eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea
BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

MÁSTER UNIVERSITARIO EN INGENIERÍA DE CONTROL, AUTOMATIZACIÓN Y ROBÓTICA

TRABAJO FIN DE MÁSTER

DESARROLLO DE UNA SOLUCIÓN HARDWARE Y SOFTWARE PARA LA RECOGIDA Y PROCESAMIENTO DE SEÑALES FISIOLÓGICAS

Estudiante *Rey Vicente, Nerea*
Director *Salazar Ramírez, Asier*
Codirectora Martínez Rodríguez, Raquel
Departamento Ingeniería de Sistemas y Automática
Curso académico *2021/2022*

Bilbao, 29 de septiembre de 2022

Resumen

Un sistema de reconocimiento de emociones requiere equipos capaces de reconocer, interpretar y responder apropiadamente a las emociones. Son cada vez más las investigaciones en este ámbito, donde la innovación y desarrollo de dispositivos ha permitido la evolución de la monitorización de las señales fisiológicas de forma no invasiva. Sin embargo, el coste que suponen estos dispositivos es muy elevado. Este proyecto desarrolla una aplicación propia para la adquisición de señales fisiológicas mediante un dispositivo hardware de bajo coste. De este modo, se consigue validar este dispositivo, así como una aplicación propia, además de plantear nuevas ubicaciones para buscar una mejor señal de la manera menos invasiva posible.

Palabras clave:

Señales fisiológicas, BITalino, Actividad Electrodermal (EDA), Electrocardiograma (EMG), estrés, emociones, Sistema Nervioso Autónomo (SNA), variabilidad del ritmo cardíaco (HRV).

Abstract

An emotion recognition system requires equipment capable of recognising, interpreting and responding appropriately to emotions. An increasing amount of research is being conducted in this area, where innovation and development of devices has enabled an evolution in the non-invasive monitoring of physiological signals. However, the cost of these devices is very high. This project develops a self-developed software application for acquiring physiological signals using a low-cost hardware device. In this way, it validates both the device and the proprietary application, as it also proposes new locations to identify a better signal in the least invasive way possible.

Key words:

Physiological signals, BITalino, Electrodermal Activity (EDA), Electrocardiography (ECG), stress, emotions, Autonomous Nervous System (SNA), Heart-Rate Variability (HRV).

Laburpena

Emozioen detekziorako sistema batek, hauek modu egokian errekonozitzeko, interpretatzeko eta erantzuteko gai den tresneria behar du. Geroz eta ikerketa gehiago egin dira eremu honetan, non gailuen berrikuntzak eta garapenak aukera eman duen seinale fisiologikoen monitorizazioan bilakaera bat izateko, modu ez-inbaditzailean. Hala ere, gailu horien kostua oso handia da. Eta horren aurrean, proiektu honek berezko aplikazio bat garatzen du seinaleak kostu txikiko hardware gailu baten bidez eskuratzeko. Horrela, gailua eta aplikazio propioa baliozkotzea ahalbidetzen da, baita kokapen berriak proposatzea ere, seinale hobea ahalik eta modu inbaditzaileenean bilatu ahal izateko.

Hitz gakoak:

Seinale fisiologikoak, BITalino, jarduera elektrodermikoa (EDA), elektrokardiograma (ECG), estresa, emozioak, Nerbio-Sistema Autonomoa (SNA), bihotz taupaden aldakortasuna (HRV)

ÍNDICE

Lista de ilustraciones

Lista de tablas

Acrónimos

EDA	Electrodermal Activity	Actividad Electrodermal
GSR	Galvanic Skin Response	Respuesta Galvánica de la Piel
SNA	Autonomous Nervous System	Sistema Nervioso Autónomo
HR	Heart-Rate	Frecuencia Cardíaca (FC)
HRV	Heart-Rate Variability	Variabilidad del ritmo cardíaco
ECG	Electrocardiography	Electrocardiograma
EMG	Electromyogram	Electromiograma
EEG	Electroencephalogram	Electroencefalograma
HW	Hardware	
SW	Software	
MAC	Media Access Control	
BCI	Brain-Computer Interface	Interfaz cerebro-computadora
ELA	Amyotrophic Lateral Sclerosis	Esclerosis Lateral Amiotrófica

MEMORIA

1. INTRODUCCIÓN

Este documento recoge el Trabajo de Fin de Máster (TFM), consistente en el diseño, desarrollo y validación de una solución hardware y software para la recogida y procesamiento de señales fisiológicas mediante un dispositivo de bajo coste. Este trabajo se ha realizado en el contexto de un proyecto de investigación de mayor envergadura, titulado "Sistemas inteligentes basados en biomarcadores para salud (BISH)" financiado por el Ministerio de Ciencia e Innovación el pasado año 2021.

Tanto el proyecto como el TFM se llevan a cabo dentro del grupo ALDAPA (ALgorithms, DAta mining and PArallelism). Este grupo de investigación centra su investigación en el área de la minería de datos (machine learning, modelos explicables, desequilibrio de clases) y en la computación fisiológica (registro y análisis de señales fisiológicas, desarrollo de sistemas vestibles y no invasivos). Actualmente la actividad central del grupo se enmarca en el contexto de la salud, en concreto en el diagnóstico y prevención de enfermedades del sistema nervioso (párkinson, alzhéimer, epilepsia) y estudio del estrés y las emociones, en colaboración con entidades sanitarias y tecnológicas públicas y privadas.

Además, el equipo inició una colaboración con Uliazpi (entidad pública que trabaja con personas con discapacidad intelectual), estableciendo objetivos comunes en cuanto a la medición y detección de eventos relacionados con el estrés. Estudios anteriores detectaron parámetros estadísticos que son indicadores efectivos del estrés y planean futuras colaboraciones utilizando dispositivos portátiles no invasivos para adquirir señales fisiológicas de personas con discapacidad cognitiva. Uliazpi está interesado en poder tener biomarcadores del sistema nervioso que ayuden a identificar cómo el ejercicio físico en bicicleta estáticas inteligentes, entre otras terapias, ayuda a los usuarios a reducir su nivel de estrés. Dado la sensibilidad del colectivo, hay que buscar un equilibrio entre la no invasividad del dispositivo de adquisición de bioseñales y fiabilidad en la recogida. Por eso, en este TFM se plantea la búsqueda de soluciones hardware que registren señales fisiológicas de la forma menos invasiva posible, y software que permitan visualizar de manera online las bioseñales recogidas.

Para la realización de este TFM ha sido necesaria la previa documentación en el estudio de la fisiología humana, además de la búsqueda de tecnologías existentes para este análisis. Se han estudiado diferentes dispositivos (HW) y plataformas software, además de técnicas de monitorización, para posteriormente definir los objetivos y beneficios del proyecto. Finalizada esta documentación, se han analizado alternativas de hardware necesario y los diferentes softwares a utilizar, decidiendo cuál podría resultar la más adecuada y realizando su implementación.

Para el desarrollo y diseño del proyecto ha sido necesario, por una parte, el hardware BITalino, desarrollando una aplicación mediante el IDE de Python para una correcta adquisición de datos; y por la otra, MATLAB, con el cual se han validado los resultados obtenidos.

Por último, se han realizado diferentes ensayos para la validación del sistema de recogida de señales fisiológicas con personas del grupo de investigación, introduciéndoles una situación de estrés para observar las señales adquiridas. Tras validar tanto el dispositivo como la aplicación creada, dada la dificultad que se ha encontrado en la adquisición de una de las señales, se han realizado más ensayos con la intención de optimizar la ubicación de los electrodos de dicha señal.

Por lo tanto, tal y como se explicará en el capítulo de las conclusiones, queda pendiente para futuras tareas el diseño de un prototipo vestible para la adquisición de datos con el dispositivo, la aplicación validada, y los puntos óptimos para la adquisición de las señales recogidas.

2. CONTEXTO

Si bien la autonomía e independencia son valores propios del ser humano, existen colectivos que no pueden disfrutar plenamente de ellos, como las personas de la tercera edad, enfermos crónicos, personas con discapacidad intelectual, etc. Promover la independencia de estas personas acarrearía dar un salto en la calidad de sus vidas, y en la de los familiares y tutores que les asisten. Por ello, son múltiples las investigaciones que estudian y trabajan con las cuestiones que afectan a la sociedad y el bienestar social, buscando mejorar el envejecimiento de la población.

Es conocido que las emociones están muy relacionadas con la salud de la persona, y que éstas ejercen una increíble y poderosa fuerza en el comportamiento humano, llegando a gobernar su día a día. El ser humano siente miles de emociones y toma decisiones en función de si está contento, triste, aburrido o frustrado. De esta manera, las situaciones estresantes y estados de depresión y/o frustración pueden afectar a dichos estados emocionales. Además, en relación a esto, gran parte de los pensamientos, emociones y aprendizaje se produce a un nivel inconsciente [1]. Por ello, crear un sistema que sea capaz de detectar dichas situaciones podría ayudar a comprender mejor cómo afectan las diferentes situaciones a los estados emocionales y así poder tratar de mejorar el bienestar de las mismas.

En cuanto a las emociones, éstas se reflejan en términos de señales fisiológicas gracias al Sistema Nervioso Autónomo (SNA), por ejemplo, en los parámetros de la frecuencia cardíaca, la respiración o la temperatura corporal. Es por eso que las diferentes respuestas fisiológicas del SNA permiten determinar el estado psicoemocional de la persona en ese momento.

Actualmente, existen múltiples grupos de investigación que desean aplicar conocimientos técnicos a sistemas que mejoren la calidad de vida de las personas, donde se requiere la integración de diferentes áreas como la inteligencia artificial, ingeniería biomédica, neurociencia, psicología y medicina. La unión de la ingeniería de sistemas a las tecnologías de la información del ámbito médico-asistencial, dan como resultado la ingeniería biomédica. Y dentro de este ámbito se encuentra la computación afectiva. Esta es una rama del estudio que hace referencia al diseño de sistemas y dispositivos que pueden reconocer, interpretar y procesar emociones humanas.

Es por eso que la innovación y el desarrollo de los dispositivos biomédicos, junto con la aplicación de diferentes principios de ingeniería, ha permitido una evolución paulatina de la monitorización de las señales fisiológicas y de la medición de los parámetros y variables más relevantes del paciente. Aun así, el control permanente del estado afectivo requiere de una monitorización constante, lo que llevaría a tener a la persona sensorizada en todo momento. Poco a poco se observan cada vez más pulseras para llevar a cabo esta tarea, que ofrecen señales para determinar el estado afectivo, y por ello, muchas investigaciones se basan en estos dispositivos que se usan a diario.

En la Figura 1 se muestra un ejemplo para la monitorización de pacientes mediante una plataforma de sensores portátiles. En este caso, la empresa Verisense ha creado una pulsera para medir la frecuencia cardíaca [2], la saturación de oxígeno y las respuestas emocionales de los participantes en ensayos clínicos.

Figura 1. *Plataforma de telemonitorización del estado de salud Verisense Pulse+* [2].

Cabe destacar que muchos de estos dispositivos son de uso clínico y por lo tanto son relativamente caros. Esto resulta un inconveniente, ya que su elevado coste económico actúa como barrera para el uso de estos dispositivos en el ámbito de la investigación. Por ello, en este proyecto se trabaja con un kit de herramienta accesible, compuesto por hardware y software de bajo coste que fue creado para reforzar el compromiso de diferentes personas en el campo de las bioseñales.

En este sentido, este TFM se va a centrar en el diseño y desarrollo de un dispositivo para la recogida de señales fisiológicas, de manera que, en un futuro, el grupo de investigación pueda diseñar un dispositivo vestible y cómodo que ayude a mejorar la calidad de vida de las personas mediante la detección de situaciones de estrés.

3. OBJETIVO Y ALCANCE DEL TRABAJO

El objetivo principal de este Trabajo de Fin de Máster es el desarrollo de un dispositivo de recogida de señales fisiológicas de fácil uso y bajo coste, que sea fácilmente adecuable a diferentes entornos experimentales, como pueden ser los experimentos de búsqueda de biomarcadores representativos de estrés en personas con discapacidad cognitiva.

Los objetivos específicos que permiten la consecución del objetivo principal son los siguientes:

1. **Revisión del estado de la tecnología existente** para la adquisición de señales fisiológicas y estudio de diferentes sensores más utilizados. Para ello, se hará una revisión bibliográfica incluyendo hojas de fabricante y documentación relacionada.

2. **Selección y validación del dispositivo hardware.** Es importante validar que el dispositivo de bajo coste elegido sea capaz de registrar correctamente las señales fisiológicas. Por ello, uno de los principales objetivos será validar en el entorno de Matlab las señales adquiridas con los dispositivos escogidos en el apartado anterior.

3. **Desarrollo y validación** en entorno de Python una **interfaz de usuario** para monitorizar las lecturas realizadas por el dispositivo. Modificar una API ya diseñada por el grupo de investigación haciéndola más intuitiva y de fácil uso.

4. **Selección** del conjunto de señales a recoger mediante los diferentes sensores. En concreto, el trabajo se centra en las señales ECG y EDA, dado que son base para poder detectar la activación y desactivación del sistema nervioso autónomo, es decir, para detectar situaciones de estrés en el cuerpo humano.

5. **La realización de diferentes pruebas** donde se analice la ubicación óptima para la recogida de las señales ECG y EDA, de forma no invasiva y de fácil accesibilidad. Dada la dificultad en el caso de la EDA, estos ensayos se centrarán más en esta señal.

6. **La validación** mediante diferentes ensayos el funcionamiento del prototipo. Analizar los datos obtenidos, y obtener conclusiones.

7. **Adecuar el diseño para un futuro prototipo vestible.** El grupo de investigación tiene como objetivo final diseñar un prototipo vestible para personas con discapacidad cognitiva. Por ello, tras evaluar las ventajas e inconvenientes de la recogida de señales en diferentes partes del cuerpo, habrá que escoger la mejor opción para ese prototipo final.

Una vez cumplidos los objetivos, el Grupo de Investigación ALDAPA contará con un diseño software y hardware que cumpla con las necesidades para una correcta recogida de señales fisiológicas mediante un dispositivo de bajo coste. Siendo el alcance de dicho trabajo, la implementación de dicha solución en contextos como la monitorización de personas con discapacidad cognitiva o ensayos donde se quiera estudiar relación fisiológica en diferentes escenarios médicos.

4. BENEFICIOS QUE APORTA EL TRABAJO

En este apartado, se procede a describir los diferentes beneficios técnicos, económicos y sociales que ofrece este proyecto.

4.1. Beneficios científicos-técnicos

Tal y como se ha definido en la introducción de esta memoria, este TFM es parte de un proyecto de investigación del Grupo ALDAPA. El objetivo principal del proyecto es crear un prototipo vestible para la monitorización de señales fisiológicas que dé soporte a detectar situaciones de estrés. Por lo que el beneficio técnico principal del TFM es conseguir un dispositivo de bajo coste que adquiera señales fisiológicas mediante una propia API creada en Python, para poder analizar y segmentar las señales de manera libre y sencilla.

Además, se va a trabajar con diferentes dispositivos para validar que el sistema de recogida de señales diseñado funciona correctamente. Para este fin, habrá que analizar y correlacionar las señales recogidas con los diferentes dispositivos y por tanto se irán adquiriendo conocimientos y técnicas para el análisis de datos.

4.2. Beneficios económicos

Actualmente, en el mercado existen diferentes dispositivos de adquisición de señales fisiológicas, pero como se ha comentado anteriormente, su coste puede ser relativamente alto, no son portables o no recogen las señales con la calidad suficiente. Todos estos inconvenientes se convierten en un obstáculo para investigar en el ámbito de este proyecto. Por ello, la demostración de que es posible recoger las señales fisiológicas de manera fiable y a bajo coste daría acceso a que cada vez más grupos de investigación tomaran la iniciativa de crear sus propias bases de datos. Así pues, los beneficios de la realización de este proyecto trascienden a los aspectos más puramente académicos y tienen también impacto económico.

4.3. Beneficios sociales

Aun así, el principal beneficio de este trabajo es social, ya que el objetivo principal reside en mejorar la calidad de vida de las personas. Como se ha comentado anteriormente, unos de los contextos donde se ha planteado la necesidad de un sistema de adquisición de bioseñales de forma no invasiva es en entornos donde viven personas con discapacidad intelectual. En este sentido, la asociación Uliazpi pretende valorar la mejora emocional de este tipo de personas tras la realización de ejercicio físico personalizado. Siguiendo en la línea de la mejora del estado emocional, el Instituto Burmuin también está interesado en la utilización de este desarrollo para poder identificar cual es la terapia o la herramienta que ayuda mejor a gestionar el estrés a cada paciente que asiste a su gabinete de psicología. Los anteriores dos ejemplos son sólo una muestra, otras aplicaciones, de casos en los que la solución propuesta en este proyecto podría utilizarse de cara a mejorar el bienestar social de la población en general.

5. ESTADO DEL ARTE

El estudio de las bioseñales ha tenido un papel transformador en múltiples aspectos de la sociedad, yendo cada vez más allá de las ciencias de la salud a los que tradicionalmente se asocian. La ingeniería biomédica es una disciplina clásica ampliamente cubierta, y hoy en día, las bioseñales son objeto de interés para estudiantes, investigadores y aficionados en áreas incluyendo la informática, la computación y la ingeniería eléctrica, entre otras muchas.

Por eso mismo, en este apartado, se estudiarán los antecedentes bibliográficos, empezando por introducir las respuestas fisiológicas más comunes. También se hará una búsqueda acerca de la recogida de estas señales fisiológicas, empezando desde proyectos donde se usen diferentes plataformas software, pasando por los dispositivos hardware, hasta la metodología usada en cada investigación.

5.1. Introducción al estudio de las respuestas fisiológicas

Muchos definen las emociones como reacciones conductuales, fisiológicas y subjetivas, activadas por una información proveniente del mundo externo y/o interno del individuo. Es decir, entienden por emoción una experiencia multidimensional con, al menos, tres sistemas de respuesta: cognitivo/subjetivo, conductual/expresivo y fisiológico/adaptativo [3]. Este planteamiento está basado en el modelo tridimensional de Lang. Estas reacciones pueden ser de rechazo, si se viven esos estímulos como peligrosos, o bien de atracción, si se sienten como atractivos o placenteros.

Es sabido que las emociones provocan fuertes reacciones en el cuerpo, como, por ejemplo, cuando el estómago se revuelve por la ansiedad o que el corazón palpita por el miedo. El sistema nervioso simpático se encarga de controlar las reacciones de lucha o huida del cuerpo. Ante una amenaza, estas respuestas preparan automáticamente al cuerpo para huir del peligro o enfrentarse a la amenaza de frente. Por el contrario, el sistema nervioso parasimpático es el que entra en acción restableciendo las condiciones normales del organismo, la cual tiende a tener un efecto relajante en el cuerpo. Es conocida como “descanso y digestión”, ya que dedica los recursos a funciones básicas de “limpieza”. De una manera gráfica, el diagrama de la Figura 2 resume cómo se desarrollarán la mayoría de las respuestas fisiológicas en el ser humano.

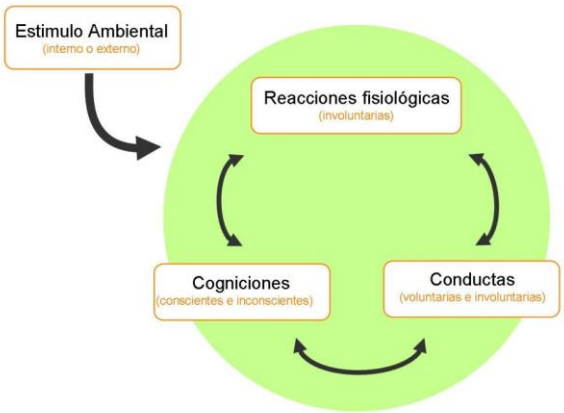


Figura 2. *Planteamiento del modelo tridimensional de Lang.*

Mientras que los primeros estudios sobre la fisiología de las emociones solían centrarse en estas respuestas autonómicas, las investigaciones más recientes se han centrado en el papel del cerebro de las emociones. Los escáneres cerebrales han demostrado que la amígdala, la cual forma parte del sistema límbico, desempeña también un amplio papel en la regulación de la respuesta del miedo tanto en humanos como en animales [4] [5]. Por ejemplo, los investigadores han utilizado imágenes cerebrales para demostrar que cuando a las personas se les muestran imágenes amenazantes, la amígdala se activa [6].

El sistema nervioso es uno de los sistemas más importantes y complejos del cuerpo humano. Aunque mucha gente lo relacione tan solo con los nervios, el sistema nervioso, junto con el sistema endocrino, tienen funciones como recibir y procesar toda la información que proviene tanto del interior del cuerpo como del entorno, con la finalidad de regular el funcionamiento de los demás órganos y sistemas. En este caso, el sistema nervioso se encarga a través de las neuronas de procesar y transmitir información, mientras que el sistema endocrino segrega hormonas a través de la sangre.

Aun siendo el sistema nervioso central un sistema complejo del cuerpo humano, la ciencia lleva siglos investigando acerca de cómo medir su actividad. La activación o inhibición del SNA durante procesos emocionales se estudia a través de diferentes señales fisiológicas, ya que la respuesta fisiológica es un mecanismo homeostático ante cualquier evento interno o externo que esté regulado por el sistema nervioso autónomo. Otras formas de medir el sistema nervioso son la monitorización de la actividad eléctrica del cerebro, los cambios en el flujo sanguíneo que ocurren con la actividad del cerebro o campos magnéticos también producidas por las corrientes eléctricas del cerebro.

Para finalizar con este apartado, destacar que el alcance de este proyecto, las señales fisiológicas que se van a recoger son la EDA y ECG. Aunque todas las señales tienen una estrecha relación con las emociones y el estrés como se está analizando en el estado del arte, estas dos señales, además de ser las menos invasivas, son fáciles de adquirir y deja abierto para el futuro el camino a diseñar un dispositivo portable con estas señales.

5.1.1. Señales fisiológicas

El estudio de las señales fisiológicas puede ayudar a tener una medida del grado de activación o inhibición del SNA durante los procesos emocionales, analizando los límites de variación y los mecanismos de regulación. Estas señales o reacciones corporales suelen ser respuestas involuntarias como, por ejemplo, cambios en el ritmo cardíaco o respiratorio, aumento de sudoración, cambios en la tensión muscular, sequedad en la boca, presión sanguínea, etc.

Con el fin de explicar de manera más detallada lo más relevante sobre las señales fisiológicas, a continuación, se presentan aquellas que se consideran más significativas, aunque no todas serán desarrolladas a lo largo de este trabajo.

- *Actividad Electrodérmica (EDA) o Resistencia Galvánica de la piel (GSR)*

La respuesta galvánica de la piel (GSR), también denominada como actividad electrodérmica (EDA), es la medida de las continuas variaciones en la conductancia de la piel causada por la variación de la sudoración del cuerpo humano [7]. Esta señal fisiológica varía conforme a la sudoración de la piel de los sujetos: las glándulas sudoríparas actúan como si se trataran de conductores eléctricos, que al llenarse de sudor aumenta su conductividad al disminuir su resistencia. Como se ha mencionado

anteriormente, el SNA controla la actividad de las glándulas, lo que permite vincular los cambios producidos en esta señal fisiológica con los estímulos que alteran el estado emocional de las personas.

La EDA mide la actividad de las glándulas sudoríparas que se encuentran en todo nuestro cuerpo en diferentes proporciones. Es importante que los electrodos EDA estén situados en una zona de la piel que contenga suficientes glándulas sudoríparas para obtener una buena calidad de la señal. Éstas pueden ser de dos tipos, apocrinas o ecrinas. Son estas últimas las que reflejan la actividad emocional, y pueden encontrarse por todo el cuerpo, aunque la densidad de estas glándulas es mayor en la cara interna de las manos (palmas), los dedos y los pies, aunque se distribuyen también con densidades variables por toda la superficie corporal, como en la frente, los hombros, el cuello y muñecas [8].

Es por eso que la EDA es una de las medidas más utilizadas para los estudios psicofisiológicos que implican excitación emocional. Aunque tradicionalmente se mide en los dedos o en las palmas de las manos, estas posiciones de registro no siempre son las preferidas para los registros ambulatorios en situaciones de la vida real. Es por eso que cada vez son más los estudios que se realizan en busca de estos sitios para leer la señal EDA, donde también depende de la intrusión que se le quiera someter al participante.

La EDA ha sido ampliamente estudiada en la historia de la psicofisiología en una gran variedad de contextos, incluyendo la atención, el procesamiento de la información y la emoción, entre otros [9]. Las aplicaciones en las que se utiliza la EDA para medir la actividad del SNA, en los artículos [10] y [11], se utiliza para examinar los estados cognitivos y emocionales y para indicar el nivel de estrés y ansiedad, en el primer caso realizando dicha investigación a personas con Trastorno Límite de la Personalidad (TLP) y en la segundo, analizando el comportamiento de las personas con Trastorno de Espectro Autista (TEA). Las señales EDA también se estudian en la robótica en la que se examinan las interacciones humano-robóticas (HRI) en relación con los estados afectivos de los usuarios [12].

Como se ha mencionado en apartados anteriores, la señal EDA va a tener un importante peso durante el trabajo en cuanto a la recogida de señales fisiológica que se refiere, dado el interés para encontrar un punto óptimo para recogerla y que sea adecuado para el diseño vestible que se desea hacer en un futuro. Por eso, en el siguiente apartado 5.1.2 se hará una revisión del estado del arte sobre la recogida de esta señal en diferentes partes del cuerpo humano.

- *Electrocardiograma (ECG)*

El electrocardiograma (ECG) es la señal que representa a la actividad eléctrica del corazón. Más concretamente representa las diferentes polarizaciones y despolarizaciones de las aurículas y ventrículos del corazón. La señal del electrocardiograma detecta información relevante sobre el nivel general de actividad de un individuo. Se puede tomar como ejemplo, las aceleraciones que sufre la frecuencia cardíaca como respuesta al ejercicio, estado emocional, sonido fuerte, excitación sexual y tensión mental [13]. Una frecuencia cardíaca más baja se asocia generalmente con un estado relajado o con la exposición a estímulos agradables [14].

El corazón está compuesto por varios tejidos, entre ellos el músculo cardíaco, que puede producir señales eléctricas detectables en la superficie de la piel mediante un sensor de ECG, utilizando al menos tres electrodos colocados en el pecho. Los electrodos recogen los datos necesarios respecto a las ondas eléctricas que describen el ciclo cardíaco, en base a los cuales se obtiene la frecuencia cardíaca (Heart-

Rate en inglés, HR) o su variabilidad del ritmo cardíaco (Heart-Rate Variability en inglés, HRV). Cada parte del corazón tiene un tiempo diferente para la excitación y la propagación de la misma, lo que afecta a los componentes de la señal del ECG.

Resumidamente, la curva de ECG consta de diferentes secciones que pueden asignarse al curso de la excitación cardíaca. Los procesos eléctricos de un ciclo se reflejan en ondas y picos característicos, de donde luego se saca el análisis del HR, por ejemplo. Los elementos de ECG se pueden asignar al proceso de excitación de la siguiente manera:

- Onda P: propagación de la excitación en las aurículas.
- Segmento PQ: transferencia de excitación de las aurículas a los ventrículos.
- Complejo QRS: propagación de la excitación en los ventrículos.
- Segmento ST: excitación completa de los ventrículos, comienzo de la re- excitación.
- Onda T: repolarización en los ventrículos.
- Intervalo TP: el periodo desde el comienzo de la propagación de la excitación en los ventrículos hasta el final de la despolarización.

La Figura 3 muestra la relación entre la propagación de la excitación en el corazón y las diversas partes de la curva de ECG [15].

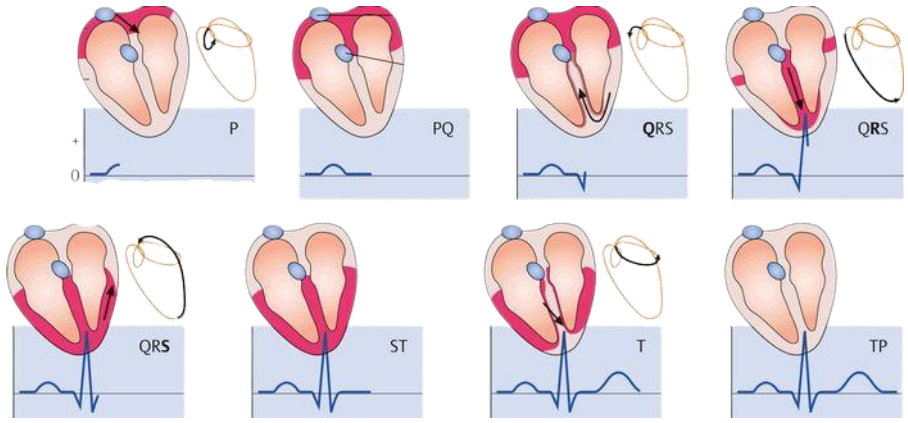


Figura 3. *Elementos de ECG en relación a la excitación del corazón.*

Debido a que la finalidad del estudio cardíaco en el presente trabajo es determinar los cambios fisiológicos, no se va a realizar un estudio profundo de la onda. Sobre todo, lo que se analizará del ECG es la variabilidad del ritmo cardíaco. Esta señal mide las variaciones temporales entre latido y latido, disminuyendo si el corazón se acelera y aumentando cuando el corazón se ralentice. Este parámetro se analizará calculando el intervalo entre los picos R de la onda ECG (ver Figura 4).

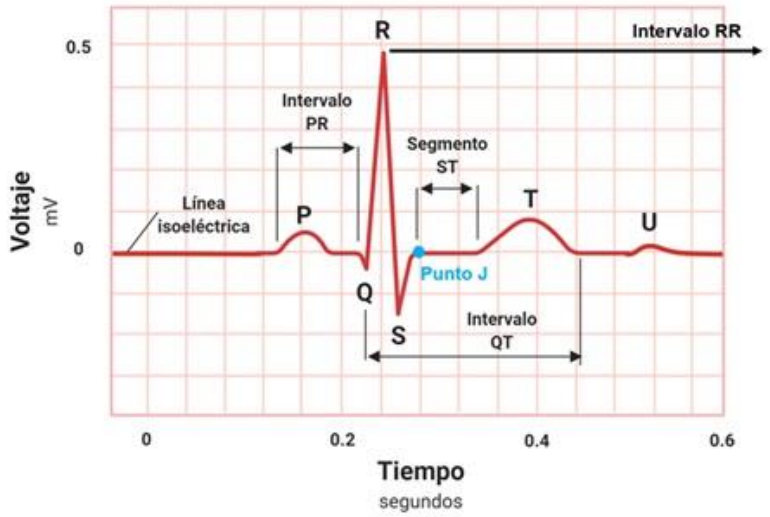


Figura 4. *Señal ECG. Nomenclatura y forma de las ondas.*

De forma similar a la EDA, el sistema cardiovascular ha sido ha sido caracterizado exhaustivamente a lo largo de décadas de investigación científica [16]. Se ha informado de que el sistema cardiovascular está implicado en ámbitos como la autorregulación, la fatiga [17] y el procesamiento emocional [18].

- *Otras señales fisiológicas*

Cuando un músculo se somete a un estímulo, voluntario o involuntario, se generan una serie de potenciales de acción que producen actividad eléctrica, y el electromiograma (EMG) es la señal que registra esta actividad mediante el uso de electrodos. Así pues, esta señal es representativa de la actividad muscular y por tanto se utiliza para diagnosticar patologías del sistema motor.

La adquisición se puede realizar colocando transductores sobre la piel o insertándolos dentro de un músculo de manera invasiva. La medida superficial es el método más empleado ya que no requiere de ninguna incisión y, por tanto, el riesgo para el paciente es mínimo. Las señales de EMG que se miden desde la superficie de la piel están en el rango de mili o microvoltios, dependiendo del tamaño del músculo. Como las señales EMG son sensibles al ruido, es necesario procesarlas para obtener una salida de buena calidad.

Una región del cuerpo muy interesante para medir las señales EMG es la cara, ya que con los músculos faciales dan información de emociones y, por tanto, es una fuente de información sobre el estado afectivo de las personas [19]. En el campo de la medicina también se utiliza para la investigación del movimiento y la rehabilitación. Algunos ejemplos en los que se aplica son las enfermedades neurodegenerativas como la apoplejía, párkinson y esclerosis lateral amiotrófica (ELA) [20].

Además del EMG, otra señal fisiológica es el electroencefalograma (EEG). Esta señal recoge los cambios de potencial eléctrico medidos entre dos puntos del cerebro mediante electrodos fijados sobre el cuero cabelludo. Las neuronas cerebrales se comunican a través de impulsos eléctricos y están activas durante todo el tiempo, incluso cuando se duerme. Por ello, esta señal sirve para detectar cambios en la actividad cerebral.

