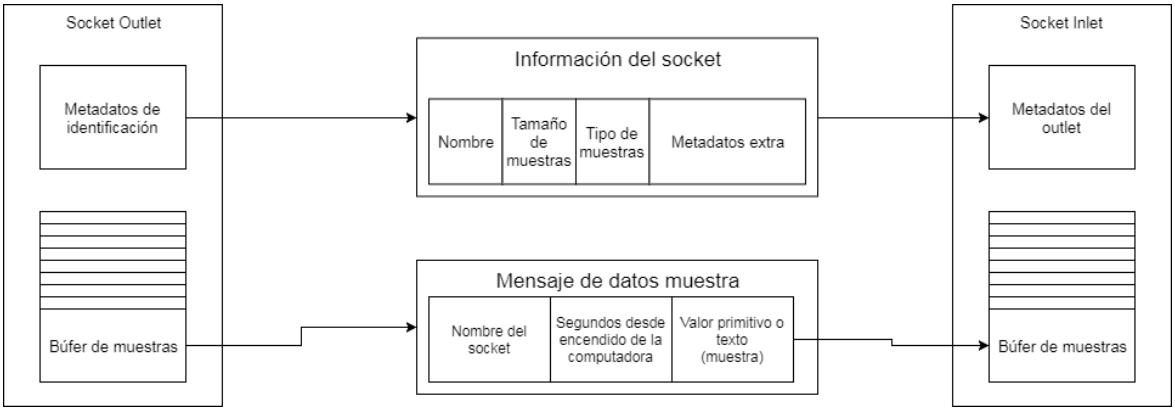


Figura 6-7. Representación de la comunicación de sockets de comunicación entre la primera fase y la segunda fase, con el contenido de los mensajes de identificación de socket de salida y mensajes de muestras de datos.

6.5 Diseño de interfaz gráfica

A continuación se muestran los bocetos base utilizados para la creación de la interfaz gráfica de usuario, la cual involucra dos ventanas: la primera ventana se asocia con el sistema de grabación completo (figura 6-9) y la segunda ventana se asocia con el reproductor paralelo multimedia (figura 6-10). Ambas son utilizadas por el usuario que realiza la evaluación de la experiencia del usuario, ya que este último no controla la grabación.

La ventana de grabación (figura 6-9) contiene una visualización de cada flujo de información de los dispositivos en tiempo real, junto con los botones de grabación y la ruta de almacenamiento de la grabación. Esta ventana es mostrada mientras se realiza la evaluación de la experiencia del usuario, donde el evaluador la visualiza e interactúa con ella desde un monitor diferente a la que utiliza el usuario que participa en la prueba (figura 6-8).

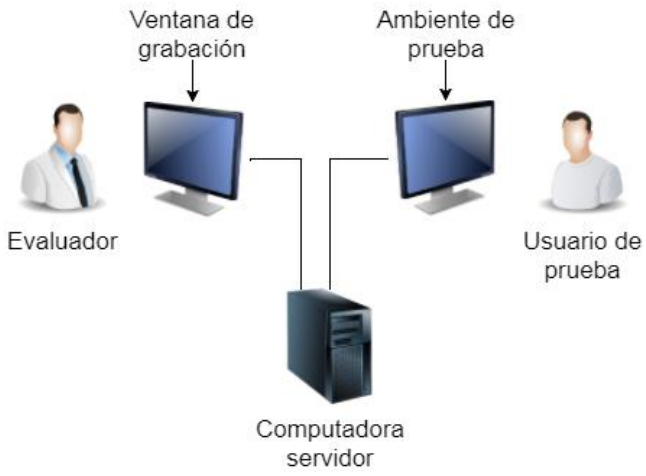


Figura 6-8. Representación de la aplicación de la evaluación de la experiencia del usuario de una herramienta de software con el sistema de grabación.

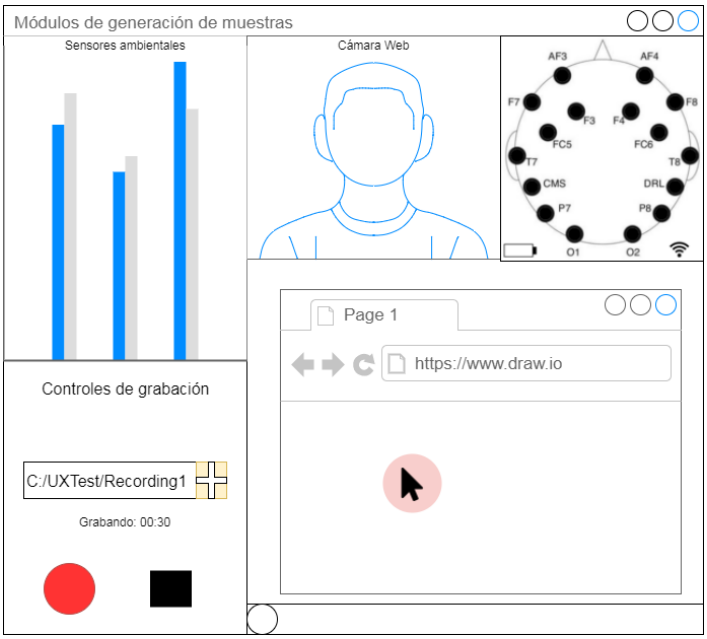


Figura 6-9. Interfaz gráfica de la ventana de módulos de grabacion en tiempo real.

Mientras tanto, la ventana de reproducción (figura 6-10) permite consultar la información grabada después de realizar la evaluación de la experiencia del usuario. Se toma como inspiración el diseño de un reproductor multimedia común, así como la interfaz de la herramienta iMotions.

Figura 6-10. Interfaz gráfica de la ventana de reproductor de grabaciones realizada, posteriormente después de haberse realizado la evaluación.

En conclusión, se diseñó un sistema con arquitectura modular que permita agregar o modificar la funcionalidad de los componentes, según la necesidad de integrar nuevas tecnologías o extender las características de las que ya se encuentren implementadas. Los módulos de grabación se encargan de obtener la información consistentemente de distintos dispositivos y almacenarla en un contenedor virtual, mientras que la parte del visualizador permite consultar posteriormente toda la información de un contenedor de manera simultánea.

Capítulo 7. Implementación de la metodología

En este capítulo se describe la manera en la que se implementó la metodología diseñada en el capítulo anterior. Se mencionan las problemáticas encontradas y la solución utilizada.

Para implementar la herramienta de grabación se utilizó el lenguaje de programación Python junto con la librería LabStreamingLayer (LSL) (Swartz Center for Computational Neuroscience (SCCN), UCSD, 2012), la cual permite generar un buffer de datos con estampas de tiempo de alta precisión en cualquier lenguaje y sistema operativo.

7.1 Arquitectura

Se tomó como referencia la especificación de los modelos de sincronización de sistemas multimedia de (Blakowski & Steinmetz, 1996), conformando la arquitectura de la siguiente manera (figura 7-1):

- Un módulo de acceso y control de información proveniente del dispositivo, el cual se encarga de obtener el flujo de información de un dispositivo específico a un ritmo constante y consistente, generando un buffer de datos.
- Un módulo de procesamiento de información que obtenga los datos, identifique la clase de procesamiento al que se deben someter para almacenarse y finalmente los localice dentro de una línea de tiempo, descartando la información que no pertenezca a la línea de tiempo seleccionada.
- Un módulo de postprocesamiento e integración de información, que permite realizar un proceso de refinación y clasificación de datos complejos, generar representaciones visuales de la información biométrica y combinación de flujos de información compatibles.

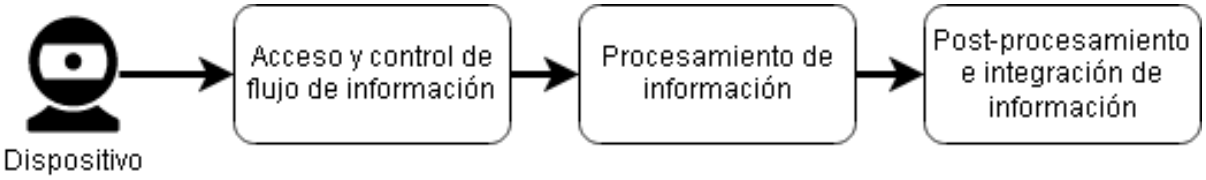


Figura 7-1. Comunicación de módulos de grabación.

7.1.1 Módulo de acceso y control de información

Se desarrolló un módulo de acceso por cada dispositivo, utilizando sus APIs para obtener las muestras generadas más actuales, y utilizando la generación de estampas de tiempo y los "outlets" de la librería LSL para generar los buffers de datos.

A continuación se describe la estructura de estos módulos de acceso y control de información (figura 7-2, 7-3, 7-4, 7-5 y 7-6):

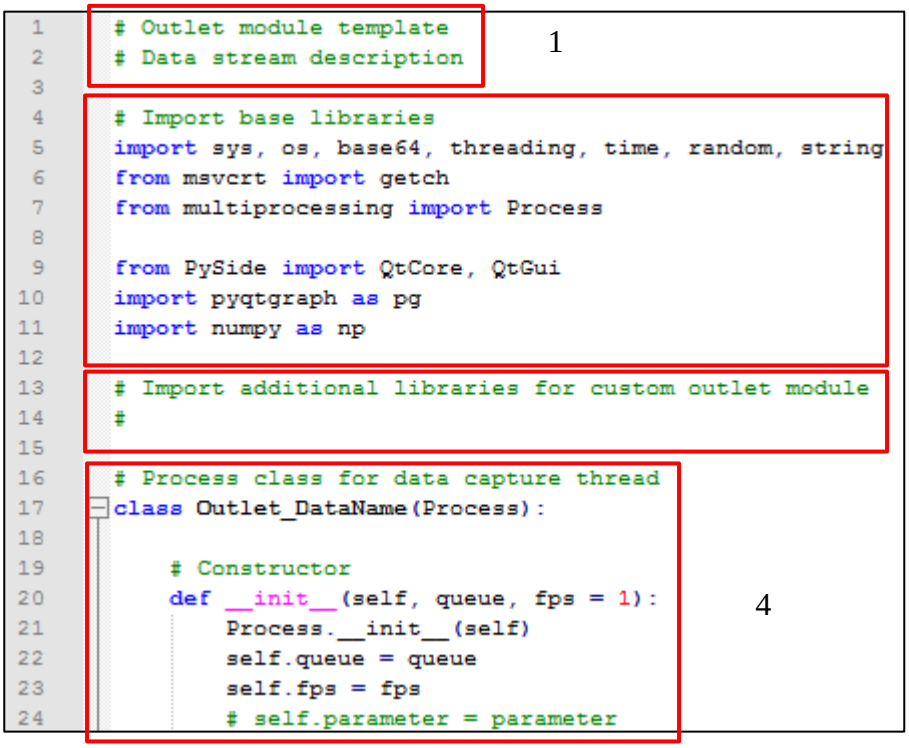
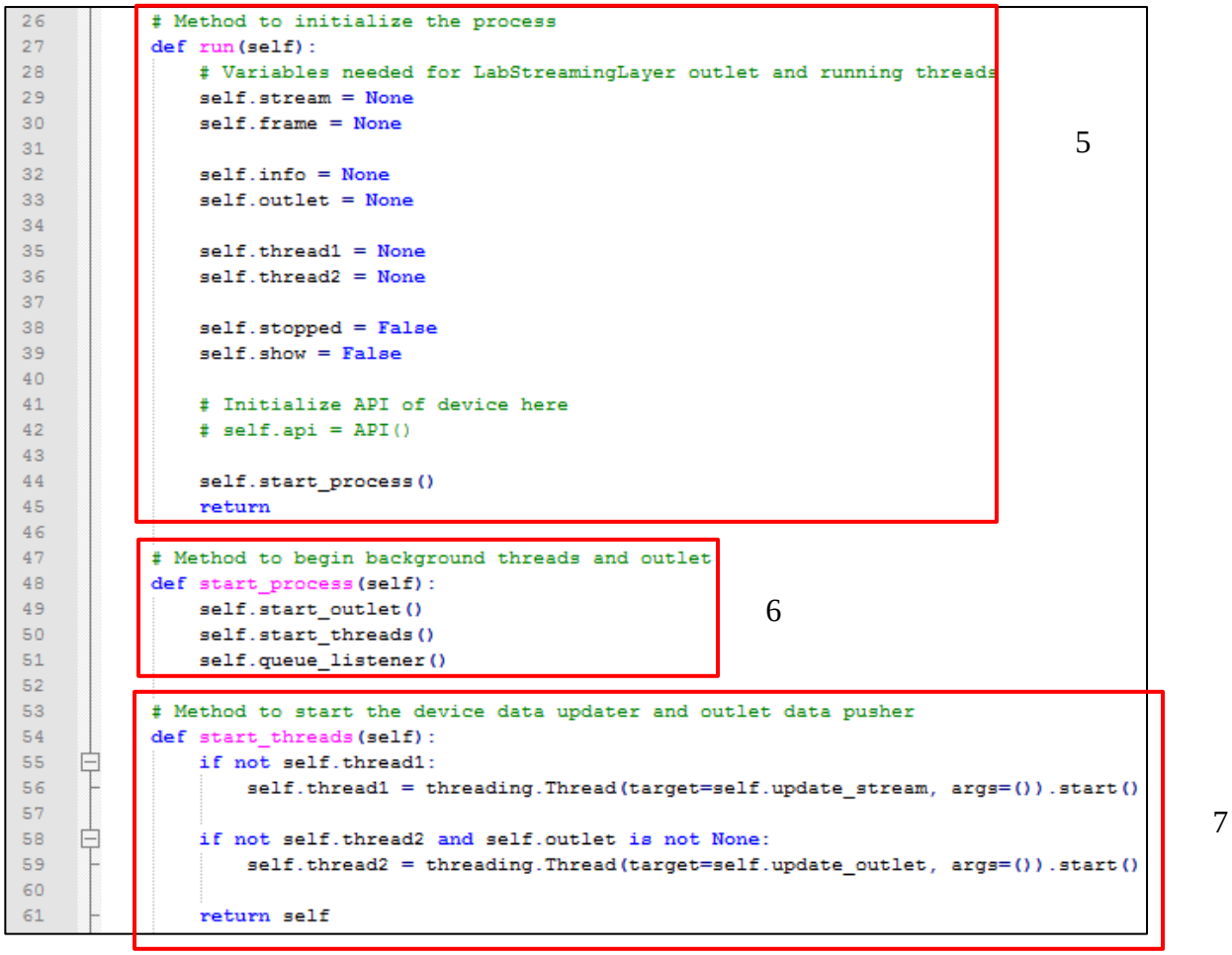


Figura 7-2. Captura de módulo de datos de generación de datos 1.

1. Descripción del dispositivo o flujo de información al que pertenece la clase.
2. Importación de clases necesarias para que funcionen los outlets de LabStreamingLayer.
3. Importación de librerías opcionales para el manejo de datos del dispositivo.
4. Clase principal que hereda de la clase 'Process' para ejecutarse como un ejecutable más. Incluye su constructor donde se deben definir los parámetros o variables configurables del módulo del dispositivo.




```
26      # Method to initialize the process
27      def run(self):
28          # Variables needed for LabStreamingLayer outlet and running threads
29          self.stream = None
30          self.frame = None
31
32          self.info = None
33          self.outlet = None
34
35          self.thread1 = None
36          self.thread2 = None
37
38          self.stopped = False
39          self.show = False
40
41          # Initialize API of device here
42          # self.api = API()
43
44          self.start_process()
45          return
46
47      # Method to begin background threads and outlet
48      def start_process(self):
49          self.start_outlet()
50          self.start_threads()
51          self.queue_listener()
52
53      # Method to start the device data updater and outlet data pusher
54      def start_threads(self):
55          if not self.thread1:
56              self.thread1 = threading.Thread(target=self.update_stream, args=()).start()
57
58          if not self.thread2 and self.outlet is not None:
59              self.thread2 = threading.Thread(target=self.update_outlet, args=()).start()
60
61          return self
```

Figura 7-3. Captura de módulo de generación de datos 2.