Una de las aplicaciones en el campo de la medicina donde se usa la señal EEG es la del diagnóstico de trastornos o enfermedades en el cerebro. Este es el caso de la epilepsia o los trastornos del sueño. La

EEG también puede ser útil cuando se utiliza con una interfaz cerebro-ordenador (BCI), por ejemplo, con pacientes con una lesión en la médula espinal o con esclerosis lateral amiotrófica (ELA). Estos pacientes con graves discapacidades motoras necesitan métodos de comunicación alternativos. Para ello, la BCI extrae características de las señales cerebrales y puede activar dispositivos externos como un interruptor, prótesis u ordenadores [21].

Por último, la respiración del sistema respiratorio es una señal que es fácil de adquirir. La respiración es el intercambio de los gases respiratorios oxígeno y dióxido de carbono. La respiración es clave en la regulación fisiológica y de los estados emocionales, y tiene estrecha relación con la variabilidad de ritmo cardíaco (HRV), donde a través de un determinado ritmo de respiración, se estimula una mayor variabilidad del ritmo cardíaco. Los médicos asocian una mayor variabilidad del ritmo cardíaco a una mayor adaptabilidad a los estímulos externos, por lo que, si se logra tener una mayor adaptabilidad a los estímulos externos, se tendrá una mayor capacidad para reponerse al estrés.

Esta última señal también se va adquirir en una fase de las pruebas que se van a realizar, ya que además de tener una estrecha relación con el HR, va a ayudar a observar si el dispositivo tiene la capacidad de recoger más de tres señales a la vez.

5.1.2. Análisis de la recogida de EDA

En la investigación psicofisiológica, la EMG, la ECG, la EDA y la EEG son medidas electrofisiológicas habituales para investigar la relación entre el comportamiento humano y su base fisiológica [22]. Como ya se ha dicho antes, este trabajo se ha centrado principalmente en la señal EDA y, por tanto, en esta sección se recogerán algunos de los trabajos más interesantes encontrados en la literatura que guardan relación con su recogida.

En los últimos años ha aumentado enormemente la popularidad de los dispositivos que dan viabilidad a la recogida de datos fisiológicos no invasivos y continuos [23]. La EDA es un ejemplo de ello, la cual se ha utilizado como marcador no invasivo del sistema nervioso autónomo en varias aplicaciones psicofisiológicas, como la excitación emocional, el estrés, el dolor, el trastorno de pánico, el autismo y la toma de decisiones [24].

Los dedos y los pies han sido aceptados durante mucho tiempo como lugares de registro óptimo para la actividad electrodérmica [25]. Sin embargo, a la hora de crear dispositivos portables para la recogida de datos, puede que estos lugares no sean prácticos. Es por eso que cada vez hay más investigaciones acerca de determinar lugares alternativos a los dedos.

De acuerdo a lo mencionado antes, los autores del artículo [26] trabajan comparando 16 diferentes sitios para colocar los electrodos para la señal EDA, en busca de una posición de registro diferente para un dispositivo portable. Los resultados indican que los pies, dedos y hombros son los más sensibles, mientras que, en los brazos, espalda, axilas y los muslos fueron menos aptas para la recogida de la EDA. En definitiva, concluye que el pie y los hombros son las mejores alternativas respecto a los dedos. El artículo [27] también analiza la colocación de los electrodos , donde llega a la misma conclusión. En la Figura 5 se muestran las 16 ubicaciones que tuvieron en cuenta para comparar la señal EDA con la de los dedos.

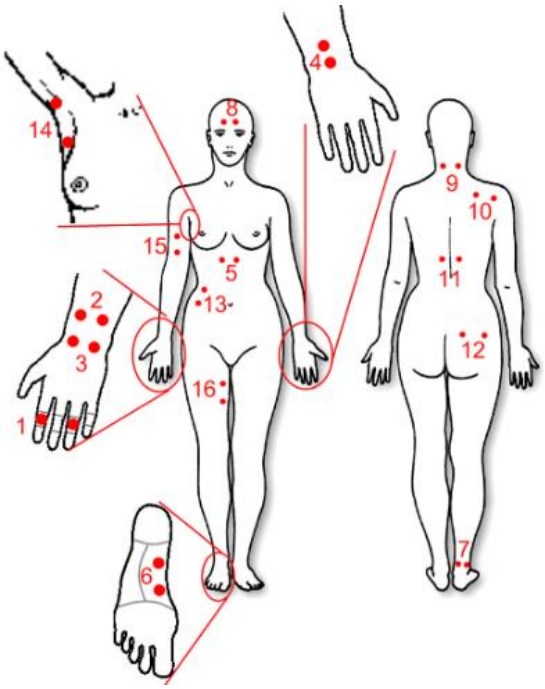


Figura 5. *Las 16 ubicaciones de los electrodos para la lectura de la EDA en el artículo* [26].

En relación a diferentes tipos dispositivos vestibles, en el artículo [28] presenta un dispositivo de monitorización de EDA para ser utilizado en sistemas informáticos relacionados con la salud. En este caso, el dispositivo se coloca en la muñeca, y recoge la señal EDA de los dedos índice y corazón. La Figura 6 muestra el dispositivo y la ubicación de los electrodos en este caso. Este método de colocación de los electrodos es el más utilizado, aunque no sea el más cómodo.

Figura 6*. Dispositivo de monitorización del artículo* [28].

En el artículo [29] también se adquiere en otra ubicación. En este caso, los electrodos están ubicados en el recto abdominal (véase la Figura 7). De esta manera se busca que la recogida sea lo más portable y cómoda posible para el usuario. Este artículo se basa en el trabajo [30], donde se analizan diferentes glándulas sudoríparas del cuerpo humano, observando la secreción de sudor.

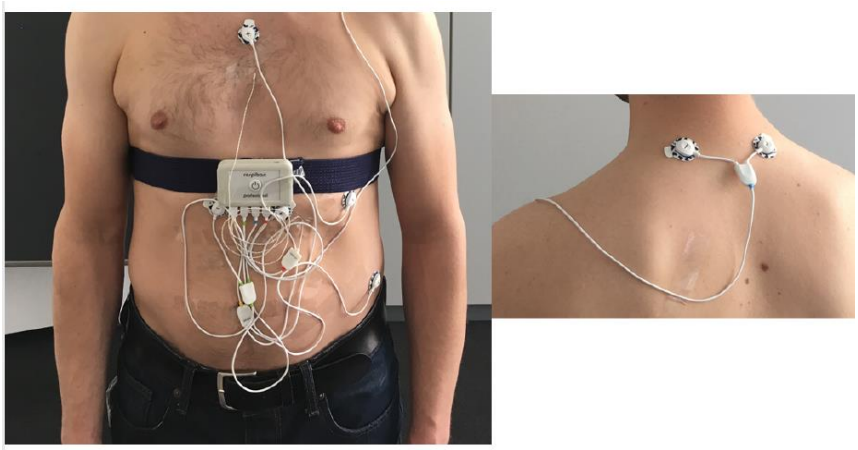


Figura 7. *Dispositivo que plantea el artículo* [29] *para la recogida de señales.*

Por otra parte, en el caso de [31], los autores miden la HRV, la EDA y la EMG del músculo masetero de un piloto profesional durante una carrera de coches real, colocando los electrodos de la EDA en las glándulas sudoríparas que hay detrás de las orejas, como se ve en la Figura 8. El objetivo del trabajo era comparar las tres señales entre sí utilizando la correlación, y tras las pruebas comprobaron que la frecuencia cardíaca y la EDA estaban vinculadas, pero la actividad del músculo no.

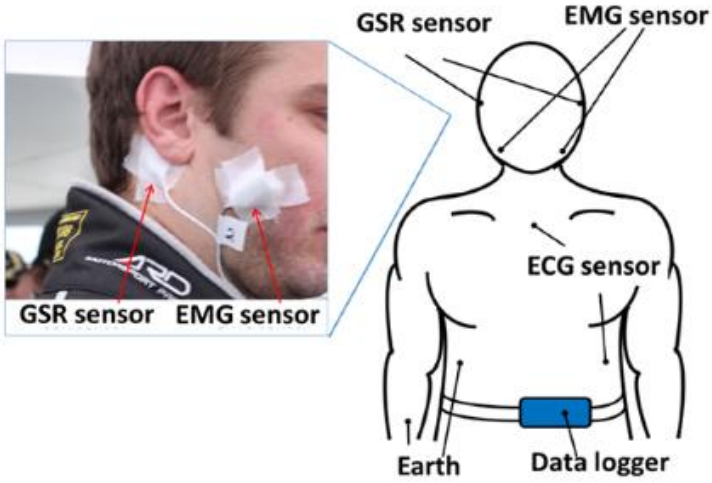


Figura 8. *Posición de los sensores GSR, EMG y ECG del artículo* [31].

6. ANÁLISIS DE ALTERNATIVAS

Una de las tendencias actuales de la ingeniería biomédica es la monitorización ubicua de la condición de salud y la búsqueda de nuevas formas progresivas de medición. Hay muchos fabricantes (gtec, BIOSEB, BIOPAC, etc.) que ofrecen bioamplificadores dedicados a medir varios tipos de señales bioeléctricas. Estos dispositivos son capaces de medir y almacenar datos con excelente precisión. Las desventajas de estos dispositivos para su uso en diversas aplicaciones son su alto precio y su sistema no abierto, cuya implementación para un tipo específico de medición puede ser un inconveniente. Debido a esta desventaja, estos dispositivos no son apropiados para aplicaciones en las que es necesario crear un software único que tenga que comunicarse con el hardware.

Como se ha comentado durante este trabajo, el Grupo de Investigación ya dispone del dispositivo de bajo coste para realizar el proyecto, junto con los sensores para leer las señales fisiológicas escogidas. Se va a trabajar con el dispositivo BITalino, creado por la empresa PLUX. Aun así, en este apartado se va a realizar un análisis de alternativas sobre los diferentes dispositivos hardware que se van a utilizar en la validación del dispositivo de bajo coste, además de analizar la placa BITalino. También se explicarán las plataformas software con la que trabaja cada dispositivo. Además, el grupo necesita una aplicación validada por ellos mismos que funcione con un software propio para unas situaciones específicas del proyecto. Por tanto, en la sección 6.1.4, se realiza una búsqueda de trabajos donde hablan de cómo validar el hardware, y, por otro lado, de qué características tiene que tener el software en la adquisición de señales.

6.1. Hardware de adquisición de datos

Como se acaba de mencionar, el grupo ya tiene escogida la placa con la que se va a trabajar, aunque se pueden encontrar más opciones de dispositivos de bajo coste para la adquisición de señales fisiológicas. Por ejemplo, en la bibliografía se pueden encontrar diferentes proyectos que tienen un objetivo común con este trabajo: encontrar o diseñar un dispositivo de bajo coste. En el artículo [32] se presentó un bio-amplificador de bajo coste para la interfaz cerebro-ordenador (BCI), o los autores del artículo [33] diseñaron otro bio-amplificador para la monitorización inalámbrica de ECG. Aun así, a continuación, se van a presentar los diferentes dispositivos para la adquisición de señales con las que se ha trabajado, además de diferentes alternativas o kits que ofrece PLUX con sus productos, dependiendo de lo que se quiera enfocar el proyecto.

6.1.1. BioPac SL

La recogida de señales hecha por BITalino que se propone en este proyecto se va a comparar con un dispositivo de referencia. En este caso, será el BIOPAC MP 36 System (Biopac Systems Inc., USA) el equipo con el que se trabaje.

La unidad MP36 tiene un microprocesador interno que permite controlar la adquisición de datos, recogiendo las señales fisiológicas y convirtiéndolas en señales digitales que pueden ser procesadas posteriormente. Como se puede observar en el panel frontal del dispositivo (Figura 9), puede recoger hasta 4 bioseñales diferentes, todas a la misma frecuencia de muestreo.

Figura 9. *Dispositivo BioPac MP36, por la parte delantera y la parte trasera.*

Los datos recogidos son enviados a un ordenador vía puerto USB. El software AcqKnowledge permite visualizar las señales fisiológicas de manera online e ir introduciendo marcas en los registros junto a comentarios explicativos. Los datos se pueden guardar con extensión ". Acq" para un posterior análisis de las mismas dentro del propio programa, o ".txt" para ser leídas desde cualquier otro

6.1.2. BITalino

BITalino es una plataforma de bioseñales asequible y de código abierto que permite a cualquier persona, desde estudiantes hasta desarrolladores profesionales, crear proyectos y aplicaciones utilizando sensores fisiológicos. Es la empresa PLUX quien ha desarrollado la plataforma en abierto BITalino, al igual que el biosignalsplux analizado en el anterior apartado. Sus productos están destinados al uso en aplicaciones de educación, investigación y creación de prototipos en el ámbito de las ciencias de la vida. Aunque cabe destacar que no son dispositivos médicos, ni están diseñados para el diagnóstico médico, cura, mitigación, tratamiento o la prevención de enfermedades.

Figura 10. *Logo del BITalino* [34].

El hardware consiste en un equipo modular e inalámbrico de bajo coste, con un formato del tamaño de una tarjeta de crédito que integra múltiples sensores de medición para datos bioeléctricos y biomecánicos. El corazón del kit es la unidad microcontroladora, que contiene un microprocesador AVR ATMEGA328P, encargada de convertir las señales analógicas recogidas por los diferentes módulos a señales digitales. El BITalino se suministra actualmente como un kit que incluye todos los componentes básicos que se necesitan para iniciarse en el mundo de la captura y procesamiento de bioseñales, es decir, los bloques de hardware; una batería de 550 mAh de LiPo recargable; un conjunto de cables de electrodos (para ECG y EDA); un conjunto de cable de electrodos de tres derivaciones (para ECG y EMG); y un paquete de cinco electrodos pregelificados de Ag/AgCl polivalentes que pueden utilizarse para la adquisición de datos de ECG, EMG o EDA.

Respecto al consumo energía de este dispositivo, en el peor de los casos en el que todos los sensores y los LEDs están conectados simultáneamente y utilizando una frecuencia de muestreo de 1.000 Hz, el BITalino consume alrededor de 65 mAh (aproximadamente el 60% del módulo Bluetooth por sí solo y aproximadamente el 15% de los LEDs). Si sólo se utilizan uno o dos sensores, se utiliza una media de aproximadamente 50 mAh, lo que permite a BITalino funcionar continuamente durante cerca de 10 horas utilizando la batería estándar incluida en el kit.

Como se ha mencionado anteriormente, en la página oficial de BITalino, se pueden encontrar diferentes tipos de placas y accesorios para elegir, agrupándose en diferentes KITs. Esto hace que se pueda elegir el KIT que mejor convenga para la aplicación que se desee crear. A continuación, se van a describir brevemente los cinco paquetes que se ofrecen con las diferentes tablas.

BITalino revolution Board: El kit BITalino unevolution Board tiene un diseño de hardware todo en uno, con todos los módulos preconectados entre ellos y listos para usar. Además, pueden acoplarse nuevos módulos o bloques a la placa.

BITalino revolution Plugged: Los kits enchufados proporcionan los bloques como una placa principal (el núcleo del BITalino) y los sensores/actuadores como componentes separados. Los conectores UC-E6 del núcleo y de los sensores permiten la conexión plug & play (por cable) de bloques BITalino o de terceros a los puertos analógicos y digitales. Los sensores que vienen con el kit son: EMG, ECG, EDA, EEG, ACC y LUX.

BITalino mini Solo: El BITalino Mini Solo representa el producto del grupo BITalino más simple y pequeño de todos ellos. Tiene un puerto UC-E6 integrado para que se pueda utilizar tanto como entrada analógica como para entrada/salida digital, seleccionando la función deseada con un puente que permite el uso de todos los sensores ya existentes, lo que hace que su uso sea rápido y sencillo.

BITalino mini Shielded: Este kit proporciona las mismas características que el mini solo, pero se podrán aprovechar todos los puertos disponibles a través de los puertos UC-E6 del BITalino Mini Shield.

BITalino R-IoT: El BITalino R-IoT incorpora un eje de 9 bits y cálculo a bordo de la orientación absoluta del módulo en el espacio. El núcleo del módulo BITalino RioT se basa en el chip CC3200 de Texas Instruments y es compatible con Energia (https://energia.nu), una herramienta de programación para los procesadores de TI con la facilidad del Arduino.

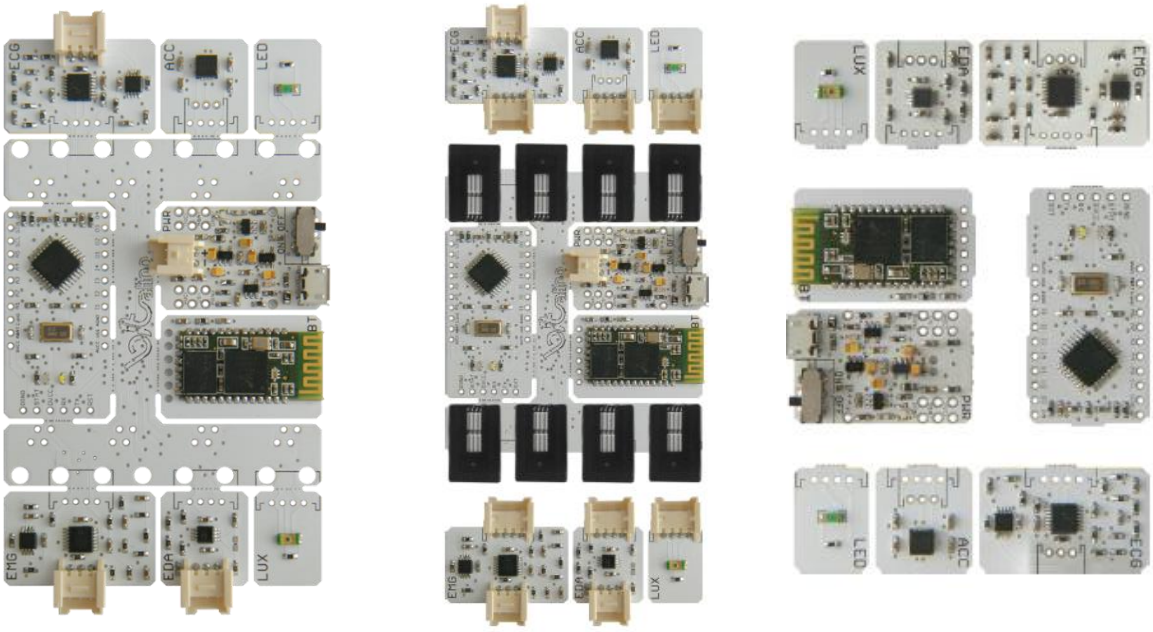
Las especificaciones de las placas vistas hasta ahora se pueden resumir en la Tabla 1.

Tabla 1. *Resumen de las características de los diferentes dispositivos BITalino.*

	Tamaño	**Sampling Rate**	**Analog Ports**	**Digital Ports**	**puerto UC-E6**	**Sensores**	**Comunicación**	**Consumo**	**Resolución**
BITalino revolution Board	100x65x6mm	1, 10, 100 or 1000 Hz	7E/1S	2E/2S		EMG, ECG, EDA, EEG, ACC, LUX Y BTN	Bluetooth y BLE	~ 65mA	
BITalino revolution Plugged	100x65x6mm	1, 10, 100 or 1000 Hz	7E/1S	2E/2S	Sí	EMG, ECG, EDA, EEG, ACC, LUX y BTN	Bluetooth y BLE	~ 65mA	
BITalino mini Solo	56x21x6mm	1,10, 100 or 1000 Hz	7E/1S	2E/2S	A1/I1O 1		BLE, SPI y UART	~ 34mA	
BITalino mini Solo Shielded	56x21x14mm	1, 10, 100 or 1000 Hz	7E/1S		A1, A2, A3, A4, A5 A6 y I1O1			~ 34mA	
BITalino R-IoT	34x21x6mm	200 Hz	2E (A1&A 2)	1E/1S		1 IMU 9DoF	WIFI y UART	~100mA	12-bit (A1&A2) + 16-bit (IMU)

Por defecto, el sistema viene como una placa única, con sensores a bordo preconectados a puertos analógicos y digitales en el bloque de control. Sin embargo, los bloques de control, alimentación y comunicación, así como el firmware, son completamente de propósito general, lo que permite que la gente lo utilice con sus propios diseños y actuadores. La versatilidad de la plataforma BITalino se extiende incluso hasta el punto de que cada bloque individual puede separarse físicamente de la placa principal, lo que permite utilizarla de muchas maneras diferentes, normalmente en las siguientes tres configuraciones:

- ***Board,*** o con la placa principal (sin modificaciones), por lo que personas pueden simplemente experimentar con los sensores de a bordo para apoyar actividades experimentales o ilustrar conceptos teóricos conceptos teóricos a través de la observación de los fenómenos fisiológicos subyacentes (Figura 11a);
- ***Plugged***, o enchufado, en el que la parte frontal analógica se separa de la placa principal del BITalino, dejando sólo el control, alimentación, comunicación y bloques de conectividad para que se puedan utilizar indistintamente diferentes combinaciones de sensores, conectando el sensor por cable (Figura 11b); y
- ***Freestyle***, o estilo libre, en el que todos los bloques digitales y analógicos se separan de la placa principal de BITalino, lo que permite que los usuarios puedan combinarlos de la de la forma que mejor se adapte a sus ideas de proyecto y aplicaciones (Figura 11c).



(a) (b) (c)

Figura 11. *Hardware de adquisición de bioseñales BITalino en sus diferentes configuraciones: (a) board, (b) plugged, (c), freestyle.*

Una vez analizadas las diferentes placas que se ofrecen y los formatos en los que se pueden presentar, el Grupo de Investigación trabaja con la placa BITalino (r)evolution board, con la configuración plugged. Así, durante el proyecto se hace más accesible su uso ya que tan solo se usan los módulos necesarios. En la Figura 12 se observa la placa con la que se ha trabajado, la cual viene con una caja para portarla.

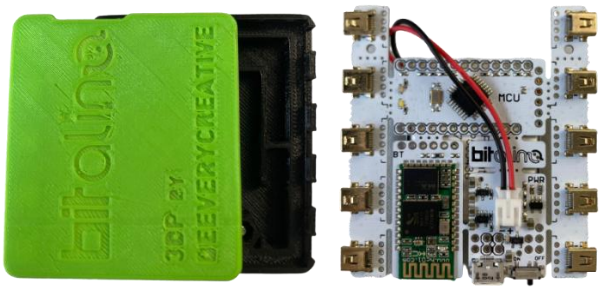


Figura 12. *Caja para portar el BITalino y la placa.*

6.1.2.1. Sensores para la adquisición de señales

El BITalino es una plataforma de desarrollo creada para la experimentación y creación rápida de prototipos basado en bioseñales, es por eso que está diseñada con entradas analógicas, lo que permite trabajar con las diferentes bioseñales, por ejemplo: ECG, EMG, EDA, ACC y LUX.

Aun así, como se ha mencionado en el estado del arte, los sensores menos invasivos para la monitorización y que tienen una relación directa con las emociones son la EDA y ECG. Además, la respiración también es un factor fácil de analizar y que tiene correlación con estas dos señales. Es por eso que son estas las señales que se van a adquirir durante las diferentes pruebas que se realicen.

En este apartado, se van a mencionar los módulos que ofrece BITalino para la recogida de las señales mencionadas anteriormente, y, además, se analizará la ubicación de los electrodos que la empresa PLUX propone y cómo interpretar los datos que obtenemos.

Módulo EDA

Para la medición de la actividad electrodérmica, se ha empleado el módulo EDA de la marca BITalino [35]. El sensor contiene una configuración de medición bipolar con dos pines de medición (IN+ e IN-). Estos dos pines se conectan a los cables para permitir la conexión entre los electrodos colocados en la superficie del cuerpo y el sensor, como se ha comentado durante el trabajo.

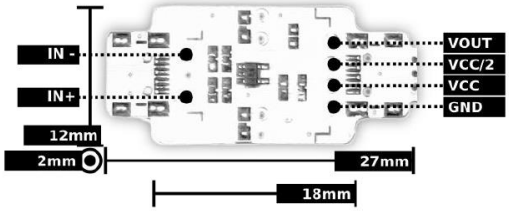


Figura 13. *Módulo EDA de BITalino.*

Las características principales se muestran en la Tabla 2:

Tabla 2. *Características principales del módulo EDA.*

Ganancia	2
Rango (µS)	0 – 24 µs (con 3.3V)
Ancho de banda (Hz)	0 - 2.8 Hz
Consumo (mA)	±0.1 mA
Electrodos	2

La función de transferencia [0.5µS - 25µS] empleada para la conversión de valores adquiridos por el canal es la siguiente:

$$EDA\ (\mu S) = \frac{\frac{ADC}{2^n} x\ VCC}{0.132} \qquad (1)$$

$$EDA\ (S) = EDA\ (\mu S)\ x\ 1\ x\ 10^{-6} \qquad (2)$$

Donde,

- VCC = 3.3 V
- EDA (µS) – Valor de EDA en micro-Siemens
- EDA (S) – Valor de EDA en Siemens
- ADC – Valor muestreado del canal
- n – Número de bits del canal, el cual dependerá de la resolución que tenga dicho canal (los primeros 4 canales tienen una resolución de 10 bits, mientras que los dos últimos solo utilizan 6 bits).

En los primeros diseños para las pruebas de validación, los dos electrodos se van a colocar en la mano, como se propone en su hoja de especificación. En este caso, las posiciones que mejor adquieren la EDA pueden ser tres, como se puede observar en la Figura 14. Cuando se realice la recogida de señales con

el BITalino y el BioPac al mismo tiempo se utilizará la primera y segunda posición de los electrodos para las primeras pruebas de correlación.

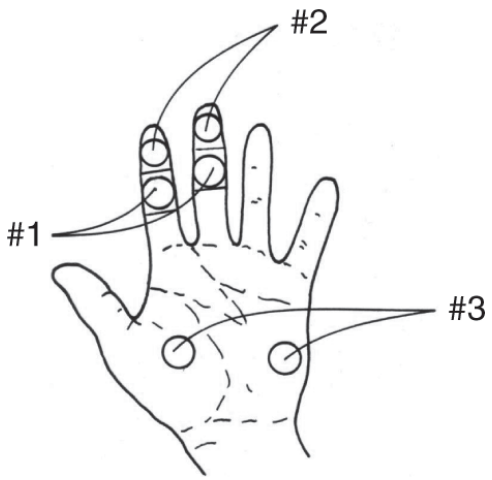


Figura 14. *Colocación de los electrodos para la lectura de EDA.*

Módulo ECG

Para la medición del electrocardiograma, se ha empleado el módulo ECG de la marca BITalino [36].

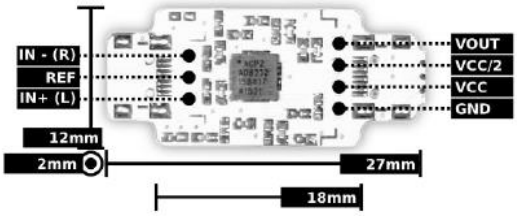


Figura 15. *Módulo EGC de BITalino.*

Las características principales se muestran en la siguiente Tabla 3:

Tabla 3. *Características principales del módulo ECG.*

Ganancia	1100
Rango (mV)	± 1.5 mV (con 3.3V)
Ancho de banda (Hz)	0.5 - 40 Hz
Consumo (mA)	± 0.17 mA
Electrodos	3

La función de transferencia [-1.5mV, 1.5mV] empleada para la conversión de valores adquiridos por el canal es la siguiente:

$$ECG\ (\mathrm{V}) = \frac{\frac{ADC}{2^n} x\ VCC}{G_{ECG}} \quad (3)$$

$$ECG\ (\mathrm{mV}) = ECG\ (\mathrm{V})\ x\ 1000 \quad (4)$$

Donde,

- VCC = 3.3 V

- G_{ECG} = 1100 (ganancia del sensor)
- ECG (mV) – Valor de ECG en minivoltios
- ECG (V) – Valor de ECG en Voltios
- ADC – Valor muestreado del canal
- n – Número de bits del canal, el cual dependerá de la resolución que tenga dicho canal (los primeros 4 canales tienen una resolución de 10 bits, mientras que los dos últimos solo utilizan 6 bits).

En este caso en la recogida de señales los tres electrodos se colocarán creando un triángulo invertido tanto con el BITalino como con el BioPac. Cabe destacar que siguiendo la ficha de datos del módulo ECG del BITalino [36], el color de los cables de los electrodos corresponden de la siguiente manera:

***Tabla 4**. Disposición de los electrodos del ECG del BITalino.*

Color	Rojo	Negro	Blanco
Cable Electrodo	+	-	referencia

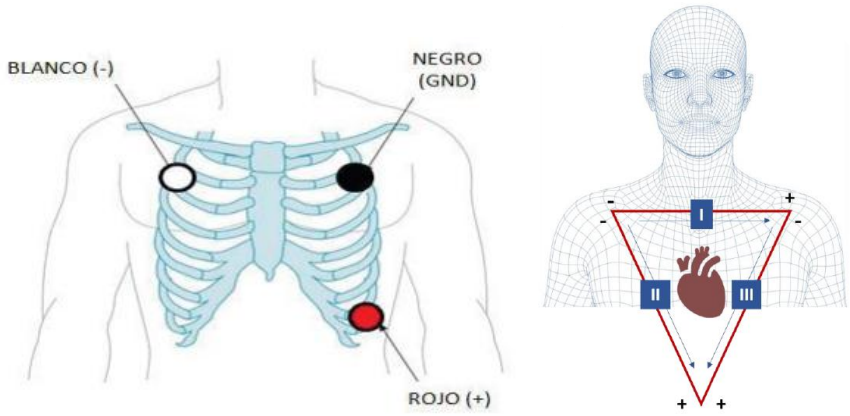


***Figura 16.** Colocación de los electrodos del módulo ECG con 3 derivaciones.*

Sensor para Respiración

Para la medición de la respiración, se ha empleado el siguiente módulo PTZ de la marca BITalino [37]. Como se puede observar en la imagen, el sensor está provisto de una correa pectoral elástica para fijarlo en el pecho, y poder adquirir los datos de una manera más sencilla. Este módulo solo se utilizará con el BITalino.

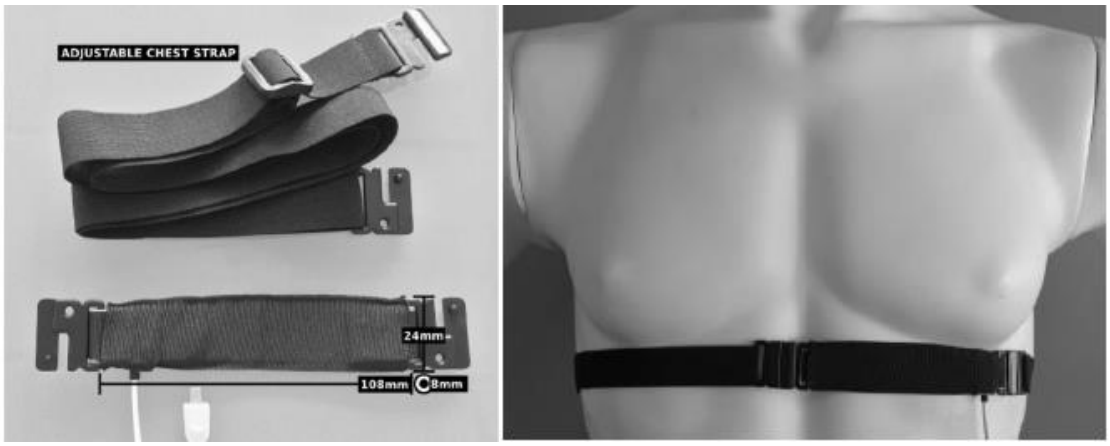


***Figura 17.** Sensor PZT provisto la cinta elástica, y ejemplo de la colocación de la misma.*

Las características de este sensor en este caso son las siguientes,

Tabla 5. *Características principales del sensor PZT.*

Ancho de banda (Hz)	0.59 – 0.9 Hz
Consumo (mA)	± 0.5 µA

La función de transferencia [-50% - 50%] en este caso, se calcula de acuerdo a la siguiente ecuación:

$$PZT\ (\%) = \left(\frac{ADC}{2^n} - \frac{1}{2}\right) x100\% \qquad (5)$$

Donde,

- PZT (%) – Valor de desplazamiento en porcentaje (%) de la escala
- ADC – Valor muestreado del canal
- n Número de bits del canal, el cual dependerá de la resolución que tenga dicho canal (los primeros 4 canales tienen una resolución de 10 bits, mientras que los dos últimos solo utilizan 6 bits).