5. Método que inicializa los valores de variables y objetos del outlet y la API del dispositivo.
6. Método que inicia los hilos del outlet, actualización de datos y la cola de mensajes.
7. Método que inicia los hilos de actualización de muestra más reciente del dispositivo y el ingreso controlado de muestras al outlet.

```
    # Starts the outlet for streaming data with the information of the device
    def start_outlet(self, DeviceName = "Emotiv", DeviceID = None):
        if self.outlet is not None:
            return

        if not DeviceID:
            DeviceID = 'EEG_{}'.format(''.join(random.choice(string.ascii_lowercase + string.digits) for _ in range(8)))

        # Generate object with information about the stream
        self.info = StreamInfo(DeviceName, "EEG", self.dataCount, self.fps, "float32", DeviceID)

        # Add metadata to the stream
        # Channel array
        chns = self.info.desc().append_child("channels")

        # Main channel
        for label in self.electrodes:
            ch = chns.append_child("channel")
            ch.append_child_value("label", label)
            ch.append_child_value("unit", "microvolts")
            ch.append_child_value("type", "EEG")

        # Device info
        self.info.desc().append_child_value("manufacturer", "Emotiv")
        cap = self.info.desc().append_child("cap")
        cap.append_child_value("name", "Epoc+")

        # CSV Headers (if applicable)
        #headers = "timestamp,signal,battery,"
        #frequency = ""
        #for area in self.electrodes:
        #    headers += "{}_quality,".format(area.lower())
        #    frequency += "{}_theta,{}_alpha,{}_lowbeta,{}_highbeta,{}_gamma,".format(area.lower(),area.lower(),area.lower(),area.lower(),area.lower())
        #headers += frequency
        #headers += "longterm_excitement,instantaneous_excitement,stress,engagement_boredom,interest,focus"
        #self.info.desc().append_child_value("csv_headers", headers)

        # Generate stream with the previous information
        self.outlet = StreamOutlet(self.info)
```

8

Figura 7-4. Captura de módulo de generación de datos 3.

8. Método que inicializa el outlet de LabStreamingLayer con los metadatos del dispositivo. En esta sección se ingresa el tipo de procesamiento por el que será sometida la información enviada por el outlet al usar la función StreamInfo. También se define el tipo de dato que se transportara (numérico, carácter, etc.) y la cantidad de elementos de la muestra (en el caso de enviar arreglos, de lo contrario se utiliza el valor 1).

```
103     # Updates the most recent data obtainable from the device's API
104     def update_stream(self):
105         # keep looping infinitely until the thread is stopped
106         while True:
107             # if the thread indicator variable is set, stop the thread
108             if self.stopped:
109                 # Stop API if needed
110                 return
111
112             # otherwise, read the next frame from the stream
113             self.frame = self.api.getFrame()
114
115     # Pushes data to the outlet
116     def update_outlet(self):
117         # keep looping infinitely until the thread is stopped
118         while True:
119             # if the thread indicator variable is set, stop the thread
120             if self.stopped:
121                 return
122
123             # otherwise, read the next frame from the stream
124             frame = self.read_sample()
125             timestamp = local_clock()
126
127             self.outlet.push_sample(sample, timestamp)
128
129             time.sleep(1 / self.fps)
130
131     # return the frame most recently read
132     def read_sample(self):
133         return self.frame
134
135     # Shows live data feed from the device
136     def show_image(self):
137         self.show = True
138
139         while True:
140             # draw most recent sample from the device
141             if not self.show:
142                 return
```

9

10

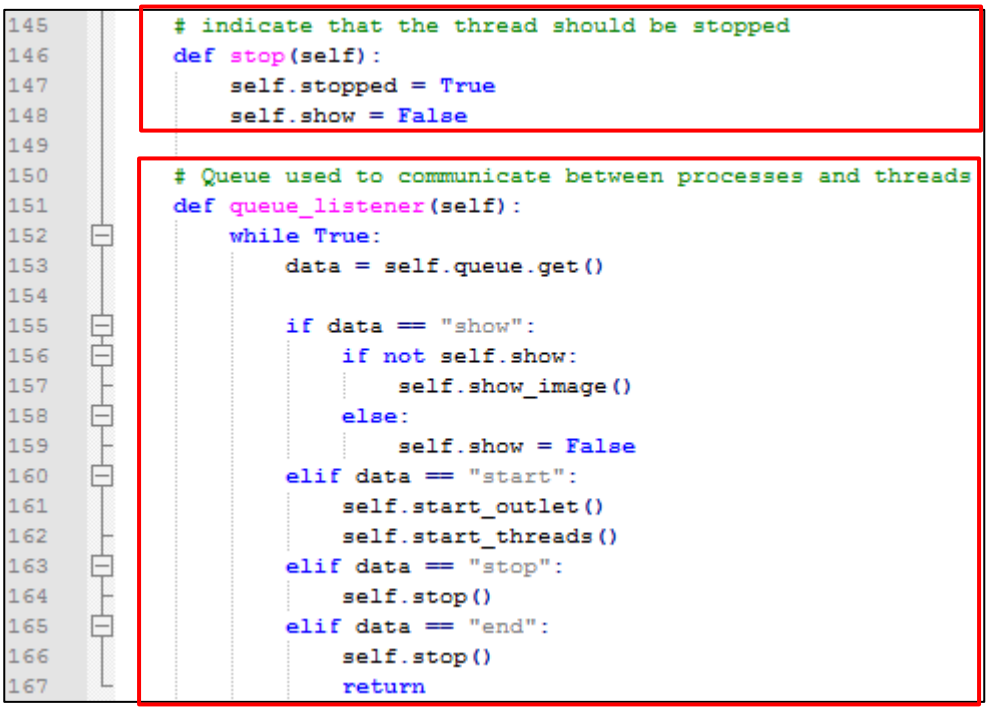
11

12

Figura 7-5. Captura de módulo de generación de datos 4.

9. Método que obtiene la última muestra del dispositivo, el cual puede ser un sólo valor o un arreglo de múltiples valores.
10. Método que agrega las muestras dentro de la cola del outlet, generalmente acompañando a la muestra con una estampa de tiempo obtenida manualmente. En este método se utiliza la sincronización “intra-objeto”.
11. Método que regresa la última muestra leída por el dispositivo.

12. Método que muestra en tiempo real la última muestra proporcionada por el dispositivo. Este método se debe de implementar de acuerdo a la información que se obtiene, ya sea en forma de gráficas o representaciones visuales.



```
145     # indicate that the thread should be stopped
146     def stop(self):
147         self.stopped = True
148         self.show = False
149
150     # Queue used to communicate between processes and threads
151     def queue_listener(self):
152         while True:
153             data = self.queue.get()
154
155             if data == "show":
156                 if not self.show:
157                     self.show_image()
158                 else:
159                     self.show = False
160             elif data == "start":
161                 self.start_outlet()
162                 self.start_threads()
163             elif data == "stop":
164                 self.stop()
165             elif data == "end":
166                 self.stop()
167                 return
```

13

14

Figura 7-6. Captura de módulo de generación de datos 5.

13. Método que detiene el proceso y sus hilos.
14. Método que administra los mensajes enviados por un proceso maestro, el cual es la interfaz gráfica que la controla u otro módulo.

En este módulo se aplica una sincronización a nivel de “intra-objeto” en la función de actualización de búfer, en la línea 129, donde cada determinado tiempo se obtiene una muestra de la información más reciente del dispositivo al que se accede (figura 7-7).

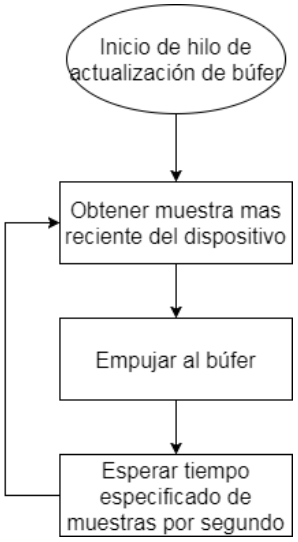


Figura 7-7. Diagrama de flujo de control de muestras para una sincronizacion de muestras "intra-objeto".

Para calcular el tiempo que debe existir entre cada muestra se utiliza la ecuación 2:

$$t = \frac{1}{Hz} * c \qquad \text{(Ecuación 2)}$$

Donde t es el tiempo en segundos, Hz es la cantidad de datos sobre segundo que soporta el dispositivo y c es una constante que es un número positivo menor o igual a 1 que permite tomar en cuenta el tiempo que se utiliza para recuperar la muestra.

7.1.2 Módulo de procesamiento e integración de información

Dentro de este módulo se utiliza la clase "inlet" de LSL para detectar y obtener el flujo de datos de los "outlets" activos. Cada "outlet" almacena metadatos del dispositivo al que se encuentra asociado, permitiendo detectar el tipo de procesamiento que le corresponde (figura 7-8).

Figura 7-8. Captura completa de estructura de módulo de integración.

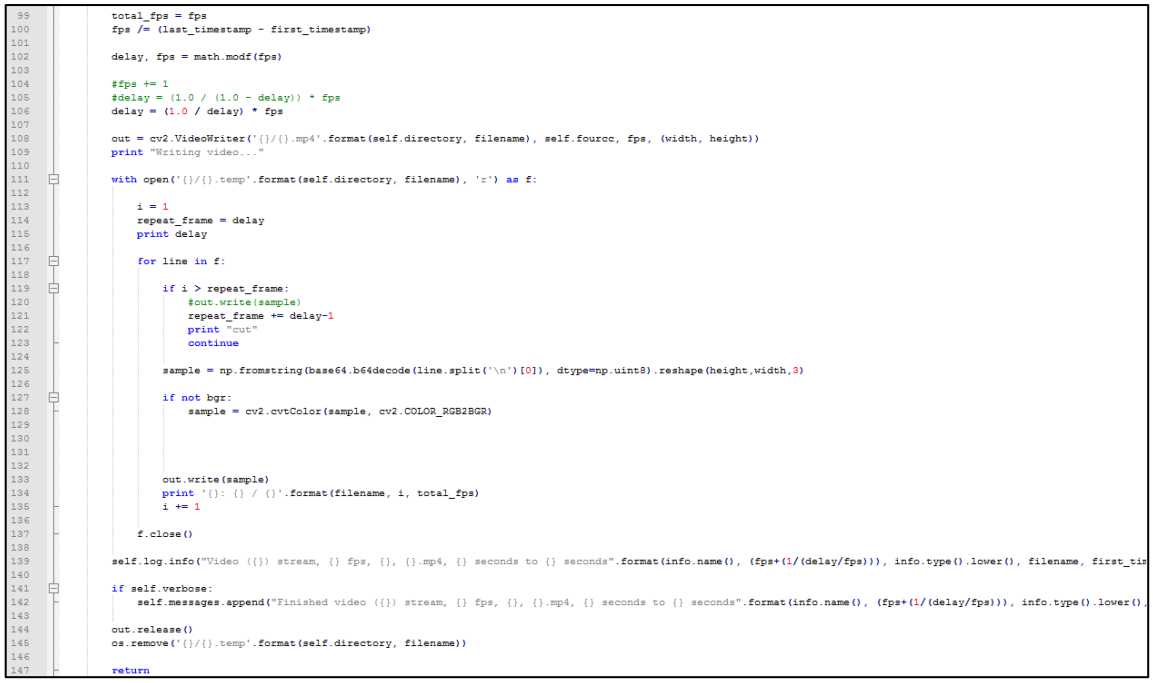
La clase principal de este módulo comprende de los siguientes métodos:

- _init_: Inicializa los valores del buscador de outlets de LabStreamingLayer.
- Video_inlet, audio_inlet y text_inlet: Métodos relacionados con el procesamiento especializado de muestras. En este caso se cuenta con el procesamiento de secuencias de imágenes a un formato de video, procesamiento de conjuntos de frecuencias auditivas a un formato de audio, y el procesamiento de información a un archivo de texto.

- Start_recording y stop_recording: Controlan el inicio y fin de la grabación, estableciendo la estampa de tiempo inicial y la estampa de tiempo final a considerar.
- Search_streams y update_streams: Buscan outlets dentro del ambiente del Sistema operativo de la computadora.
- Set_directory: Establece el directorio donde se guardarán los archivos procesados de la grabación.
- Read_messages: Escucha y lee los mensajes de los outlets.
- Start_inlets: Detecta los outlets disponibles, obtiene sus metadatos y distribuye cada outlet a un hilo de procesamiento de datos especifico según la información del outlet.

```
52      def video_inlet(self, inlet, filename = "output", timeout = 10):
53
54          info = inlet.info()
55          first_second = True
56          fps = 0
57          video_frames = []
58          first_timestamp = None
59          bgr = True
60
61          ch = info.desc().child("channels").child("channel")
62          width = int(float(ch.child_value("width")))
63          height = int(float(ch.child_value("height")))
64
65          if not ch.child_value("color_space").lower() == "bgr":
66              bgr = False
67
68          temp_file = open('{}/{}.temp'.format(self.directory, filename), 'w')
69          temp_file.truncate()
70
71          while True:
72
73              sample, timestamp = inlet.pull_sample(timeout)
74
75              if timestamp >= self.starttime:
76
77                  if timestamp <= self.endtime:
78
79                      if not first_timestamp:
80                          first_timestamp = timestamp
81
82                      temp_file.write('{}\n'.format(sample[0]))
83                      fps += 1
84                      last_timestamp = timestamp
85
86                      del sample
87
88                  else:
89                      break
90
91              elif not sample and not self.recording:
92                  break
93
94          temp_file.close()
95          inlet.close_stream()
96
97          del inlet
```

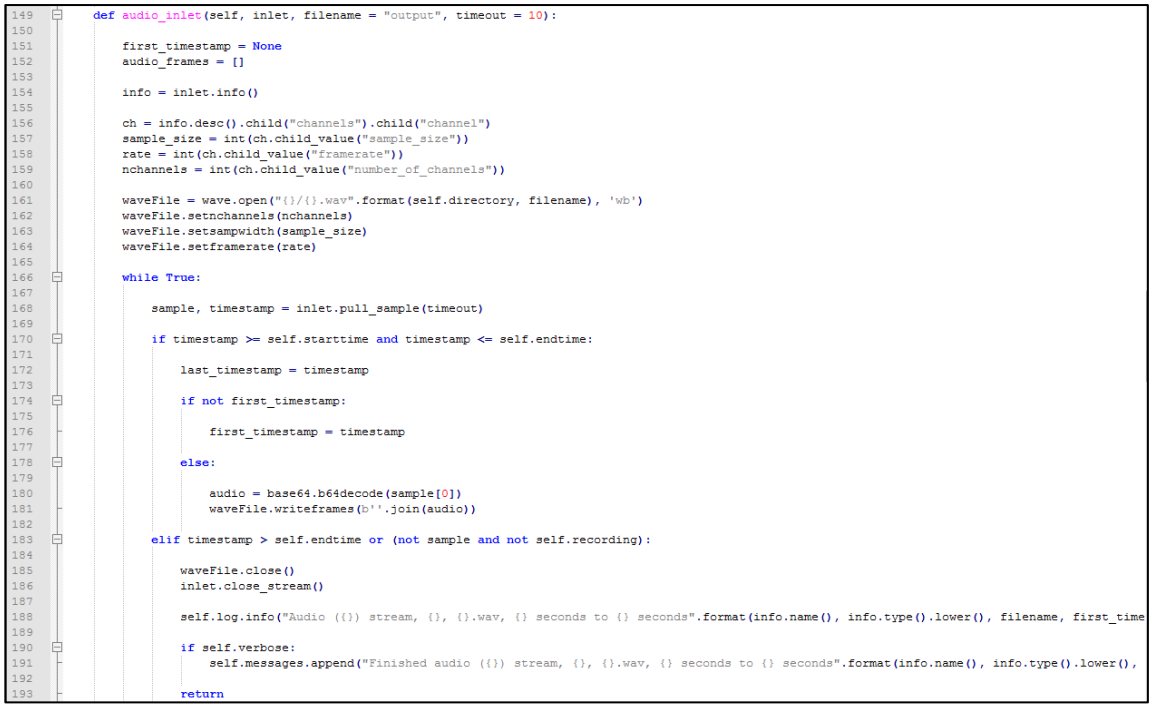
Figura 7-9. Captura del metodo de procesamiento de video 1.



```
        total_fps = fps
        fps /= (last_timestamp - first_timestamp)

        delay, fps = math.modf(fps)

        #fps += 1
        #delay = (1.0 / (1.0 - delay)) * fps
        delay = (1.0 / delay) * fps

        out = cv2.VideoWriter('{}/{}.mp4'.format(self.directory, filename), self.fourcc, fps, (width, height))
        print "Writing video..."

        with open('{}/{}.temp'.format(self.directory, filename), 'r') as f:

            i = 1
            repeat_frame = delay
            print delay

            for line in f:

                if i > repeat_frame:
                    #out.write(sample)
                    repeat_frame += delay-1
                    print "cut"
                    continue

                sample = np.fromstring(base64.b64decode(line.split('\n')[0]), dtype=np.uint8).reshape(height,width,3)

                if not bgr:
                    sample = cv2.cvtColor(sample, cv2.COLOR_RGB2BGR)




                out.write(sample)
                print '{}: {} / {}'.format(filename, i, total_fps)
                i += 1

            f.close()

        self.log.info("Video ({}) stream, {} fps, {}, {}.mp4, {} seconds to {} seconds".format(info.name(), (fps+(1/(delay/fps))), info.type().lower(), filename, first_tim

        if self.verbose:
            self.messages.append("Finished video ({}) stream, {} fps, {}, {}.mp4, {} seconds to {} seconds".format(info.name(), (fps+(1/(delay/fps))), info.type().lower(),

        out.release()
        os.remove('{}/{}.temp'.format(self.directory, filename))

        return
```

Figura 7-10. Captura del metodo de procesamiento de video 2.

El método de procesamiento de video obtiene los metadatos que identifican el tamaño y colores de la imagen. Después se inicia un ciclo continuo que verifica si la estampa de tiempo de cada muestra se encuentra dentro del rango de grabación. Debido a que la información de las imágenes es muy pesada, se guardan los datos en un archivo temporal de texto (figura 7-9), para después de terminar la grabación se inicia la conversión a un archivo de video MP4, calculando las imágenes o muestras por segundo y agregando una copia de imagen cada determinado segundo, con el objetivo de agregar la muestra faltante resultado de la eliminación de los dígitos flotantes del redondeo de muestras por segundo (figura 7-10).



```
149     def audio_inlet(self, inlet, filename = "output", timeout = 10):
150
151         first_timestamp = None
152         audio_frames = []
153
154         info = inlet.info()
155
156         ch = info.desc().child("channels").child("channel")
157         sample_size = int(ch.child_value("sample_size"))
158         rate = int(ch.child_value("framerate"))
159         nchannels = int(ch.child_value("number_of_channels"))
160
161         waveFile = wave.open("{}/{}.wav".format(self.directory, filename), 'wb')
162         waveFile.setnchannels(nchannels)
163         waveFile.setsampwidth(sample_size)
164         waveFile.setframerate(rate)
165
166         while True:
167
168             sample, timestamp = inlet.pull_sample(timeout)
169
170             if timestamp >= self.starttime and timestamp <= self.endtime:
171
172                 last_timestamp = timestamp
173
174                 if not first_timestamp:
175
176                     first_timestamp = timestamp
177
178                 else:
179
180                     audio = base64.b64decode(sample[0])
181                     waveFile.writeframes(b''.join(audio))
182
183             elif timestamp > self.endtime or (not sample and not self.recording):
184
185                 waveFile.close()
186                 inlet.close_stream()
187
188                 self.log.info("Audio ({}) stream, {}, {}.wav, {} seconds to {} seconds".format(info.name(), info.type().lower(), filename, first_time
189
190                 if self.verbose:
191                     self.messages.append("Finished audio ({}) stream, {}, {}.wav, {} seconds to {} seconds".format(info.name(), info.type().lower(),
192
193                 return
```

Figura 7-11. Captura del metodo de procesamiento de audio.

Dentro del método de procesamiento de audio se realiza el mismo procedimiento, se crea un ciclo continuo que va validando si las muestras pertenecen al rango de tiempo de grabación, pasando las muestras del búfer a través de un codificador de audio, el cual genera archivos WAV (figura 7-11).

```
def text_inlet(self, inlet, filename = "output", timeout = 10):

    first_timestamp = None

    info = inlet.info()

    text_file = open("{}/{}.csv".format(self.directory, filename), 'w')
    text_file.truncate()

    if info.type().lower() == "eeg" or info.type().lower() == "gaze":
        ch = info.desc().child_value("csv_headers")
        text_file.write("{}\n".format(headers))

    while True:

        sample, timestamp = inlet.pull_sample(timeout)

        if timestamp >= self.starttime:

            last_timestamp = timestamp

            if timestamp <= self.endtime:

                if not first_timestamp:

                    first_timestamp = timestamp

                text_file.write("{}".format(timestamp-self.starttime))

                for item in sample:
                    text_file.write(",{}".format(item))

                text_file.write("\n")

            else:
                break

        elif not sample and not self.recording:
            break

    text_file.close()
    inlet.close_stream()

    self.log.info("Text ({}) stream, {}, {}.csv, {} seconds to {} seconds".format(info.name(), info.type().lower(), filename, first_timestamp, last_timestamp))

    if self.verbose:
        self.messages.append("Finished text ({}) stream, {}, {}.csv, {} seconds to {} seconds".format(info.name(), info.type().lower(), filename, first_timestamp, last

    return
```

Figura 7-12. Captura del metodo de procesamiento de texto.

El método de procesamiento de texto abarca cada dispositivo que regresa información que no requiere de procesamiento multimedia, o que necesitan regresar valores brutos para un futuro postprocesamiento o uso diferente al previsto en este sistema (figura 7-12). Al igual que en funciones similares, se inicia un ciclo continuo que verifica que la estampa de tiempo de las muestras se encuentre dentro del rango de grabación, y se guarda cada registro en un archivo de texto separado por comas (CSV).

```
295         for info in self.streams:
296
297             inlet = StreamInlet(info)
298
299             if info.type().lower() == "video" or info.type().lower() == "display":
300
301                 filename = "{}_{}".format(info.type(), i)
302                 t = threading.Thread(name='video_inlet{}'.format(i), target=self.video_inlet, args=(inlet, filename, self.timeout))
303                 t.start()
304                 i += 1
305
306                 self.threads.append(t)
307
308                 if info.type().lower() == "display":
309                     self.desk_file = filename
310                 else:
311                     self.video_file = filename
312
313             elif info.type().lower() == "eeg":
321
322             elif info.type().lower() == "audio":
332
333             elif info.type().lower() == "gaze":
344
345             elif info.type().lower() == "serial":
```

Figura 7-13. Fragmento de código de detección de tipo de procesamiento de información a partir de los metadatos del búfer.

Dentro del ultimo método, ‘start_inlets’, se obtiene el tipo de dispositivo que representa el outlet que se detecta y se le asigna un tipo de procesamiento. Dependiendo del dispositivo, es probable que sea necesario realizar un postprocesamiento para generar información más representativa o combinarla con otro tipo de información, por lo que es necesario asignarle un nombre único que lo diferencie de archivos similares (figura 7-13).

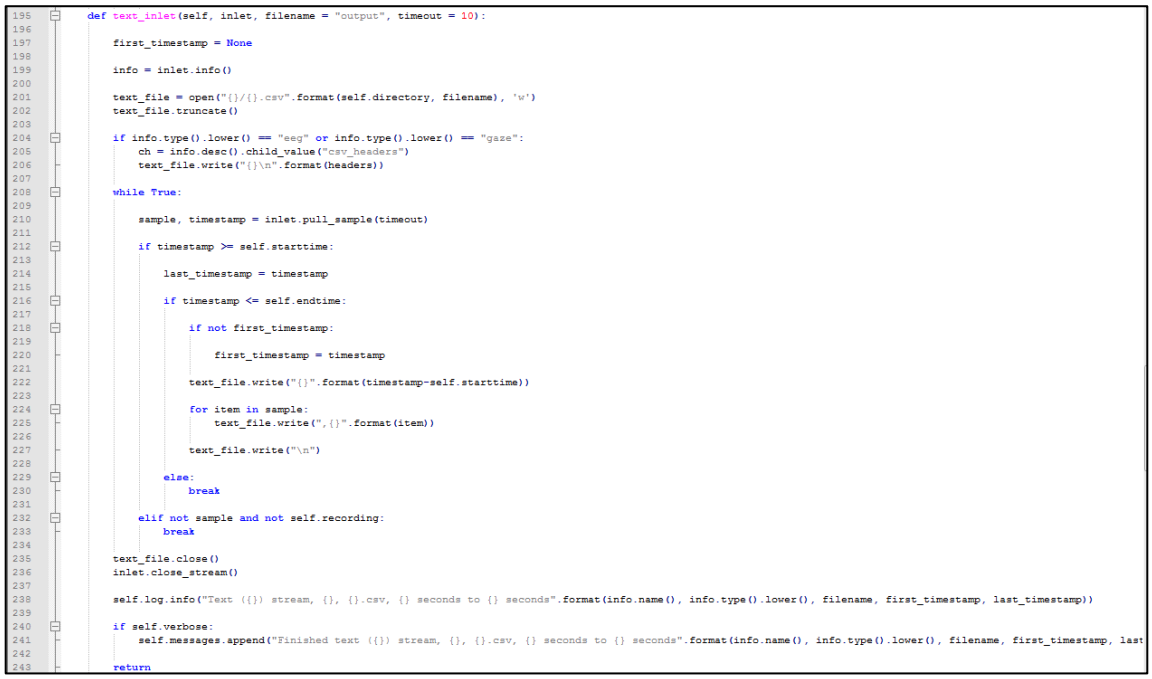
El usuario decide el rango de tiempo que desea grabar, utilizando botones de inicio y fin de la grabación. Esto aplica una sincronización “inter-objetos”, donde cada muestra se localiza en una línea de tiempo basado en el contador de alta precisión localizado en procesadores de computadoras modernas (Zhou, Zheng, H., & Yang, 2013).

7.1.3 Módulo de postprocesamiento de información

Finalmente, en el último módulo se implementó creación de representaciones visuales de la información de movimiento ocular, además de realizar el procesamiento y combinación de todos los archivos multimedia.

Dentro del método de postprocesamiento de información de movimiento ocular se encuentran dos categorías: la creación de animación de datos crudos, la cual genera un

video de fondo negro que dibuja el camino que recorre el movimiento ocular con una cola de muestras anteriores de un tamaño por defecto de 30 datos (figura 7-14), y la creación de animación de mapas de calor, el cual utiliza la librería PyGaze para generar cada imagen con la cola de muestras hasta un momento especifico, generando la secuencia y recorriendo la cola con cada dato (figura 7-15).



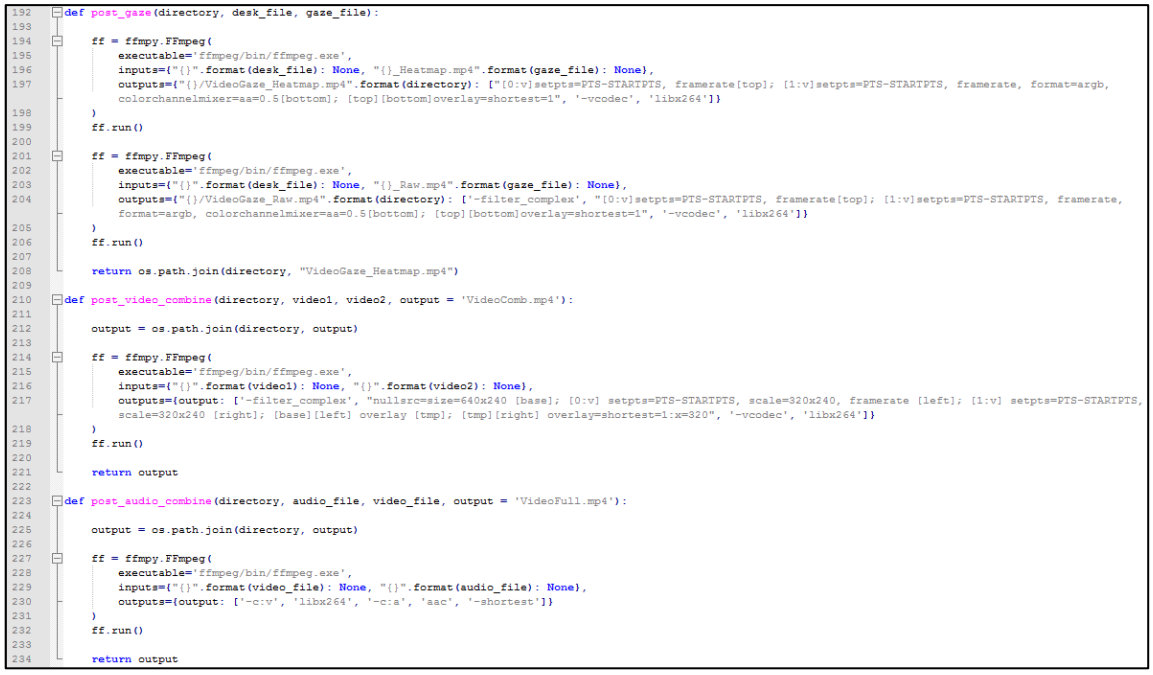
```
    def text_inlet(self, inlet, filename = "output", timeout = 10):

        first_timestamp = None

        info = inlet.info()

        text_file = open("{}/{}.csv".format(self.directory, filename), 'w')
        text_file.truncate()

        if info.type().lower() == "eeg" or info.type().lower() == "gaze":
            ch = info.desc().child_value("csv_headers")
            text_file.write("{}\n".format(headers))

        while True:

            sample, timestamp = inlet.pull_sample(timeout)

            if timestamp >= self.starttime:

                last_timestamp = timestamp

                if timestamp <= self.endtime:

                    if not first_timestamp:

                        first_timestamp = timestamp

                    text_file.write("{}".format(timestamp-self.starttime))

                    for item in sample:
                        text_file.write(",{}".format(item))

                    text_file.write("\n")

                else:
                    break

            elif not sample and not self.recording:
                break

        text_file.close()
        inlet.close_stream()

        self.log.info("Text ({}) stream, {}, {}.csv, {} seconds to {} seconds".format(info.name(), info.type().lower(), filename, first_timestamp, last_timestamp))

        if self.verbose:
            self.messages.append("Finished text ({}) stream, {}, {}.csv, {} seconds to {} seconds".format(info.name(), info.type().lower(), filename, first_timestamp, last

        return
```

Figura 7-14. Bloque de código de la generación de animación de informacion cruda del movimiento ocular.

Mientras tanto, se desarrollaron tres funciones que llaman la herramienta de software *FFmpeg* para combinar automáticamente los flujos multimedia en uno solo con una lista de argumentos de línea de consola (figura 7-16). Esta herramienta es un framework multimedia que cuenta con distintas librerías para codificar, convertir, manipular y reproducir contenido multimedia (FFmpeg team, s.f.).

```
    else:
        def init_anim():
            pass

        def animate_gaze(frameno):
            print "{} / {}\r".format(frameno, len_frames),
            ax.cla()

            if frameno < length:
                heatmap = gaze_processor.get_heatmap(frameno, frameno+1)
            else:
                heatmap = gaze_processor.get_heatmap(frameno, length)

            ax.set_axis_off()
            ax.axis([0,width,0,height])
            ax.imshow(heatmap, cmap='jet', alpha=0.5)
            ax.invert_yaxis()
            fig.subplots_adjust(left=0, bottom=0, right=1, top=1, wspace=None, hspace=None)
            #mat.set_data(heatmap)

        # construct screen (black background)
        screen = np.zeros((height, width, 3), dtype='uint8')
        # dots per inch
        dpi = 100.0
        # determine the figure size in inches
        figsize = (width/dpi, height/dpi)

        # create a figure
        fig = plt.figure(figsize=figsize, dpi=dpi, frameon=False)
        ax = fig.add_axes([0,0,1,1])
        ax.set_axis_off()
        #ax.invert_yaxis()
        ax.axis([0,width,0,height])
        #fig.subplots_adjust(left=0, bottom=0, right=1, top=1, wspace=None, hspace=None)
        #mat = ax.imshow(screen, cmap='jet', alpha=0.5)

        ani = animation.FuncAnimation(fig, animate_gaze, init_func=init_anim, blit=False, interval=1000/gaze_processor.raw_fps, frames=len(gaze_processor.dataset), repeat=

        ani.save('{}_Heatmap.mp4'.format(filename_csv))

        print "Heatmap finished"
```

Figura 7-15. Bloque de código de la generación de animación de mapas de calor del movimiento ocular.



```
def post_gaze(directory, desk_file, gaze_file):

    ff = ffmpy.FFmpeg(
        executable='ffmpeg/bin/ffmpeg.exe',
        inputs={"{}".format(desk_file): None, "{}_Heatmap.mp4".format(gaze_file): None},
        outputs={"{}/VideoGaze_Heatmap.mp4".format(directory): ["[0:v]setpts=PTS-STARTPTS, framerate[top]; [1:v]setpts=PTS-STARTPTS, framerate, format=argb,
        colorchannelmixer=aa=0.5[bottom]; [top][bottom]overlay=shortest=1", '-vcodec', 'libx264']}
    )
    ff.run()

    ff = ffmpy.FFmpeg(
        executable='ffmpeg/bin/ffmpeg.exe',
        inputs={"{}".format(desk_file): None, "{}_Raw.mp4".format(gaze_file): None},
        outputs={"{}/VideoGaze_Raw.mp4".format(directory): ['-filter_complex', "[0:v]setpts=PTS-STARTPTS, framerate[top]; [1:v]setpts=PTS-STARTPTS, framerate,
        format=argb, colorchannelmixer=aa=0.5[bottom]; [top][bottom]overlay=shortest=1", '-vcodec', 'libx264']}
    )
    ff.run()

    return os.path.join(directory, "VideoGaze_Heatmap.mp4")

def post_video_combine(directory, video1, video2, output = 'VideoComb.mp4'):

    output = os.path.join(directory, output)

    ff = ffmpy.FFmpeg(
        executable='ffmpeg/bin/ffmpeg.exe',
        inputs={"{}".format(video1): None, "{}".format(video2): None},
        outputs={output: ['-filter_complex', "nullsrc=size=640x240 [base]; [0:v] setpts=PTS-STARTPTS, scale=320x240, framerate [left]; [1:v] setpts=PTS-STARTPTS,
        scale=320x240 [right]; [base][left] overlay [tmp]; [tmp][right] overlay=shortest=1:x=320", '-vcodec', 'libx264']}
    )
    ff.run()

    return output

def post_audio_combine(directory, audio_file, video_file, output = 'VideoFull.mp4'):

    output = os.path.join(directory, output)

    ff = ffmpy.FFmpeg(
        executable='ffmpeg/bin/ffmpeg.exe',
        inputs={"{}".format(video_file): None, "{}".format(audio_file): None},
        outputs={output: ['-c:v', 'libx264', '-c:a', 'aac', '-shortest']}
    )
    ff.run()

    return output
```

Figura 7-16. Metodos de postprocesamiento de archivos multimedia con ayuda del framework multimedia FFmpeg.