6.1.3. Biosignalsplux

Este hardware de adquisición de señales esta desarrollado también por la empresa PLUX. Biosignalplux es una plataforma de adquisición de bioseñales con sistemas de sensores portátiles y vestibles para aplicaciones avanzadas de I+D de bioseñales.

Mientras que BITalino es una plataforma de bioseñales asequible, plug & play y versátil, diseñada para la educación y la creación de prototipos en un entorno de laboratorio o aula, biosignalsplux está más enfocado a proyectos de bioseñales más avanzadas. Asume este reto proporcionando todas las herramientas necesarias para utilizar casi cualquier combinación de sensores que se necesite para los proyectos de bioseñales, y hay que destacar también que hay una diferencia considerable en el precio de este kit.

Los sistemas biosignalsplux están disponibles en 4 versiones de diferentes kits, dependiendo del proyecto que se quiera realizar o de las señales que se quieran adquirir. En este caso, el Grupo de Investigación dispone de un kit *biosignalplux Researcher,* el cual contiene un equipo biosignalsplux HUB con ocho canales, 8 sensores profesionales a elegir (en este caso son, EDA, 2 EEG, EMG, EEG, PZT, EMG y ACC), 1 Bluetooth dongle y 24 electrodos entre otros elementos.

Figura 18. *HUB biosignalplux de 8 canales.*

El HUB biosignalsplux, véase Figura 18, es el dispositivo de 4 u 8 canales que recoge y digitaliza todas las señales de los sensores y accesorios y las transmite por Bluetooth al ordenador donde se registran y visualizan online. Los canales admiten una resolución de hasta 16 bits resolución y una frecuencia de muestreo de 3000 Hz por canal, es decir, hasta 8 canales con 3000 de 16 bits por canal y por segundo, o una frecuencia de muestreo de 4000 Hz por canal cuando utilizando sólo hasta 3 canales simultáneamente. Estas frecuencias de muestreo son mucho mayores que los de BITalino.

Biosignalsplux es compatible con todos los sensores profesionales biosignalsplux disponibles, sensores avanzados y accesorios. Los dispositivos con memoria interna también permiten opciones de registro de datos a bordo, sin la necesidad de una conexión Bluetooth permanente.

Las características principales de este dispositivo son las siguientes: está diseñado para la investigación avanzada de bioseñales, la adquisición de datos brutos es de calidad médica, está listo para usar dentro y fuera del laboratorio, es capaz de autodetectar el sensor que se incorpora al canal y está previsto de memoria interna para la adquisición de datos fuera de línea. El uso previsto en el caso de aplicaciones de investigación podría ser en estudios de ciencias de la vida, investigación biomédica, investigaciones acerca de la iteración persona-ordenador, en las ciencias del deporte y biomecánica y un largo etcétera.

6.1.4. Análisis de proyectos realizados con BITalino

Como se ha mencionado, la fiabilidad de los sistemas móviles de bajo coste para el registro de datos fisiológicos es mayoritariamente desconocida, lo que plantea dudas sobre la calidad de los datos registrados y la validez de los parámetros fisiológicos extraídos. Por ello, tras analizar las diferentes opciones que da la empresa PLUX en torno a sus productos y las diferentes alternativas respecto a dispositivos de referencia, se ha realizado una búsqueda de diferentes trabajos donde, a través de la adquisición de datos desde BITalino, se han obtenido resultados acordes con lo esperado. Cabe destacar, que, para el análisis de los datos recogidos, también existen diferentes softwares y plataformas de código abierto donde poder hacerlo, por lo que también se hará una breve investigación acerca de ello más adelante.

En el análisis del BITalino se ha destacado que la plataforma está basada en un microcontrolador ATmega, la cual también está basada en Arduino. Esta última es actualmente la plataforma hardware de bajo coste más popular dentro de la comunidad del Do-it-Yourself (DiY). Y aunque el Arduino no está diseñado específicamente para tener en cuenta los requisitos de la informática fisiológica, en el

artículo [38] se hace una comparación entre estos dos dispositivos. Los autores de [38] realizan un análisis comparativo entre las plataformas Arduino y BITalino con el objetivo de evaluar su rendimiento para la adquisición de bioseñales. También describe algunas aplicaciones de computación fisiológica construidas con el BITalino mostrando el potencial de esta plataforma dentro de la comunidad de ingenieros.

Como se ha mencionado anteriormente, es importante estudiar cómo los datos adquiridos por estos dispositivos se comparan con los recogidos por los equipos de referencia. Esta comparación permitirá evaluar la calidad de las señales recogidas y ayudar a validar el trabajo de investigación que se desarrolla con las bioseñales adquiridas con las plataformas BITalino. Uno de estos estudios es el que se realiza en [39], demostrando la correlación entre los datos electrocardiográficos adquiridos de forma "no presencial" con BITalino y un sistema de grado clínico.

También se analiza en el siguiente trabajo la comparación experimental de los sensores BITalino y BITalino (r)evolution contra un dispositivo de referencia (en este caso un sistema de grado de investigación Biosignalsplux) [40]. Este estudio analiza el conjunto de sensores estándar de BITalino, en particular la EMG, la EDA, la ECG, la ACC y la EEG. Aunque se detecta que en la versión más antigua de BITalino a veces había un problema de saturación de la señal EDA, el estudio concluye que no hay diferencias importantes entre las señales adquiridas de este tipo. Respecto a las demás señales, el estudio concluyó que había una gran similitud morfológica y por lo tanto se consideró el dispositivo BITalino revolution como apto para la mayoría de las aplicaciones por su buena concordancia con el dispositivo de referencia (Biosignalplux).

Otro estudio donde se han conseguido resultados positivos respecto a la validación de la recogida de la señal ECG mediante el BITalino es en el artículo [41]. En el presente se ha comparado el kit de herramientas BITalino con un sistema de ECG de grado médico establecido (BrainAmp-ExG). Para el análisis, los participantes se sometieron a registros simultáneos de ECG con los dos instrumentos mientras veían imágenes agradables y desagradables. Se extrajeron los parámetros comunes del ECG y se compararon entre los dos sistemas. En los resultados, todos los parámetros, excepto uno, mostraron una excelente concordancia (>80%) entre ambos dispositivos en el análisis demostrando que el sistema BITalino puede considerarse como un dispositivo de registro equivalente para los registros estacionarios de ECG en experimentos psicofisiológicos. En la Figura 19 se observa el esquema de la prueba mencionada.

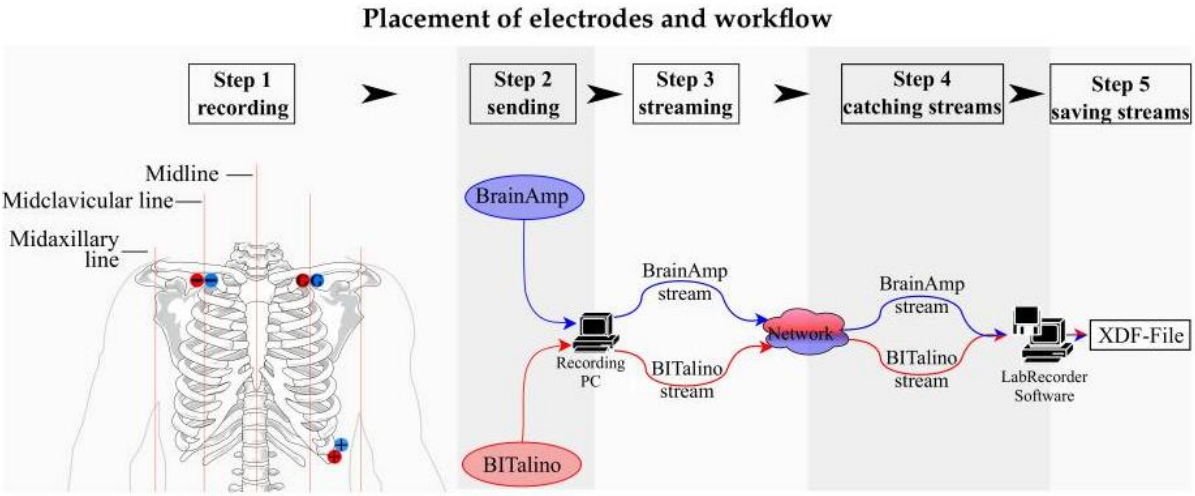


Figura 19. *Flujo de trabajo para adquisición de datos y posterior comparación de señales del artículo* [41].

Los artículos mencionados hasta ahora, validan al BITalino para la investigación biomédica, pero, aun así, ponen en relieve varios aspectos metodológicos y prácticos que requieren una evaluación adicional. Se señala la necesidad de realizar una evaluación comparativa con un estándar de referencia más establecido y ampliamente utilizado. Por ello, en la siguiente investigación se ha proporcionado una evaluación del BITalino para un rendimiento mayor mediante el BioPac [42], el sistema más establecido y reconocido para la investigación y la educación biomédica. Este estudio se ha centrado en particular en las siguientes cuatro señales fisiológicas: ECG, EMG, EDA y EEG. Para la adquisición de datos con BITalino se ha utilizado el software OpenSignals (r)evolution, que es el recomendado para ser utilizado con BITalino. Para el sistema BioPac elegido como patrón de referencia con el que se compara el BITalino se utilizó la aplicación de software BSL Pro. Tras recoger los datos de ambos dispositivos en diferentes sujetos y posprocesar las señales para obtener una comparación más justa, el artículo concluye que las señales de EDA eran muy similares para ambos dispositivos. En el caso del ECG, una simple comparación reveló una gran similitud entre los dispositivos. Es decir, las diferencias entre BITalino y BioPac fueron de nuevo pequeñas y los resultados presentados muestran una gran similitud entre los datos adquiridos por el BITalino y por el dispositivo de referencia.

En la Figura 20 se exponen los resultados de este último artículo, pudiendo observar la similitud entre ambas señales.

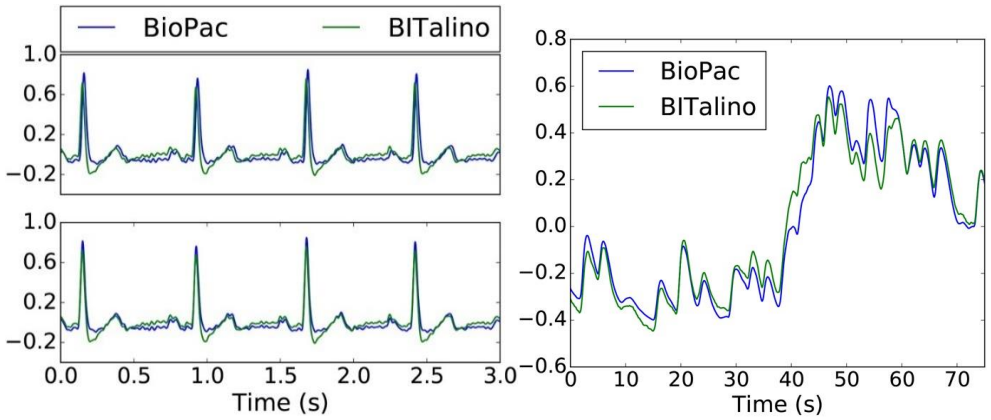


Figura 20. *Resultados del artículo* [42]. *A la izquierda comparación de la señal ECG y a la derecha, la señal EDA.*

6.2. Software de adquisición de datos

A la hora de realizar la lectura de datos de los sensores de la placa BITalino, se tiene que tener en cuenta el procesamiento de los datos recogidos por este. Es por ello que, aunque BITalino tiene su marco de software OpenSignals (antes conocido como SignalBIT) donde se puede descargar y utilizar para visualizar y registrar datos de bioseñal online o para revisar datos previamente registrados; también es posible elegir entre un número creciente de APIs de programación para lenguajes como Python, Java y Matlab, entre otros. Esto permite a los usuarios comunicarse con BITalino y acceder a los datos de los sensores en sus propias aplicaciones.

Realizando un análisis de alternativas para la adquisición de datos, a la hora de procesar las señales que adquieren los dispositivos, también se encuentran diferentes softwares que hacen el trabajo más sencillo. Aun así, muchos de ellos carecen de funcionalidades específicas dependiendo del análisis a realizar. Estos programas informáticos pueden ser Open Sesame [43] y PsychoPy [44]. Por ello, en el artículo [45], presentan el diseño de una herramienta de investigación neuroIS basada en BITalino, una plataforma de captura y procesamiento de bioseñales. El prototipo que proponen se ha derivado como una respuesta a la escasez de un sistema conveniente y autónomo, que tenga la capacidad de construir un experimento, controlar su flujo, adquirir y visualizar los datos, y potencialmente procesar y analizar las señales almacenadas, ya que, disponer de un sistema autónomo reduce la complejidad de la realización de experimentos, además de reducir sus costes.

Cabe destacar que no solo se va a hacer uso del BITalino, sino que este dispositivo se va a comprar con otros de referencia. Por ello, se van a analizar estas alternativas también. En el esquema de la Figura 21 se exponen las opciones para la adquisición de datos para los diferentes dispositivos. Como punto a destacar, todos los softwares propuestos tienen la capacidad de guardar los datos en txt, lo que facilita el posterior análisis de estos en Matlab. Queda pendiente para una futura investigación, el integrar a la interfaz de Python el análisis de los datos que se leen de forma online. A continuación, se analizan los diferentes softwares.

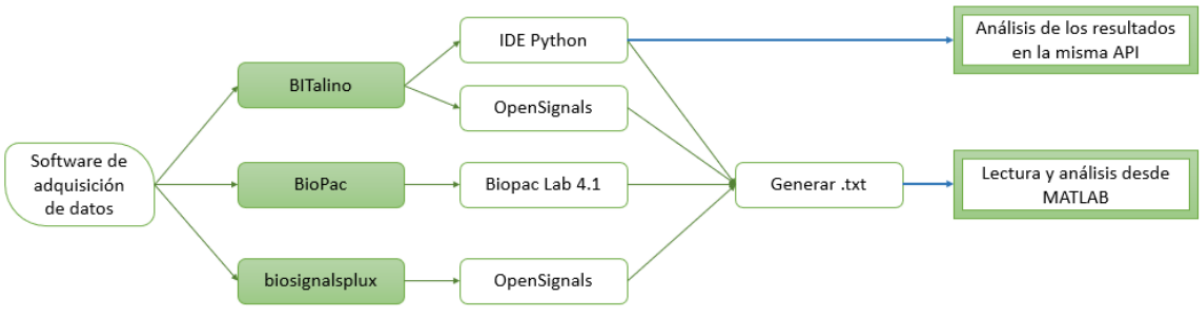


Figura 21. *Esquema de los SW para la adquisición de datos.*

6.2.1. OpenSignals

Como se ha mencionado anteriormente, a nivel de usuario, BITalino ofrece su software OpenSignals, un software fácil de usar y versátil para la visualización de bioseñales online, compatible con todos los dispositivos PLUX.

Las funcionalidades principales incluyen la adquisición de datos de sensores de múltiples canales y dispositivos simultáneamente, la visualización y el registro de datos. Esto hará posible la validación del BITalino con el biosignalsplux. Además, OpenSignals cuenta con un conjunto de complementos de análisis de datos para crear informes a partir de los datos registrados y extraer características directamente de las señales sin tener que hacer ninguna codificación. Este software está disponible para diferentes sistemas operativos como Windows, macOS y Linux; incluso se podrían analizar los datos mediante un dispositivo Android.

Figura 22. *Logotipo del software OpenSignals* [46].

La Figura 23 muestra la interfaz gráfica de usuario para el menú principal, donde se observan dos dispositivos hardware ya reconocidos por el software. El BITalino está vinculado al ordenador mediante la dirección MAC.

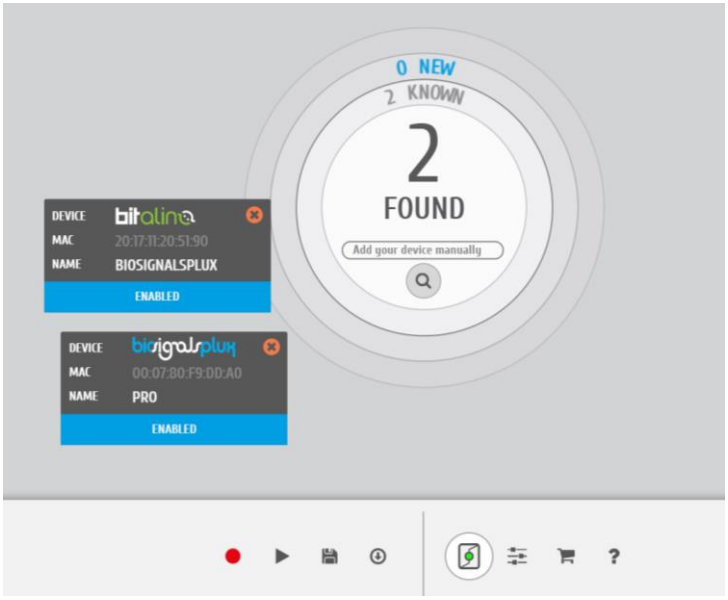


Figura 23. *Pantalla principal del software OpenSignals (r)evolution.*

Resumidamente, estas son las ventajas del software que va a aportar al proyecto [47]:

- Es compatible con todos los dispositivos PLUX con Bluetooth, en este caso BITalino y el biosignalsplux.
- Adquisición y visualización de señales online de hasta 24 canales simultáneamente, es decir, se pueden adquirir datos de hasta tres dispositivos a la vez.
- Exportación de datos en formatos .txt, H5 y .EDF.

6.2.2. Biopac Student Lab 4.1

El sistema Biopac Student Lab es un dispositivo y método de enseñanza que integra software y hardware para la adquisición y análisis de ciencias [48]. Fue el reemplazo digital para los registradores y osciloscopios de gráficos antiguos. Este dispositivo registra la información de señales sobre su estado fisiológico, como la temperatura de la piel, la señal del corazón o la sudoración de la piel.

Figura 24. *Biopac Student Lab.*

A diferencia del OpenSignals, esta plataforma de análisis trae consigo lecciones sobre diferentes señales donde se explican y se muestran diferentes ejemplos de adquisiciones.

En el ANEXO I se presenta un ejemplo de adquisición de datos mediante las plataformas OpenSignals y Biopac Student Lab 4.1, explicado paso a paso el procedimiento que se debe seguir para configurar ambos softwares.

6.2.3. IDE Python

Todos los sensores y periféricos de BITalino son compatibles con las plataformas de plataformas de hardware de DiY, y la comunicación puede ocurrir a nivel de flujo de bytes, utilizando el protocolo de comunicación implementado en el firmware. Por ello, es posible elegir entre APIs de programación en diferentes lenguajes. En este trabajo, se ha escogido trabajar con Python, dado su versatilidad en este ámbito y la facilidad de implementar diferentes librerías tanto para la conexión vía Bluetooth o la lectura de ficheros de BITalino.

Figura 25. *Logotipo Python* [49].

Python es un lenguaje de alto nivel de programación de código abierto, que permite desarrollar software sin límite, donde dispone fuentes incorporadas como diferentes tipos de librerías [49]. Estas librerías hacen más sencilla la creación de aplicaciones, y en este caso, BITalino Plux tiene su propia librería donde facilita la recogida de datos. También se necesitan las librerías correspondientes para la conexión Bluetooth, o realizar diferentes ecuaciones matemáticas o plotear gráficas. Por ello, se va a crear un entorno virtual en el sistema, donde se instalarán las dependencias de este proyecto sin tener la obligación de instalarlas en todo el sistema.

Cabe destacar que otro de los motivos de haber escogido el IDE de Python es que gracias a la gran comunidad que genera, se pueden encontrar ejemplos de otros proyectos.

En el ANEXO II., se explica paso a paso cómo se realiza la instalación de la librería BITalino, y de todos los componentes que hacen falta para su correcto uso en un entorno virtual.

Tras la instalación, con la librería del BITalino viene un simple ejemplo para una primera toma de adquisición de datos, y poder entender lo que recoge y cómo se guarda. Por eso, a continuación, se explica este ejemplo, con el objetivo de analizar cómo trabaja. Es importante que se entienda como hace tanto la conexión con el dispositivo, como la lectura de datos.

Lo primero de todo código Python es importar las librerías necesarias, y luego definir las variables que se irán usando. Se debe definir la dirección MAC del dispositivo al que se quiere conectar. El threshold que se define es para la batería, es decir, si el porcentaje de batería es menor al 30%, el dispositivo no se conectará, como luego se verá en el código.

En este ejemplo, se recogerán un número de 10 muestras durante 5 segundos, con una frecuencia de muestreo de 1000Hz. '*acqChannels'*, son los canales analógicos que se van a recoger, siendo en este caso los 6 disponibles. El BITalino también dispone de canales digitales, como se ha mencionado en el apartado de la anatomía del dispositivo.

```
import time

from bitalino import BITalino

# The macAddress variable on Windows can be "XX:XX:XX:XX:XX:XX" or "COMX"
# while on Mac OS can be "/dev/tty.BITalino-XX-XX-DevB" for devices ending
# with the last 4 digits of the MAC address
# or "/dev/tty.BITalino-DevB" for the remaining
macAddress = "00:00:00:00:00:00"

# This example will collect data for 5 sec.
running_time = 5

batteryThreshold = 30
acqChannels = [0, 1, 2, 3, 4, 5]
samplingRate = 1000
nSamples = 10
digitalOutput_on = [1, 1]
digitalOutput_off = [0, 0]
```

Entrando en la librería 'bitalino', se accede al código donde se definen todos los comandos a ejecutar, tanto para acceder al dispositivo, como para cerrar la conexión o leer los datos. Ese apartado es importante dado que da la información acerca de cómo se accede o que datos de acceso necesita.

En las siguientes líneas se realiza la conexión, asignando a *device* el dispositivo BITalino con su correspondiente dirección MAC, y ejecutando el .start es cuando comienza a recoger datos.

```
# Connect to BITalino
device = BITalino(macAddress)

# Set battery threshold
device.battery(batteryThreshold)

# Read BITalino version
print(device.version())

# Start Acquisition
device.start(samplingRate, acqChannels)
```

Una vez establecida la conexión, como se ha mencionado anteriormente, como se desea que se recojan datos durante 5 segundos, se crea un bucle *while* para la lectura de datos.

```
start = time.time()
end = time.time()
while (end - start) < running_time:
    # Read samples
    print(device.read(nSamples))
    end = time.time()
```

Durante estos cinco segundos que dura el bucle, se van a mostrar los resultados, y después con las siguientes líneas se activa el *trigger* durante otros 5 segundos, y se cerrará el acceso al dispositivo.

```
# Turn BITalino led and buzzer on
device.trigger(digitalOutput_on)

# Script sleeps for n seconds
time.sleep(running_time)

# Turn BITalino led and buzzer off
device.trigger(digitalOutput_off)

# Stop acquisition
device.stop()

# Close connection
device.close()
```

Una pequeña parte del resultado que se obtiene se puede ver en la siguiente imagen, donde en este caso el BITalino no tiene ningún módulo conectado a las salidas analógicas.

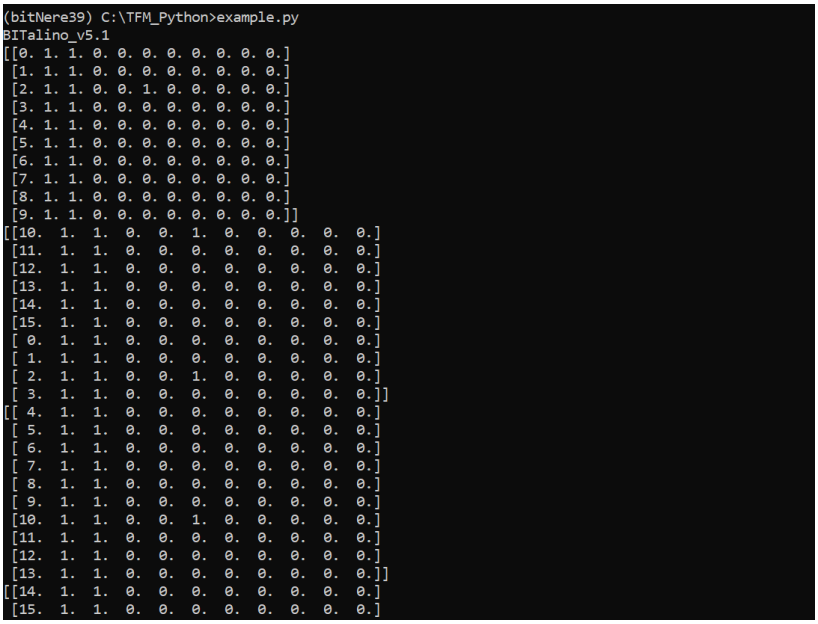


***Figura 26**. Lectura del programa example.py del BITalino.*

Para entender esto que se lee, es importante acceder a la librería que se está utilizando, y ver cómo se define '*.read*', como se ha mencionado anteriormente.

Cuando el código principal llama a la lectura, tan solo se deben definir el número de muestras que se desea, y si no se define nada, el programa tiene como valor predefinido tomar 100 muestras. Además, el código explica el tipo de dato que debe ser, y también aconseja el número a asignar, dado que recoger 1 muestra puede ser computacionalmente caro.

Una vez explicado el número de muestras que se recogen, explica la matriz de datos que se crea con la lectura. Como menciona, la primera columna se refiere al número de secuencia en el que se encuentra, y las siguientes 4 columnas, las cuales no se pueden eliminar, representan los canales digitales del dispositivo. Una vez en la sexta columna, tenemos los datos que deseamos, siendo los valores muestreados del canal. Aquí se pueden escoger los canales a añadir en la matriz, dependiendo de los que se quieran leer. Cabe destacar que estos valores luego habrá que transformarlos con su correspondiente función de transferencia dependiendo de la señal que se esté leyendo.

```
def read(self, nSamples=100):
    """
    :param nSamples: number of samples to acquire
    :type nSamples: int
    :returns: array with the acquired data
    :raises Exception: device not in acquisition (in IDLE)
    :raises Exception: lost communication with the device when data is corrupted

    Acquires `nSamples` from BITalino. Reading samples from BITalino implies the use of the method :meth:`receive`.

    Requiring a low number of samples (e.g. ``nSamples = 1``) may be computationally expensive;
    it is recommended to acquire batches of samples (e.g. ``nSamples = 100``).

    The data acquired is organized in a matrix whose lines correspond to samples and the columns are as follows:
        * Sequence Number
        * 4 Digital Channels (always present)
        * 1-6 Analog Channels (as defined in the :meth:`start` method)
    Example matrix for ``analogChannels = [0, 1, 3]`` used in :meth:`start` method:

    ==================  ========= ========= ========= ========= ======== ======== ========
    Sequence Number*    Digital 0 Digital 1 Digital 2 Digital 3 Analog 0 Analog 1 Analog 3
    ==================  ========= ========= ========= ========= ======== ======== ========
    0
    1
    (...)
    15
```

Por lo que, si se vuelve a observar los resultados de la Figura 26, se entiende mejor que la primera columna es el número de la secuencia que se está leyendo, separándose en matrices de 10 filas, por 11 columnas, ya que se han definido los seis canales analógicos (más los cinco que son obligatorios). Los valores analógicos dan cero dado que no se ha leído nada en el momento de la ejecución.

7. DISEÑO DE LA SOLUCIÓN PROPUESTA

Con el objetivo de conseguir un dispositivo capaz de monitorizar tanto las señales fisiológicas y éste esté validado con uno de referencia, se describe la solución propuesta en el siguiente apartado.

En primer lugar, se presenta la aplicación software propia creada para la recogida de señales. Segundo, se expondrá el protocolo experimental que se va a seguir para la validación del tándem del BITalino y la aplicación propia creada, utilizando un dispositivo considerado de referencia en investigación, recogiendo las mismas señales y usándolas de referencia en una comparación justa posterior. Tercero, se presentará la metodología seguida para la validación mediante Matlab. Por último, se explicarán las tres diferentes pruebas que se han realizado con sus resultados.

7.1. Diseño del software propio para la recogida de señales

El objetivo principal del proyecto ha sido construir una aplicación que monitorice online las tres señales fisiológicas (en este caso), para poder analizarlas y visualizarlas, pudiendo crear diferentes puntos para señalizar cambios bruscos en las emociones. Además, está la opción de poder acceder a esos datos generados de una forma práctica.

Como se ha mencionado anteriormente, el grupo de investigación contaba con una aplicación ya diseñada para la adquisición de datos mediante el BITalino, pero se propuso la mejora de esta. El desarrollo de la aplicación está basado en la API de Python creada para la tarjeta BITalino. Además, para el diseño de la interfaz, diseño de ventanas y la visualización de señales online se utilizó Qt [50].

Qt es un conjunto de herramientas de widgets gratuito y de código abierto para crear aplicaciones GUI multiplataforma, lo que permite que las aplicaciones se dirijan a múltiples plataformas de Windows, macOS, Linux y Android con un único código base. Además, cuenta con soporte integrado para multimedia, bases de datos, gráficos vectoriales e interfaces. PyQt5 es una versión en Python del Qt desarrollado por Riverbank [51].

Por ello, en este apartado se irá comentando paso a paso, los aspectos más relevantes de la aplicación y los cambios realizados para llegar al diseño final.

7.1.1. Diseño de la aplicación

La aplicación se ha diseñado para que sea intuitiva para el usuario, y no de problemas a la hora de su utilización. También es verdad, que luego se han ido haciendo modificaciones, para intentar crear una aplicación además de segura, con un coste de cómputo menor de forma que al recoger la señal se pierda la menor información posible.

La aplicación está desarrollada mediante programación orientada a objetos con pyQt5. Los objetos definidos son cuatro, aunque de esas una es la principal donde se instancian las demás. La clase u objeto principal es *MainWindow*, y las que se instancian en ella serán *Ventana_Ploteo, Ventana_Ploteo_Online* y *Ventana_config.* Estas cuatro clases se observan en el esquema de la Figura 27, donde dentro de ellas presentan las funciones que la completan.

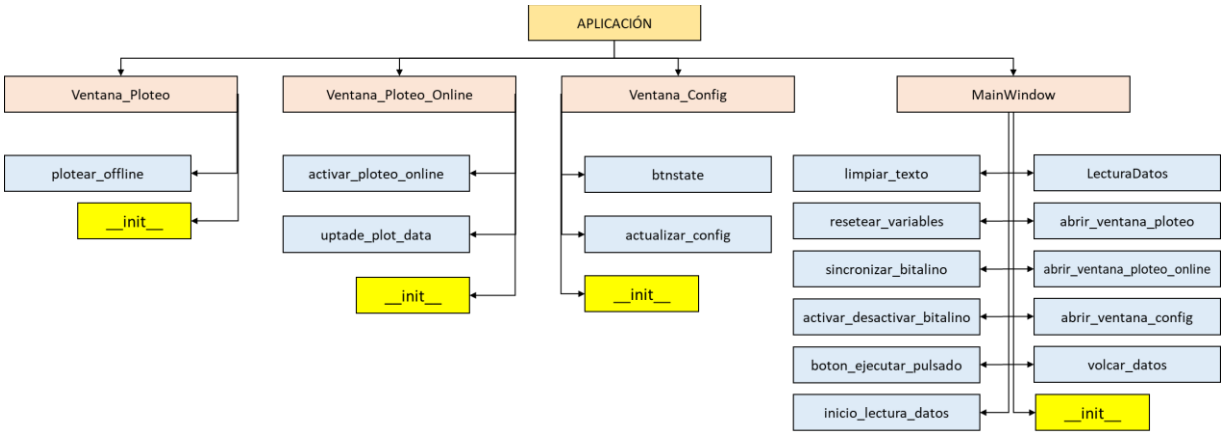


***Figura 27**. Esquema de las clases y funciones que contiene la aplicación.*

La ventana *MainWindow* es la principal nada más ejecutar la aplicación (véase Figura 28), y permite acceder a las demás ventanas. El objetivo es dar al usuario facilidad para conectarse al correspondiente dispositivo, y poder comenzar así con el proceso de la realización de la prueba. Es por ello que da accesibilidad a todo respecto a la recogida de datos, como se ha mencionado en el esquema general del apartado anterior.

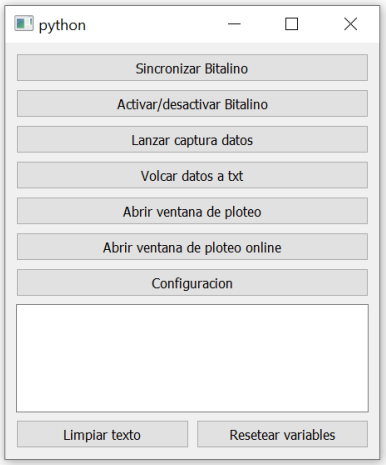


***Figura 28.** Pantalla principal de la aplicación.*

7.1.1.1. Diseño de la Ventana_Config

Dentro de este objeto, una de las instancias más relevante es la de configuración *Ventana_Config*. La configuración es el bloque principal, ya que da el acceso a diferentes parámetros del dispositivo, como actualizar la dirección MAC a la que conectarse o escoger los canales que se desean leer. En la Figura 29 se presenta el esquema general del funcionamiento de este objeto. Nada más ejecutar la aplicación, se debe pulsar el botón de configuración de la pantalla principal, es en ese momento donde se accederá a la configuración del dispositivo HW, donde se podrá actualizar la dirección MAC además de escoger los canales que se desean recoger.