7.2 Tipos de procesamiento de información

Entre los dispositivos utilizados para realizar la grabación paralela de una evaluación UX se encuentran tres tipos de flujos de información:

7.2.1 Flujo de imágenes

Se distinguen por transportar arreglos de 2 o 3 dimensiones, las cuales almacenan valores numéricos que indican la concentración de colores que lleva una imagen cuadrangular. Cada imagen representa un momento en específico, permitiendo almacenar una secuencia consistente que permite generar la ilusión de movimiento y así observar la manera en la que ocurre un evento.

En la figura 7-17 se observa la manera en la que se implementó el procesamiento de una muestra hasta la creación del archivo multimedia.

Figura 7-17. Diagrama secuencial de procesamiento de datos de video.

7.2.2 Flujo de frecuencias acústicas

Este flujo de información transporta un buffer unidimensional con valores numéricos acerca de la amplitud de las ondas acústicas captadas por un micrófono. Cada muestra almacena el conjunto de sonidos escuchados en determinado momento, tomando en cuenta que un sonido genera vibraciones a una determinada frecuencia.

En la figura 7-18 se observa la manera en la que se implementó el procesamiento de una muestra hasta la creación del archivo multimedia.

Figura 7-18. Diagrama secuencial de procesamiento de datos de audio.

7.2.3 Flujo de mensajes secuenciales

En el último flujo de información se reciben mensajes secuenciales de datos, los cuales pueden estar encriptados y necesitan de una API o librería que permita obtener dicho contenido. El mensaje debe contener al menos una estampa de tiempo y un valor relacionado con uno de los sensores utilizados por el dispositivo.

Generalmente, las muestras relacionadas con este tipo de flujo tienen la característica de almacenar valores "crudos" o sin refinar, mientras que dispositivos propietarios como el Emotiv Epoc y el Eyetribe necesitan de software especializado que ocasionalmente debe de pagarse.

En la figura 7-19 se observa la manera en la que se implementó el procesamiento de una muestra hasta la creación del archivo de texto utilizado para almacenar los valores.

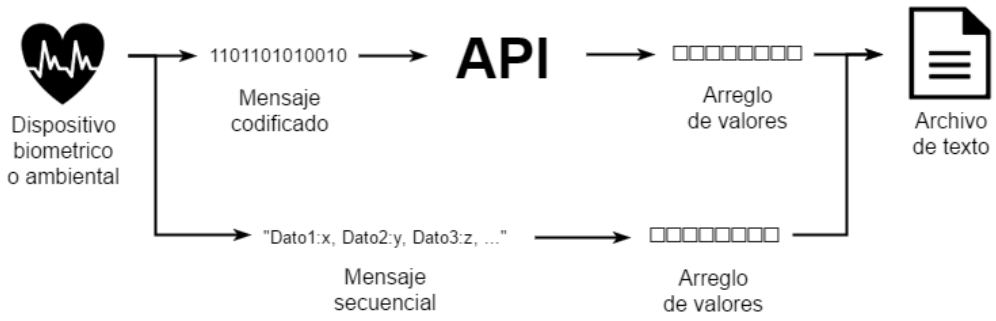


Figura 7-19. Diagrama secuencial de procesamiento de datos ambientales y biométricos.

Para datos no codificados, como los mensajes que genera el Arduino, se definió utilizar como formato el siguiente texto: "NombreDato1: ValorDato1, NombreDato2: ValorDato2, NombreDato3: ValorDato3, …", donde debe de colocarse el nombre del dato seguido por dos puntos y su valor numérico, separando cada dato por una coma.

7.3 Diseño de la interfaz gráfica

Se desarrollaron dos interfaces gráficas principales para el sistema de grabación paralela con la librería *PySide* y *Qt*:

7.3.1 Módulo de dispositivos (outlets), el cual permite generar y activar los flujos de información de cada dispositivo, con la opción de visualizar el estado o la muestra actual (figura 7-20 y 7-21).

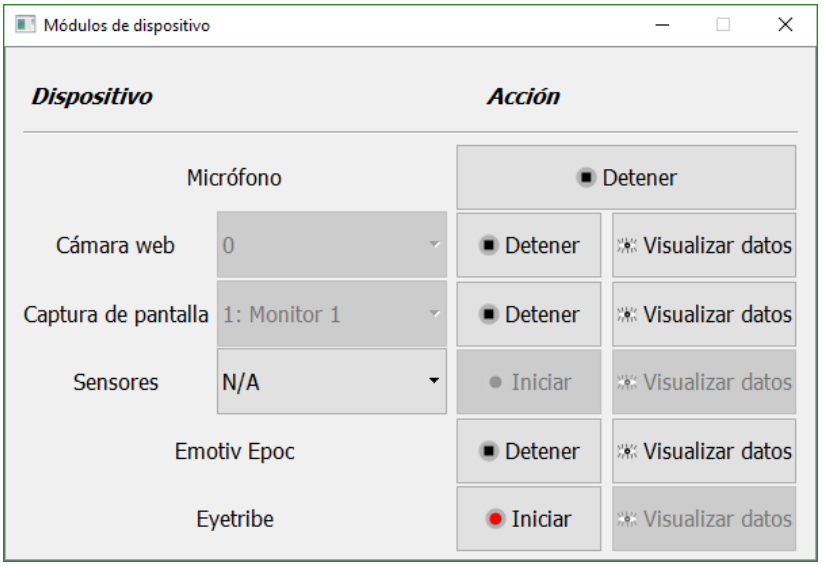


Figura 7-20. Captura de interfaz gráfica de módulo de dispositivos.

Cada botón que inicia o detiene un flujo de información genera un proceso que trabaja en su propio núcleo de procesador, manteniendo un determinado estado de paralelismo.

El botón de visualización de datos llama a la función de visualización de dicho proceso.

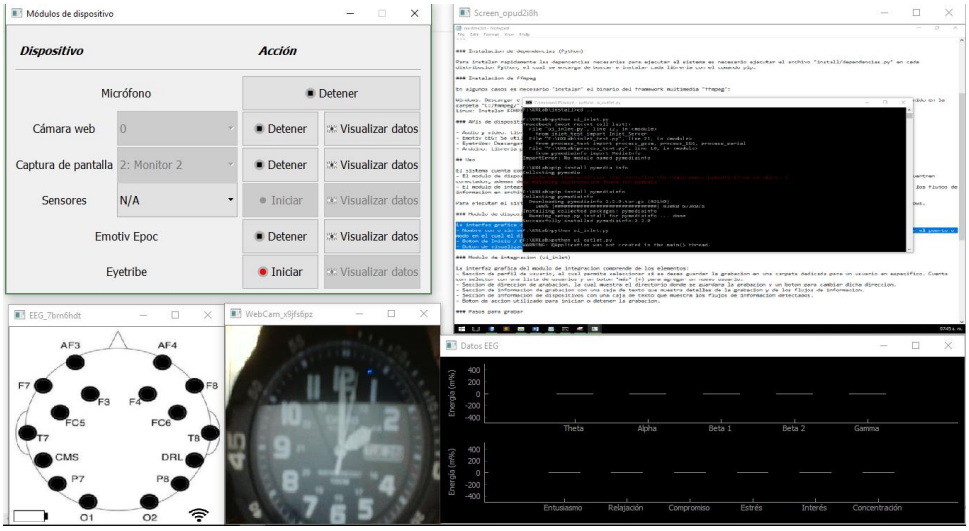


Figura 7-21. Captura de interfaz gráfica de módulo de dispositivos con visualizaciones abiertas.

7.3.2 Módulo de integración (inlet), el cual detecta los flujos de información activos, controla el directorio de almacenamiento de la grabación, y permite definir el rango de tiempo manualmente por medio de un botón de inicio y fin de grabación (figura 7-22).

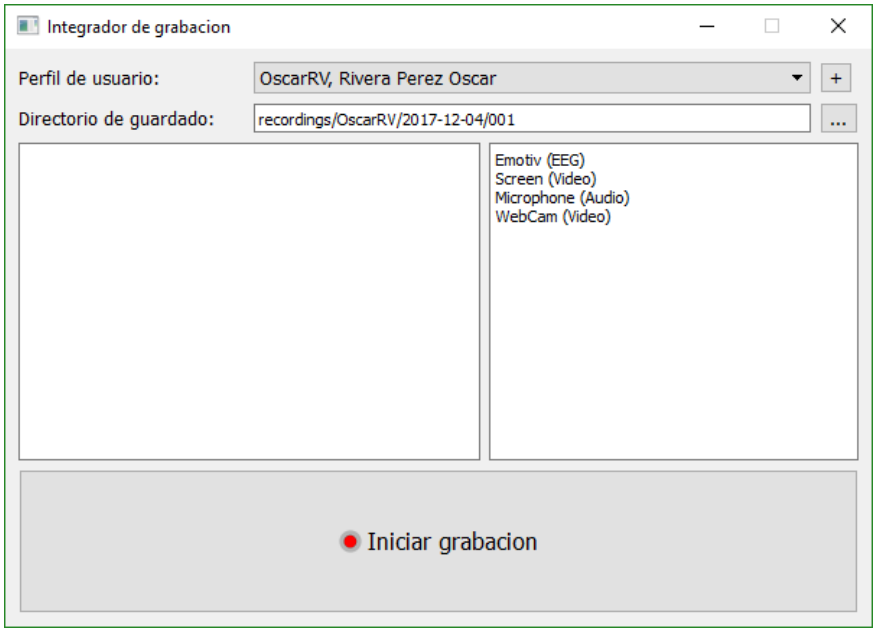


Figura 7-22. Captura de interfaz gráfica de módulo de integración.

La sección de “perfil de usuario” se agregó con la finalidad de identificar el participante del experimento realizado, la fecha y el número de prueba del día.

Los flujos de información se detectan de manera automática, permitiendo identificar que dispositivos generan información correctamente.

7.4 Reproducción y visualización de la información grabada

Para consultar los archivos resultantes de la grabación se implementó un reproductor multimedia en lenguaje C# con visualización de datos en conjunto con una línea de tiempo (figura 7-23).

Se toma en cuenta la manera en la que se grabaron los flujos de información, para así identificar la manera en la que se deben de manipular y visualizar de manera paralela (figura 7-24).

Para visualizar los datos de video y audio se utiliza la API del reproductor *VLC*, el cual permite obtener metadatos de los archivos multimedia y genera una medida de tiempo confiable que se utiliza como base para crear una “línea de tiempo”.

Mientras tanto, se utiliza la librería *OxyPlot* para generar con facilidad gráficas de líneas a partir de los datos provenientes de sensores ambientales y biométricos.

Figura 7-23. Captura de interfaz gráfica del reproductor.

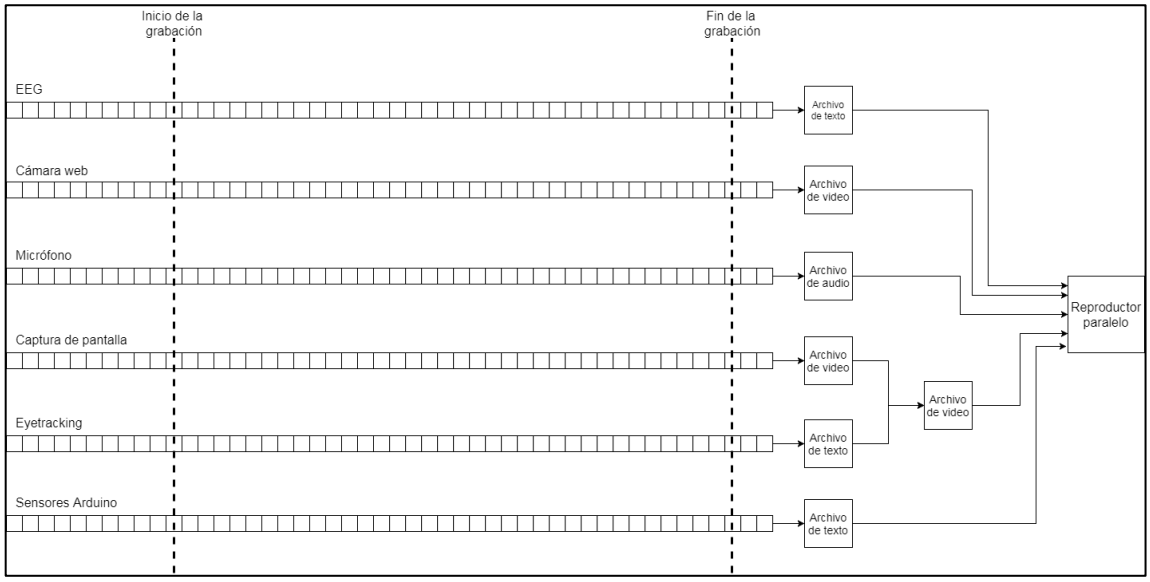


Figura 7-24. Representación gráfica de grabación sincronizada, procesamiento y alimentación de datos al reproductor paralelo.

7.4.1 Generación de mapas de calor

La visualización de datos provenientes del dispositivo de Eyetracking se realiza por medio de la generación de secuencias de imágenes que contienen la trayectoria de la mirada en forma de puntos o la creación de una secuencia de mapas de calor que permiten identificar las áreas visuales más observadas. El primer tipo de secuencia se genera con información cruda, mientras que los mapas de calor necesitan ser generados utilizando una variación de la fórmula de distribución gaussiana multivariable como se especifica en la ecuación 3:

$$f(x,y) = e^{-\left(\frac{(x-x\mu)^2}{2x\sigma^2}+\frac{(y-y\mu)^2}{2y\sigma^2}\right)} \qquad \text{(Ecuación 3)}$$

Donde x es la coordenada sobre el plano horizontal, $x\mu$ es el promedio sobre la coordenada *x*, $x\sigma^2$ es la varianza sobre la coordenada *x*, y es la coordenada sobre el plano

vertical, $y\mu$ es el promedio sobre la coordenada y, y $y\sigma^2$ es la varianza sobre la coordenada y.

Para la implementación de la generación de mapas de calor se realizó la modificación de la librería PyGaze, desarrollada por (Dalmaijer, Mathôt, & Van der Stigchel, 2014), el cual genera el mapa de calor de todo el historial completo en una imagen estática. Dentro del código original se utiliza la librería *MatPlotLib* para graficar una matriz bidimensional, la cual inicialmente cuenta con valor cero y con tamaño similar al de la pantalla de computadora utilizado; sin embargo, por cada coordenada registrada por el movimiento ocular, se realiza la suma de los valores de una pequeña matriz de *n*n* a la cual se le aplicó una función gaussiana de dos dimensiones con un valor grande en el centro de la matriz y se decrementa el valor uniformemente en forma circular hasta llegar a las orillas. En otras palabras, si se viera en una gráfica de tres dimensiones, en cada posición de la mirada dentro de un plano cartesiano se adicionan los valores de una matriz "campana" de poca área (figura 7-25).

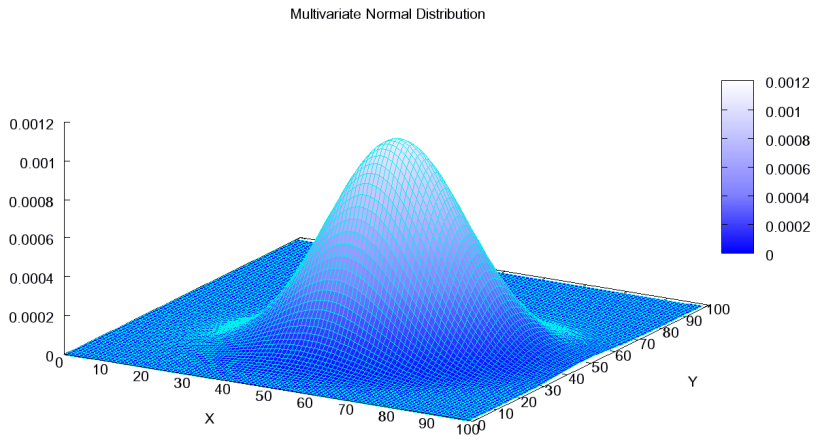


Figura 7-25. Ejemplo visual 3D de matriz con características de mapa de calor (Wikipedia).

La librería *MatPlotLib* se encarga de obtener una media de los valores de la matriz final del movimiento ocular, coloreando en azul las áreas con valores que se encuentren debajo, y rojo las áreas que se encuentran arriban de la media.

En la modificación se aplica de la siguiente manera: por cada cuadro de un video, se debe de tomar una cola de alrededor 5 a 10 segundos de muestras previas de movimiento ocular, con las cuales se les aplica la formula antes mencionada a cada muestra, agregándose a una matriz bidimensional utilizada para dibujar el mapa (figura 7-26).

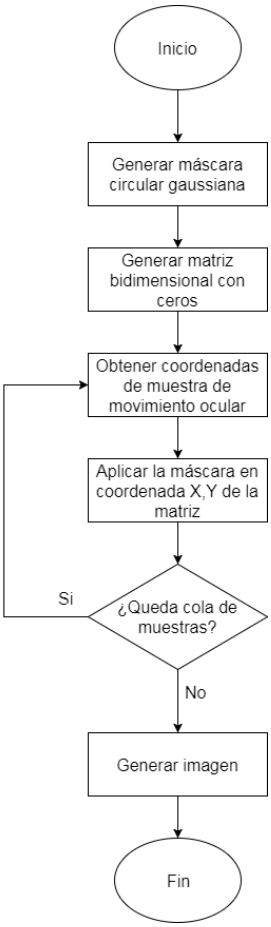


Figura 7-26. Algoritmo de la creacion de un mapa de calor. Se genera una matriz bidimensional vacia y una matriz mascara con valores resultantes de la funcion circular gaussiana, se aplica a cada coordenada X y Y de la informacion de movimiento ocular y se genera una imagen con la librería gráfica de estadisticas.

Para generarla es necesario obtener la matriz bidimensional con valores de la función gaussiana, sacar el valor promedio y el máximo, y dibujar una tonalidad azul-amarilla-roja según un rango de valores conformado por los obtenidos en el paso anterior (figura 7-27).

Figura 7-27. Ejemplo de mapa de calor creado con datos de Eyetracking.

Capítulo 8. Pruebas y resultados

En este capítulo se presentan los experimentos realizados con el sistema de grabación paralela y los resultados de los mismos.

8.1 Descripción de las pruebas

A continuación, se describen las pruebas que se realizaron en esta fase para asegurar el correcto funcionamiento de los módulos de grabación que se desarrollaron en el prototipo.

8.1.1 Pruebas funcionales

En esta sección se presentan las pruebas funcionales realizadas según el estándar 829-1998 (IEEE, 2008) con el sistema de grabación paralela. Se utilizó una cámara web con micrófono para grabar el rostro del usuario y sus opiniones del prototipo de software que evalúa, la captura de la pantalla que utilizo para manipularlo, una cámara EyeTribe para obtener información de zonas de interés visual o secciones de la interfaz con las que interactúa, la diadema Emotiv EPOC para detectar su actividad eléctrica cerebral, y sensores ambientales conectados a un Arduino para encontrar posibles modificadores de conducta a partir de la cual se pueden justificar acciones del usuario, como se muestra en la figura 8-1.

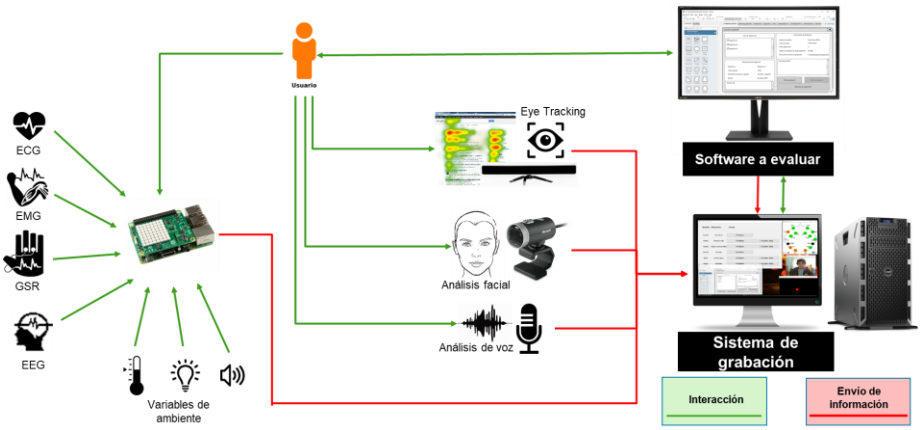


Figura 8-1. Ejemplo de ambiente de pruebas con múltiples dispositivos y sensores.

El proceso de evaluación de la experiencia del usuario se realizó de la siguiente manera:

- El evaluador instala los dispositivos biométricos en el sujeto de prueba al que se le realiza la evaluación de la UX.

- El evaluador inicia los módulos de captura de información, iniciando su visualización y verificando que se encuentren arrojando los datos correctos según el dispositivo (por ejemplo, movimiento ocular con las coordenadas de la mirada, la actividad cerebral con los estados mentales, o la cámara web con imagen del rostro).
- Se ejecuta el prototipo de software a evaluar, colocándose en el monitor que utiliza el usuario de prueba.
- Antes de permitir al usuario de prueba iniciar con la evaluación, es necesario iniciar el proceso de grabación en el módulo de integración.
- El evaluador observa su monitor donde se encuentran las visualizaciones de datos, analizando todas las fuentes de información mientras el sujeto de prueba realiza su evaluación aplicando la "Metodología para Evaluación de SRSC Centrada en el Usuario" de Julia Yazmín Arana Llanes (Arana Llanes, 2014) con las técnicas:
    - Evaluación heurística
    - Think-Aloud
    - Cognitive Walkthrough
- Al finalizar la evaluación, el evaluador debe de detener la grabación, permitiendo al sistema procesar la información obtenida en formatos comunes de archivo.

Los prototipos a evaluar se muestran en la tabla 3:

Tabla 3. Prototipos probados con sistema de grabación paralela para la evaluación de la UX.

Nombre del prototipo	Tipo	Descripción
Buscador ontológico	Mockup	Prototipo en papel de sistema web dedicado a la búsqueda y recuperación de información a partir de ontologías.
Directorio de instituciones	Mockup	Prototipo en papel de aplicación móvil de geolocalización de institutos, con directorio de personal y localización de áreas en tiempo real.
Monitor biométrico	Mockup	Prototipo en papel de aplicación móvil de captura de medidas biométricas relacionadas con la salud del usuario, con bitácora de registros y servicios de emergencia.

Visor de instrucciones AR	Aplicación móvil	Aplicación móvil de realidad aumentada que muestra instrucciones animadas en 3D para manipular herramientas o dispositivos.
Escenario VR	Aplicación de realidad virtual	Aplicación de realidad virtual de escritorio que permite a un usuario interactuar en un ambiente virtual similar a la sala de una casa, objetos interactivos y opciones de personalización de avatares.
Ambiente de inducción de estados emocionales VR	Navegador web con realidad virtual	Sistema web con contenido de realidad virtual diseñado para inducir estados emocionales específicos por medio de la metodología que se desarrolló con ese propósito.

En la tabla 4 se muestran los dispositivos aplicables en cada prueba:

Tabla 4. Flujos de información disponibles en las pruebas.

Nombre del prototipo	Ambiente de prueba	Dispositivos utilizados					
		Cámara web	Micrófono	Captura de pantalla	Eyetracking	EEG	Sensores Arduino
Ontología	Computadora tradicional	X	X	X	X	X	X
Mapa	Computadora tradicional	X	X	X	X	X	X
Monitor biométrico	Computadora tradicional	X	X	X	X	X	X
Visor de instrucciones AR	Celular con cámara trasera	X	X	X		X	X
Escenario VR	Lentes y controles de realidad virtual	X	X	X		X	X
Ambiente de inducción de estados emocionales VR	Lentes y controles de realidad virtual	X	X	X		X	X

8.1.2 Pruebas de sincronización de flujos de información

En esta prueba se analizaron las grabaciones resultantes de las pruebas funcionales y se realizaron pruebas complementarias a dispositivos de video con referencias de medida de tiempo. Estas últimas pruebas utilizan recursos utilizados por empresas de televisión para verificar la correcta transmisión de su señal de video y de audio.

Primero se analiza el nivel de desfase de medios visuales y auditivos por medio de la cantidad positiva o negativa de segundos de diferencia entre los flujos de información. Esto permite elegir uno de los flujos visuales como base a utilizar para verificar el desfase de las otras fuentes de información restantes.

Para ello se utilizó un video prueba utilizado por la cadena BBC para probar la sincronización de sus receptores de señal HD. Como lo indica "Andy Quested" en el blog de internet de la BBC (Andy Quested, 2008), en el video se muestran los siguientes elementos (figura 8-2):

1. "Measurment Bar": Una barra con 24 marcas a los lados, los cuales indican la cantidad de fotogramas ocurridos antes o después de llegar al fotograma clave que reproduce un sonido.
2. "Sync Flash Bars": Tres líneas que aparecen en la parte superior al reproducir el sonido.
3. "Plunger" o "Clapper bar": Una barra martillo que indica que se reproduce el sonido al chocar con la barra base.
4. El sonido de dos bloques de madera al chocar una contra otra, la cual se reproduce en el fotograma clave.

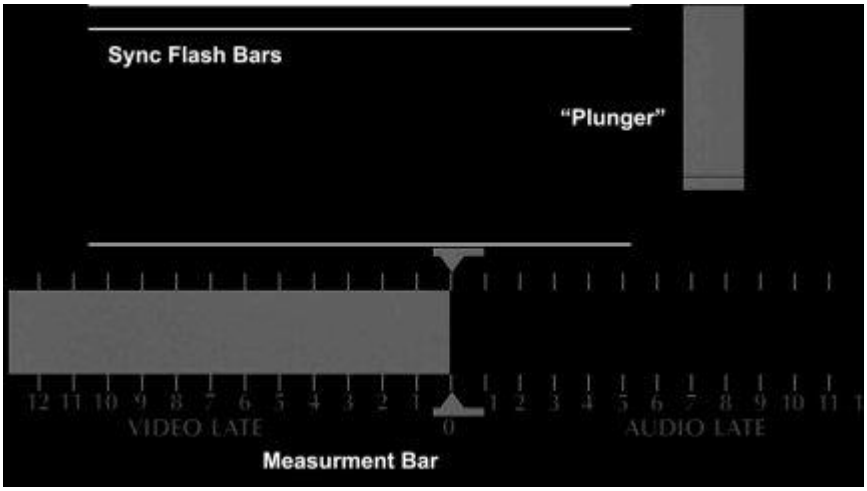


Figura 8-2. Imagen punto de sincronización del video de BBC.

A partir de este video se encuentra el desfase de los flujos visuales y auditivos con sólo ver la posición en la que se encuentra la barra a la hora de activarse el sonido del fotograma clave. Si la barra de medida se encuentra a la izquierda de la marca se dice que la imagen tiene un desfase negativo o retardo, de lo contrario, si la barra se encuentra a la derecha de la marca se dice que tiene un desfase positivo o adelanto (figura 8-3).

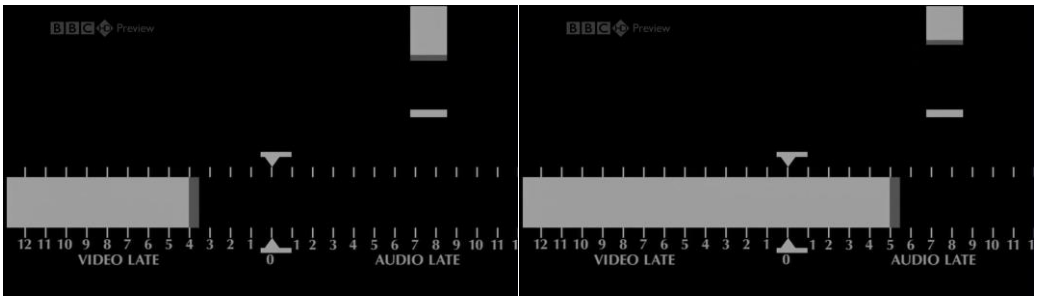


Figura 8-3. Ejemplo de desfase negativo y positivo de imagen.

Mientras tanto, para verificar el desfase de los flujos de información se analizan los eventos más significativos que se detecten en la grabación de la evaluación. El objetivo principal es encontrar coincidencias entre dichos eventos con cambios relacionados en el sensor correspondiente. Por ejemplo, utilizando la información de movimiento ocular en combinación con la captura de pantalla y la cámara web, se encuentra una aproximación

del momento en el que observa una determinada sección de la interfaz de usuario en base a sus acciones en la interfaz y sus expresiones faciales.

8.2 Resultados

Para validar la grabación simultánea de múltiples fuentes de información provenientes de varios dispositivos primero se realizó la ejecución del plan de pruebas funcionales antes especificado. Posteriormente, los evaluadores revisaron que la información obtenida fuera suficiente para realizar la evaluación de la experiencia de los usuarios utilizados como sujetos de prueba.

Finalmente se utilizan las grabaciones para analizar el grado de sincronización basándose en el desfase de eventos importantes para cada flujo resultante. Eventos como acciones poca repetición o referencias de tiempo precisas son las que se eligieron como marcadores para medir el desfase en milisegundos.

En total se realizaron dos rondas de pruebas:

- En la primera ronda, 5 personas realizaron la evaluación UX de sus prototipos de software con 4 sujetos de prueba cada uno. Uno de ellos tiene como prototipo una aplicación móvil de Android, y otro utiliza lentes de realidad virtual, por lo que no es posible capturar el movimiento ocular con el dispositivo relacionado.
- En la segunda ronda, una persona realizó la evaluación de su metodología con la ayuda de un ambiente de realidad virtual. En total realizaron pruebas con 10 diferentes sujetos.

Durante la primera ronda se presentaron algunos casos en los que la prueba se tuvo que suspender u omitir debido a problemas con el sistema de grabación y los prototipos evaluados, mientras que en otros casos el problema no fue crítico y se permitió terminar la prueba:

- El caso más predominante fue un error de desbordamiento de memoria que no se logró solucionar sino hasta el final de la primera etapa de pruebas. Debido a esto se

debieron de limitar las evaluaciones a una duración de 10 minutos con los dos flujos de video activos, o a 30 minutos con sólo uno de ellos.

- El dispositivo de captura de movimiento ocular mostró algunos problemas en su operación en las evaluaciones de prototipos tradicionales, ya que en la mayoría de los casos el flujo de información se congelaba después de varios minutos.
- Otro problema fue la omisión de la activación del dispositivo de audio, el cual evitó incluir la grabación sincronizada de las opiniones de los usuarios. Esto ocurrió por la poca visibilidad del botón de activación y su estado.
- Existieron otros detalles relacionados con el uso de las tecnologías no planeadas para ser usadas en el sistema, como la realidad virtual o dispositivos móviles. En estos casos se debió experimentar con software que permitiera reflejar lo que observaba el usuario (figura 8-4).
- Finalmente se encontró un error en la captura de video de la cámara web, ya que el módulo de grabación recuperaba la última imagen obtenida, siendo diferente a la anterior o no. Esto provocó que a las grabaciones relacionadas se le aplique un efecto de cámara lenta/cámara rápida, la cual se presenta al modificar la densidad de datos por segundo.

  Por ejemplo, del segundo 00:00.00 al 00:01.00 se obtienen alrededor de 50 imágenes, de las cuales 30 son copias, mientras que en el siguiente segundo obtiene 20 imágenes no repetidas. Esto provoca que al sacar el promedio de fotogramas por segundo se cuenten las imágenes repetidas como un momento que dura un rango de milisegundos específico, desfasando las imágenes siguientes y acelerando la aparición de fotogramas no repetidos con un efecto conocido como “slow-down” y “catch-up”.

  Para este caso se utilizó una función que duerme el proceso relacionado por un corto tiempo, permitiendo al dispositivo generar una nueva imagen antes de ser recuperada.

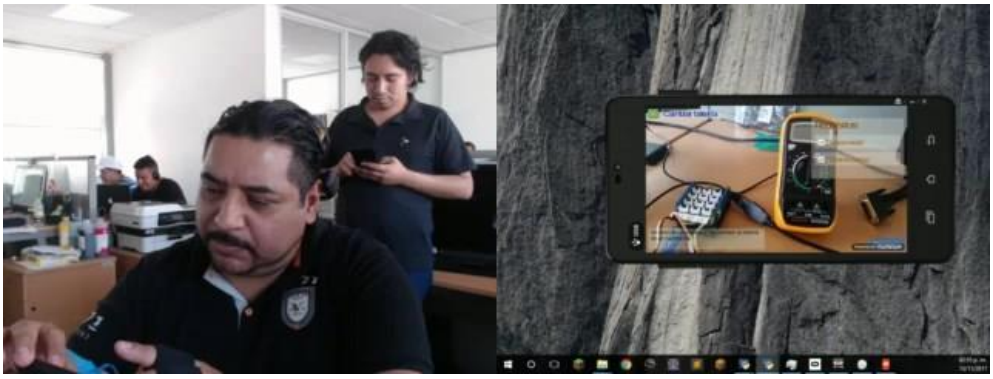


Figura 8-4. Caso de prueba de etapa 1, primera ronda.