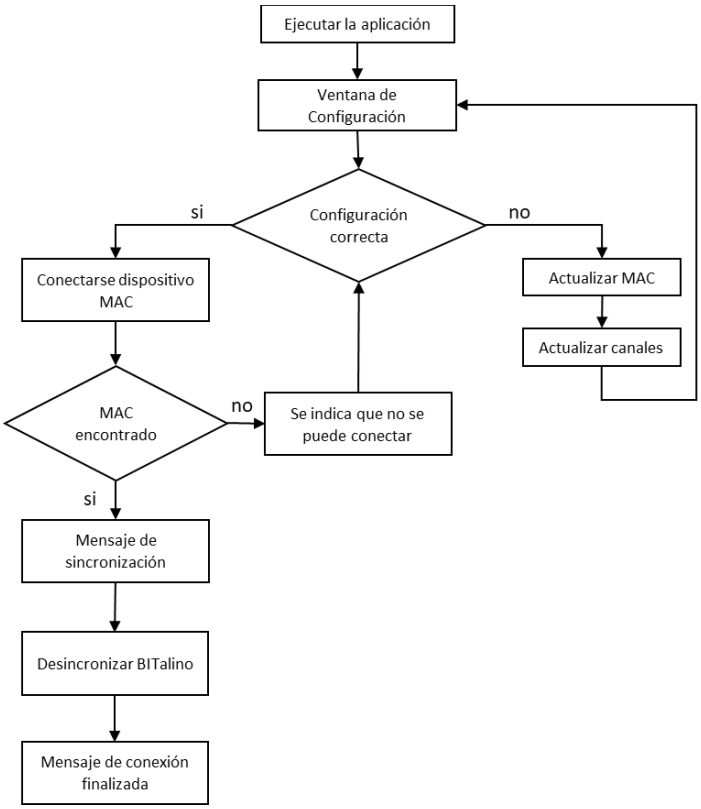


Figura 29. *Esquema de la ventana de configuración.*

La Figura 30 muestra la ventana de configuración que se ha diseñado. En ella, como se ha mencionado, escribe la dirección MAC del dispositivo HW al que se va a conectar la aplicación. Además, se puede escoger los canales que se quieren adquirir en la prueba a realizar. Este último es uno de los cambios que se ha realizado para el diseño de la nueva aplicación.

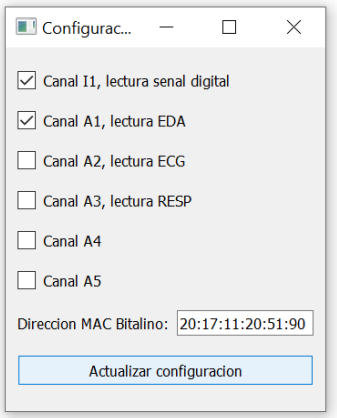


Figura 30. *Ventana de configuración de la aplicación.*

Cuando se empezó a trabajar con la aplicación ya diseñada del Grupo de Investigación, se observó que a la hora de graficar las señales el tiempo de cómputo que se consumía era bastante más elevado a cuando no se estaba graficando. Uno de los motivos era que se recogían y mostraban todos los canales del BITalino, en vez de capturar tan solo la de los canales que se deseaban.

Por eso mismo, se diseñó un algoritmo donde de primeras tan solo tuviera tres canales analógicos más el canal digital adquiriendo señales, pero diera opción al usuario de, o añadir más canales o quitar los canales seleccionados.

Antes de esto, cabe destacar que a la aplicación también se le sumó la entrada de la señal digital, dado que hasta ahora era un dato que no se utilizaba. Por lo que, el array de datos útiles con el que se trabajaba y se guardaban los datos de los 6 canales más la del tiempo, pasó a ser 6 canales, más el canal digital más el canal del tiempo, definido en el siguiente fragmento de código. Luego se presentará el formato con el que se guardan los datos para un posterior volcado a archivo de texto.

```
## VARIABLES GLOBALESS ###

NUMERO_CANALES = 6;
DATOS_UTILES = np.zeros((2,NUMERO_CANALES+2), dtype='float64')  # El +1 viene de la columna de tiempo.
                                                                # El +1 viene de la columna digital
```

Una vez definidos los canales deseados, se ha creado un nuevo array de valores booleanos. Este array estará preconfigurado siempre de la misma manera al ejecutar la aplicación, pero el usuario podrá cambiar el estado booleano seleccionando/deseleccionando el canal. La estructura inicial del array se observa en el esquema generado en la Figura 31, donde se observa también el esquema a seguir en el código para crear el algoritmo que solo guarde las señales deseadas.

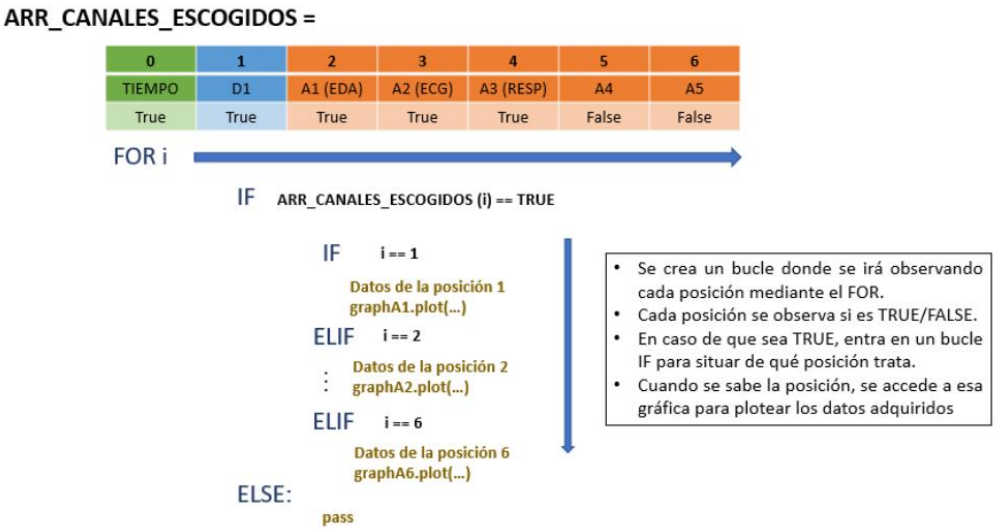


Figura 31. *Esquema de la selección de datos a adquirir.*

Este esquema general se puede resumir en el siguiente fragmento de código. Se observa la variable global *ARR_CANALES_ESCOGIDOS* definida tal y como se ha mencionado en el esquema. En el siguiente fragmento se muestra uno de los casos donde se usa en el código, donde en este caso, lo que se desea es plotear los datos escogidos por el usuario.

```
# ARR_CANALES_ESCOGIDOS = [Tiempo,'D1' 'EDA', 'ECG', 'RESP', A4, A5]
ARR_CANALES_ESCOGIDOS = [True, True, True, True, True, False, False]
```

```
        datos = DATOS_UTILES
        self.eje_x = datos[:,0]

        for i in range(1, len(ARR_CANALES_ESCOGIDOS)):
            if ARR_CANALES_ESCOGIDOS[i] == True:
                if i == 1: #estamos en el grafico 1
                    self.eje_y = datos[:,1]
                    self.graphA1.plot(self.eje_x, self.eje_y)
                elif i == 2:
                    self.eje_y = datos[:,2]
                    self.graphA2.plot(self.eje_x, self.eje_y)
                elif i == 3:
                    self.eje_y = datos[:,3]
                    self.graphA3.plot(self.eje_x, self.eje_y)
                elif i == 4:
                    self.eje_y = datos[:,4]
                    self.graphA4.plot(self.eje_x, self.eje_y)
                elif i == 5:
                    self.eje_y = datos[:,5]
                    self.graphA5.plot(self.eje_x, self.eje_y)
                elif i == 6:
                    self.eje_y = datos[:,6]
                    self.graphA6.plot(self.eje_x, self.eje_y)
            else:
                pass
```

Por otra parte, el estado del botón TRUE/FALSE sigue la siguiente estructura. Resumidamente, en el siguiente fragmento se presenta la casilla diseñada para la entrada digital, donde coincidiendo con el array anteriormente explicado, se fija en TRUE. El segundo paso es agregar acción a las cajas, por lo que se crea una función donde se verifique el estado del botón. Finalmente, la función lo que hace es observar si el botón está en TRUE o no, y cambiar la configuración del array según el caso.

```
# casilla D1 activado
self.layoutconfig_fila_1 = QHBoxLayout() # Layout para primera fila de objetos.
self.tick_A1 = QCheckBox('Canal I1, lectura senal digital')
self.tick_A1.setChecked(True) #en principio tenemos todos activados
self.layoutconfig_fila_1.addWidget(self.tick_A1)
```

```
 # agregar accion a los checkbox
 self.tick_A1.stateChanged.connect(lambda:self.btnstate(self.tick_A1)) # Digital
```

```
def btnstate(self, A):
  global ARR_CANALES_ESCOGIDOS
  if A.text() == "Canal I1, lectura senal digital":
     if A.isChecked() == True:
        print (A.text()+" is selected")
        ARR_CANALES_ESCOGIDOS[1] = True
     else:
        print (A.text()+" is deselected")
        ARR_CANALES_ESCOGIDOS[1] = False
```

7.1.1.2. Diseño del bloque de captura de datos

Por otra parte, en el segundo bloque principal se agrupan las ventanas relacionadas con la sincronización del dispositivo, activación/desactivación, la captura de datos o el volcar los datos obtenidos a un archivo de texto.

Una vez la aplicación está configurada, el usuario puede realizar la sincronización y la activación del dispositivo. Para facilitar el uso, en la caja de texto se va informando al usuario de cada momento, por ejemplo, si la conexión se ha realizado o no, o si la se ha lanzado la captura de datos. Se muestra el esquema general en la Figura 33.

El cambio en el código que se ha realizado en este bloque ha sido en la lectura de datos. En el apartado 6.2.3 se explica el ejemplo de cómo trabaja la librería del bitalino, y esto es importante para la lectura de los datos, ya que solo se desea acceder a los datos que son útiles en la investigación. El bitalino cuando recoge las señales, obligatoriamente recoge el número de la secuencia, y las cuatro entradas digitales. Las entradas analógicas las elige el usuario. El ejemplo de la matriz se observa en la Figura 32, donde se observa que de toda la data que recoge el BITalino, tan solo el programa se queda con los canales que van a ser útiles.

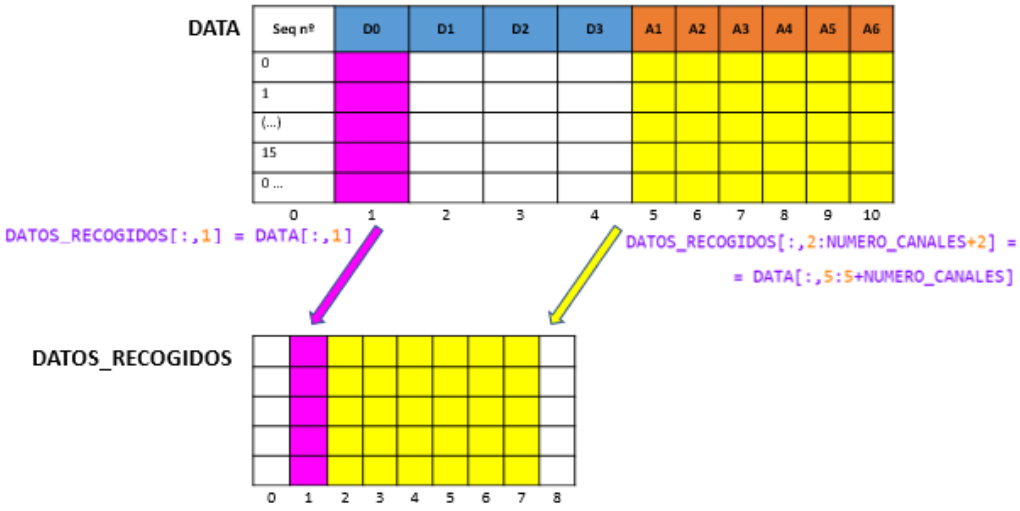


Figura 32. *Transferencia de todos los datos leídos a datos recogidos.*

Siguiendo la tabla, y sabiendo que la función de la aplicación es que recoja los seis canales analógicos más uno digital, el fragmento del código completo es el siguiente:

```
def LecturaDatos(self, DEVICE, DATOS_RECOGIDOS):
    # Start acquisition and reading 300 samples:
    global PERMISO_RELECTURA

    PERMISO_RELECTURA = False

    DATA = DEVICE.read(DATA_CHUNK_SIZE)
    DATOS_RECOGIDOS[:,2:NUMERO_CANALES+2] = DATA[:,5:5+NUMERO_CANALES]
    DATOS_RECOGIDOS[:,1] = DATA[:,1]
```

Una vez ordenados los datos, solo se tendrá que tener en cuenta las nuevas posiciones y cambiarlas durante el código. A la hora de volcar los datos a un archivo de texto, habrá que tener en cuenta que la segunda columna pertenece a la entrada digital, y a partir de la tercera a las entradas analógicas.

Mantiene de igual forma la opción de resetear los datos adquiridos hasta el momento o la limpieza de la caja de texto.

7.1.1.3. Diseño del bloque de visualización de datos

Finalmente, el *MainWindow* también da acceso a las ventanas de visualización de datos. En este caso, el usuario dispone de dos opciones, la visualización online, es decir, poder observar los canales que se están recogiendo en el mismo momento, o tan solo visualizar las señales adquiridas hasta ese momento. En la Figura 33 se muestra un esquema general donde se explica el bloque de la captura de datos y la visualización de ellos una vez el dispositivo HW está sincronizado.

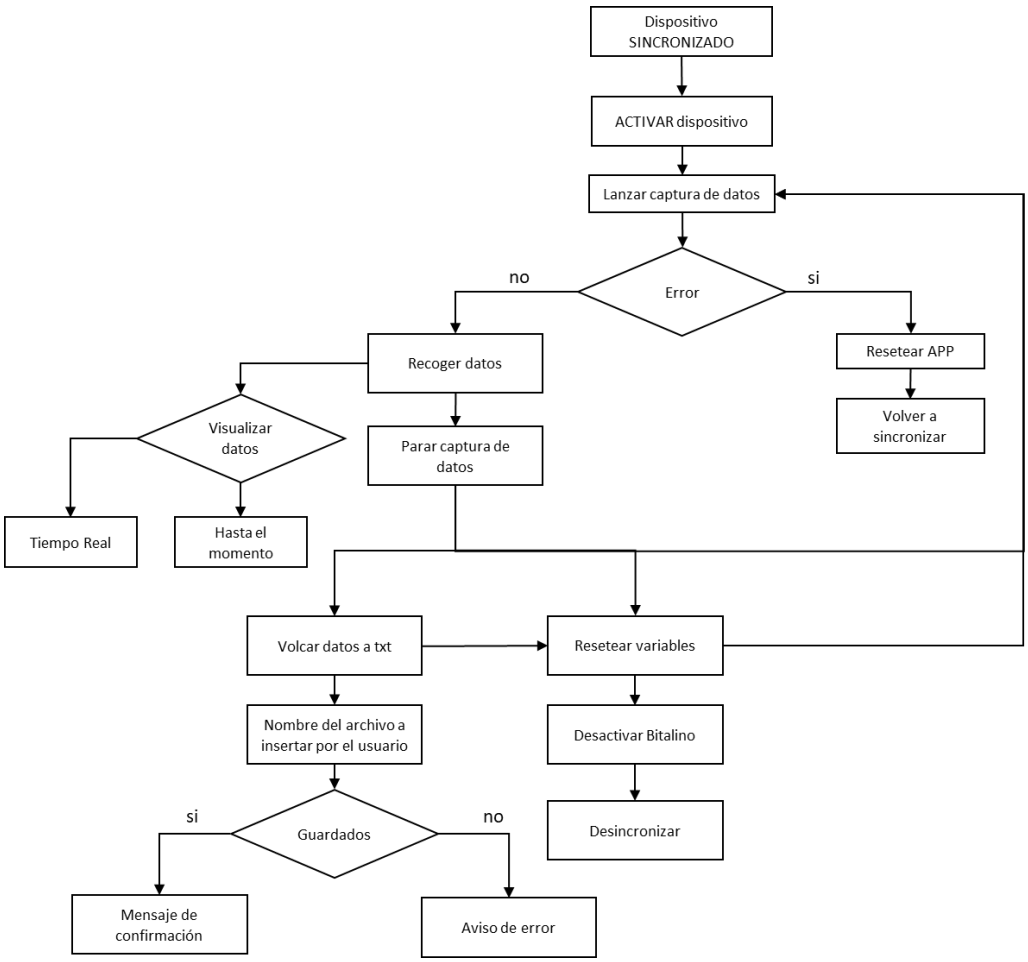
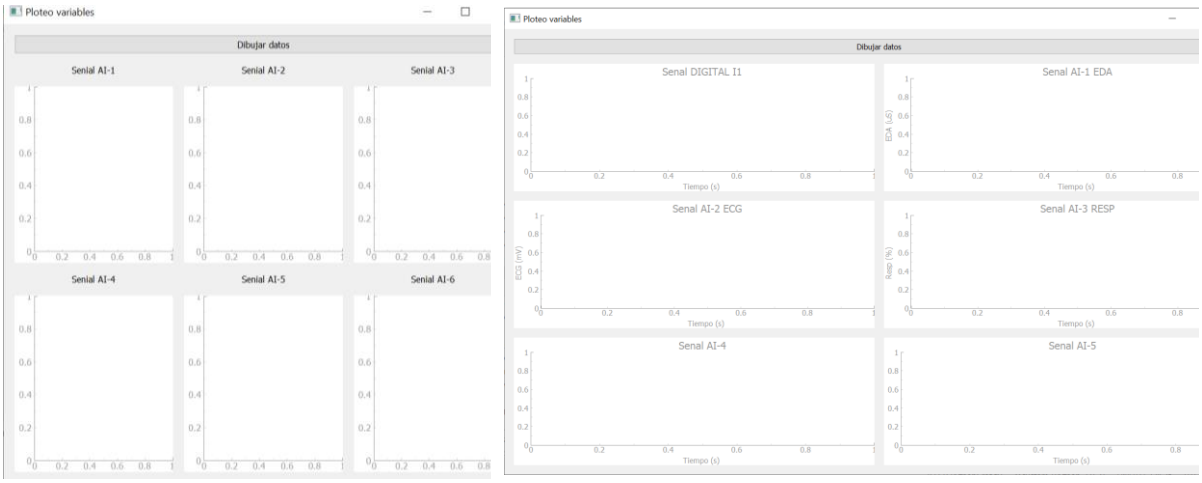


Figura 33. *Esquema de la parte principal de la aplicación.*

En este bloque solo se han realizado cambios en la estructura de las gráficas. En la Figura 34 se observan estos cambios, donde se puede ver que la disposición de las gráficas ha cambiado teniendo las nuevas más capacidad horizontal para presentar los datos. Además, se ha insertado la señal digital como primera gráfica. El código de este fragmento aparece en el ANEXO III. junto con el código completo.



a b

Figura 34. *Ventana de visualización antes (a) y con el cambio realizado (b).*

7.2. Protocolo experimental

Tras analizar la aplicación propia creada para la adquisición de datos, ésta se debe validar junto con el dispositivo. Para ello, es importante crear un protocolo de actuación frente a cada ensayo, creando un protocolo de actuación frente a cada ensayo y diseñando la prueba de estrés. También se presentará el esquema general que se va a seguir en las posteriores pruebas. Cabe destacar que, aunque las pruebas se han realizado principalmente a participantes del Grupo de Investigación, han sido informados del proceso a seguir y se ha pedido su consentimiento. Se adjunta dicho consentimiento en el ANEXO IV.

7.2.1. Diseño de experimento: Prueba cálculo matemático

A continuación, se propone la prueba para inducir estrés al voluntario. Además de la prueba en sí, será muy importante generar un ambiente de estrés alrededor de la misma y realizar una evaluación emocional de los sentimientos de la persona mientras hace el experimento.

En esta prueba el experimentador va proponiendo secuencialmente cálculos matemáticos con una dificultad creciente. Según se va respondiendo correctamente se le propone otro cálculo con una mayor dificultad. El objetivo es responder el máximo número de cálculos en un tiempo limitado.

Antes de la considerada etapa del cálculo matemático, en el diseño del experimento se decidió llevar al sujeto a un estado emocional basal a través de un video de naturaleza con música relajante de dos minutos de duración. Así, una vez transcurridos los dos minutos y viendo que el sujeto se encuentra relajado, se procederá con el cálculo matemático. Es importante que nada más terminar el video comience la prueba de estrés, así se podrá apreciar el cambio emocional, pasando de relajación directamente al estrés. En la Figura 35 se observa el diagrama de tiempo de lo que será el experimento.

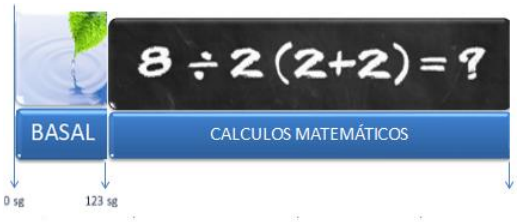


Figura 35. *Diagrama de tiempo del experimento.*

Para el cálculo matemático se seguirá la Tabla 6, con sus respectivas anotaciones, siendo para todos los sujetos la misma prueba de estrés.

Tabla 6. *Diseño del cálculo matemático.*

1+3 = 4	**+2** = 6	**-1** = 5	**x2** = 10	**+8** = 18	**+2** = 20	**/10** = 2	**+14** = 16
En las primeras dar tiempo más que suficiente y asentir con la cara afirmando que va bien o diciendo "bien"							
-5 = 11	**-1** =10	**+7** = 17	**+22** = 39	**-10** = 29	**-13** = 16	**+12** = 28	**x3** = 84
dar tiempo suficiente pero un poco más rápido						a) *!!	
-7 = 77	**+34** = 111	**-8** = 103	**+21** = 124	**/2** =62	**+79** = 141	**+55** = 196	**-1** = 195
/3 =65	**+3**=68	***7**= 476		**/2** = 238	**+12** = 250	**250** = 500	**-4** = 496
		b) 466	**+10** = 476	seguimos normal			
-38 = 458	**+16** = 474	**-202** = 272	***6** = 1632	**/12** =136	**+425** = 561	***2** = 1122	**+573** = 1695
	AQUI TERMINARIA	Por si esta calculín o no está muy estresado o ha pasado poco tiempo					
***3** = 5085	**/5** = 1017						

Si en alguna falla, poner cara y dejarle intentar otra vez, si vuelve a fallar, ayudarle con el resultado de partida y repetir la operación. Ejemplo 5x2=__? En algunas otras, no decirles el número de partida, solo…. x2=?

Cuando llegue a la operación a) aunque acierte poner cara como que ha fallado y decir… 28x3=?, esperar a que conteste y decirle a si… estaba bien.

Cuando lleguen a la operación b) decirle que no, aunque acierte, si acierta decirle... "no 466" si falla darle otra oportunidad y decirle "68*7=?".

7.2.2. Protocolo de experimentación

A continuación, se detallan las fases se deben seguir para cumplir el protocolo de experimentación.

1. Preparación.

Disponer en una mesa los dispositivos, electrodos, sincronizador, hoja de consentimiento, documento del cálculo matemático y papel para limpiar los restos de gel si hubiera.

2. Apertura del archivo.

Se abre el archivo de configuración de los dispositivos:

- **NOMBRE FICHERO CONFIGURACIÓN:**
- **Configuracion_Dispositivo_EXPERIMENTO.EXT**
  EXPERIMENTO: BioBit
  EXT: extensión del dispositivo de adquisición de datos

y se guarda como:

- **NOMBRE FICHERO DATOS: EXPERIMENTO_AAAAMMDD_ID-SUJETO.EXT**
  EXPERIMENTO: BioBit001
  SESION: AAAAMMDD: 20210526
  ID-SUJETO = NNN: asignar un identificador al sujeto (001)
  EXT: extensión del dispositivo de adquisición de datos

3. Recepción del sujeto.

Se le cuenta al sujeto que va a realizar un experimento donde se quiere comprobar si las señales recogidas por el dispositivo BITalino son las mismas que por el dispositivo BioPac. Se le explica también, muy por encima, las pruebas a realizar. Y finalmente, se le da la hoja de consentimiento para que la firme.

4. Colocación de electrodos.

Se colocan los electrodos en el sujeto, dependiendo del número de prueba que estemos, corresponderá a una ubicación u otra.

5. Comprobación de la recogida de señales.

Una vez colocados los electrodos, antes de comenzar con la prueba, se observará que las señales se registran correctamente. Se pedirá al sujeto que sentado se mueva hacia delante y los lados para comprobar que todo va bien.

6. Inicio de la prueba

Iniciar la prueba y pulsar el botón de la placa de sincronización. Así aparecerá un cambio en la señal digital del registro.

7. Visualización del video de relajación.

Proyectar en una pantalla el video ya preparado para que el sujeto se relaje.

8. Cálculo matemático.

Una vez finalizado el vídeo, sin perder tiempo se procederá con el cálculo matemático como se ha explicado en el apartado anterior.

9. Parar el experimento.

Cuando llegue al final de la cadena de cálculo, o se vea que ya el sujeto tiene muchas dificultades, se finalizará la recogida de datos.

10. Guardar el archivo de datos.
Tras parar la aplicación, se guardarán los datos adquiridos con el nombre correspondiente, siguiendo el paso 2: **EXPERIMENTO_AAAAMMDD_ID-SUJETO.EXT**

11. Entrevista.
Se le pregunta al sujeto: ¿Qué tal te has sentido?, ¿se te ha movido algún electrodo en particular? En este caso, al no ser importante la emoción del sujeto, no se hace una entrevista profunda.

12. Añadir información del sujeto al documento.
La información relevante respecto al experimento o el sujeto, se escribirá en un documento, que se guardará con el siguiente formato: **Informacion_Sesion_EXPERIMENTO_AAAAMMDD.docx**

13. Guardar el archivo .acq en .txt dentro de la carpeta **0.Registro_Total_EXPERIMENTO.txt**

14. Fin de la prueba.
Tras finalizar la prueba, se quitarán los electrodos y desecharán.

7.2.3. Diseño experimental para la recogida de señales

Una de las partes importantes del proyecto es la validación del dispositivo de bajo coste BITalino respecto a un dispositivo de referencia, en este caso, será el BIOPAC MP 36 System (Biopac Systems Inc., USA) la unidad con la que se trabajará. Una vez validado el dispositivo, se realizará el diseño del dispositivo tan solo con el BITalino, realizando diferentes pruebas con las ubicaciones de los electrodos, para conseguir un diseño final vestible y portable.

En la Figura 36 se muestra el esquema general del diseño, el cual se divide en tres fases. La primera, y muy importante, será la sincronización de ambos dispositivos de recogida de señales, para que sea una recogida simultánea. Esta sincronización se realiza mediante un módulo que se ha creado, la cual se analizará más a fondo en el siguiente apartado. La segunda fase es la recogida de señales y el envío de datos a la interfaz. Ambas interfaces tienen la opción de ver online las señales que se están adquiriendo. La última fase trata de analizar los datos que se han adquirido. Una vez terminada la prueba, se guardarán los datos en formato '.txt', y se analizarán con el software de Matlab.

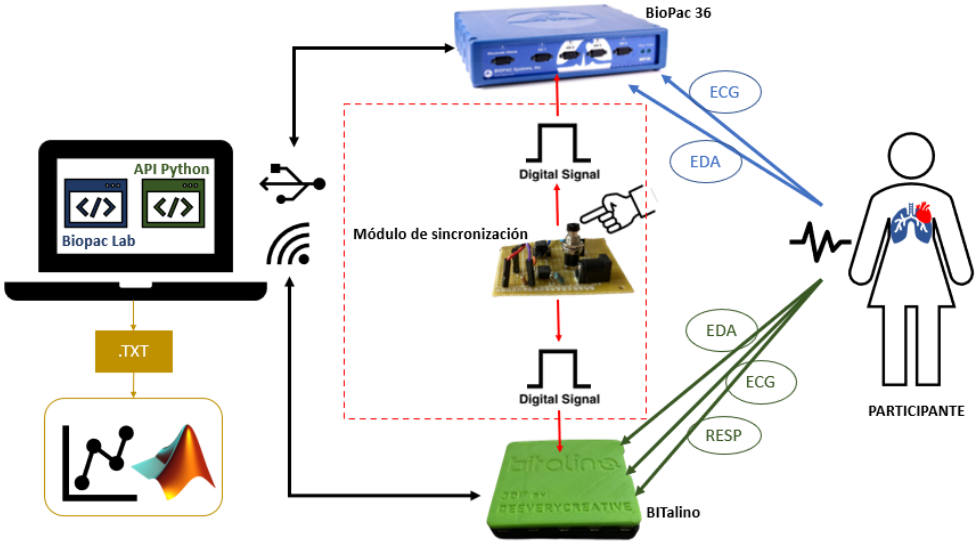


Figura 36. *Esquema general para la validación del BITalino.*

Como se ha mencionado durante el trabajo, el dispositivo de referencia será el hardware de adquisición de señales Biopac MP36. Aunque tiene la capacidad de recoger hasta cuatro señales, solo se van a recoger las señales ECG y EDA en los canales 1 y 2 respectivamente, y haciendo uso de los sensores de Biopac.

Por otra parte, como se ha mencionado anteriormente, la placa que se va a emplear en este proyecto es BITalino (r)evolution Plugged. En este caso se recogerán las señales ECG y EDA, aunque en alguna prueba también se recoja la respiración.

7.2.3.1. Módulo de sincronización

Parte del proyecto consiste en validar que el dispositivo de bajo coste recoge las señales de forma adecuada respecto al dispositivo de referencia BIOPAC. Para realizar el posterior análisis de los datos, es importante que las señales estén sincronizadas y se guarden estos datos desde el mismo punto de inicio.

Aprovechando que ambos dispositivos también procesan señales digitales, se ha creado un módulo donde gracias al pulsador es capaz de enviar un franco de subida o de bajada a ambos dispositivos, para luego en su posterior análisis tener esta señal como referencia en el tiempo. Esta placa se puede observar en la Figura 37, donde además del pulsador, se observan los dos cables que conectan tanto con el BITalino (blanco) y el Biopac (azul y marrón).

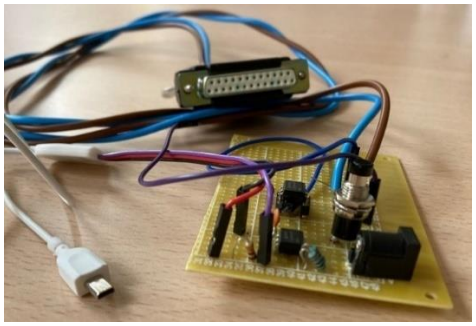


Figura 37*. Módulo de sincronización para el BioPac y BITalino.*

El módulo sigue el siguiente esquema eléctrico (véase **Figura 38**).

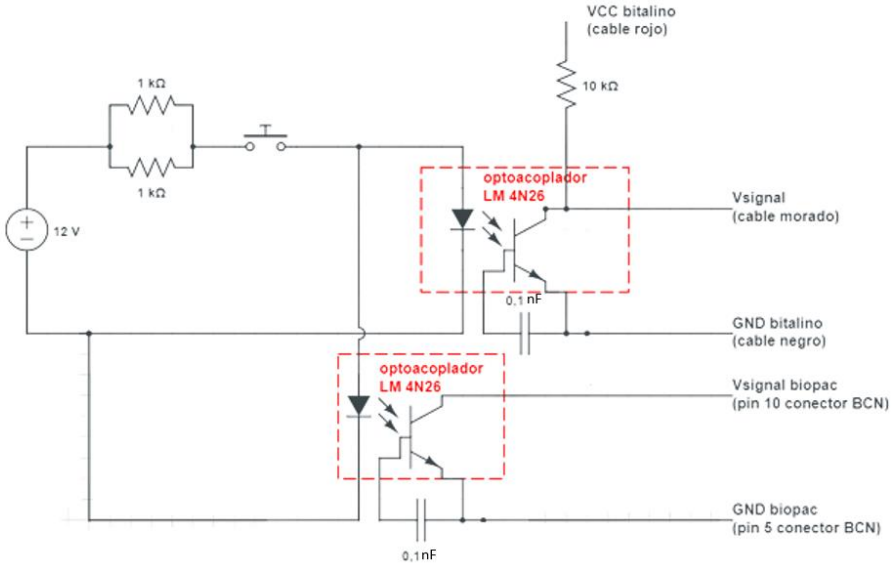


Figura 38. *Esquema eléctrico del módulo de sincronización.*

Para poder incluir ambos dispositivos HW en la placa, ha sido necesario un previo análisis de la hoja de características del Biopac [52]. El conector necesario para el Biopac es un DSUB 25 puertos hembra (Figura 39), y como se especifica, los puntos a los que se ha conectado el optoacoplador del módulo son el 10 (positivo) y el 5 (negativo).