En el transcurso de la segunda ronda (figura 8-5) se verificó que se hayan corregido los errores presentados en la ronda anterior.

- El error de desbordamiento de memoria fue provocado por el control equivocado de captura de imágenes del dispositivo de video, el cual generaba un buffer de imágenes más rápido de lo que podía procesarla.
- A los botones para la activación de captura de información se les agrego una imagen de color rojo que indicaba su estado.

Figura 8-5. Caso de prueba de etapa 1, segunda ronda.

En la tabla 5 se engloban los problemas principales de la primera etapa y el número de veces que ocurrieron:

Tabla 5. Registro de ocurrencias.

Problema	Número de ocurrencias
Desbordamiento de memoria / Error fatal de memoria	1
Falla en el dispositivo de captura de movimiento ocular a mitad de la prueba.	18
Error de calibración de dispositivo de captura de movimiento ocular	3
Falta de activación de micrófono	1
Error en la grabación de captura de pantalla	3
Desfase provocado por falta de control en el video de la cámara web	20

Finalmente, en la segunda etapa, se realizó la grabación del video de prueba de la cadena de televisión BBC para verificar el grado de desfase entre los flujos de video y audio, ya que los primeros necesitan de un algoritmo de control de muestras para la generación de archivo multimedia comúnmente utilizado, ya que utilizan codificadores de video que necesitan de un número redondeado de muestras o cuadros por segundo para crear el medio correctamente.

Se utilizan los conceptos de desfase positivo si la información de un flujo se encuentra adelantada a comparación de otra, y desfase negativo para cuando la información aparece retrasada en esta comparación.

Todos los flujos de video utilizan un algoritmo de corrección de la cantidad de muestras por segundo, la cual se describe mediante las ecuaciones 4 y 5:

$$Cuadros\ por\ segundo = \frac{Total\ de\ segundos}{Total\ de\ muestras} \qquad (Ecuación\ 4)$$

$$Correción\ positiva\ de\ secuencia = \frac{1}{1 - Decimales\ de\ cuadros\ por\ segundo} * Redondeo(Cuadros\ por\ segundo + 1) \quad (Ecuación\ 5)$$

Donde:

- Cuadros por segundo: es el promedio de muestras generadas por un dispositivo en una grabación.
- Corrección de secuencia: es el número de muestras en las que se debe agregar una muestra copia para mantener la consistencia en grabaciones de video con cuadros por segundos enteros.

También se experimentó con su versión inversa, la cual indica que se debe de ignorar una muestra después de determinado número de muestras calculado con la ecuación 4 y 6:

$$Correcion\ negativa\ de\ secuencia\ de\ muestras = \frac{1}{Decimales\ de\ cuadros\ por\ segundo} * Redondeo(Cuadros\ por\ segundo) \quad (Ecuación\ 6)$$

Ambos algoritmos son utilizados para permitir generar el video con un número de muestras por segundo constante, ya que, como se mencionó anteriormente, los codificadores de video más utilizados sólo aceptan valores de números enteros para procesar la información (figura 8-6).

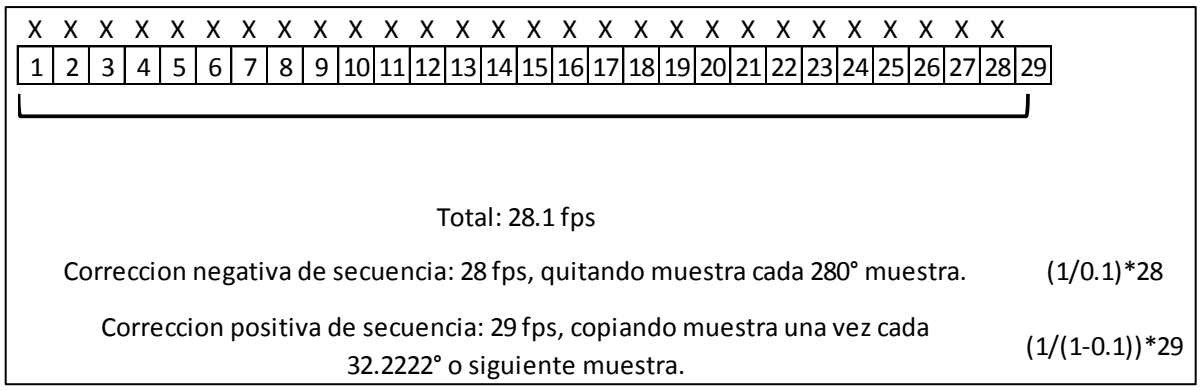


Figura 8-6. Ejemplo de aplicación de fórmula de corrección de secuencia positiva y negativa.

Utilizando la aplicación "Shotcut" y la captura de audio como referencia temporal, se analizan la cantidad de desfase calculado entre las muestras de la captura de pantalla utilizando ambas fórmulas de corrección de secuencia:

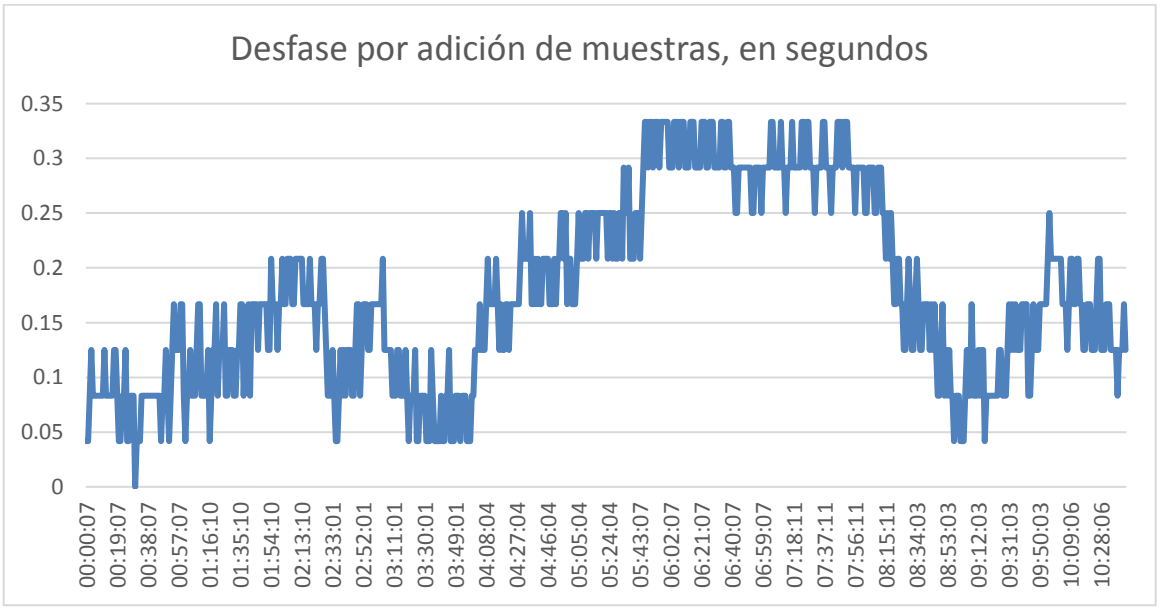


Figura 8-7. Gráfica con datos detallados del desfase resultante provocado por la formula de correccion positiva de muestras de un video de 10 minutos.

- La ecuación de corrección positiva aplicada en un video de 10 minutos (figura 8-7), la cual agrega muestras copiadas cada determinado tiempo, mostrando un rango de desfase de entre 0 y 0.0.3333 segundos. Esto indica que la fórmula de corrección positiva mantiene un bajo nivel de desfase, lo cual permite asegurar que las muestras obtenidas por el sistema se encuentran sincronizadas.

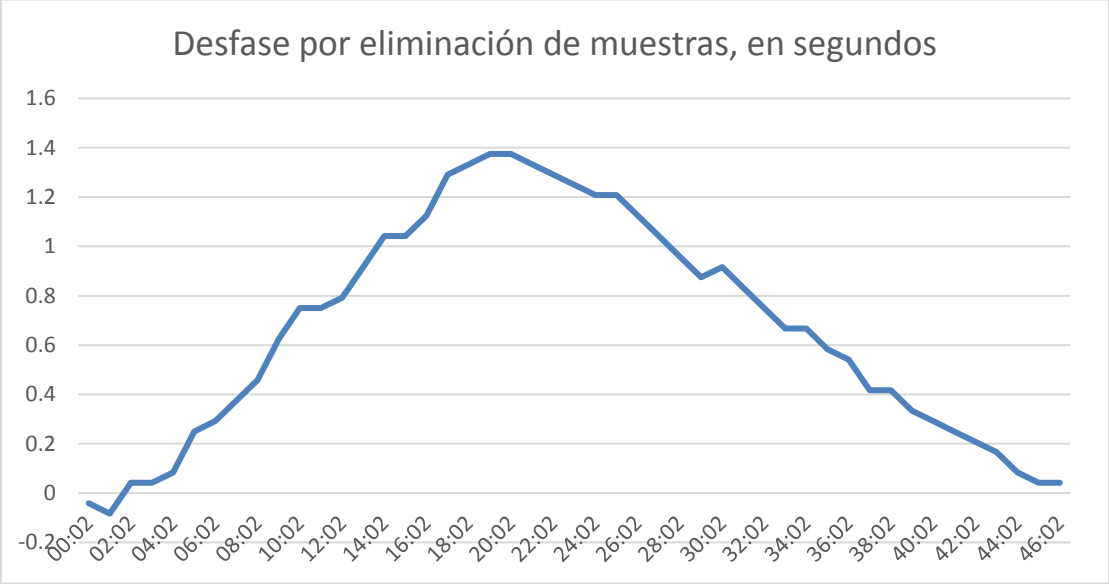


Figura 8-8. Gráfica con datos detallados del desfase resultante provocado por la formula de correccion negativa de muestras de un video de 1 minuto.

- Mientras tanto, el uso de la ecuación de corrección negativa, la cual se utiliza para quitar muestras que afecten al redondeo de muestras por segundo (figura 8-8), muestra un rango de desfase de entre -0.0416 y 1.375 segundos en un video de 1 minuto. Estos resultados indican que en este caso la implementación de esta ecuación genera un mayor desfase en un rango de tiempo más corto, lo cual es problemático al intentar sincronizarlo con eventos de dispositivos biométricos y ambientales.

Tomando en cuenta que el flujo de datos de audio, no cuenta con anomalías temporales al revisarla en la aplicación “Shotcut” (el sonido se ejecuta exactamente cada segundo),

encontrando que su secuencia de muestras no requiere de una corrección, demostrando que cualquier desfase existente debe de ser constante (figura 8-9).

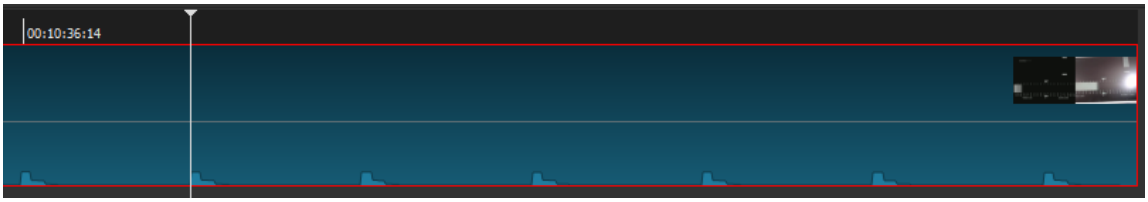


Figura 8-9. Fragmento de muestra de audio de la prueba de correcion donde se muestra el sonido repetido generado por la grabacion de audio. Las barras azules representan el nivel de sonido.

A partir de esta información se verifican las grabaciones de la primera ronda de pruebas funcionales para revisar el desfase de la información de movimiento ocular. Primero se genera la animación de mapas de calor a partir del archivo CSV resultante de la grabación con registros acerca de la posición de la mirada del usuario y se combina con el video de la captura de pantalla usando la aplicación *FFmpeg* (figura 8-10 y 8-11).

```
timestamp,fixation,rawx,rawy,avgx,avgy,psize,Lrawx,Lrawy,Lavgx,Lavgy,Lpsize,Lpupilx,Lpupily,Rrawx,Rrawy,Ravgx,Ravgy,Rpsize,Rpupilx,Rpupily
0.0198129050004,14557.8097136,False,911.1516,271.0933,898.8091,272.8265,11.9618,912.8967,264.2068,875.0155,261.2561,11.9639,0.2473,0.7117,909.4064,277.9799,909.4064,277.979
0.053351120001,14557.8432519,False,869.3158,237.8165,878.3379,269.5518,12.6537,891.283,214.4878,878.396,251.7567,13.0266,0.2475,0.7118,847.3485,261.1453,878.3379,269.5518,1
0.0868739400012,14557.8767747,False,900.459,270.2112,880.9407,266.7774,12.3948,906.2084,264.6999,883.2315,253.8824,12.8708,0.2478,0.7126,894.7095,275.7227,883.7657,271.6032
0.120404608,14557.9103053,False,903.8022,264.7078,883.0056,265.4594,12.542,913.0821,264.4605,887.7661,255.3978,12.8303,0.248,0.7132,894.5223,264.9652,886.4605,269.9236,12.2
0.153937692001,14557.9438384,False,907.3719,281.6309,884.9108,265.1497,12.19165,933.9828,247.9004,893.9703,254.3995,12.4197,0.2482,0.7133,880.7609,315.3615,885.3053,279.138
0.187453266,14557.977354,False,895.7756,260.128,886.1035,265.2689,12.38695,896.4504,255.3934,894.5027,254.4849,12.2721,0.2482,0.7134,895.1008,264.8625,886.9683,276.7704,12.
0.22099118,14558.0108919,True,883.751,243.1898,886.5614,265.4544,12.4233,890.3669,223.195,894.278,251.1023,12.6682,0.2484,0.7132,877.135,263.1847,885.5286,274.7965,12.1784,
0.254559281,14558.04446,True,890.8594,267.4709,887.0676,265.2172,12.80875,893.2397,275.117,894.3613,253.3919,13.1775,0.2485,0.7131,888.4792,259.8249,885.9081,272.8421,12.44
0.288031687001,14558.0779324,True,890.0685,266.6227,887.4839,265.0064,13.02875,893.5767,271.0313,894.4526,254.9927,13.3333,0.2486,0.7126,886.5602,262.2142,885.9806,271.563,
0.321680691,14558.1115814,True,904.7213,281.7913,888.0291,264.9595,12.81305,911.8334,278.3921,896.1052,257.0385,13.0209,0.2487,0.7124,897.6091,285.1904,887.2283,272.9759,12
0.355062837001,14558.1449636,True,903.3728,256.3877,888.4922,265.058,12.6379,908.6066,237.8309,897.3001,255.5293,12.6931,0.2486,0.7121,898.1391,274.9446,888.3341,273.1553,1
0.388587166,14558.1784879,True,885.1985,265.672,889.0102,264.8125,14.32,899.9396,249.2606,897.6866,255.059,14.4588,0.2484,0.7122,870.4575,282.0835,886.7304,273.97,14.1812,0
0.422618644001,14558.2125194,True,894.283,260.7175,889.3949,264.7148,14.68355,898.5108,260.584,897.9167,255.4778,15.0335,0.248,0.7126,890.0551,260.861,887.022,272.8427,14.3
0.456142671001,14558.2460434,True,877.7015,236.0811,889.6317,264.4505,14.11045,883.9691,230.8665,897.0641,253.7204,14.3987,0.2477,0.7126,871.4341,241.2957,885.7513,270.2317,
0.489954988001,14558.2798557,True,897.61,249.535,889.9495,264.0189,12.92365,884.8693,250.031,896.3172,253.4201,13.598,0.2473,0.7129,910.3506,249.039,887.6492,268.482,12.249
0.523689723001,14558.3135905,True,900.3984,241.4757,890.2332,263.601,12.87655,891.3124,235.1408,896.039,252.1334,13.5592,0.2469,0.713,909.4844,247.8107,889.3183,266.7974,12
0.557215260002,14558.347116,True,900.3984,241.4757,890.2332,263.601,12.87655,891.3124,235.1408,896.039,252.1334,13.5592,0.2469,0.713,909.4844,247.8107,889.3183,266.7974,12.
0.590731740002,14558.3806325,True,900.3984,241.4757,890.2332,263.601,12.87655,891.3124,235.1408,896.039,252.1334,13.5592,0.2469,0.713,909.4844,247.8107,889.3183,266.7974,12
0.624268748001,14558.4141695,True,900.3984,241.4757,890.2332,263.601,12.87655,891.3124,235.1408,896.039,252.1334,13.5592,0.2469,0.713,909.4844,247.8107,889.3183,266.7974,12
0.657784323001,14558.4476851,True,900.3984,241.4757,890.2332,263.601,12.87655,891.3124,235.1408,896.039,252.1334,13.5592,0.2469,0.713,909.4844,247.8107,889.3183,266.7974,12
0.691314689,14558.4812154,True,792.7358,392.4612,890.0814,262.5101,12.4296,792.7358,392.4612,792.7358,392.4612,13.0511,0.2446,0.7127,0.0,0.0,0.0,0.0,0.0,0.0,11.8081,0.369,0.72
0.72486498,14558.5147657,True,899.8849,271.7142,891.2089,262.5186,12.39565,915.7789,260.7768,915.7789,260.7768,12.9844,0.2441,0.7131,883.991,282.6516,889.0171,267.7651,11.8
0.758365461001,14558.5482662,True,899.6854,283.6279,891.5941,263.1828,12.7159,885.0629,294.8607,900.4012,277.8405,13.3822,0.2434,0.7136,914.308,272.3951,890.8463,267.9626,12
0.791881941001,14558.5817827,True,896.1313,279.4798,892.0145,263.8263,12.57745,898.4242,270.2768,899.7134,275.3375,13.0173,0.243,0.7136,893.8385,288.6828,891.1711,269.2603,
0.825902551001,14558.6158033,True,892.2058,257.4215,892.4106,264.3171,12.6782,903.6153,268.795,900.6757,273.6991,13.354,0.2425,0.7136,880.7963,246.048,890.6102,267.6873,12.
0.859429597002,14558.6493303,True,882.6483,273.9257,892.6617,264.6543,12.6032,889.8226,251.299,898.4606,269.1498,13.122,0.2419,0.7137,875.4739,296.5524,889.7244,269.4485,12
0.893092486001,14558.6829932,True,892.2393,252.3195,892.8476,264.858,12.6223,899.3383,254.4234,898.5826,266.5991,13.2384,0.2414,0.7141,885.1403,250.2156,889.4777,268.2222,1
```

Figura 8-10. Archivo CSV de grabación de información de movimiento ocular. Los datos guardados son: estampa de tiempo, fijación, las coordenadas X y Y detectadas de la mirada en la pantalla por cada ojo y una aproximación con ambos, el ángulo de la mirada ocular y el tamaño de la pupila.

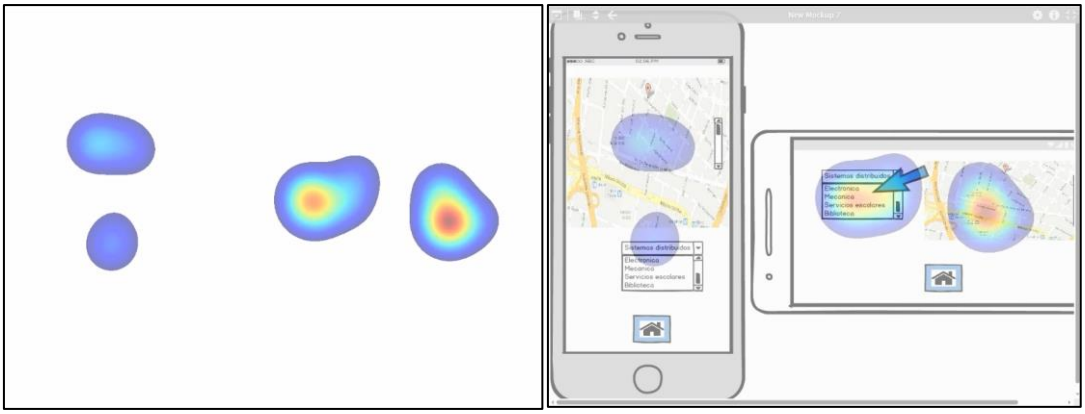
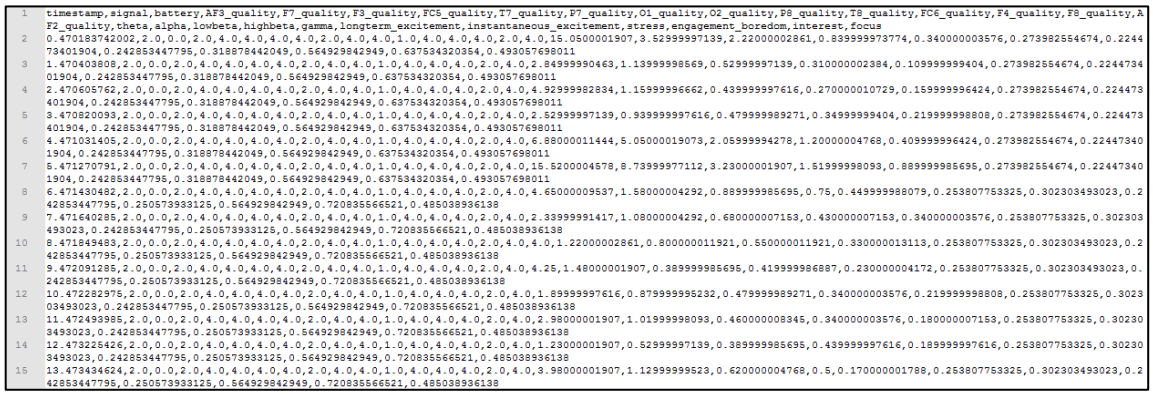


Figura 8-11. Generación de la animación de mapa de calor, y su posterior combinación con la captura de pantalla.

Utilizando el grado de desfase obtenido por la corrección positiva utilizada en esta prueba, se observa que la información del movimiento ocular se acopla a acciones donde los usuarios dan clic para interactuar con un objeto y ver el evento resultante, comentando la acción que realizaban por el dispositivo de audio (figura 8-12). Sin embargo, los usuarios no contaban con un nivel de atención ideal, provocando que generalmente el movimiento ocular revisará toda la pantalla mientras interactuaba con objetos de la interfaz gráfica, haciendo difícil vincular la posición de la mirada a un evento.

Figura 8-12. Puntos detectados de sincronización de movimiento ocular relacionados con eventos visuales y auditivos.

Finalmente, se utilizan las grabaciones de la segunda ronda de pruebas funcionales para revisar el grado de desfase aproximado de la diadema EEG. Sin embargo, debido a que la información que se obtiene del dispositivo es el promedio de las ondas de frecuencia de cada 5 segundos, se concluyó que no es posible verificar la precisión de la sincronización ya que es necesario obtener los datos crudos e instantáneos para vincularlos a un evento en específico (figura 8-13).



```
timestamp,signal,battery,AF3_quality,F7_quality,F3_quality,FC5_quality,T7_quality,P7_quality,O1_quality,O2_quality,P8_quality,T8_quality,FC6_quality,F4_quality,F8_quality,AF2_quality,theta,alpha,lowbeta,highbeta,gamma,longterm_excitement,instantaneous_excitement,stress,engagement_boredom,interest,focus
0.470183742002,2.0,0.0,2.0,4.0,4.0,4.0,4.0,2.0,4.0,4.0,1.0,4.0,4.0,4.0,2.0,4.0,15.0500001907,3.52999997139,2.22000002861,0.839999973774,0.340000003576,0.273982554674,0.224473401904,0.242853447795,0.318878442049,0.564929842949,0.637534320354,0.493057698011
1.470403808,2.0,0.0,2.0,4.0,4.0,4.0,4.0,2.0,4.0,4.0,1.0,4.0,4.0,4.0,2.0,4.0,2.84999990463,1.13999998569,0.52999997139,0.310000002384,0.109999999404,0.273982554674,0.224473401904,0.242853447795,0.318878442049,0.564929842949,0.637534320354,0.493057698011
2.470605762,2.0,0.0,2.0,4.0,4.0,4.0,4.0,2.0,4.0,4.0,1.0,4.0,4.0,4.0,2.0,4.0,4.92999982834,1.15999996662,0.439999997616,0.270000010729,0.159999996424,0.273982554674,0.224473401904,0.242853447795,0.318878442049,0.564929842949,0.637534320354,0.493057698011
3.470820093,2.0,0.0,2.0,4.0,4.0,4.0,4.0,2.0,4.0,4.0,1.0,4.0,4.0,4.0,2.0,4.0,2.52999997139,0.939999997616,0.479999989271,0.34999999404,0.219999998808,0.273982554674,0.224473401904,0.242853447795,0.318878442049,0.564929842949,0.637534320354,0.493057698011
4.471031405,2.0,0.0,2.0,4.0,4.0,4.0,4.0,2.0,4.0,4.0,1.0,4.0,4.0,4.0,2.0,4.0,6.88000011444,5.05000019073,2.05999994278,1.20000004768,0.409999996424,0.273982554674,0.224473401904,0.242853447795,0.318878442049,0.564929842949,0.637534320354,0.493057698011
5.471270791,2.0,0.0,2.0,4.0,4.0,4.0,4.0,2.0,4.0,4.0,1.0,4.0,4.0,4.0,2.0,4.0,15.5200004578,8.73999977112,3.23000001907,1.51999998093,0.889999985695,0.273982554674,0.224473401904,0.242853447795,0.318878442049,0.564929842949,0.637534320354,0.493057698011
6.471430482,2.0,0.0,2.0,4.0,4.0,4.0,4.0,2.0,4.0,4.0,1.0,4.0,4.0,4.0,2.0,4.0,4.65000009537,1.58000004292,0.889999985695,0.75,0.449999988079,0.253807753325,0.302303493023,0.242853447795,0.250573933125,0.564929842949,0.720835566521,0.485038936138
7.471640285,2.0,0.0,2.0,4.0,4.0,4.0,4.0,2.0,4.0,4.0,1.0,4.0,4.0,4.0,2.0,4.0,2.33999991417,1.08000004292,0.680000007153,0.430000007153,0.340000003576,0.253807753325,0.302303493023,0.242853447795,0.250573933125,0.564929842949,0.720835566521,0.485038936138
8.471849483,2.0,0.0,2.0,4.0,4.0,4.0,4.0,2.0,4.0,4.0,1.0,4.0,4.0,4.0,2.0,4.0,4.0,1.22000002861,0.800000011921,0.550000011921,0.330000013113,0.253807753325,0.302303493023,0.242853447795,0.250573933125,0.564929842949,0.720835566521,0.485038936138
9.472091285,2.0,0.0,2.0,4.0,4.0,4.0,4.0,2.0,4.0,4.0,1.0,4.0,4.0,4.0,2.0,4.0,4.25,1.48000001907,0.389999985695,0.419999986887,0.230000004172,0.253807753325,0.302303493023,0.242853447795,0.250573933125,0.564929842949,0.720835566521,0.485038936138
10.472282975,2.0,0.0,2.0,4.0,4.0,4.0,4.0,2.0,4.0,4.0,1.0,4.0,4.0,4.0,2.0,4.0,1.89999997616,0.879999995232,0.479999989271,0.340000003576,0.219999998808,0.253807753325,0.302303493023,0.242853447795,0.250573933125,0.564929842949,0.720835566521,0.485038936138
11.472493985,2.0,0.0,2.0,4.0,4.0,4.0,4.0,2.0,4.0,4.0,1.0,4.0,4.0,4.0,2.0,4.0,2.98000001907,1.01999998093,0.460000008345,0.340000003576,0.180000007153,0.253807753325,0.302303493023,0.242853447795,0.250573933125,0.564929842949,0.720835566521,0.485038936138
12.473225426,2.0,0.0,2.0,4.0,4.0,4.0,4.0,2.0,4.0,4.0,1.0,4.0,4.0,4.0,2.0,4.0,1.23000001907,0.52999997139,0.389999985695,0.439999997616,0.189999997616,0.253807753325,0.302303493023,0.242853447795,0.250573933125,0.564929842949,0.720835566521,0.485038936138
13.473434624,2.0,0.0,2.0,4.0,4.0,4.0,4.0,2.0,4.0,4.0,1.0,4.0,4.0,4.0,2.0,4.0,3.98000001907,1.12999999523,0.620000004768,0.5,0.170000001788,0.253807753325,0.302303493023,0.242853447795,0.250573933125,0.564929842949,0.720835566521,0.485038936138
```

Figura 8-13. Archivo CSV de grabación de información EEG. Los datos guardados son: estampa de tiempo, estado de la señal, batería, estado de los electrodos, valores promedio de cinco segundos Alpha, Beta baja, Beta alta, Gamma y Theta, y porcentajes de "métricas de rendimiento" excitación, aburrimiento, estrés y atención.

La información de los sensores del Arduino no requiere de un análisis muy detallado ya que reciben información del ambiente, cuyo cambio no es lo suficientemente drástico como para vincularlo con un evento externo. Existieron casos de prueba en los que se agregaron sensores biométricos como EMG, ECG y GRS para demostrar que es posible agregar más tipos de sensores al sistema (figura 8-14).

```
0.231786824,EMG:334,GRS:410,ECG:353,Temp:27
1.231936251,EMG:343,GRS:409,ECG:340,Temp:27
2.232212767,EMG:364,GRS:408,ECG:354,Temp:27
3.232373062,EMG:351,GRS:409,ECG:333,Temp:27
4.23257562,EMG:338,GRS:408,ECG:354,Temp:27
5.232791158,EMG:348,GRS:407,ECG:341,Temp:27
6.23300096,EMG:348,GRS:406,ECG:338,Temp:28
7.233229478,EMG:336,GRS:408,ECG:346,Temp:28
8.233446827,EMG:347,GRS:409,ECG:349,Temp:28
9.233675044,EMG:347,GRS:407,ECG:339,Temp:28
10.233851942,EMG:345,GRS:407,ECG:348,Temp:28
11.234466557,EMG:343,GRS:407,ECG:345,Temp:28
12.234775978,EMG:338,GRS:406,ECG:335,Temp:28
13.235526436,EMG:339,GRS:406,ECG:363,Temp:28
```

Figura 8-14. Archivo CSV de grabación de información de sensores ambientales y biométricos. Se almacenan datos como la estampa de tiempo y el valor obtenido por cada sensor.

Todos los flujos de información obtenidos después de cada evaluación se consultan simultáneamente a través del reproductor de información paralela (figura 8-15).

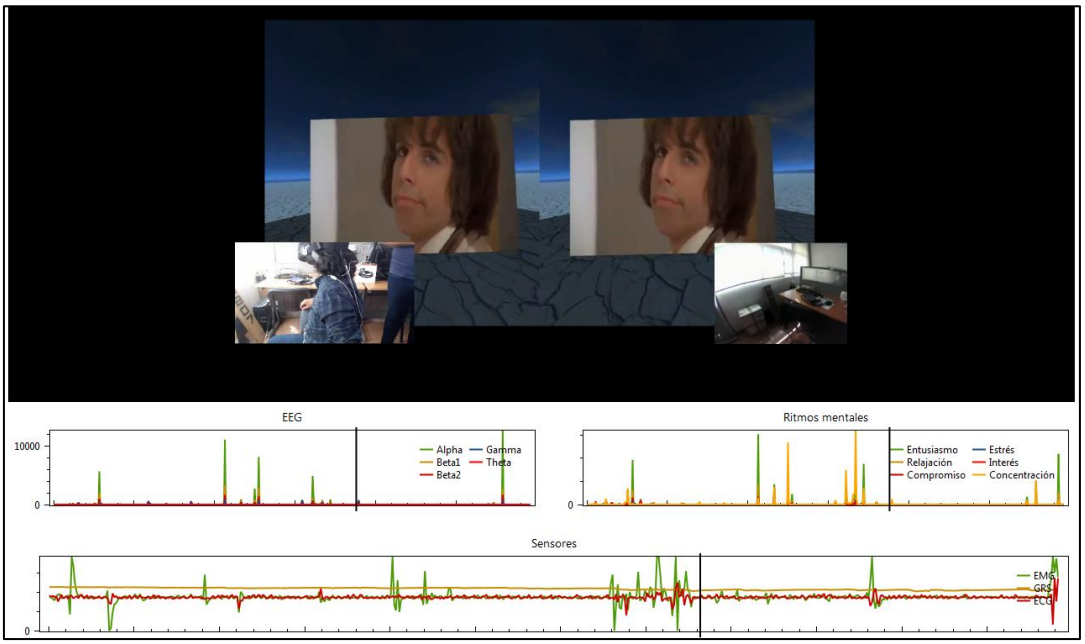


Figura 8-15. Visualización de información de una evaluación de la segunda etapa.