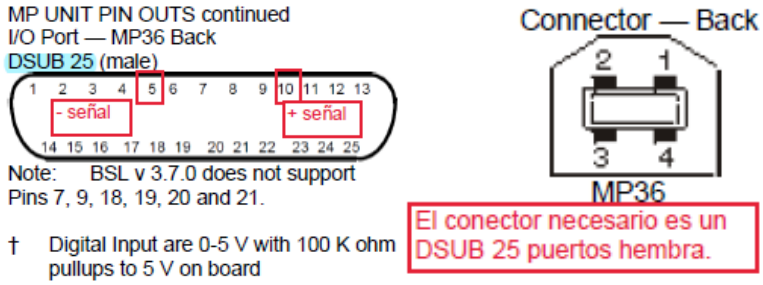


Figura 39. *Conector DSUB 25 para la conexión Biopac-módulo de sincronización.*

7.3. Metodología para la validación del dispositivo BITalino utilizando Matlab

Tal y como se ha mencionado, el proceso de validación del HW BITalino y del software propio pasa por comparar y obtener un alto grado de similitud respecto a las señales recogidas con un dispositivo HW de referencia. Dado que los dos formatos de los datos recogidos son diferentes, se ha decidido hacer el proceso de comparación a través de Matlab. Esta decisión se ha tomado por su facilidad de uso a la hora de correlacionar datos de fuentes diferentes. Una vez terminada la prueba, se obtendrán dos archivos con formato de archivo de texto (txt), donde tan solo habrá que observar a qué señal corresponde cada lectura o columna, dependiendo de cómo estén configurados los canales, para así poder coger los datos adecuados en el código.

Una vez se carga el archivo a Matlab, se coge la columna deseada y se grafica en el tiempo, así se obtendrá la señal. Con esos datos también se podrá hacer una correlación vs la misma señal recogida con el otro dispositivo.

En el caso de la ECG, se tendrá que realizar un análisis más profundo para sacar la correlación entre las dos señales recogidas. Por lo que, se sacará la HRV calculando el intervalo entre dos picos R de la onda ECG, para luego sacar la frecuencia cardíaca (HR) de ella. Con esta nueva señal sí que se podrá analizar la correlación de las dos señales.

Como ejemplo de lo mencionado hasta ahora, la Figura 40 muestra el contenido de un archivo de una prueba recogida mediante el Biopac, donde el primer canal corresponde a la señal ECG, el segundo a la EDA y el tercero a la entrada digital de la señal de sincronización, la cual se pondrá a cero en el momento que se pulse el botón.

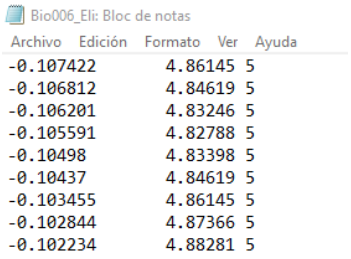


Figura 40. *Ejemplo de los datos obtenidos de una prueba con el BioPac.*

Sin embargo, la Figura 41, muestra los datos recogidos con BITalino para el mismo ensayo, y tal como se ve, este dispositivo recoge muchas más señales. Esto es debido a que la aplicación está diseñada para que guarde los datos de todos los canales. En este caso, el primero es el tiempo transcurrido, los siguientes tres, EDA, ECG y RESP respectivamente, y la quinta columna la entrada de sincronización, la cual se pondrá a cero cuando se pulse el botón.

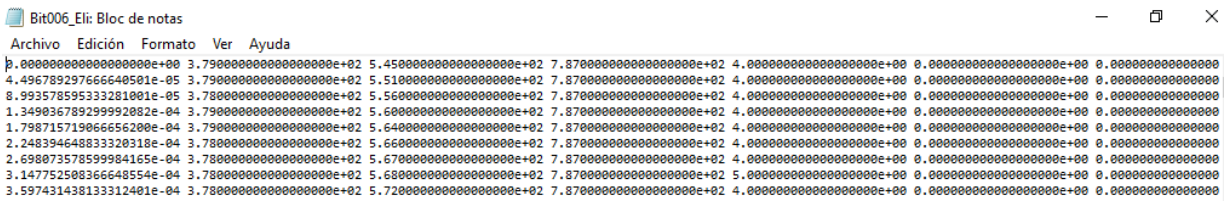


Figura 41. *Ejemplo de los datos obtenidos de una prueba con el BITalino.*

Una vez se entienden los datos que se obtienen de la captura de señales, el código para graficar las señales sería así.

```
%% Lectura de los ficheros y encontrar último cero
file_BITALINO = 'Bit006_Eli'; %nombre del fichero del BITalino
file_BIOPAC = 'Bio006_Eli'; %nombre del fichero del Biopac

% Abrir fichero .txt y cargar datos
var_bitalino = load([file_BITALINO, '.txt']);
var_biopac = load([file_BIOPAC, '.txt']);

% bitalino
% tiempo CH1    CH2   CH3   CH4
%        EDA    ECG   Resp  trigger
EDA_bit = var_bitalino(:,2);
ECG_bit = var_bitalino(:,3);
trigger_bit = var_bitalino(:,5);

for i=1:length(EDA_bit)
    if trigger_bit(i) == 0
```

```
        pos_bit = i;
    end
end

% Biopac
% CH1 CH2 CH3
% ECG EDA trigger
ECG_bio = var_biopac(:,1);
EDA_bio = var_biopac(:,2);
trigger_bio = var_biopac(:,3);

for i=1:length(EDA_bio)
    if trigger_bio(i) == 0
        pos_bio = i;
    end
end
```

Cabe destacar que los datos que recoge el BITalino hay que convertirlos a valores reales con sus respectivas funciones de transferencias mencionadas en el apartado 6.1.2.1. Eso es lo que se observa en el siguiente fragmento, cómo se guarda en una nueva variable los datos de cada señal empezando desde la última posición de sincronización.

```
%% datos conversión
% Pasar los datos de ADC a EDA (uS), ECG (mV), RESP (%)
n = 10; %los primeros 4 canales 10 bits de resolución, y los demás 6bits
VCC = 3.3;
Gecg = 1100; %ganancia del sensor

%% recortar los array de datos a partir de pos + 1
L_bio = length(EDA_bio)-pos_bio;
L_bit = length(EDA_bit)-pos_bit;

EDA_bit_new = zeros(L_bit, 1);
ECG_bit_new = zeros(L_bit, 1);
EDA_bio_new = zeros(L_bio, 1);
ECG_bio_new = zeros(L_bio, 1);

j = 1;
for i=(pos_bit+1):length(EDA_bit)
    EDA_bit_new(j) = (VCC*(EDA_bit(i)/2^n))/0.132;
    ECG_bit_new(j) = 1000*((ECG_bit(i)/2^n)*VCC)/Gecg;
    j = j+1;
end

j = 1;
for i=(pos_bio+1):length(EDA_bio)
    EDA_bio_new(j) = EDA_bio(i);
    ECG_bio_new(j) = ECG_bio(i);
    j = j+1;
end
```

Una vez aquí, sabiendo que las señales se recogen con una frecuencia de muestreo de 1000 Hz, habría que sacar el tiempo transcurrido durante la prueba para poder graficar la señal.

```
fs = 1000;   %frecuencia de muestreo (1000 Hz)
t = (0:L_bit-1)/fs; %sacamos el tiempo total del muestreo

figure
subplot(2,1,1); plot(t, EDA_bio_new);
title('EDA BioPac');
xlabel('tiempo (s)'); ylabel('EDA (uS)'); grid;

subplot(2,1,2); plot(t, EDA_bit_new);
```

```
title('EDA BITalino')
xlabel('tiempo (s)'); ylabel('EDA (uS)'); grid;
```

Y el resultado de este ejemplo se puede observar en la Figura 42, donde se puede apreciar que ambas señales siguen una misma tendencia. El participante en este caso, ha logrado relajarse mucho durante dos minutos, y cuando ha comenzado la segunda fase de la prueba, donde el participante es sometido a la prueba del cálculo matemático, se observa que se ha puesto nervioso.

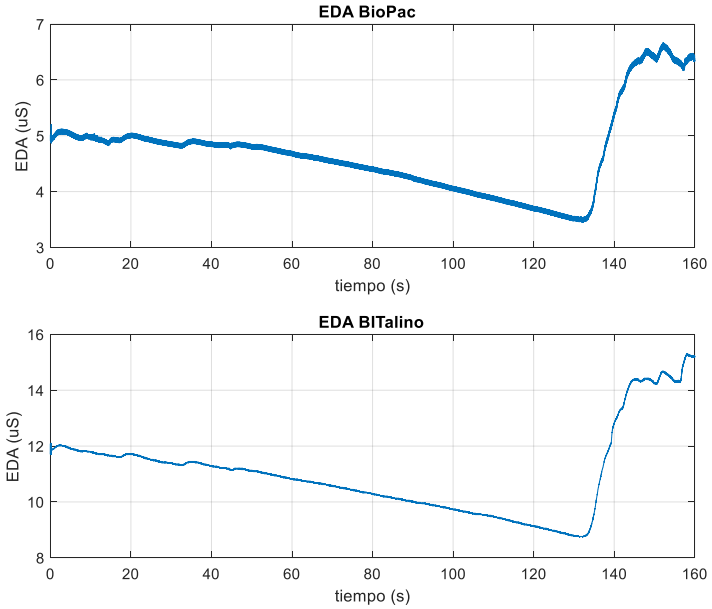


Figura 42. *Gráfica de la señal EDA del ejemplo.*

Si se grafica de la misma manera y en base al mismo tiempo la señal ECG, el resultado se observa en la gráfica de la Figura 43. Al contrario que en la señal EDA, intentar analizar esta señal sería mucho más complicado.

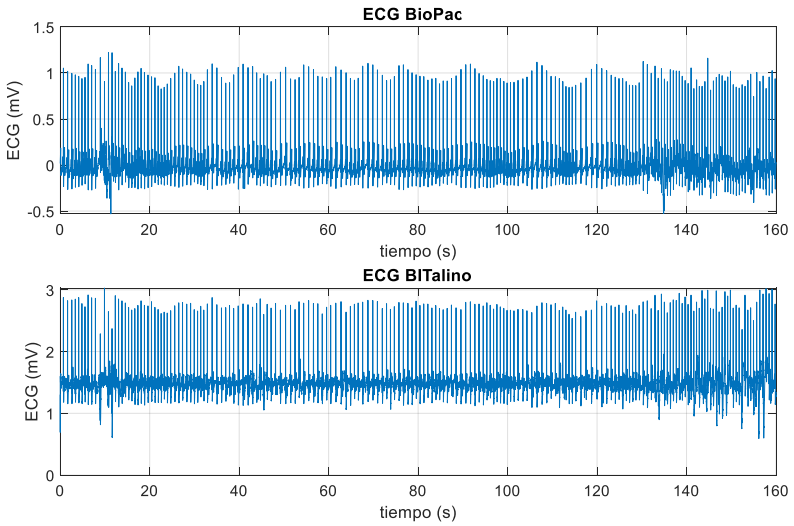


Figura 43. *Gráfica de la señal ECG del ejemplo.*

Si se reduce la escala del tiempo, se puede observar la señal con su complejo QRS en el intervalo temporal [50, 55] (véase la Figura 44), además de ver que ambas señales se parecen. Analizar así las señales no sería tarea fácil, es por eso que de esta señal se buscará el ritmo cardíaco para poder hacer una correlación con ella.

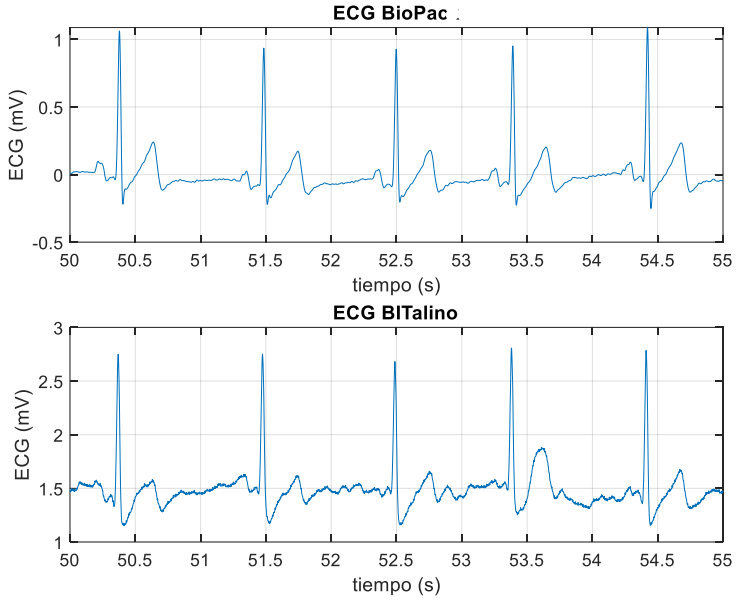


Figura 44. *Fragmento de la señal ECG del ejemplo en el intervalo temporal [50-55].*

Para el cálculo del HRV, primero hay que calcular el intervalo RR, el cual se define como el tiempo transcurrido entre dos ondas R consecutivas en el cardiograma. Es decir, el tiempo que transcurre desde los picos más altos de la onda. Teniendo el ejemplo del BITalino, en la Figura 45 se observa como el programa de Matlab encuentra cada pico.

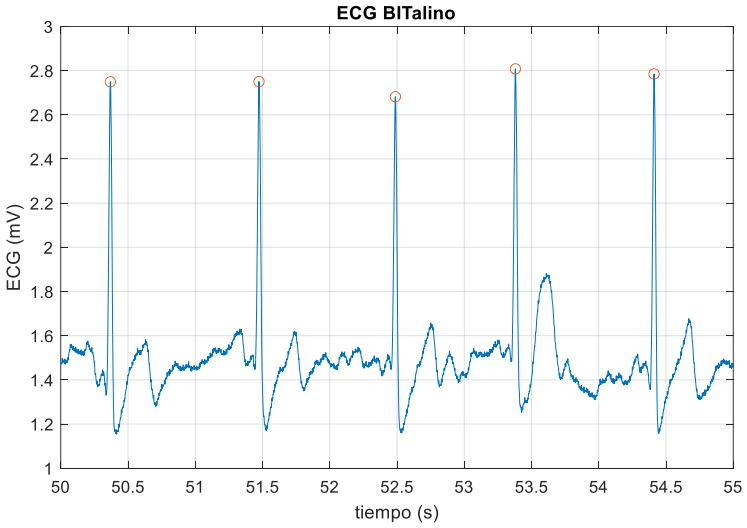


Figura 45. *Búsqueda del intervalo R-R en la onda ECG del BITalino.*

Una vez encontrado el intervalo RR en la onda, se calcula la diferencia temporal entre los picos R conseguidos y, si se plotea mediante la función *stem*, se consiguen las gráficas de la Figura 46.

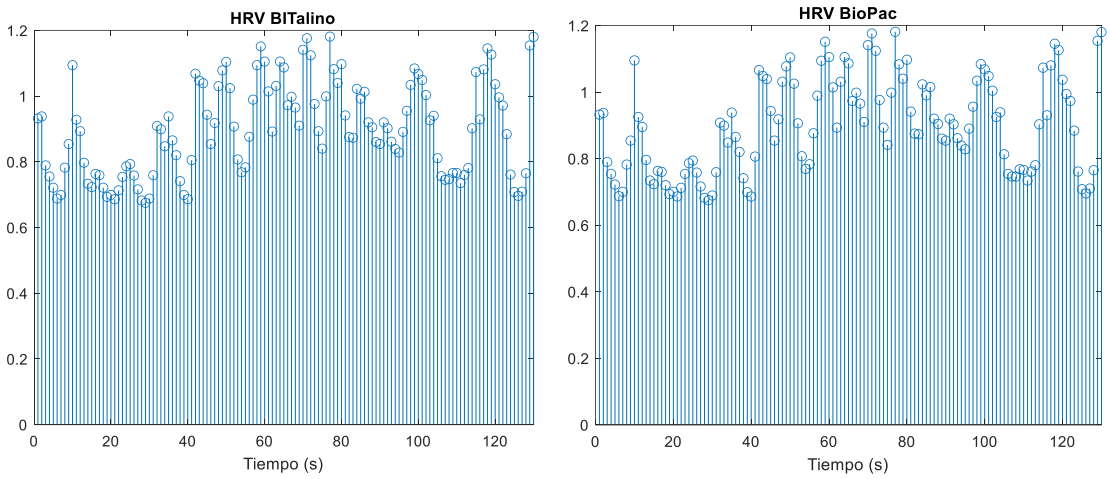


Figura 46. *Variabilidad del Ritmo Cardíaco recogida con el BITalino y Biopac.*

Para evitar trazos sin datos, se realiza una interpolación bajando la frecuencia a 2 Hz. Una vez se obtiene el HRV a frecuencia baja, se convierte a latidos por minuto. En la Figura 47 se muestran los latidos por minuto respecto al tiempo tanto del BITalino como del BioPac, es decir, la HR de ambas señales donde se aprecia que están superpuestas. Esto significa que, en esta prueba, ambos dispositivos han recogido las señales de manera similar.

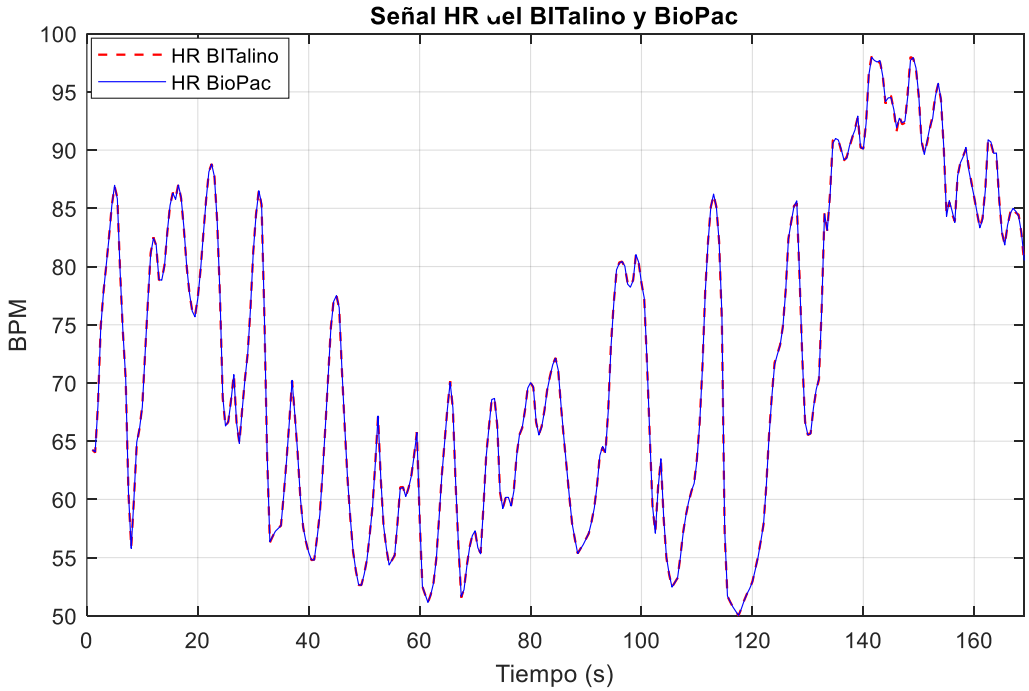


Figura 47. *Señal de HR conseguida con BITalino y Biopac.*

El código de lo expuesto hasta ahora sería el siguiente:

```
%% Cálculo del HRV: BITalino
%findpeaks: buscamos los picos más altos para ver la distancia entre ellos
figure
plot(t,ECG_t); xlim([0 4])
title('ECG BITalino')
xlabel('tiempo'); ylabel('ECG (mV)'); grid; hold on;
```

```
umbral_y_t= 1.5*mean(abs(ECG_t));
umbral_x_t = 0.3*fs; %60/200 por el número de muestras

[pks_t, locs_t] =
findpeaks(ECG_t,'MinPeakHeight',umbral_y_t,'MinPeakDistance',umbral_x_t);
scatter(t(locs_t), pks_t)

%% Calculo del HRV: BioPac
figure
plot(t,ECG_o); xlim([0 4])
title('ECG BioPac')
xlabel('tiempo'); ylabel('ECG (mV)'); grid; hold on;

umbral_y_o= 4*mean(abs(ECG_o));
umbral_x_o = 0.3*fs; %60/200 por el número de muestras

[pks_o, locs_o] =
findpeaks(ECG_o,'MinPeakHeight',umbral_y_o,'MinPeakDistance',umbral_x_o);
scatter(t(locs_o), pks_o)

%% HRV
HRV_bit = diff(t(locs_t));
figure
stem(HRV_bit); title('HRV BITalino'); xlabel('Tiempo (s)'); xlim([0 130]);

HRV_bio = diff(t(locs_o));
figure
stem(HRV_bio); title('HRV BioPac'); xlabel('Tiempo (s)');xlim([0 130]);

% interpolar
new_t = downsample(t, 500);
%Biopac
HRV_bio_f = interp1(t(locs_o(1:end-1)), HRV_bio, new_t);

BPM_bio = (1./HRV_bio_f)*60; %latidos por minuto
BPM1_bio = BPM_bio(~isnan(BPM_bio)); %quitamos los valores NaN
new_t1 = new_t(~isnan(BPM_bio));

%Bitalino
HRV_bit_f = interp1(t(locs_t(1:end-1)), HRV_bit, new_t);

BPM_bit = (1./HRV_bit_f)*60; %latidos por minuto
BPM1_bit = BPM_bit(~isnan(BPM_bit));
new_t1 = new_t(~isnan(BPM_bit));

%Gráficas
figure
subplot(2,1,1); plot(new_t1, BPM1_bio, 'r'); xlim([0 130]);
xlabel ('Tiempo (s)'); ylabel ('BPM'); title('HR BIOPAC')
subplot(2,1,2); plot(t,ECG_o); xlim([0 130])
title('ECG BioPac')
xlabel('tiempo (s)'); ylabel('Amplitud (mV)'); grid; hold on;

figure
subplot(2,1,1); plot(new_t1, BPM1_bit, 'r'); xlim([0 130]);
xlabel ('Tiempo (s)'); ylabel ('BPM'); title('HR BITalino')
subplot(2,1,2); plot(t,ECG_t); xlim([0 130])
title('ECG BITalino')
xlabel('tiempo (s)'); ylabel('Amplitud (mV)'); grid; hold on;
```

Aunque en el ejemplo expuesto ahora la onda de la señal es relativamente buena, durante algunos análisis se han encontrado señales con mucho ruido. En el apartado de conclusiones se hará referencia

a las interferencias que puede sufrir en la lectura el BITalino, las cuales han llegado a meter una onda de 50 Hz de ruido en la señal que se está adquiriendo. Por eso, ha sido necesario la implementación de un filtro notch para eliminar la onda de 50 Hz.

A continuación, se va a analizar este filtro paso a paso. En la Figura 48 se aprecia un ejemplo de la adquisición de una señal ECG que se ha recogido de manera sincronizada con el BioPac y BITalino. Aunque sin un procesamiento en ninguna de las dos gráficas se pueda observar la onda ECG de manera correcta, es decir, diferenciar el complejo QRS, en la señal del BITalino se puede apreciar el ruido.

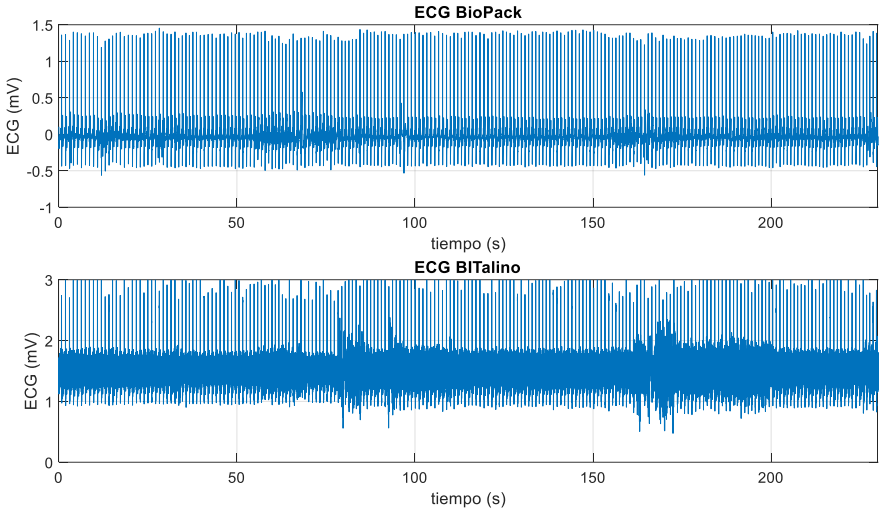


Figura 48. *Ejemplo de una señal ECG adquirida mediante BioPac y BITalino.*

En la Figura 49 se observa el zoom realizado a la anterior señal en el fragmento temporal [100, 108], y aquí se puede apreciar que la señal del BITalino tiene mucho ruido.

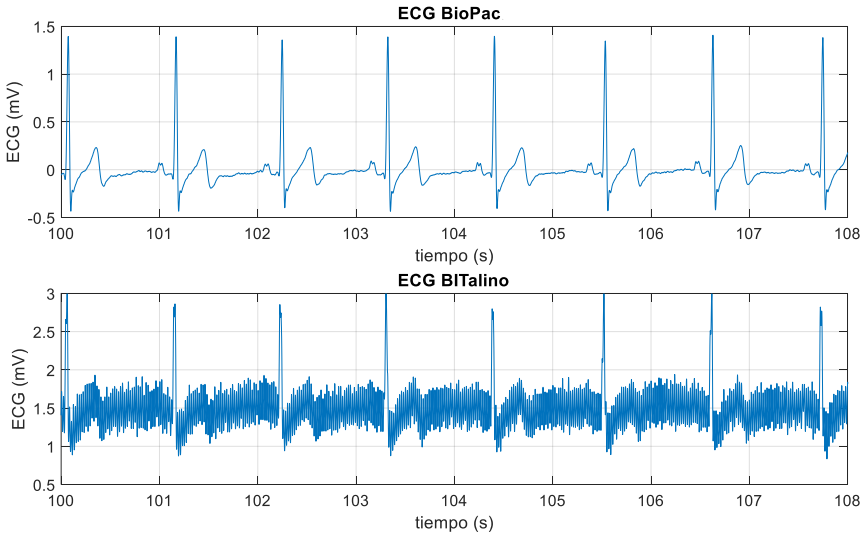


Figura 49. *Secuencia de 8 segundos de la señal adquirida ECG.*

En un segmento aún más preciso, se observa que el ruido del BITalino es debido a ondas sinusoidales de 50Hz, Figura 50, que probablemente sea inducido por el campo electromagnético producido por la red eléctrica, o algún otro ruido del laboratorio.

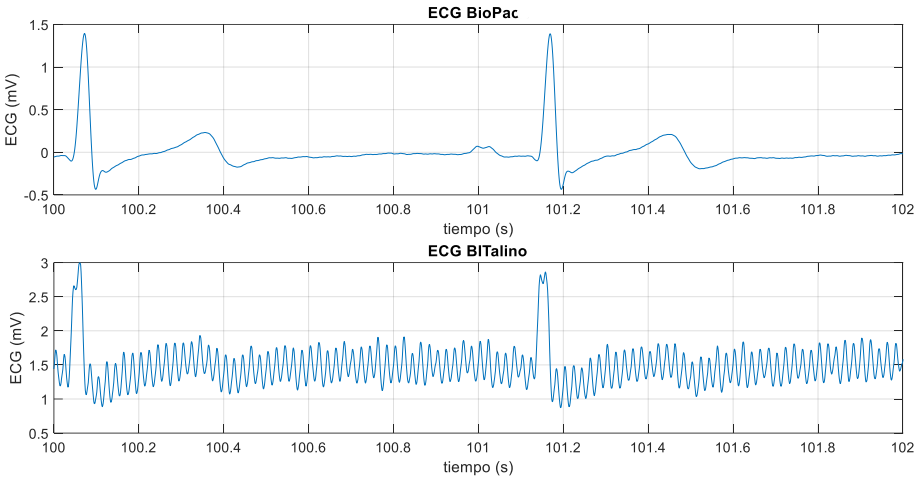


***Figura 50**. Secuencia de 2 segundos de la señal ECG para apreciar la onda de 50 Hz de ruido.*

Para eliminar esta onda, se va a aplicar un filtro notch de 50 Hz. El filtro en Matlab quedaría de la siguiente manera:

```
%% filtro NOTCH
% aplicamos el filtro notch para quitar la onda de 50 hz
w=50/(fs/2);
bw=w;
[num,den]=iirnotch(w,bw); % notch filter implementation
ECG_filter= filter(num, den, ECG_t);
N1=length(ECG_filter);
t1=[0:N1-1]/fs;
```

Y si se grafica el resultado de la misma secuencia de onda, se observa que esas ondas sinusoidales de 50Hz han desaparecido, aunque sigan quedando un pequeño ruido en la señal (véase Figura 51 y Figura 52). Aun así, el complejo de la onda se diferencia, por lo que se podrán obtener datos para el análisis de correlación de ambas señales.

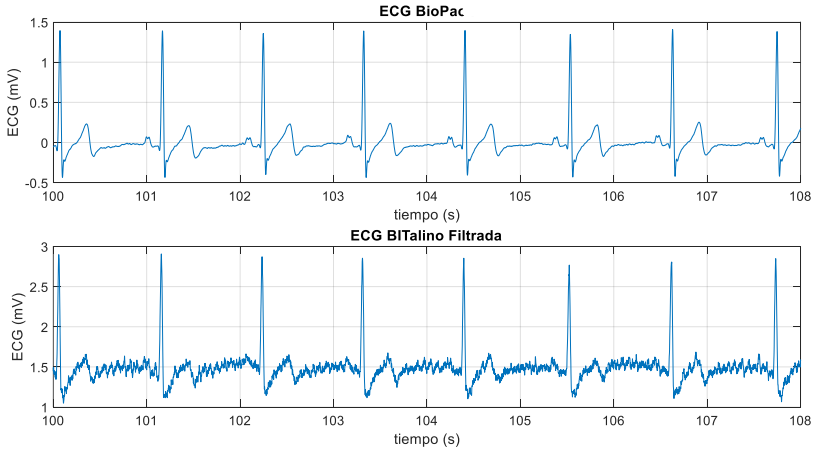


***Figura 51.** Resultado de la secuencia de 8 segundos tras aplicar el filtro notch a la señal ECG.*

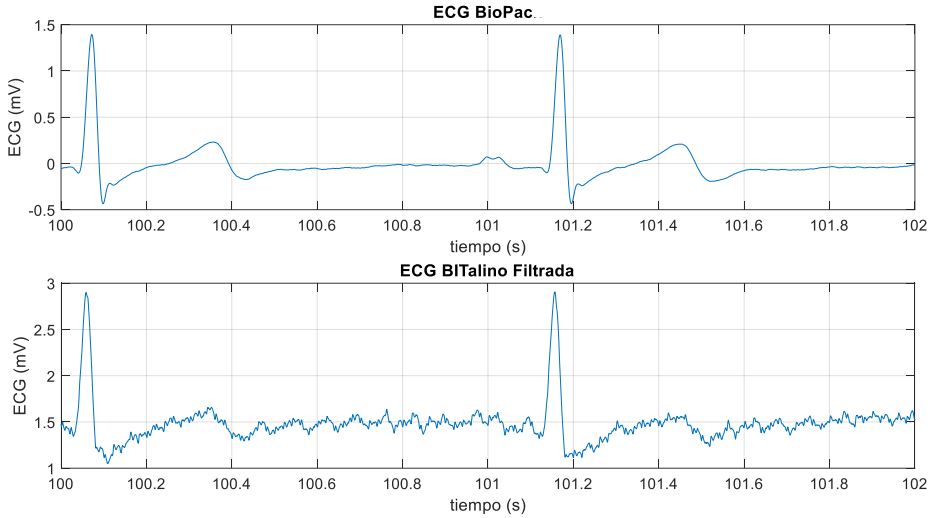


Figura 52. *Resultado de la secuencia de 2 segundos tras aplicar el filtro notch a la señal ECG.*

7.4. Diseño de pruebas

Por último, se debe diseñar un itinerario de pruebas para una correcta validación del dispositivo HW y aplicación propia. En cada prueba, se deben ir logrando los objetivos propuestos, hasta ver que los resultados obtenidos son relativamente buenos respecto al dispositivo de referencia, y crear el diseño final vestible y portable para el usuario.