Capítulo 9. Conclusiones

En este capítulo se dan a conocer las conclusiones obtenidas a partir de los resultados de las pruebas del sistema de grabación paralela desarrollado para realizar una captura simultánea y una grabación sincronizada de flujos de información de varios dispositivos conectados en manera paralela, con el objetivo de crear un sistema multimedia y obtener pruebas que permiten asociar información biométrica con eventos captados por dispositivos de audio y video de una evaluación de la experiencia del usuario de una herramienta de software, así como guardar las opiniones y las reacciones de un sujeto en el transcurso de dicha prueba.

9.1 Conclusiones

El objetivo general establecido al inicio de esta tesis, el cual indica diseñar un método para procesar, integrar y sincronizar la información generada a partir de distintos dispositivos multimedia en una evaluación de la experiencia del usuario, se cumplió al crear e implementar el modelo de un sistema modular de grabación paralela, el cual genera un contenedor virtual con información vinculada por estampas de una misma medida de tiempo y una consistencia estable de los flujos de información, sin pérdida de sincronización en estas referencias temporales.

De igual manera, los objetivos específicos se cumplen al implementar y utilizar el sistema de grabación con la técnica de sincronización por etiquetado de muestras con estampas de tiempo de alta precisión, evaluando las grabaciones resultantes de la etapa de pruebas. La evaluación de esta técnica de sincronización se realizó por medio de la consistencia de las muestras obtenidas a partir de cada flujo de información y su visualización en un ambiente gráfico. Por ejemplo, el flujo de audio requiere de una conversión a datos que se transmiten a dispositivos de audio, los cuales se generan sin mostrar desfase. Mientras tanto, los flujos de video presentaron problemas al convertirse a su forma de secuencia de imágenes por el redondeo forzado de cálculo de muestras por segundo, necesitando de un algoritmo de corrección de desfase para mantener una consistencia en la localización temporal de cada imagen, agregando la copia de una muestra cada determinado tiempo.

Como se aprecia en el capítulo 4, se encontraron algunos métodos con los que se realiza la sincronización de grabaciones multimedia, de los cuales se buscó una que no necesitará de dispositivos externos costosos y cuyo uso sea menos restrictivo. Se eligió el uso de estampas de tiempo para realizar la captura sincronizada de muestras gracias a los esfuerzos del Centro Swartz de Neurociencias Computacionales de la Universidad de San Diego, California, el cual puso a disposición una librería de código abierto, multiplataforma y no dependiente a un lenguaje de programación en específico (LabStreamingLayer) con la finalidad de facilitar la construcción de ambientes con múltiples flujos paralelos de información. Esta librería permite generar estampas de

tiempo de alta precisión con el contador de los procesadores de computadoras modernas, etiquetar muestras con ellas, y crear buffers que se pueden compartir con otros programas.

Esto permitió desarrollar un sistema con una arquitectura basada en módulos, el cual permite integrar fácilmente nueva funcionalidad con soporte a futuras tecnologías y dispositivos. Su ejecución fue satisfactoria en una computadora tipo servidor, proporcionando una opción más accesible al instituto para montar un ambiente de evaluación de la experiencia del usuario con la grabación simultanea de múltiples flujos de información adicional con dispositivos multimedia, biométricos y ambientales.

Dentro de los flujos de video fue necesario aplicar un método de corrección de secuencia para normalizar el número de muestras por segundo que aceptan los codificadores de video, utilizados para generar los archivos multimedia relacionados con su visualización estandarizada en una computadora, de los cuales se encontró que la implementación de la corrección positiva de muestras de video fue la que tenía menor margen de desfase.

Finalmente, como actividad complementaria, se adaptaron las funciones de creación de mapas de calor de la librería *Pygaze* para la creación de animaciones sobrepuestas sobre el video de la captura de pantalla de la prueba. En el transcurso de la investigación no se encontraron tecnologías de libre acceso que permitieran generar este tipo de representación visual, el cual es utilizado comúnmente en aplicaciones comerciales.

9.2 Aportaciones

Como aportación principal se encuentra el sistema resultante para realizar grabaciones de evaluaciones de la experiencia del usuario orientados a la prueba de prototipos de software convencionales. Este sistema se diseñó a manera que permita futuras modificaciones para integrar nuevas tecnologías o características en la funcionalidad en el procesamiento de información de los dispositivos de diferentes tipos.

También, como resultado de esta tesis se publicó un artículo titulado “Metodología para la grabación paralela de flujos de datos multimedia y biométricos para la evaluación de la experiencia del usuario”, el cual explica acerca del sistema de grabación paralela y la manera en la que se implementó el método de sincronización de información en cada uno de sus módulos siguiendo la especificación de (Blakowski & Steinmetz, 1996), en el Congreso de Ingeniería en Sistemas Computacionales y sus Aplicaciones (CISCA) 2017.

9.3 Trabajos futuros

9.3.1 Distribución de trabajo a computadoras conectadas en una misma red local

Una computadora moderna cuenta con la capacidad de manejar un número limitado de flujos de información de manera paralela. Cuando se rebasa ese límite es posible empezar a notar un desfase en la recuperación de información al alternar órdenes de procesos que comparten un mismo hilo de computadora.

La librería LabStreamingLayer indica en su repositorio de GitHub que cuenta con la capacidad de enviar y recibir flujos de información de una red local con una referencia temporal global, sin embargo, este proceso no cuenta con la documentación necesaria para identificar la manera en la que se implementa a un sistema.

9.3.2 Integración de otros sensores biométricos

La información del movimiento ocular de un usuario nos permite identificar los elementos que se encuentra mirando al momento de reaccionar u opinar acerca del mismo. Mientras tanto, la diadema EEG permite analizar patrones de la actividad cerebral para obtener una aproximación de lo que va sintiendo. Esta información se complementa con el uso de otros dispositivos biométricos, por ejemplo, el ritmo cardiaco tiene vínculos con la adrenalina y las emociones que la generan, la respiración con el nivel de excitación, entre otros.

Capítulo 10. Referencias

Agarwal, S., Starobinski, D., & Trachtenberg, A. (2002). On the scalability of data synchronization protocols for PDAs and mobile devices. *IEEE Network, 16*(4), 22-28. doi:10.1109/MNET.2002.1020232

Alejandres Sanchez, H. (2016). Evaluación centrada en el usuario de sistemas de recomendación sensibles al contexto: efecto de interfaces multimodales interactivas y esquemas de explicación en la experiencia del usuario. Cuernavaca, Morelos, México: Centro Nacional de Investigación y Desarrollo Tecnológico.

Andy Quested. (17 de Diciembre de 2008). *BBC - BBC Internet Blog: A Christmas Present from the HD Channel!* Obtenido de BBC - BBC Internet Blog: http://www.bbc.co.uk/blogs/bbcinternet/2008/12/a_christmas_present_from_the_h.html

Arana Llanes, J. (Junio de 2014). Metodología para Evaluación de SRSC Centrada en el Usuario, Basada en Características de Efectividad, Confianza y Satisfacción Mediante Interfaces Multimodales sobre Dispositivos Móviles Multisensoriales. Cuernavaca, Morelos, Mexico: Centro Nacional de Investigación y Desarrollo Tecnológico.

Audio-Technica U.S., Inc. (2005). *¿Qué Hace un Micrófono? || Audio-Technica U.S.* Obtenido de Audio-Technica U.S: https://www.audio-technica.com/cms/site/9f3f5c571dcbded8/

Blakowski, G., & Steinmetz, R. (Enero de 1996). A Media Synchronization Survey: Reference Model, Specification, and Case Studies. *IEEE Journal on Selected Areas in Communications, 14*(1), 5-35.

Boronat, F., Lloret, J., & García, M. (Marzo de 2009). Multimedia group and inter-stream synchronization techniques: A comparative study. *Information Systems, 34*(1), 108-131. doi:10.1016/j.is.2008.05.001

Brocious, C., Machulis, K., Lemaignan, S., Olorin, S., & Schumacher, B. (s.f.). *GitHub - openyou/emokit: Open source driver for accessing raw data from the Emotiv EPOC EEG headset.* Recuperado el 2017, de GitHub: https://github.com/openyou/emokit

C. Topper, N., N. Burke, S., & Porter Maurer, A. (30 de Diciembre de 2014). Multiple frequency audio signal communication as a mechanism for neurophysiology and

video data synchronization. *Journal of Neuroscience Methods, 238*, 35-42. doi:10.1016/j.jneumeth.2014.09.018

Çağlayan, O. (s.f.). *GitHub - ozancaglayan/python-emotiv: A Python module to access Emotiv EPOC EEG headset on Linux.* Recuperado el 2017, de GitHub: https://github.com/ozancaglayan/python-emotiv

Chow, E. (13 de Julio de 2012). *Multimedia Synchronization Specification.* Obtenido de University of Colorado Colorado Springs: http://cs.uccs.edu/~cs525/synmm/synmm.htm

Comité Coordinador Permanente de la Infraestructura de Datos Espaciales del Perú. (2017). *¿Qué son los Metadatos?* Obtenido de Portal de la Infraestructura de Datos Espaciales del Perú: http://www.geoidep.gob.pe/conoce-las-ides/metadatos/que-son-los-metadatos

Dalmaijer, E., Mathôt, S., & Van der Stigchel, S. B. (2014). PyGaze: An open-source, cross-platform toolbox for minimal-effort programming of eyetracking experiments. *Behavior research methods, 46*(4), 913-921. doi:10.3758/s13428-013-0422-2

Dawes, B., Abrahams, D., & Rivera, R. (1 de Septiembre de 1999). *Boost C++ Libraries*. Obtenido de Boost C++ Libraries: https://www.boost.org/

Eyetracking, Incorporated. (s.f.). *Quad Server, real-time data integration.* Recuperado el Enero de 2017, de http://www.eyetracking.com/Software/Quad-Server

Eyetracking, Incorporated. (s.f.). Quad Server: Real Time Data Fusion Brochure. San Diego, California: Eyetracking, Incorporated. Recuperado el 2017, de http://www.eyetracking.com/Portals/et/Documents/Quad_Server.pdf

FFmpeg team. (s.f.). *About FFmpeg*. Recuperado el 2017, de FFmpeg: https://www.ffmpeg.org/about.html

Hearn, T. (s.f.). *GitHub - thearn/pyemotiv: A Python library for data acquisition from the Emotiv Epoc EEG headset, using the research SDK.* Recuperado el 2017, de GitHub: https://github.com/thearn/pyemotiv

Howarth, E., & Hoffman, M. S. (1984). A multidimensional approach to the relationship between mood and weather. *British Journal of Psychology*(75), 15-23. doi:10.1111/j.2044-8295.1984.tb02785.x

IEEE. (2008). *IEEE Standard for Software Test*. Obtenido de http://faculty.ksu.edu.sa/mohamedbatouche/SWE%20434/IEEE%20Std%20829%20-%201998.pdf

iMotions A/S. (2017). *iMotions Biometric Research Platform*. (iMotions) Recuperado el 29 de Mayo de 2017, de https://imotions.com/

International Organization for Standardization. (15 de Marzo de 2010). ISO 9241: Ergonomics of human-system interaction -- Part 210: Human-centred design for interactive systems. *1*, 32.

Jacko, J. A. (2012). *Human Computer Interaction Handbook: Fundamentals, Evolving Technologies, and Emerging Applications, Third Edition* (3rd ed.). CRC Press.

Lambert, L., Hachicha, K., Ahmed, S. Z., Pinna, A., & Garda, P. (2015). Synchronizing physiological data and video in a telemedicine application: A multimedia approach. *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)* (págs. 181-185). Milan: IEEE Engineering in Medicine and Biology Society (EMBC). doi:10.1109/EMBC.2015.7318330

Lopez-Calderon, J., & Luck, S. J. (14 de Abril de 2014). ERPLAB: an open-source toolbox for the analysis of event-related potentials. *Frontiers in Human Neuroscience, 8*, 213. doi:10.3389/fnhum.2014.00213

Madore, D. A. (27 de Diciembre de 2010). *The Unix leap second mess.* Recuperado el 12 de Mayo de 2017, de http://www.madore.org/~david/computers/unix-leap-seconds.html

Makeig, S., Delorme, A., & Brunner, C. (7 de Septiembre de 2013). EEGLAB - an Open Source Matlab Toolbox for Electrophysiological Research. *Biomed Tech, 58*. doi:10.1515/bmt-2013-4182

Matroska. (2016). *Matroska Media Container*. (Matroska) Obtenido de https://www.matroska.org/

Microsoft. (2012). *Kinect Sensor*. Obtenido de Windows Dev Center: https://msdn.microsoft.com/en-us/library/hh438998.aspx

Microsoft. (2 de Mayo de 2017). *Acquiring high-resolution time stamps (Windows) - MSDN*. Obtenido de https://msdn.microsoft.com/en-us/library/windows/desktop/dn553408(v=vs.85).aspx

MikeGrusin. (s.f.). *TSL2561 Luminosity Sensor Hookup Guide - learn.sparkfun.com.* Recuperado el 2017, de learn.sparkfun.com: https://learn.sparkfun.com/tutorials/tsl2561-luminosity-sensor-hookup-guide

NeuroSky. (s.f.). *NeuroSky Developer*. Recuperado el 2017, de NeuroSky: developer.neurosky.com/docs/doku.php

Niedermeyer E., d. S. (2004). *Electroencephalography: Basic Principles, Clinical Applications, and Related Fields.* Lippincot Williams & Wilkins.

Ojeda, A., Bigdely-Shamlo, N., & Makeig, S. (5 de Marzo de 2014). MoBILAB: an open source toolbox for analysis and visualization of mobile brain/body imaging data. *Frontiers in Human Neuroscience, 8*, 121. doi:10.3389/fnhum.2014.00121

OpenCV team. (s.f.). *Official OpenCV library site*. Recuperado el 2017, de OpenCV library: https://opencv.org/

Peña, R., Ávila, A., Muñoz, D., & Lavariega, J. (24 de Enero de 2015). A Data Hiding Technique to Synchronously Embed Physiological Signals in H.264/AVC Encoded Video for Medicine Healthcare. *BioMed Research International, 2015*, 10. doi:10.1155/2015/514087

Pham, H. (2006). *PyAudio: PortAudio v19 Python Bindings*. Recuperado el 2017, de PyAudio: https://people.csail.mit.edu/hubert/pyaudio/

Seeedstudio. (2012). *Grove - Digital Light Sensor.* Obtenido de Seeed Product Document: http://wiki.seeedstudio.com/Grove-Digital_Light_Sensor/

Seeedstudio. (2017). *Grove - Sound Sensor.* Recuperado el 2017, de Seeed Product Document: http://wiki.seeedstudio.com/Grove-Sound_Sensor/

Seeedstudio. (s.f.). *Grove - Temperature&Humidity Sensor.* Recuperado el 2017, de Seeed Product Document: http://wiki.seeedstudio.com/Grove-TemperatureAndHumidity_Sensor/

Swartz Center for Computational Neuroscience (SCCN), UCSD. (20 de Julio de 2012). *labstreaminglayer - Multi-modal time-synched data transmission over local network*. Obtenido de GitHub - sccn: https://github.com/sccn/labstreaminglayer

The Emotiv Team. (s.f.). *GitHub - Emotiv/community-sdk: Emotiv SDK Community Edition.* Recuperado el 2017, de GitHub: https://github.com/Emotiv/community-sdk

The EyeTribe. (2016). *The Eye Tribe*. (The Eye Tribe Aps) Recuperado el Octubre de 2016, de https://theeyetribe.com/

The OpenKinect community. (2000). *OpenKinect*. Recuperado el 2017, de OpenKinect: https://openkinect.org/wiki/Main_Page

Tobii Technology AB. (2 de Enero de 2014). *An introduction to Tobii EyeX - Tobii Developer Zone.* Obtenido de Tobii Developer Zone: https://developer.tobii.com/an-introduction-to-the-tobii-eyex-sdk/

Torniainen, J., & Henelius, A. (26 de Septiembre de 2016). A short review and primer on online processing of multiple signal sources in human computer interaction applications. Obtenido de arXiv preprint arXiv:1609.02339

University of California San Diego. (04 de Marzo de 2010). *Swartz Center for Computational Neuroscience*. Obtenido de https://sccn.ucsd.edu/

Venthur, B. a. (Agosto de 2012). Mushu, a free- and open source BCI signal acquisition, written in Python. *Engineering in Medicine and Biology Society (EMBC), 2012 Annual International Conference of the IEEE*, 1786-1788. doi:10.1109/EMBC.2012.6346296

W Homan, R., Herman, J., & Purdy, P. (1987). Cerebral location of international 10–20 system electrode placement. *Electroencephalography and Clinical Neurophysiology*, *66*(4), 376-382. doi:10.1016/0013-4694(87)90206-9

Zhou, Z., Zheng, R., H., J., & Yang, G. (2013). EVS: An EEG-Video Synchronization System Using High Performance Counter. *2013 International Conference on Information Technology and Applications* (págs. 105-108). Chengdu: Information Technology and Applications (ITA). doi:10.1109/ITA.2013.30