7.4.1. PRUEBA 1: validación inicial de la recogida de ECG y EDA mediante Biopac y BITalino

El objetivo de esta primera prueba es simplemente observar que tanto el BITalino como el BIOPAC recogen las señales de una forma adecuada y que se puede trabajar y crear más pruebas a partir de estos dispositivos.

Señales a recoger:

- BIOPAC: ECG pecho + EDA mano.
- BITalino: ECG pecho + EDA mano + módulo sincronización.

En este caso, el módulo de sincronización se inserta en uno de los canales analógicos del BITalino.

Una vez realizadas las pruebas, se debe prestar atención especialmente en dos puntos: la primera de ellas si el ECG se refleja sobre la señal de sudoración, es decir, se observa un acoplamiento de ambas señales por el mismo dispositivo; además de observar si la dinámica de las gráficas es igual.

Para ello, se va a realizar un procesamiento de comprobación, analizando las correlaciones entre las señales del BIOPAC y BITalino y calculando también a partir de la señal ECG el HR de ambos dispositivos, como se ha mencionado en el anterior apartado.

El diseño de adquisición de datos de esta primera prueba se aprecia en la Figura 53. En este diseño se

han seguido las recomendaciones para las ubicaciones de los electrodos, ya que lo interesante de esta prueba es que se validen ambas señales siendo las mismas, y no buscar un diseño vestible.

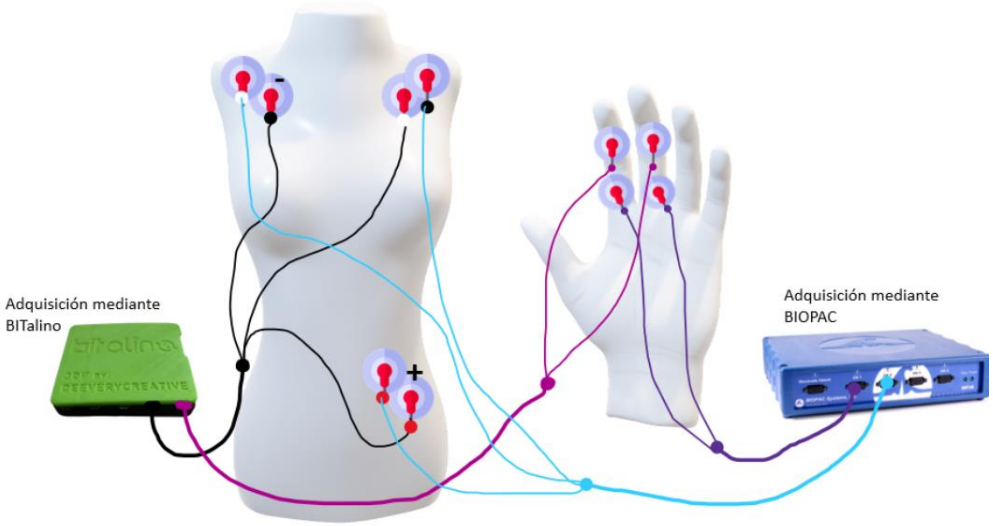


Figura 53. *Diseño para la prueba 1.*

Resultados de la PRUEBA 1

La primera prueba realizada para observar la sincronización de las señales mediante ambos dispositivos, se realizó con la señal fisiológica EDA. Esta señal mide la conductividad que varía conforme a la sudoración de la piel de los sujetos, como se ha comentado anteriormente. Se escogió este dado por su simplicidad para observar los cambios en la respuesta ante diferentes cambios de estrés, como se va a observar en uno de los ejemplos del sujeto 004 en la Figura 54.

Para ello, se ha realizado esta primera prueba a 4 diferentes sujetos, tan solo con la intención de que se relajaran mediante un video durante los dos primeros minutos, y luego una pequeña prueba de estrés mediante cálculo matemático, prueba que se ha mencionado en el apartado 7.2.1. Como se puede observar, las correlaciones entre ambas señales son todas mayores del 90%, y observando las gráficas que aparecen en los resultados del ANEXO V, se observa que se respeta la tendencia en cada momento de estrés o relajación.

Tabla 7. *Resultados de la correlación entre el dispositivo Biopac y BITalino.*

Sujeto	001	002	003	004
Correlación EDA	0.9427	0.9930	0.9640	0.9890

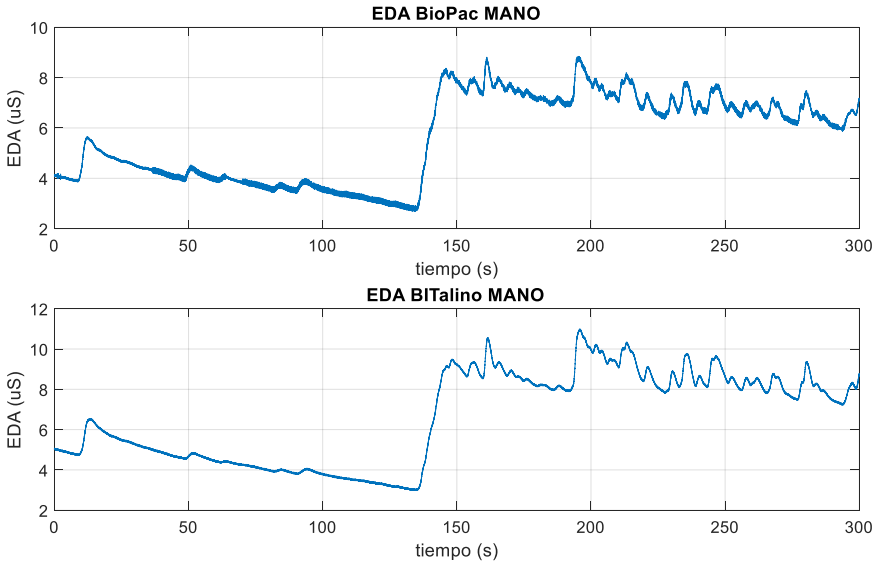


Figura 54. *Resultado de la recogida de la señal EDA del sujeto 004.*

Tras ver la alta correlación entre las señales EDA, el siguiente paso ha sido adquirir la señal EDA junto con la ECG, como la prueba que estaba preestablecida desde el primer momento. Este paso es importante dado que puede haber un acoplamiento de señales en el dispositivo de recogida y una se vea superpuesta en la otra.

En este caso, tras procesar los resultados en Matlab, se ha observado que los resultados también han sido favorables. La EDA ha seguido teniendo una alta correlación, y tras analizar el HR de la señal ECG, también se ha observado que el BITalino respecto al Biopac, es un dispositivo fiable.

Tabla 8. *Resultados de la correlación entre el Biopac y BITalino.*

Sujeto	005	006	007	008	009	010
Correlación EDA		0.5581	0.9923	0.9957	0.8922	0.9763
Correlación ECG (HR)	0.9997	1	0.7276	0.8724	0.9991	0.1571

En la Tabla 8 se pueden observar los resultados de 6 voluntarios en este caso. Cabe destacar que, los dos últimos casos son obtenidos de los resultados de la prueba 2, ya que, tras ir analizando la prueba 1 y ver que los resultados eran favorables respecto a la validación en los 4 voluntarios anteriores, se procedió a realizar la siguiente prueba, que está más relacionada con la prueba del rendimiento del BITalino.

Respecto a los resultados, se puede observar que, aunque en general son buenos, hay momentos en los que la correlación es más mala. En el caso de la EDA del voluntario 006, es debido a que se le movió algún electrodo de la mano, por eso hay una zona en la que no coincide. Y en el caso del voluntario 010, los electrodos no se le pegaron adecuadamente al pecho, y es por eso que la señal es muy mala. Más adelante se hará hincapié en las conclusiones de estos resultados.

Como ejemplo de los resultados, aunque todos ellos se encuentran en el ANEXO V, está la Figura 55 que representa la señal HR del voluntario 007. Se observa que en todo momento siguen la misma tendencia.

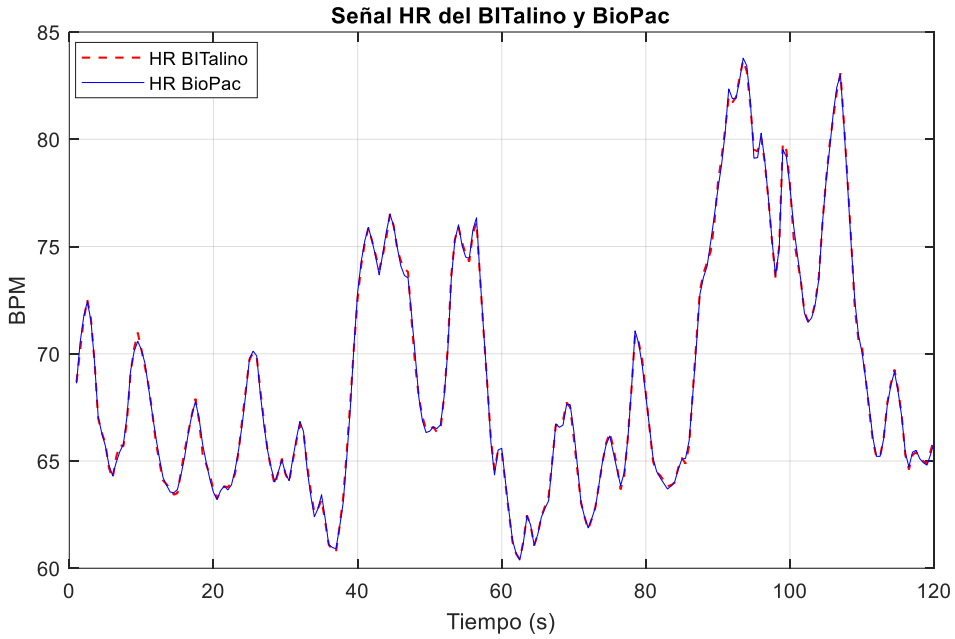


Figura 55. *Señal HR del voluntario 007.*

7.4.2. PRUEBA 2: Prueba de rendimiento de la plataforma recogiendo el máximo de señales posibles.

Una vez se ha comprobado que el BITalino y el BIOPAC tienen una correlación aceptable entre las señales, el siguiente objetivo, además de ir creando un diseño vestible para la recogida de señales, es observar cómo se comporta el BITalino a medida que se añaden canales para la recogida de señales. Por eso mismo, en esta prueba se van a correlacionar dos señales de sudoración más la señal ECG en el BITalino, comprobando así que el BITalino no se satura, y además comprobando si colocando los electrodos en otras ubicaciones, la señal sigue la misma dinámica.

Señales a recoger:

- BIOPAC: ECG pecho + EDA mano
- BITalino: ECG pecho + EDA mano + EDA pecho + módulo de sincronización

En este caso, el módulo de sincronización se añade como una entrada digital y no haciendo uso de los canales analógicos.

Por ello, en este caso se tendrá que prestar atención en si se sigue manteniendo un sincronismo entre las señales adecuado: si se observa en las señales adquiridas por dispositivos la señal ECG reflejada en la sudoración como en el caso anterior; o si la dinámica de las señales cambia respecto a la otra.

Para ello, siguiendo el procesamiento de la anterior prueba, se observarán las correlaciones entre las señales del BIOPAC y BITalino, además de calcular el HR derivado del ECG de ambos dispositivos y comparar sus valores. En esta prueba también se analizará por separado los resultados obtenidos tan solo del BITalino, para observar si tiene capacidad de corriente para monitorizar bien todos los canales

escogidos y si existe una correlación de la señal EDA entre la ubicación de los electrodos en la mano y en el pecho.

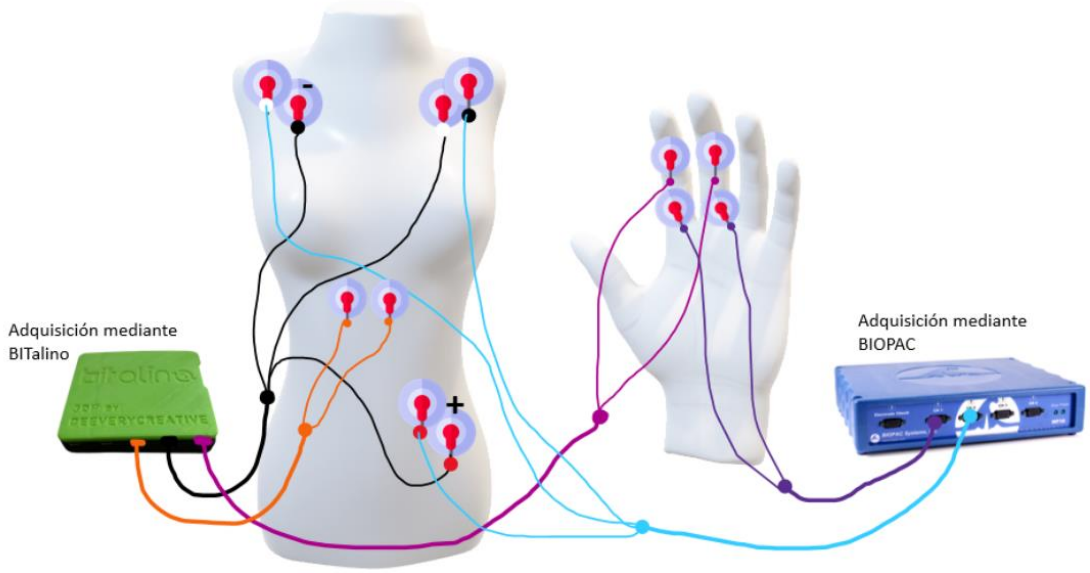


Figura 56. *Diseño de la prueba 2.*

Resultados de la PRUEBA 2

En este caso, los resultados han vuelto a ser favorables respecto a la capacidad que tiene el BITalino de recoger señales a la vez. Aun así, tras analizar las pruebas, los resultados con la comparativa de la EDA de la mano respecto a la del pecho no fue buena.

Esta prueba tenía como objetivo principal el observar cómo trabajaba el BITalino, es decir, añadirle más canales analógicos trabajando, para poder ver si seguía trabajando de una manera correcta, o si al meterle muchas señales dejaba de trabajar de la misma forma que en la prueba 1. Los resultados obtenidos en este sentido fueron buenos, dado que la correlación entre la ECG y la señal EDA capturada en la mano fueron buenas.

Además de este objetivo, se empezó a plantear el diseño vestible que tiene como objetivo futuro el grupo de investigación. Por eso se insertó otra señal a adquirir también de la EDA, pero en este caso en el pecho. Es en este caso donde el análisis comparativo entre las dos señales EDA no coincide, y la señal adquirida en el pecho es mala.

A partir de este momento se modifican las pruebas, y se decide realizar la prueba 3, añadiendo al TFM el objetivo de encontrar un punto óptimo de la recogida de la señal EDA al TFM.

7.4.3. PRUEBA 3: Recogida de la señal EDA de diferentes partes del cuerpo

Como se ha mencionado en el apartado del ESTADO DEL ARTE, son muchas las investigaciones que se llevan a cabo con el objetivo de buscar un lugar adecuado donde recoger la señal EDA. Este proyecto tiene como finalidad diseñar un dispositivo vestible, por lo que las siguientes pruebas a realizar, tras

observar que el dispositivo con el que trabajamos ha respondido adecuadamente hasta ahora, será ir observando diferentes lugares para recoger la señal EDA, teniendo siempre como referencia la mano.

Los principales lugares que se van a analizar se pueden observar en la siguiente figura. Las doce diferentes ubicaciones se analizarán de una en una mediante el dispositivo BITalino, adquiriendo en el canal uno la señal de referencia EDA de la mano, versus la nueva ubicación.

Cabe destacar que estas ubicaciones se han escogido, o bien referenciadas por los artículos mencionados anteriormente, los recomendados por PLUX, o también observando el mapa de calor del cuerpo humano, viendo la ubicación de las glándulas sudoríparas; pensando en un posible diseño de una cinta alrededor del pecho para encontrar esta señal.

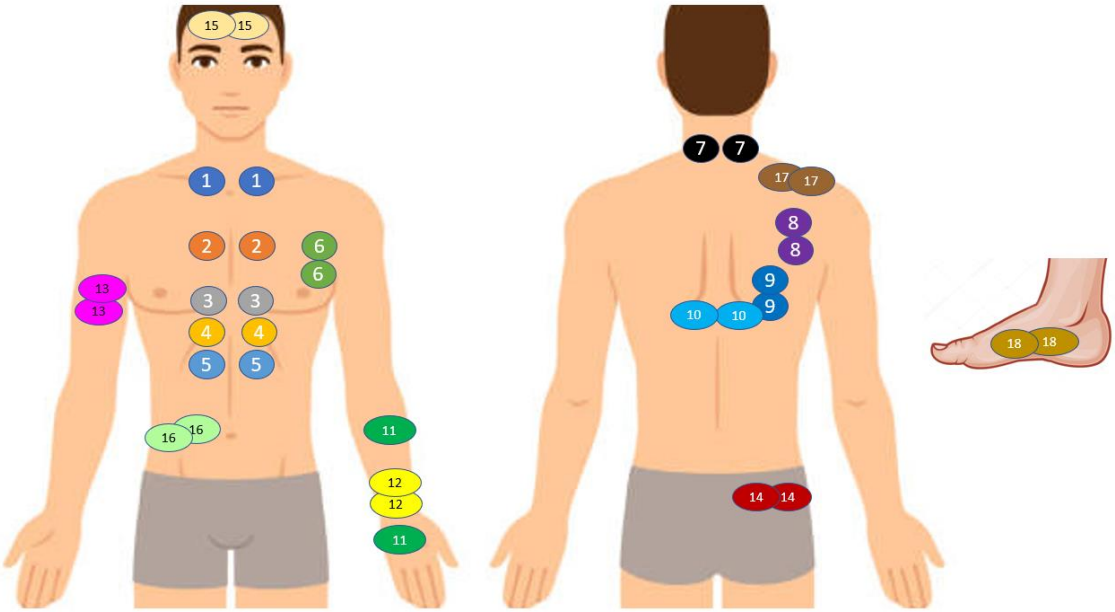


Figura 57. *Ubicación de los electrodos para la prueba 3.*

En la Tabla 9 se nombra el lugar de cada ubicación correspondiente al número. Esta enumeración se mantendrá durante todo el proyecto.

Tabla 9*. Nombres de las ubicaciones de los electrodos*

1	Cuello delantero	10	Espina dorsal
2	Pectoral superior	11	Antebrazo
3	Pectoral inferior	12	Muñeca
4	Abdomen superior	13	Bíceps
5	Recto abdominal	14	Glúteo superior
6	Axila	15	Frente
7	Cuello trasero	16	Apéndice
8	Escapula	17	Trapecio
9	Dorsal	18	Pie

Resultados de la PRUEBA 3

Tras realizar esta prueba en diferentes voluntarios, se han obtenido las correlaciones de la EDA de las ubicaciones de la Tabla 9, respecto de la mano. Los resultados aparecen en la siguiente Tabla 10.

Tabla 10. *Resultados de las correlaciones de la prueba 3.*

Ubicación	Correlación 011	Correlación 012	Ubicación	Correlación 011	Correlación 012
1	0.3551	0.5997	10	-0.7055	0.8033
2	0.4094	-0.4857	11	0.9921	0.8426
3	*0.4246	-0.1403	12	0.6266	0.7590
4	*0.9028	-0.7368	13	0.7444	0.4881
5	*0.8921	0.1909	14	0.7735	0.6484
6	0.9520	0.1909	15	0.8337	0.3690
7	-0.0542	0.1393	16	0.6594	-0.0516
8	0.1207	0.0523	17	0.9598	-0.1017
9	0.0523	-0.1791	18	0.9729	0.9660

*Estas pruebas se realizaron con la cinta apretando el pecho, o la muñeca en su caso.

7.5. Conclusiones de los ensayos realizados

Tras observar los diferentes ensayos y pruebas realizadas, y su posterior análisis, cabe destacar que, aunque el BITalino trabaja de manera correcta para la recogida de señales, y que tras la validación es apto para el uso que se le quiere dar en la investigación, se han encontrado diferentes inconvenientes: una de ellas es respecto a la sensibilidad que tiene respecto al ruido. En el anterior apartado se ha explicado el filtro notch que se ha tenido que aplicar a las señales que se recogían con el BITalino. Otra dificultad se ha encontrado cuando los voluntarios tenían demasiado vello en el pecho, dado que los electrodos no se pegaban adecuadamente.

Respecto a la recogida de EDA, las correlaciones obtenidas no son buenas en comparación con la palma de la mano, pie o ubicaciones donde en los artículos mencionados en el estado del arte validaba. Queda abierta una línea futura en este ámbito para seguir mirando diferentes ubicaciones.

8. METODOLOGÍA SEGUIDA EN EL DESARROLLO DEL TRABAJO

En el siguiente apartado se describe la metodología seguida en el desarrollo de llevar a cabo el TFM. Se van a describir las tareas y las subtareas que se han realizado para conseguir el objetivo de desarrollar una aplicación software con un dispositivo de bajo coste para la adquisición de señales fisiológicas. Por otro lado, también se hará mención a la planificación llevada a cabo; y finalizará con las posibles mejoras y proyectos futuros a raíz de este proyecto.

8.1. Descripción de tareas y fases

A continuación, se describen las tareas necesarias y fases a seguir para conseguir los objetivos previamente definidos en este TFM. También se describirán los procedimientos llevados a cabo, con el respectivo equipo. Como resumen, las tareas realizadas son las siguientes:

- Búsqueda de información, definición de la investigación a resolver
- Definición de especificaciones y requisitos
- Formación
- Estado del arte y análisis de alternativas
- Desarrollo del código y comunicación entre dispositivos
- Interfaz de usuario
- Implementación y ensayos
- Documentación

Tarea 1: Búsqueda de información, definición del proyecto, y definición de las especificaciones y requisitos (semana 1).

Descripción: En esta tarea se ha realizado una búsqueda detallada de información acerca de diferentes investigaciones sobre las emociones, y su estrecha relación con las señales fisiológicas, además de buscar información de cómo adquirir dichas señales con un dispositivo de coste menor. Por lo tanto, se ha definido el proyecto, y las funciones y especificaciones que tendrá la aplicación que se cree. Cabe destacar, que el objetivo final descrito al principio puede estar sujeto a modificaciones durante el proyecto.

Medios humanos: Ingeniera Junior y Responsables de Proyecto.

Tarea 2: Formación (semanas 2 y de 5 a 9).

Descripción: Para llevar a cabo el proyecto, es indispensable familiarizarse con las herramientas de trabajo. Como el desarrollo se ha llevado a cabo con BITalino y BioPac, gran parte de las subtareas se han centrado en estudiar cómo trabajan ambos dispositivos. La lectura de datos e interfaz, se va a crear en Python, por lo que otra parte de formación está destinada a esta subtarea. Las subtareas, o diferentes formaciones, que se realizan:

- Tarea 2.1.: Formación BITalino (semana 2)
- Tarea 2.2.: Formación BioPac (semana 2)
- Tarea 2.3.: Formación básica lectura de datos mediante Python online (semana 5 y 6)
- Tarea 2.4.: Formación básica en diseño de interfaz mediante Python (PyQt5) (semana 5 a 9)

Medios humanos: Ingeniera Junior.

Tarea 3. Estado del arte y análisis de alternativas (semanas 3 a 9)

Descripción: Se ha estudiado acerca de diferentes investigaciones sobre la recogida de señales fisiológicas para expresar emociones, mediante diferentes dispositivos o softwares de bajo coste. Además, se ha estudiado el comportamiento humano y la fisiología, para saber cuáles son las mejores opciones para elegir qué señales y donde medirlas.

Medios humanos: Ingeniera Junior.

Tarea 4. Desarrollo del código (semanas 8 a 13)

Descripción: Esta tarea se ha dividido en diferentes subtareas, para poder realizar el código de la aplicación de una forma en la que se vayan aplicando los conocimientos que poco a poco la tarea de formación 2 nos da.

- Tarea 4.1: Entender la aplicación ya diseñada por el Grupo de Investigación (semanas 8 y 9)
- Tarea 4.2: Diseñar una nueva interfaz desde cero para entender cómo trabaja, con modificaciones respecto a la aplicación del grupo (semanas 9 y 10)
- Tarea 4.3: Insertar las nuevas modificaciones a la aplicación ya diseñada (semana de 10 a 13)

Medios humanos: Ingeniera Junior.

Tarea 5. Implementación y ensayos (semanas 14 a 22)

Descripción: En esta tarea se destacan los ensayos y pruebas realizadas mediante la nueva aplicación y el dispositivo.

- Tarea 5.1: Diseño de pruebas a realizar: 1. Validación del dispositivo vs de referencia; 2. Búsqueda de ubicaciones para los electrodos EDA… (semanas 14 y 15)
- Tarea 5.2: Ensayos experimentales de las pruebas a diferentes voluntarios del grupo de investigación (semanas de 15 a 20)
- Tarea 5.3: Análisis de los resultados obtenidos. (semanas de 17 a 20)

Medios humanos: Ingeniera Junior y Grupo de Investigación.

Tarea 6. Documentación (semanas 3 a 24)

Descripción: Esta tarea ha consistido en documentar el TFM, y se ha realizado a medida que el proyecto ha ido avanzando, intentando no dejarlo para el último momento.

Hito: Entrega final del documento, día 18 de septiembre.

Medios humanos: Ingeniera Junior y Responsables del Proyecto.

8.2. Diagrama de Gantt

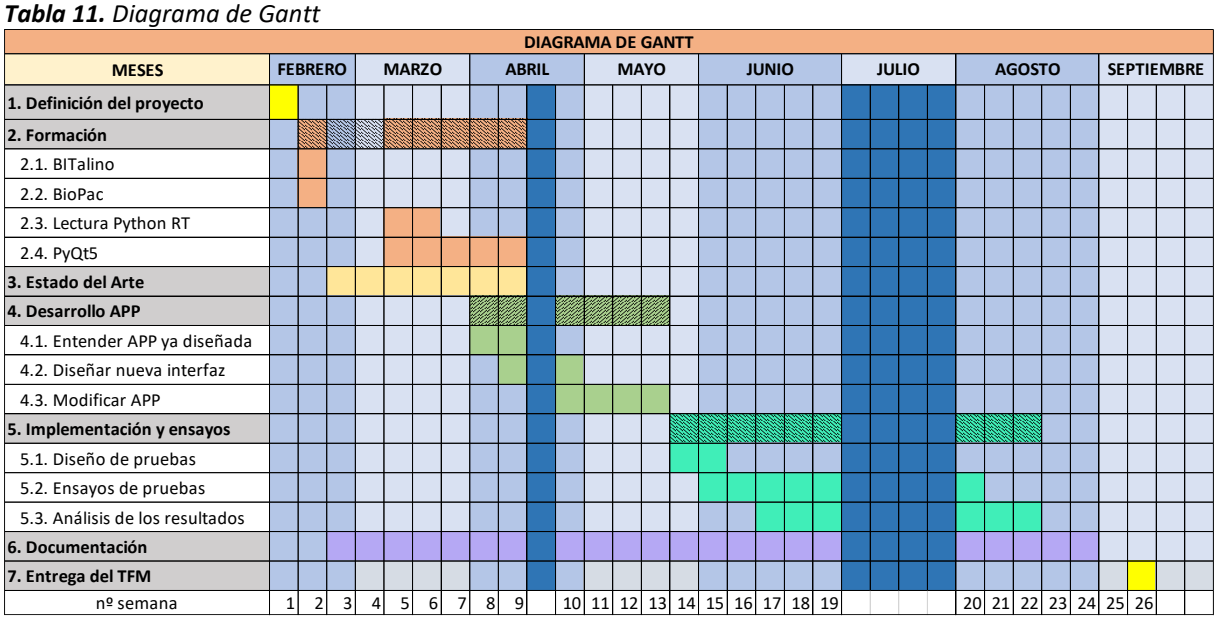
En este apartado, se muestra la planificación temporal de las tareas anteriormente descritas. En la siguiente figura se observa el resultado final del Diagrama de Gantt. Aunque se hizo una planificación inicial, se ha ido modificando respecto a los contratiempos surgidos, dado que, durante una parte de la realización de este proyecto se ha estado intercalando con las clases del Máster, lo que ha llevado a estas semanas, las horas invertidas por días hayan sido menores.

Semana de inicio: 14/02/2022

Semana final: 31/08/2022

En total suman 23 semanas de trabajo desde que se inició, teniendo en cuenta fechas de exámenes o vacaciones que se han restado del total. Además, al inicio del proyecto, hasta el 13 de abril de 2022, al aun estar compaginando las clases del Máster con el proyecto, las horas que se invertían en el TFM eran de media jornada. Tras los exámenes, a partir del 25 de abril de 2022, se ha invertido en el proyecto una jornada completa de lunes a viernes. Durante el mes de julio no se han contabilizado las semanas. En total, el tiempo invertido en el desarrollo del proyecto ha sido de 600 horas.

Tabla 11. *Diagrama de Gantt*



9. ASPECTOS ECONÓMICOS

En este apartado se desarrolla el estudio económico necesario para llevar a cabo este trabajo. Se desglosará el valor de los dispositivos que se han utilizado, para poder observar que se ha cumplido con el objetivo de diseñar una solución HW y SW de bajo coste, pudiendo comparar los precios de los diferentes dispositivos HW. Se desglosa por actividades: horas internas, amortizaciones y gastos. También incluye un resumen con la suma de las tres partidas y el coste total.

Horas internas: Para el cálculo del coste asociado al número de horas se tienen en cuenta las horas de trabajo asociadas a un ingeniero junior, así como las dedicadas por el director del proyecto. Como se ha mencionado en el apartado anterior, el proyecto tuvo una duración de 600 horas para el ingeniero junior. Las horas del director del proyecto suman 50 en total.

Tabla 12. *Aspectos económicos: horas internas.*

Concepto	n º horas	Coste por horas €/h	Coste total (€)
Ingeniero Junior	600	30	18.000
Director del proyecto	50	70	3.500
Total			**21.500**

Amortizaciones: Además, es necesario incluir la partida correspondiente a las amortizaciones de los recursos utilizados durante la realización del proyecto.

Tabla 13. *Aspectos económicos: amortizaciones.*

Concepto	Coste (€)	Vida útil (meses)	Duración Proyecto (meses)	Coste total (€)
PC	800	60	6	74,2
MATLAB	800	12	6	400
Microsoft 365	126	12	6	63
Total				**537,2 €**

Gastos: A continuación, es necesario incluir la partida de gastos relacionada con el material necesario para la ejecución del proyecto.

Tabla 14. *Aspectos económicos: gastos.*

Concepto	Cantidad	Coste unitario €/unit	Coste total (€)
KIT BITalino revolution Plugged	1	256,89	256,89
Caja BITalino	1	12,10	12,10
Optoacoplador LM 4N26	2	2,29	4,58
BioPac MP36	1	300	300
Biosignalsplux Researcher Kit	1	3.617,90	3.617,90
Gastos varios	1	100	100
Total			**4.291,47 €**

Total: Por último, se muestra el resumen del presupuesto total, al que se añade el 5% de los costes indirectos de la realización del proyecto, como la infraestructura o electricidad.

Tabla 15. *Aspectos económicos: gastos totales.*

Concepto	Coste (€)
Horas internas	21.500
Amortizaciones	537,20
Gastos	4.291,47
Costes directos	26.328,67
Costes indirectos (5%)	1.316,43
Total	**27.645,10 €**

Cabe destacar que este sería un presupuesto si el Grupo de Investigación lleva trabajando con estos dispositivos más tiempo en diferentes investigaciones. En esta investigación no se trabaja con un prototipo que se quiera vender, por lo que todo el material utilizado pertenece, o bien al Grupo de Investigación, o bien al departamento de Ingeniería de Sistemas y Automática.

Teniendo en cuenta los costes directos como los indirectos, el presupuesto total del proyecto del Máster *"Desarrollo de una solución hardware y software para la recogida y procesamiento de señales fisiológicas"* asciende a **27.645,10 €**.

10. CONCLUSIONES Y LÍNEAS FUTURAS

El desarrollo de este TFM ha pasado por las fases de estudio, desarrollo y validación de una solución hardware y software para la recogida de señales fisiológicas.

El objetivo principal era validar que el sistema hardware y software diseñado en el grupo de investigación es apto para recoger las señales. Para ello, se han hecho ensayos en los que se recogían las señales fisiológicas con dos dispositivos, con el objetivo de contrastar el diseño propio contra un equipo de referencia. En el análisis de los resultados se ha concluido que el dispositivo de bajo coste es apto para el uso que se le desea dar en este proyecto, ya que recoge las señales de manera adecuada.

En cuanto al objetivo centrado en la mejora del programa software ya existente, se puede decir que la aplicación ha mejorado y que la interfaz, además de ser más intuitiva para el usuario, se ha demostrado mediante las diferentes pruebas realizadas funciona adecuadamente y es apta para el funcionamiento que se le quiere dar en un futuro. Además, con los cambios realizados el tiempo que se pierde en la visualización de datos ha disminuido, teniendo un tiempo de cómputo menor.

Respecto a la recogida de la señal EDA, se han analizado diferentes puntos para ubicar los electrodos y ver el sitio óptimo, teniendo en referencia otras investigaciones analizadas en el estado del arte. Este objetivo no se ha cumplido completamente, dado que no se ha sacado una conclusión certera acerca de donde recoger esta señal de manera segura que no sea colocando los electrodos en la mano, posición poco deseable por el estorbo que le supondría al usuario durante el uso del dispositivo en su vida cotidiana. Por lo tanto, se deja el objetivo de encontrar el punto óptimo para la recogida de esta señal como una línea a desarrollar en el futuro cercano del proyecto.

Finalmente, este trabajo sirve como base para que Grupo de Investigación pueda diseñar un prototipo vestible y de bajo coste en el futuro con el cual poder adquirir las señales fisiológicas y detectar el estrés.

11. BIBLIOGRAFÍA

[1] L. F. Barrett, B. Mesquita, K. N. Ochsner, and J. J. Gross, "The experience of emotion," *Annu. Rev. Psychol.*, vol. 58, pp. 373–403, 2007, doi: 10.1146/annurev.psych.58.110405.085709.

[2] "Verisense - Wearable Sensors for Clinical Trials - Clinical Research - Activity & Sleep." https://verisense.net/ (accessed Sep. 02, 2022).

[3] M. C. Montañés, "Psicología De La Emoción: El Proceso Emocional," pp. 1–34, 2005, [Online]. Available: www.uv.es/=choliz%0Ahttps://s3.amazonaws.com/academia.edu.documents/34266078/2._Psicologia_de_la_emocion._El_proceso_emocional.pdf?A.

[4] L. Pessoa, "Emotion and cognition and the amygdala: From 'what is it?' to 'what's to be done?,'" *Neuropsychologia*, vol. 48, no. 12, pp. 3416–3429, 2010, doi: 10.1016/j.neuropsychologia.2010.06.038.

[5] J. E. LeDoux, "The neurobiology of emotion. Mind and brain: Dialogs in cognitive neuroscience," *Cambridge Univ. Press*, pp. 301–354, 1986, Accessed: Jan. 22, 2022. [Online]. Available: https://nyuscholars.nyu.edu/en/publications/neurobiology-of-emotion.

[6] K. J. Ressler, "Amygdala Activity, Fear, and Anxiety: Modulation by Stress," *Biol. Psychiatry*, vol. 67, no. 12, pp. 1117–1119, 2010, doi: 10.1016/j.biopsych.2010.04.027.

[7] M. Benedek and C. Kaernbach, "A continuous measure of phasic electrodermal activity," *J. Neurosci. Methods*, vol. 190, no. 1, pp. 80–91, 2010, doi: 10.1016/j.jneumeth.2010.04.028.

[8] W. Boucsein *et al.*, "Publication recommendations for electrodermal measurements SOCIETY FOR PSYCHOPHYSIOLOGICAL RESEARCH AD HOC COMMITTEE ON ELECTRODERMAL MEASURES," 2012, doi: 10.1111/j.1469-8986.2012.01384.x.

[9] R. R. Dynes and H. Rodriguez, "Affective reactions to acoustic stimuli," *Psychophysiology*, vol. 37, no. 2, pp. 204–215, 2000.

[10] R. Bortolla, M. Cavicchioli, M. Galli, P. F. M. J. Verschure, and C. Maffei, "A comprehensive evaluation of emotional responsiveness in borderline personality disorder: A support for hypersensitivity hypothesis," *Borderline Personal. Disord. Emot. Dysregulation*, vol. 6, no. 1, pp. 1–16, May 2019, doi: 10.1186/S40479-019-0105-4/FIGURES/2.

[11] Y. Wang *et al.*, "Heart Rate Variability and Skin Conductance During Repetitive TMS Course in Children with Autism," doi: 10.1007/s10484-015-9311-z.

[12] I. Leite, R. Henriques, C. Martinho, and A. Paiva, "Sensors in the Wild: Exploring Electrodermal Activity in Child-Robot Interaction," Accessed: Sep. 06, 2022. [Online]. Available: http://www.plux.info/EDA.

[13] T. Takahashi *et al.*, "Changes in EEG and autonomic nervous activity during meditation and their association with personality traits," *Int. J. Psychophysiol.*, vol. 55, no. 2, pp. 199–207, 2005, doi: 10.1016/j.ijpsycho.2004.07.004.

[14] C. D. Katsis, G. Ganiatsas, and D. I. Fotiadis, "An integrated telemedicine platform for the assessment of affective physiological states," *Diagn. Pathol.*, vol. 1, no. 1, pp. 1–9, 2006, doi: 10.1186/1746-1596-1-16.

[15] R. A. Rhoades and D. R. Bell, *Medical Physiology: Principles for Clinical Medicine, 5e*. 2017.

[16] F. Shaffer and J. P. Ginsberg, “An Overview of Heart Rate Variability Metrics and Norms,” *Front. Public Heal.*, vol. 5, no. September, pp. 1–17, 2017, doi: 10.3389/fpubh.2017.00258.

[17] S. C. Segerstrom and L. S. Nes, “Heart rate variability reflects self-regulatory strength, effort, and fatigue,” *Psychol. Sci.*, vol. 18, no. 3, pp. 275–281, 2007, doi: 10.1111/j.1467-9280.2007.01888.x.

[18] B. M. Appelhans and L. J. Luecken, “Heart rate variability as an index of regulated emotional responding,” *Rev. Gen. Psychol.*, vol. 10, no. 3, pp. 229–240, 2006, doi: 10.1037/1089-2680.10.3.229.

[19] J. T. Cacioppo, L. K. Bush, and L. G. Tassinary, “Microexpressive Facial Actions as a Function of Affective Stimuli: Replication and Extension,” *Personal. Soc. Psychol. Bull.*, vol. 18, no. 5, pp. 515–526, Oct. 1992, doi: 10.1177/0146167292185001.

[20] J. T. Cacioppo, R. E. Petty, M. E. Losch, and H. S. Kim, “Electromyographic Activity Over Facial Muscle Regions Can Differentiate the Valence and Intensity of Affective Reactions,” *J. Pers. Soc. Psychol.*, vol. 50, no. 2, pp. 260–268, 1986, doi: 10.1037/0022-3514.50.2.260.

[21] S. Machado *et al.*, “EEG-based brain-computer interfaces: An overview of basic concepts and clinical applications in neurorehabilitation,” *Rev. Neurosci.*, vol. 21, no. 6, pp. 451–468, 2010, doi: 10.1515/REVNEURO.2010.21.6.451.

[22] A. Jimenez-Molina, C. Retamal, and H. Lira, “Using Psychophysiological Sensors to Assess Mental Workload During Web Browsing,” *Sensors (Basel).*, vol. 18, no. 2, Feb. 2018, doi: 10.3390/S18020458.

[23] A. Kamišali, I. Fister, M. Turkanovi, and S. I. Karakatič, “Sensors and Functionalities of Non-Invasive Wrist-Wearable Devices: A Review,” doi: 10.3390/s18061714.

[24] G. G. Cacioppo, John; Tassianary, Louis G.; Berntson, *The handbook of psychophysiology*. 2007.

[25] A. F. H. Payne, M. E. Dawson, A. M. Schell, K. Singh, and C. G. Courtney, “Can you give me a hand? A comparison of hands and feet as optimal anatomical sites for skin conductance recording,” 2013, doi: 10.1111/psyp.12093.

[26] M. van Dooren, J. J. G. G. J. de Vries, and J. H. Janssen, “Emotional sweating across the body: Comparing 16 different skin conductance measurement locations,” *Physiol. Behav.*, vol. 106, no. 2, pp. 298–304, 2012, doi: 10.1016/j.physbeh.2012.01.020.

[27] M. B. Hossain, Y. Kong, H. F. Posada-Quintero, and K. H. Chon, “Comparison of Electrodermal Activity from Multiple Body Locations Based on Standard EDA Indices’ Quality and Robustness against Motion Artifact,” *Sensors*, vol. 22, no. 9, 2022, doi: 10.3390/s22093177.

[28] R. Zangróniz, A. Martínez-Rodrigo, J. Manuel Pastor, M. T. López, and A. Fernández-Caballero, “Electrodermal Activity Sensor for Classification of Calm/Distress Condition,” doi: 10.3390/s17102324.

[29] P. Schmidt, A. Reiss, R. Duerichen, and K. Van Laerhoven, “Introducing WeSAD, a multimodal dataset for wearable stress and affect detection,” *ICMI 2018 - Proc. 2018 Int. Conf. Multimodal Interact.*, pp. 400–408, 2018, doi: 10.1145/3242969.3242985.

[30] M.-M. Taylor, N. A. Taylor, and C. A. Machado-Moreira, “Regional variations in transepidermal water loss, eccrine sweat gland density, sweat secretion rates and electrolyte composition in resting and exercising humans,” 2013. doi: 10.1186/2046-7648-2-4.

[31] W. Wen *et al.*, “Continuous Estimation of Stress Using Physiological Signals during a Car Race,” *Psychology*, vol. 8, no. 7, pp. 978–986, May 2017, doi: 10.4236/PSYCH.2017.87064.

[32] C. M. Mccrimmon *et al.*, “Performance Assessment of a Custom, Portable, and Low-Cost Brain-Computer Interface Platform HHS Public Access,” *IEEE Trans Biomed Eng*, vol. 64, no. 10, pp. 2313–2320, 2017, doi: 10.1109/TBME.2017.2667579.

[33] A. Hadad and A. Braidot, “Development of a tele-ECG device,” *IFMBE Proc.*, vol. 49, no. January, 2015, doi: 10.1007/978-3-319-13117-7.

[34] “PLUX Biosignals.” https://www.pluxbiosignals.com/ (accessed Sep. 06, 2022).

[35] BITalino, “Electrodermal Activity (EDA) Sensor Data Sheet,” pp. 0–1, 2015, [Online]. Available: http://bitalino.com/.

[36] BITalino, “Electrocardiography (ECG) Sensor Data Sheet,” pp. 5–6, 2020, [Online]. Available: http://bitalino.com/.

[37] BITalino, “Respiration (PZT) Sensor Data Sheet,” pp. 9–10, 2020, [Online]. Available: http://bitalino.com/.

[38] J. Guerreiro, A. Lourenço, H. Silva, and A. Fred, “Performance Comparison of Low-cost Hardware Platforms Targeting Physiological Computing Applications,” *Procedia Technol.*, vol. 17, pp. 399–406, 2014, doi: 10.1016/j.protcy.2014.10.204.

[39] H. P. da Silva, C. Carreiras, A. Lourenço, A. Fred, R. C. das Neves, and R. Ferreira, “Off-the-person electrocardiography: performance assessment and clinical correlation,” *Health Technol. (Berl).*, vol. 4, no. 4, pp. 309–318, 2015, doi: 10.1007/s12553-015-0098-y.

[40] D. Batista, H. Silva, and A. Fred, “Experimental characterization and analysis of the BITalino platforms against a reference device,” *Proc. Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. EMBS*, pp. 2418–2421, 2017, doi: 10.1109/EMBC.2017.8037344.

[41] R. E. Wagner, H. P. da Silva, and K. Gramann, “Validation of a low-cost electrocardiography (Ecg) system for psychophysiological research,” *Sensors*, vol. 21, no. 13, pp. 1–17, 2021, doi: 10.3390/s21134485.

[42] D. Batista, H. P. da Silva, A. Fred, C. Moreira, M. Reis, and H. A. Ferreira, “Benchmarking of the BITalino biomedical toolkit against an established gold standard,” *Healthc. Technol. Lett.*, vol. 6, no. 2, pp. 32–36, 2019, doi: 10.1049/htl.2018.5037.

[43] S. Mathôt, D. Schreij, and J. Theeuwes, “OpenSesame: An open-source, graphical experiment builder for the social sciences,” *Behav. Res. Methods*, vol. 44, no. 2, pp. 314–324, 2012, doi: 10.3758/s13428-011-0168-7.

[44] J. W. Peirce, “PsychoPy-Psychophysics software in Python,” *J. Neurosci. Methods*, vol. 162, no. 1–2, pp. 8–13, 2007, doi: 10.1016/j.jneumeth.2006.11.017.

[45] H. Ibrahim, S. Ewais, and S. Chatterjee, “A novel, low-cost NeuroIS prototype for supporting bio signals experimentation based on BITalino,” in *Lecture Notes in Information Systems and Organisation*, vol. 10, Springer Heidelberg, 2015, pp. 77–83.

[46] “PLUX Biosignals | OpenSignals & Data analysis Add-Ons.” https://www.pluxbiosignals.com/collections/opensignals (accessed Sep. 07, 2022).

[47] “OpenSignals (r)evolution (Download) – Support PLUX Biosignals official.”

https://support.pluxbiosignals.com/knowledge-base/introducing-opensignals-revolution/ (accessed Aug. 23, 2022).

[48] C. O. Systems, “Biopac Student Lab,” 2005.

[49] “Welcome to Python.org.” https://www.python.org/ (accessed Sep. 07, 2022).

[50] S. Sepúlveda, P. Reyes, and A. Weinstein, “Visualizing physiological signals in real-time,” *Proc. 14th Python Sci. Conf.*, no. Scipy, pp. 182–186, 2015, doi: 10.25080/majora-7b98e3ed-01c.

[51] M. Fitzpatrick, “Create GUI Applications with Python & Qt5 (PyQt5 Edition): The hands-on guide to making apps with Python,” p. 825, 2020.

[52] I. BIOPAC Systems, “BIOPAC Hardware, MP36/45.” pp. 1–6.

[53] PLUX Wireless Biosignals S.A., “OpenSignals (r)evolution - user manual.”

[54] “GitHub - BITalinoWorld/revolution-python-api: Python API for BITalino (r)evolution.” https://github.com/BITalinoWorld/revolution-python-api (accessed May 04, 2022).

[55] “GitHub - pybluez/pybluez: Bluetooth Python extension module.” https://github.com/pybluez/pybluez (accessed Feb. 16, 2022).

ANEXO I.

MANUAL PARA LA RECOGIDA DE SEÑALES MEDIANTE LAS PLATAFORMAS SOFTWARE OPENSIGNALS Y BIOPAC STUDENT LAB 4.1

ANEXO I. MANUAL PARA LA RECOGIDA DE SEÑALES MEDIANTE LAS PLATAFORMAS SOFTWARE OPENSIGNALS Y BIOPAC STUDENT LAB 4.1

En este ANEXO se detalla el paso a paso para la recogida de señales mediante las dos plataformas con las que se trabaja durante el proyecto.

Paso a paso del OpenSignals

A continuación, se hará un resumen sobre la monitorización utilizando BITalino y OpenSignals [53]. El dispositivo biosignalplux también se monitoriza de igual manera, aunque da más opciones.

1. Primero, el dispositivo BITalino tendrá que estar emparejado con el ordenador con el que se analizarán los datos. Para ello, se encenderá el dispositivo, y desde el ordenador se irá al menú Bluetooth para buscar un nuevo dispositivo. Una vez que aparezca el BITalino, se selecciona la opción de emparejar y se introduce el pin que se solicita, 1234.

2. Una vez emparejados los dos dispositivos, se abre la plataforma OpenSignals, y se clica en el icono de la lupa que se muestra nada más iniciar la aplicación. En esta pestaña debería aparecer la placa que se ha emparejado, y pinchando sobre esta, se abrirá un abanico de opciones para seleccionar los canales de los que se quiere obtener información (ECG, EMG…). También habrá que seleccionar la frecuencia de muestreo a la que se quieren obtener los resultados.

***Figura 58**. Interfaz de configuración, siendo el de la izquierda, el del BITalino, y el de la derecha, Biosignalsplux.*

3. El último paso será clicar el icono de grabar (punto rojo). Así, se abrirá una interfaz gráfica en la que se podrá observar la señal escogida.

Figura 59. *Inicio de la adquisición de los dos canales escogidos.*

Una vez finalizada la grabación, se obtendrán dos archivos procedentes de cada señal, ya que BITalino crea dos extensiones diferentes por cada grabación (.h5 y .txt).

En la Figura 60 se observa el ejemplo de la adquisición de datos. Aunque en la parte de abajo aparece toda la señal, hay opción de coger tan solo la parte que te interesa para analizarla más en concreto. En este caso se ha cogido una parte para observar mejor todas las señales. El primer canal corresponde al EDA, donde se observa que el sujeto en esa parte se ha puesto más nervioso, y es por eso que la gráfica tiene una tendencia ascendente al principio, donde luego se va relajando. El segundo canal es la respiración; y el tercer canal se recoge el ECG, donde se puede observar el complejo PQR de la señal.

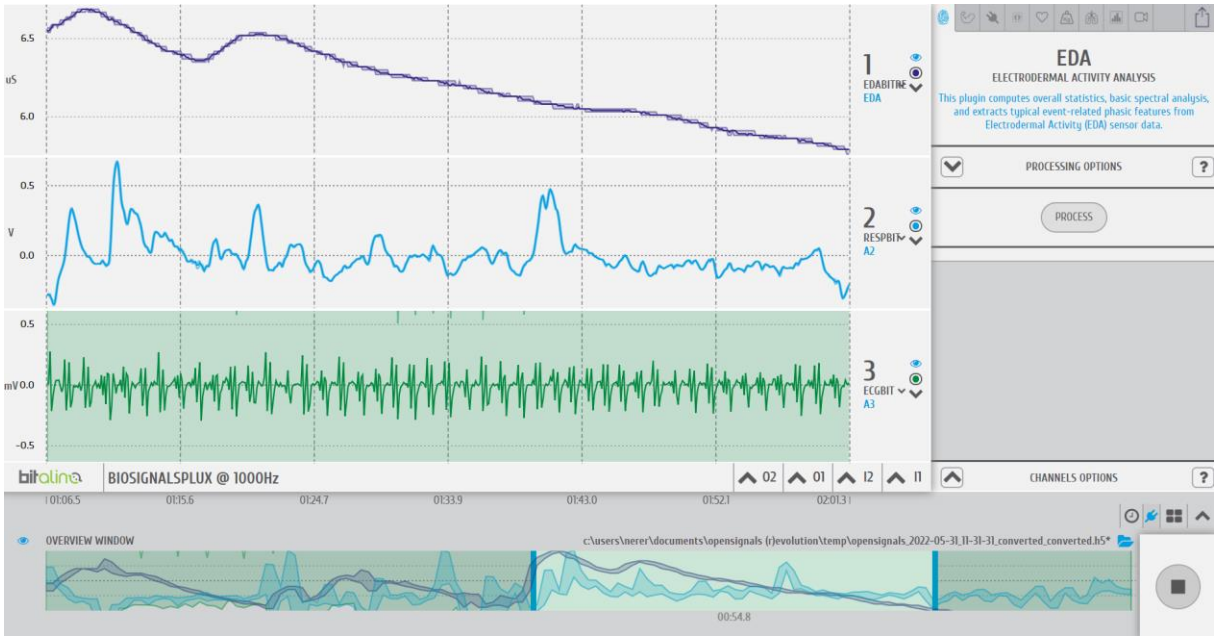


Figura 60. *Ejemplo de la recogida de señales EDA, ECG y RESP con BITalino.*

Paso a paso Biopac Student Lab 4.1.

En el caso de la plataforma Biopac Student Lab 4.1, esta trabaja de una forma muy similar al estudiado anteriormente, OpenSignals. Aun así, en vez de introducir la MAC, en este caso el hardware de adquisición debe estar conectado mediante USB. Es más, debe estar conectado y encendido el Biopac para que deje al usuario acceder a la plataforma. Si no está conectada, dará la opción de realizar solo análisis de pruebas adquiridas anteriormente. En la Figura 61 se observa el mensaje que aparece si el Biopac no está conectado al ordenador donde se van a analizar las señales.

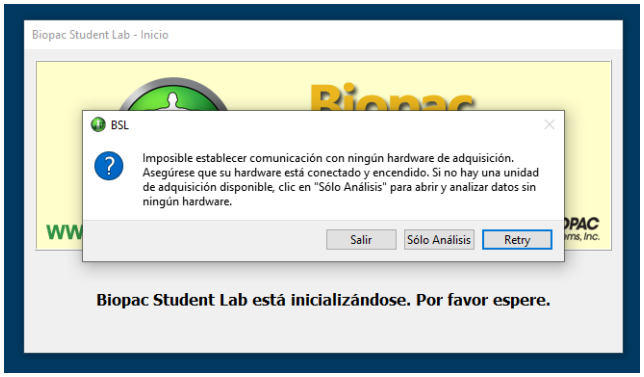


***Figura 61**. Aviso de comunicación errónea en el software Biopac SL.*

Una vez se establece comunicación con el hardware de adquisición, se accede al menú principal (véase Figura 62). Este software está enfocado a la educación e investigación, es por eso que la primera opción disponible trata de registrar o analizar una lección disponible en BSL. Es la segunda opción la que se usará en el caso del proyecto, donde se crea o registra un nuevo experimento. Además, accediendo a ella, da la opción de crear un gráfico vacío, abrir una plantilla de gráfico desde el disco o utilizar una plantilla reciente. La tercera opción se utilizará en el caso que tan solo se desee analizar un registro anterior, y las dos últimas opciones son para, o bien personalizar una lección disponible del software BSL, o para acceder a manuales y ayuda para el usuario.

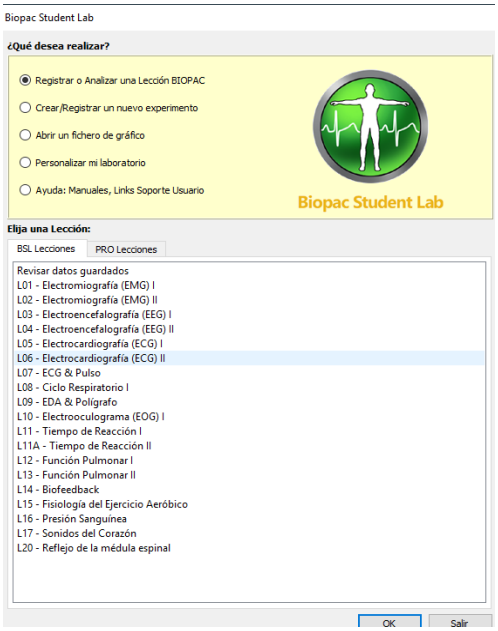


***Figura 62.** Pantalla principal del hardware Biopac SL.*

En el caso del proyecto, esta plataforma solo se va a hacer uso para la adquisición de señales, por lo que se escogerá la segunda opción mencionada anteriormente, *Crear/Registrar un nuevo experimento*. En ella se ofrecen tres nuevas opciones más, Figura 63. La primera vez que se desee hacer una nueva adquisición, se deberá crear un gráfico vacío. En esta opción se deberán configurar todos los canales que se deseen adquirir. Es por esto que se recomienda crear una plantilla vacía cuando se va a realizar una misma prueba varias veces. Es decir, antes de realizar ninguna prueba, crear un nuevo experimento configurando todos los parámetros, y guardarla. Así, una vez se comiencen con las pruebas se abre la misma plantilla y se guarda con el nombre apropiado para guardarla.

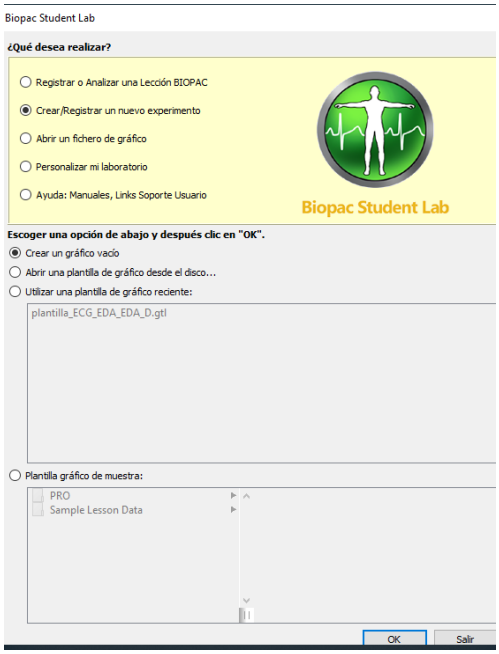


***Figura 63**. Pantalla para crear un nuevo experimento desde cero en Biopac SL.*

Una vez se accede a crear un gráfico vacío, como se ha mencionado, hay que configurar todas las entradas. En la Figura 64 se observa los diferentes canales que se pueden escoger. Da la opción de visualizar o no la gráfica de manera online y de mostrar el valor. También de seleccionar la frecuencia de muestreo que se adecua a cada caso.

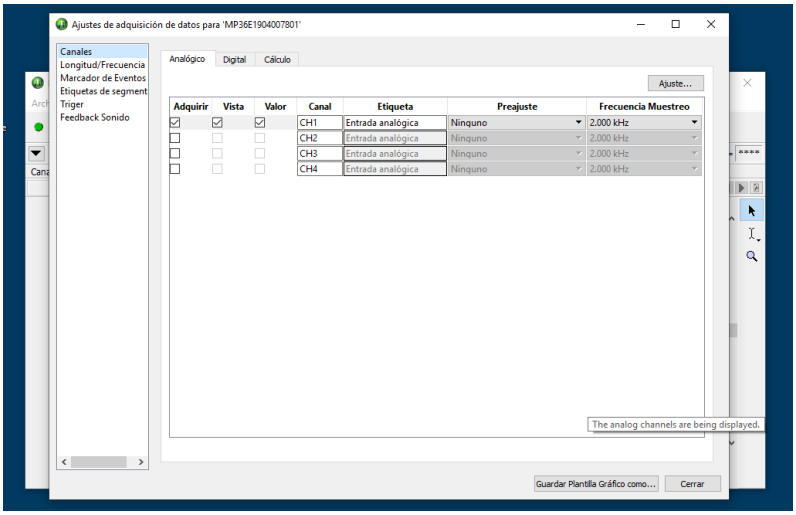


***Figura 64.** Pantalla principal de configuración en Biopac SL.*

En el caso del preajuste, esta plataforma da la opción de adquirir muchas señales, y en diferentes rangos de frecuencia. Esto se puede apreciar en la Figura 65. En este proyecto se configurará en el canal 1 '*Electrocardiograma (ECG), .5-35Hz*' y en el canal 2, '*Actividad Electrodermal (EDA, SS3L/SS3LA/SS57L), 0-35Hz*'.

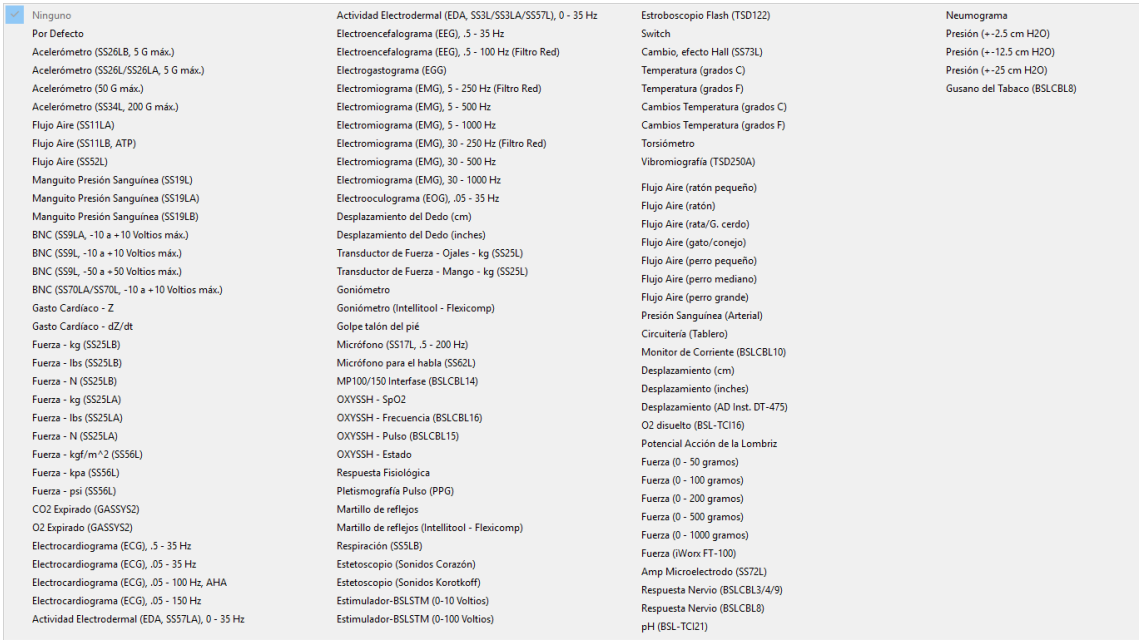


Figura 65. *Pantalla de preajuste de la configuración del Biopac SL.*

Por lo que en la Figura 66 se observa cómo queda la configuración, después de ajustar también la frecuencia de muestreo a 1.000 Hz.

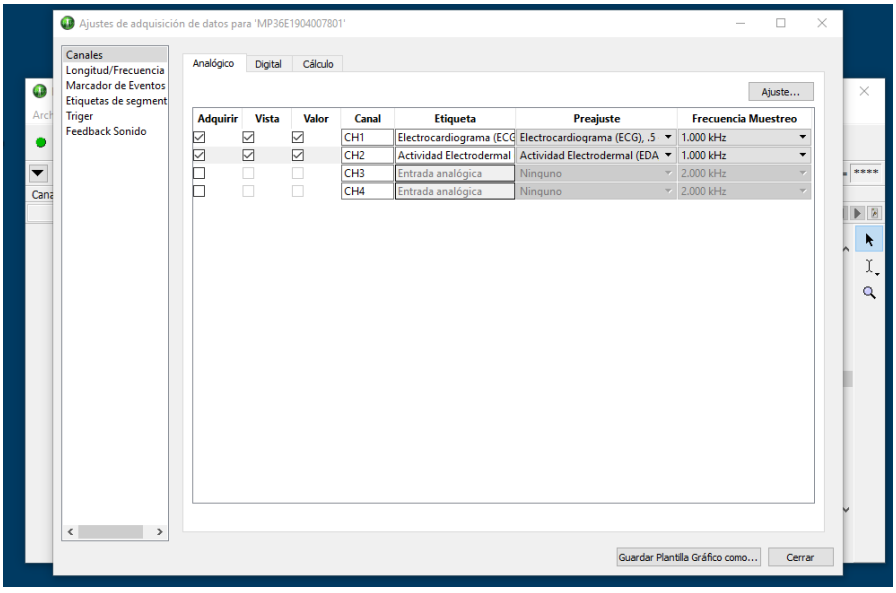


Figura 66*. Pantalla de configuración de los canales analógicos del Biopac.*

Durante este proyecto, el uso del Biopac se centra en validar el dispositivo BITalino, y como se ha mencionado, ambos dispositivos se sincronizan mediante una entrada digital. Por lo cual, en la Figura 67 se aprecia la configuración de dicha entrada, con la frecuencia de muestreo acorde a las entradas analógicas configuradas anteriormente.

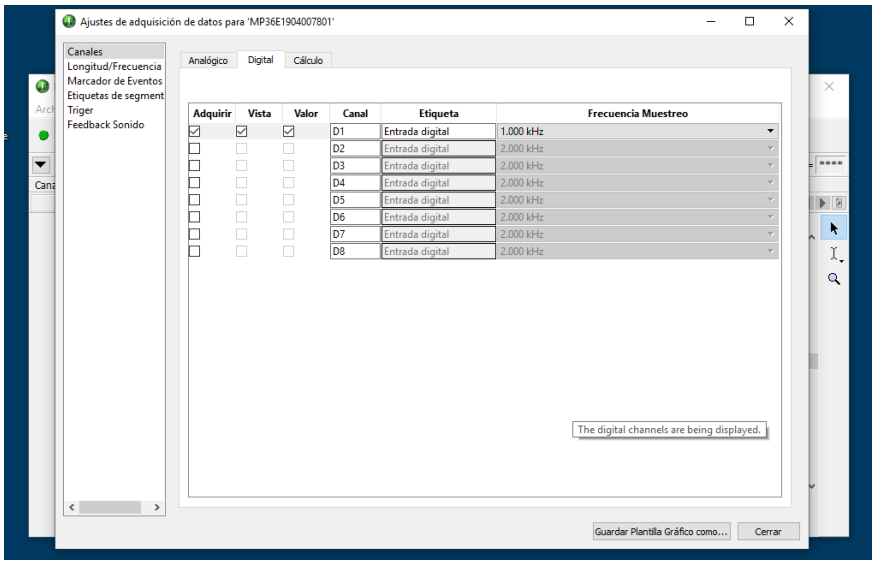


Figura 67. *Configuración de la entrada digital del Biopac.*

Además, esta plataforma da al usuario facilidad para realizar cálculos, dado que en la pestaña '*Calculo',* se puede asignar a un canal que realice la integración de esta señal o calcular el intervalo RR de la señal ECG. Todas estas opciones se observan en la Figura 68.

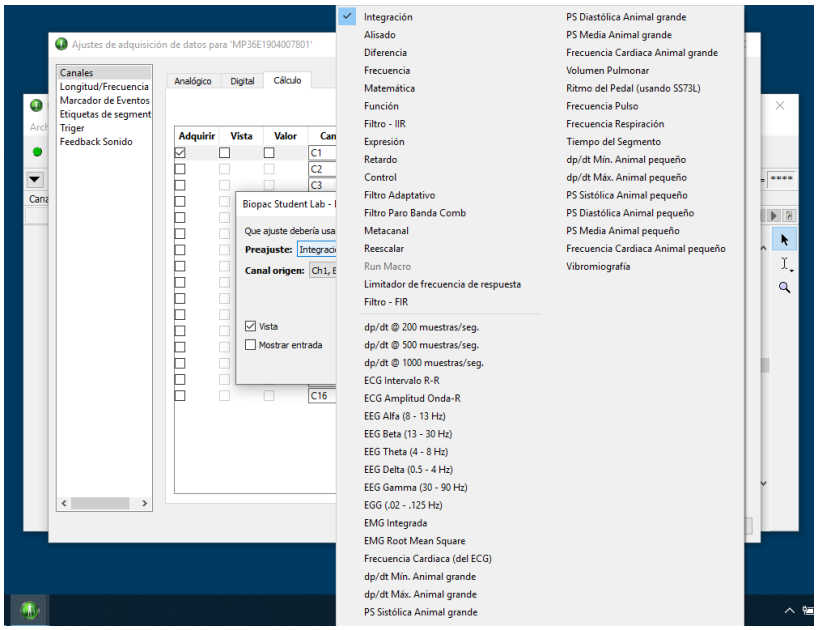


Figura 68. *Configuración de cálculos que ofrece Biopac SL.*

Una vez configurado todo, se procede a la adquisición de las señales. La pantalla principal se observa en Figura 69, y una vez se le da a *start*, comenzará con la adquisición de señales.

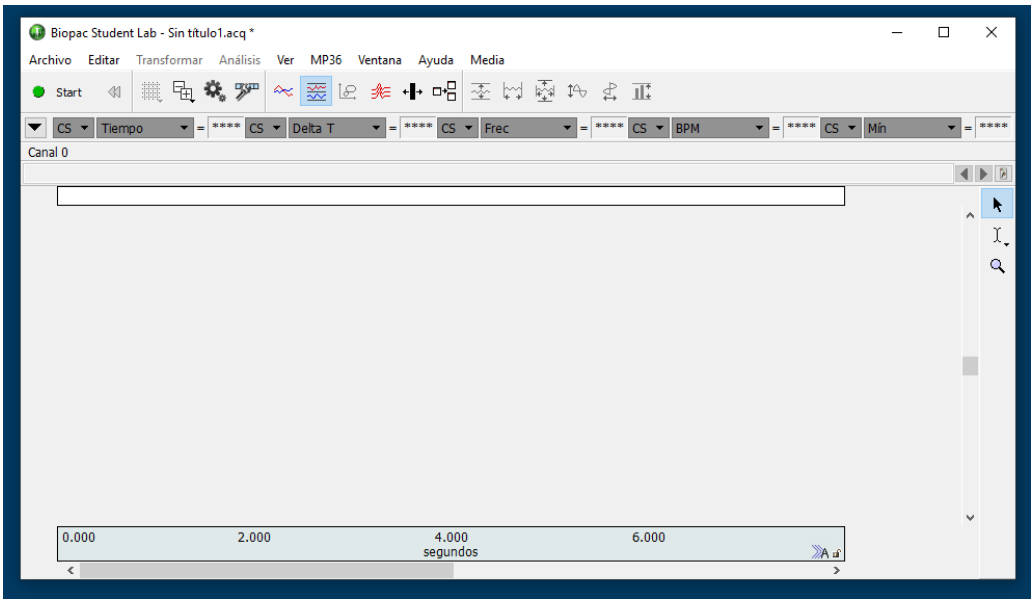


Figura 69. *Pantalla principal para la recogida de señales del Biopac SL.*

En la Figura 70 se muestra un ejemplo de un trozo de la adquisición de señales de una prueba realizada mediante Biopac. En el primer canal se observa la señal ECG, en la segunda la EDA y en la tercera se muestra la señal digital, es decir, el momento en el que se sincronizan el BITalino y el Biopac.

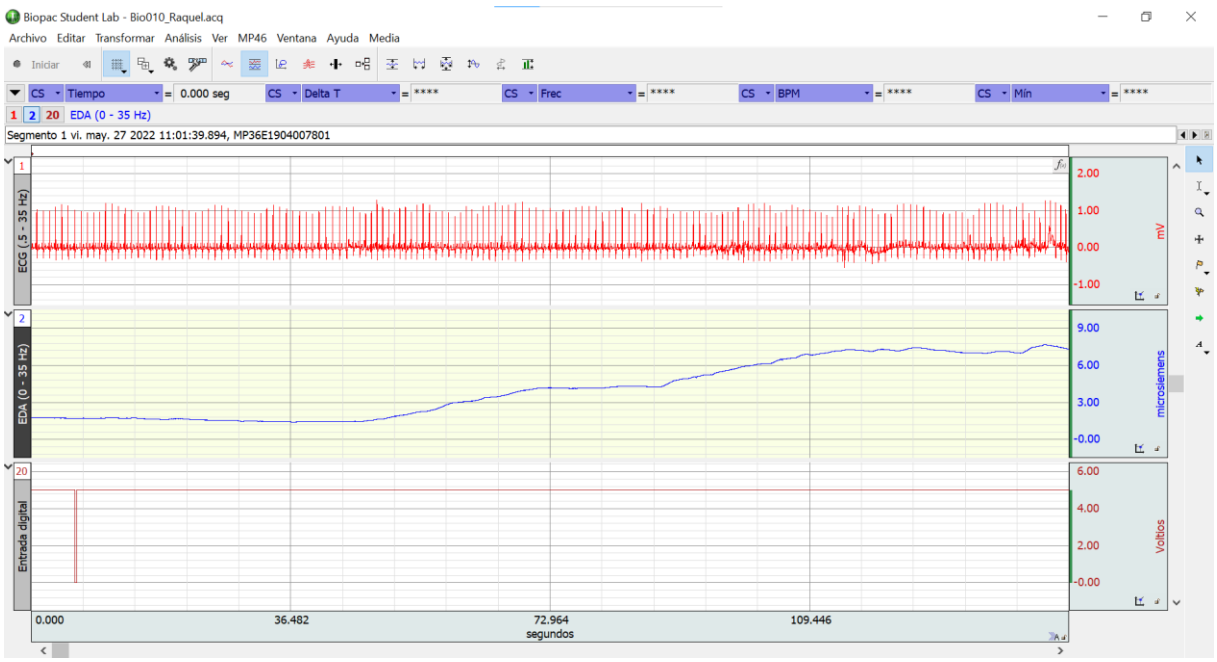


Figura 70. *Ejemplo de la adquisición de señales mediante Biopac.*

ANEXO II.

INSTALACIÓN DE PYTHON Y LAS LIBRERÍAS PERTINENTES PARA LA CONEXIÓN CON EL BITALINO

ANEXO II. INSTALACIÓN DE PYTHON Y LAS LIBRERÍAS PERTINENTES PARA LA CONEXIÓN CON EL BITALINO

En este ANEXO II se va a explicar los pasos seguidos para la configuración del BITalino en un ordenador de uso personal. Para ello, primeramente, se tiene que tener instalado el IDE Python, y, aunque actualmente existen múltiples versiones de Python, se recomienda la instalación de Python 3.9. Esto es debido a que algunas librerías pueden dar problemas dependiendo de la versión Python que se use y, en este caso, la librería para la conexión Bluetooth PyBluez, aún no está disponible para la última versión.

Para facilitar la posterior instalación de librerías con las que se va a trabajar, se recomienda agregarlo al path cuando se está instalando Python, y sino agregarlo manualmente después.

Como se acaba de mencionar, las aplicaciones en Python usualmente hacen uso de paquetes y módulos que no forman parte de la librería estándar. Las aplicaciones a veces necesitan una versión específica de una librería, debido a que dicha aplicación requiere que un bug particular haya sido solucionado o bien la aplicación ha sido escrita usando una versión obsoleta de la interfaz de la librería. Es por eso que se recomienda crear un entorno virtual donde estén todas las librerías con las que se va a trabajar, creando así un espacio de trabajo sin la necesidad de modificar el sistema.

Para la creación de entornos virtuales se recomienda utilizar el módulo *venv*, el cual viene instalado por defecto con la librería estándar de Python desde la versión 3.3. Por otro lado, la instalación de paquetes se realiza mediante la herramienta pip, la cual viene incluida por defecto a partir de Python 3.4. Por lo que se instalará el entorno virtual mediante el siguiente comando en el CMD:

```
> pip install virtualenv
```

Se creará una carpeta para el destino de los entornos que se quieran crear, y una vez situados en esa carpeta, ejecutando las siguientes líneas se creará el entorno con el nombre '*entornoBITALINO',* con la versión 3.9 de Python.

```
> cd C:\EntornosVirtuales
C:\EntornosVirtuales > virtualenv entornoBITALINO -p python39
```

Una vez creado, para activar o desactivar este entorno, se deberá ir a la ruta de la carpeta, y en los scrips ejecutar *'activate'* o '*deactivate'.*

```
> cd C:\EntornosVirtuales\entornoBITALINO\Scripts
C:\EntornosVirtuales\entornoBITALINO\Scripts > actívate #o deactivate en su caso
(entornoBITALINO) C:\EntornosVirtuales\entornoBITALINO\Scripts >
```

- BITALINO REVOLUTION PYTHON API

La API Python de BITalino (r)evolution proporciona las herramientas necesarias para interactuar con BITalino (r)evolution utilizando Python [54].

Para poder usar el paquete proporcionado por PyPI de BITalino, es necesario tener instaladas las siguientes librerías en el entorno virtual que se va a programar.

- Python3.7 +
- NumPy
- pySerial
- PyBluez (en caso de trabajar con macOS, no es necesaria)

Es la última librería de estas la que da problemas a la hora de intentar instalarla. Este módulo es el que va a permitir al código Python acceder a los recursos Bluetooth de la placa a la que nos queremos conectar.

PyBluez puede instalarse en sistemas GNU/Linux, Windows y macOS. Dependiendo del sistema que desde donde se trabaje, el entorno deberá contar con unas dependencias u otras. En este caso, se trabaja con Windows, por lo que se hará una breve explicación de como se ha conseguido instalar dicha librería en este entorno de trabajo. Además de estar trabajando con Windows 7/8/8.1/10, será necesario tener instalado Python 3.5 o más reciente (aunque con la última versión 3.10 no funciona).

PyBluez requiere un compilador C++ instalado en el sistema para construir módulos CPython [55]. Para ello, siguiendo las instrucciones, se ha instalado Visual Studio Community 2019. Este, es un completo IDE gratuito, donde, además de poder descargarse los compiladores C++ necesarios para Python, se pueden crear aplicaciones modernas para Windows, Android e iOS, además de aplicaciones web y servicios en la nube.

Las dependencias que pide PyBluez para Visual Studio son las siguientes:

- Microsoft Visual C++ 14.0 standalone: Build Tools for Visual Studio 2017 (x86, x64, ARM, ARM64)
- Microsoft Visual C++ 14.0 with Visual Studio 2017 (x86, x64, ARM, ARM64)
- Microsoft Visual C++ 14.0 standalone: Visual C++ Build Tools 2015 (x86, x64, ARM)
- Microsoft Visual C++ 14.0 with Visual Studio 2015 (x86, x64, ARM)

Además, es necesario descargar e instalar Windows 10 SDK, el cual también se instalará individualmente como componente de VS.

Por lo tanto, los componentes individuales que se deben instalar son los siguientes (cabe destacar que, al no saber seguro, para asegurar se han instalado de diferentes versiones, teniendo en cuenta que no colisionan un paquete con otro):

- Compiladores de Roslyn para C# y Visual Basic
- C# y Visual Basic
- Características principales de C++
- Herramientas de compilación de MSVC v142 – VS 2019 C++ para x64/x86 (más recientes)
- Compiladores y bibliotecas más recientes de Visual C+ + (v142) para x64/x86
- Windows 10 SDK (10.0.19041.0)

- Python 3 64-bit (3.9.7)
- Python 3 32-bit (3.9.7)
- Actualización de C++ 2019 Redistributable
- MSVC v 141 – VS 2017 C++ Build Tools para ARM (v14.16)
- MSVC v 141 – VS 2017 C++ Build Tools para ARM64 (v14.16)
- Herramientas de Desarrollo de Android en C++
- Herramientas de Desarrollo de iOS en C++
- SDK de .NT Compiler Platform
- C++ for Linux Development
- Herramientas de CMake en C++ para Linux
- MSVC v142 – VS 2019 Build Tools para ARM (v14.20)
- MSVC v142 – VS 2019 Build Tools para ARM64 (v14.20)
- MSVC v142 – VS 2019 Build Tools para x64/x86 (v14.20)
- MSVC v142 – VS 2019 Build Tools para ARM (v14.21)
- MSVC v142 – VS 2019 Build Tools para ARM64 (v14.20)
- MSVC v142 – VS 2019 Build Tools para x64/x86 (v14.21)
- MSVC v142 – VS 2019 Build Tools para x64/x86 (v14.24)
- Herramientas de compilación de MSVC v142 – VS 2019 para ARM64 (v14.27)
- Bibliotecas con mitigaciones de Spectre de MSVC v142 – VS 2019 C++ ARM (v14.27)

Una vez instalado el VS y todos sus componentes, con el siguiente comando sería suficiente para la una instalación correcta.

```
(entornoBITALINO) C:\EntornosVirtuales\entornoBITALINO\Scripts > pip install pybluez
```

Si sigue sin funcionar, se podría hacer manualmente, descargando del github el master.zip, y una vez extraído en el directorio del entorno virtual, ejecutando el script '*setup.py*'.

```
(entornoBITALINO) C:\EntornosVirtuales\entornoBITALINO\master > python setup.py install
```

Una vez descargadas estas librerías, se instala el paquete bitalino de PyPI mediante el comando pip, o clonando la librería de github.

```
(entornoBITALINO) C:\EntornosVirtuales\entornoBITALINO\ Scripts > pip install bitalino
```

Una vez aquí, el entorno virtual tendrá instalado el paquete para la adquisición de datos, y se podrá conectar el dispositivo BITalino al ordenador mediante Bluetooth, recogiendo datos en tiempo real.

ANEXO III.

CÓDIGO PYTHON DE LA APLICACIÓN

ANEXO III. CÓDIGO PYTHON DE LA APLICACIÓN

En este ANEXO III se expone el código completo de la aplicación creada para la recogida de señales con BITalino.

```
"""
Created by Asier Salazar

Aplicacion de recogida de señales fisiologicas mediante BITalino

Modificaciones: Nerea Rey

- la entrada digital se lee desde I1, y aparece en el TXT en la columna 2, después del tiempo

- solo plotea las señales que se escogen, por definición, se plotearan la señal digital, y los tres primeros
canales analogicos, siendo estos EDA, ECG y RESP (en ese orden)

- se han modificado las graficas para que se vea mejor

"""
import numpy as np
import time
import datetime as dt

from PyQt5.QtWidgets import (QApplication, QMainWindow, QPushButton, QPlainTextEdit, QLabel, QInputDialog,
                             QVBoxLayout, QWidget, QProgressBar, QDesktopWidget, QHBoxLayout, QCheckBox, QLineEdit)
from PyQt5.QtCore import (QTimer, Qt)
import sys

from bitalino import BITalino
from math import pow, ceil

#from pyqtgraph import PlotWidget, plot
import pyqtgraph as pg

import multiprocessing

from scipy import signal

from timeit import default_timer as timer



## VARIABLES GLOBALESS ###

NUMERO_CANALES = 6;
DATA_CHUNK_SIZE = 300
ONLINE_WINDOW_LENGTH = 4000
DATA = np.ndarray(shape=(DATA_CHUNK_SIZE, 9), dtype='float64')
DATOS_UTILES = np.zeros((2,NUMERO_CANALES+2), dtype='float64')  # El +1 viene de la columna de tiempo.


CH_MASK = [1, 2, 3, 4, 5, 6]    #Analog channels that have to be read
SRATE = 1000                    #Sampling rate
#MAC_ADDR = "20:17:09:18:60:03" #MAC Adress of the BITalino DEVICE
MAC_ADDR = "20:17:11:20:51:90"  #MAC Adress of the BITalino DEVICE NEREA
#MAC_ADDR = "5C:02:72:9F:50:F6" # MAC address fo the BITalino Mini Device
#MAC_ADDR = "00:07:80:F9:DD:A0"  #mac del bitalino caro

# labels = ["'nSeq'","'I1'","'I2'","'O1'","'O2'","'A1'","'A2'","'A3'","'A4'","'A5'","'A6'"]
#   EXAMPLE: A1 - channel = 5
CHANNELS = [5, 6, 7, 8] # Esta es la columna de la matriz de información leida.
IDENTS = [0, 1, 2, 3] # 0 = EMG DER, 1 = EMG IZQ, 2 = ECG, 3 = GSR

# PARAMETROS PARA EL FILTRO DE BUTTERWORTH

F_CORTE = 100  # Cut-off frequency of the filter
W = F_CORTE / (SRATE / 2) # Normalize the frequency
F_NOTCH = 50 # Frecuencia de la red
Q = 30 # Factor de calidad del filtro notch


# Coeificientes de los filtros para la senal
B, A = signal.butter(5, W, 'low')
D, C = signal.iirnotch(F_NOTCH, Q, SRATE)


POSITION = np.ndarray(shape=(1, ))
TIMESTAMP_POS = np.ndarray(shape=(1, ))
TIMESTAMP_EMG = np.ndarray(shape=(1, ))

TIEMPOS_RECOGIDA = np.ndarray(shape=(1, ))
PERMISO_RELECTURA = True
DATOS_NUEVOS_A_PLOTEAR = False

# PRUEBA
# ARR_CANALES_ESCOGIDOS = [Tiempo,'D1' 'EDA', 'ECG', 'RESP', A4, A5]
ARR_CANALES_ESCOGIDOS = [True, True, True, True, True, False, False]


```

```
class Ventana_Ploteo(QMainWindow):
    # https://stackoverflow.com/questions/40622095/pyqt5-closeevent-method
    def __init__(self):
        super(Ventana_Ploteo, self).__init__()

        screen = QDesktopWidget().screenGeometry()
        self.contenedor = QWidget()
        pos_ancho = int(screen.width()/4)
        pos_alto = int(screen.height()/8)
        self.setGeometry(pos_ancho,pos_alto,1600,950) # Tamano y posicion de la ventana (coord_x, coord_y, ancho, alto)


        self.setWindowTitle("Ploteo variables") # Extraemos el texto de la etiqueta para nombrar asi la ventana.

        # definicion de las graficas
        self.graphA1 = pg.PlotWidget()
        self.graphA1.setBackground('w')
        self.graphA1.setTitle("Senal DIGITAL I1")
        self.graphA1.setLabel(axis='bottom', text = 'Tiempo (s)')
        self.graphA2 = pg.PlotWidget()
        self.graphA2.setBackground('w')
        self.graphA2.setTitle("Senal AI-1 EDA")
        self.graphA2.setLabel(axis='left', text = 'EDA (uS)')
        self.graphA2.setLabel(axis='bottom', text = 'Tiempo (s)')
        self.graphA3 = pg.PlotWidget()
        self.graphA3.setBackground('w')
        self.graphA3.setTitle("Senal AI-2 ECG")
        self.graphA3.setLabel(axis='left', text = 'ECG (mV)')
        self.graphA3.setLabel(axis='bottom', text = 'Tiempo (s)')
        self.graphA4 = pg.PlotWidget()
        self.graphA4.setBackground('w')
        self.graphA4.setTitle("Senal AI-3 RESP" )
        self.graphA4.setLabel(axis='left', text = 'Resp (%)')
        self.graphA4.setLabel(axis='bottom', text = 'Tiempo (s)')
        self.graphA5 = pg.PlotWidget()
        self.graphA5.setBackground('w')
        self.graphA5.setTitle("Senal AI-4" )
        self.graphA5.setLabel(axis='bottom', text = 'Tiempo (s)')
        self.graphA6 = pg.PlotWidget()
        self.graphA6.setBackground('w')
        self.graphA6.setTitle("Senal AI-5" )
        self.graphA6.setLabel(axis='bottom', text = 'Tiempo (s)')


        self.layout_fila_0 = QHBoxLayout()
        self.boton_plot = QPushButton("Dibujar datos")
        self.boton_plot.clicked.connect(self.plotear_offline)
        self.layout_fila_0.addWidget(self.boton_plot)

        # primera fila : titulos SERIAL AI-1 - 2 - 3
        self.layout_fila_11 = QHBoxLayout()
        self.titulo_A1 = QLabel("Senial AI-1")
        self.titulo_A1.setAlignment(Qt.AlignLeft)
        self.layout_fila_12 = QHBoxLayout()
        self.titulo_A2 = QLabel("Senial AI-2")
        self.titulo_A2.setAlignment(Qt.AlignLeft)
        self.layout_fila_13 = QHBoxLayout()
        self.titulo_A3 = QLabel("Senial AI-3")
        self.titulo_A3.setAlignment(Qt.AlignLeft)
        self.layout_fila_11.addWidget(self.titulo_A1)
        self.layout_fila_12.addWidget(self.titulo_A2)
        self.layout_fila_13.addWidget(self.titulo_A3)

        # VAMOS A PONER LAS GRAFICAS EN HORIZONTAL

        #fila dos, los graficos 1,2,3 creados anteriormente
        self.layout_fila_21 = QHBoxLayout()
        self.layout_fila_21.addWidget(self.graphA1)
        self.layout_fila_21.addWidget(self.graphA2)

        self.layout_fila_22 = QHBoxLayout()
        self.layout_fila_22.addWidget(self.graphA3)
        self.layout_fila_22.addWidget(self.graphA4)

        self.layout_fila_23 = QHBoxLayout()
        self.layout_fila_23.addWidget(self.graphA5)
        self.layout_fila_23.addWidget(self.graphA6)

        # Creamos la combinacion de widgets para que parezca que hay una rejilla
        # de cada grafico con sus correspondientes titulos.

        self.layout_general = QVBoxLayout()
        self.layout_general.addLayout(self.layout_fila_0)
        #self.layout_general.addLayout(self.layout_fila_11)
        self.layout_general.addLayout(self.layout_fila_21)
        #self.layout_general.addLayout(self.layout_fila_12)
        self.layout_general.addLayout(self.layout_fila_22)
        #self.layout_general.addLayout(self.layout_fila_13)
        self.layout_general.addLayout(self.layout_fila_23)
        #self.layout_general.addLayout(self.layout_fila_24)
        #self.layout_general.addLayout(self.layout_fila_3)

        self.layout_general.setContentsMargins(20,20,20,20) # (izq,arriba,der,abajo)
        # Tambien se puede definir el espacio entre objetos dentro del layout: setSpacing(0)
        self.layout_general.setSpacing(20)

        self.contenedor.setLayout(self.layout_general)
        self.setCentralWidget(self.contenedor)


```

```
    def plotear_offline(self):

#         NOMBRE_ARCHIVO = "Datos_recogidos.txt"
#         datos = np.loadtxt(NOMBRE_ARCHIVO)
        self.graphA1.clear()
        self.graphA2.clear()
        self.graphA3.clear()
        self.graphA4.clear()
        self.graphA5.clear()
        self.graphA6.clear()

        datos = DATOS_UTILES
        self.eje_x = datos[:,0]

        for i in range(1, len(ARR_CANALES_ESCOGIDOS)):
            if ARR_CANALES_ESCOGIDOS[i] == True:
                if i == 1: #estamos en el grafico 1
                    self.eje_y = datos[:,1]
                    self.graphA1.plot(self.eje_x, self.eje_y)
                elif i == 2:
                    self.eje_y = datos[:,2]
                    self.graphA2.plot(self.eje_x, self.eje_y)
                elif i == 3:
                    self.eje_y = datos[:,3]
                    self.graphA3.plot(self.eje_x, self.eje_y)
                elif i == 4:
                    self.eje_y = datos[:,4]
                    self.graphA4.plot(self.eje_x, self.eje_y)
                elif i == 5:
                    self.eje_y = datos[:,5]
                    self.graphA5.plot(self.eje_x, self.eje_y)
                elif i == 6:
                    self.eje_y = datos[:,6]
                    self.graphA6.plot(self.eje_x, self.eje_y)
            else:
                pass


class Ventana_Ploteo_Online(QMainWindow):
    # Esta clase es una ventana que filtra con un paso de bajos (Fcorte=100Hz) y un notch a 50Hz
    # las senales recogidas por los sensores. Al ser una representacion online, tiene su carga computacional.
    # Por esto, el tiempo que se pierde entre muestreos cuando la visualizacion online esta activa es de
    # alrededor de 90ms, mientas que sin el ploteo online es de 10ms. Por eso, es recomendable solo usar esto
    # para la comprobacion inicial de las senales. Pero luego, conviene desactivarlo.
    def __init__(self):
        super(Ventana_Ploteo_Online, self).__init__()

        screen = QDesktopWidget().screenGeometry()
        self.contenedor = QWidget()
        pos_ancho = int(screen.width()/4)
        pos_alto = int(screen.height()/8)
        self.setGeometry(pos_ancho,pos_alto,1600,950) # Tamano y posicion de la ventana (coord_x, coord_y, ancho, alto)

        self.setWindowTitle("Ploteo online") # Extraemos el texto de la etiqueta para nombrar asi la ventana.

        self.ploteo_online_activado = False
        self.layout_fila_0 = QHBoxLayout()
        self.boton_activar_ploteo = QPushButton("Activar ploteo online")
        self.boton_activar_ploteo.clicked.connect(lambda checked: self.activar_ploteo_online(self.boton_activar_ploteo))
        self.layout_fila_0.addWidget(self.boton_activar_ploteo)

        self.graphA1 = pg.PlotWidget()
        self.graphA1.setTitle("Senal DIGITAL I1")
        self.graphA1.setLabel(axis='bottom', text = 'Tiempo (s)')
        self.graphA2 = pg.PlotWidget()
        self.graphA2.setBackground('w')
        self.graphA2.setTitle("Senal AI-1 EDA")
        self.graphA2.setLabel(axis='left', text = 'EDA (uS)')
        self.graphA2.setLabel(axis='bottom', text = 'Tiempo (s)')
        self.graphA3 = pg.PlotWidget()
        self.graphA3.setBackground('w')
        self.graphA3.setTitle("Senal AI-2 ECG")
        self.graphA3.setLabel(axis='left', text = 'ECG (mV)')
        self.graphA3.setLabel(axis='bottom', text = 'Tiempo (s)')
        self.graphA4 = pg.PlotWidget()
        self.graphA4.setBackground('w')
        self.graphA4.setTitle("Senal AI-3 RESP" )
        self.graphA4.setLabel(axis='left', text = 'Resp (%)')
        self.graphA4.setLabel(axis='bottom', text = 'Tiempo (s)')
        self.graphA5 = pg.PlotWidget()
        self.graphA5.setBackground('w')
        self.graphA5.setTitle("Senal AI-4" )
        self.graphA5.setLabel(axis='bottom', text = 'Tiempo (s)')
        self.graphA6 = pg.PlotWidget()
        self.graphA6.setBackground('w')
        self.graphA6.setTitle("Senal AI-5" )
        self.graphA6.setLabel(axis='bottom', text = 'Tiempo (s)')
```

```
274
275 ## ESTE ES EL FORMATO NUEVO DE LA REPRESENTACION POR FILAS
276          # primera fila : titulos SERIAL AI-1 - 2 - 3
277         self.layout_fila_11 = QHBoxLayout()
278         self.titulo_A1 = QLabel("Senial AI-1")
279         self.titulo_A1.setAlignment(Qt.AlignLeft)
280         self.layout_fila_12 = QHBoxLayout()
281         self.titulo_A2 = QLabel("Senial AI-2")
282         self.titulo_A2.setAlignment(Qt.AlignLeft)
283         self.layout_fila_13 = QHBoxLayout()
284         self.titulo_A3 = QLabel("Senial AI-3")
285         self.titulo_A3.setAlignment(Qt.AlignLeft)
286         self.layout_fila_11.addWidget(self.titulo_A1)
287         self.layout_fila_12.addWidget(self.titulo_A2)
288         self.layout_fila_13.addWidget(self.titulo_A3)
289
290         # VAMOS A PONER LAS GRAFICAS EN HORIZONTAL
291         #fila dos, los graficos 1,2,3 creados anteriormente
292         self.layout_fila_21 = QHBoxLayout()
293         self.layout_fila_21.addWidget(self.graphA1)
294         self.layout_fila_21.addWidget(self.graphA2)
295
296         self.layout_fila_22 = QHBoxLayout()
297         self.layout_fila_22.addWidget(self.graphA3)
298         self.layout_fila_22.addWidget(self.graphA4)
299
300         self.layout_fila_23 = QHBoxLayout()
301         self.layout_fila_23.addWidget(self.graphA5)
302         self.layout_fila_23.addWidget(self.graphA6)
303
304         # Creamos la combinacion de widgets para que parezca que hay una rejilla
305         # de cada grafico con sus correspondientes titulos.
306
307         self.layout_general = QVBoxLayout()
308         self.layout_general.addLayout(self.layout_fila_0)
309         self.layout_general.addLayout(self.layout_fila_21)
310         self.layout_general.addLayout(self.layout_fila_22)
311         self.layout_general.addLayout(self.layout_fila_23)
312
313         self.layout_general.setContentsMargins(20,20,20,20) # (izq,arriba,der,abajo)
314         # Tambien se puede definir el espacio entre objetos dentro del layout: setSpacing(0)
315         self.layout_general.setSpacing(20)
316
317         self.contenedor.setLayout(self.layout_general)
318         self.setCentralWidget(self.contenedor)
319
320
321         self.eje_x = DATOS_UTILES[:,0]
322         self.eje_y_A1 = DATOS_UTILES[:,1]
323         self.linea_A1 = self.graphA1.plot(self.eje_x, self.eje_y_A1)
324
325         self.eje_y_A2 = DATOS_UTILES[:,2]
326         self.linea_A2 = self.graphA2.plot(self.eje_x, self.eje_y_A2)
327
328         self.eje_y_A3 = DATOS_UTILES[:,3]
329         self.linea_A3 = self.graphA3.plot(self.eje_x, self.eje_y_A3)
330
331         self.eje_y_A4 = DATOS_UTILES[:,4]
332         self.linea_A4 = self.graphA4.plot(self.eje_x, self.eje_y_A4)
333
334         self.eje_y_A5 = DATOS_UTILES[:,5]
335         self.linea_A5 = self.graphA5.plot(self.eje_x, self.eje_y_A5)
336
337         self.eje_y_A6 = DATOS_UTILES[:,6]
338         self.linea_A6 = self.graphA6.plot(self.eje_x, self.eje_y_A6)
339         self.buffer_lleno = False
340
341         self.eje_x = self.eje_x[len(self.eje_x):]
342         self.eje_y_A1 = self.eje_x[len(self.eje_y_A1):]
343         self.eje_y_A2 = self.eje_x[len(self.eje_y_A2):]
344         self.eje_y_A3 = self.eje_x[len(self.eje_y_A3):]
345         self.eje_y_A4 = self.eje_x[len(self.eje_y_A4):]
346         self.eje_y_A5 = self.eje_x[len(self.eje_y_A5):]
347         self.eje_y_A5 = self.eje_x[len(self.eje_y_A6):]
348         self.linea_A1.setData(self.eje_x, self.eje_y_A1)  # Update the data.
349         self.linea_A2.setData(self.eje_x, self.eje_y_A2)  # Update the data.
350         self.linea_A3.setData(self.eje_x, self.eje_y_A3)  # Update the data.
351         self.linea_A4.setData(self.eje_x, self.eje_y_A4)  # Update the data.
352         self.linea_A5.setData(self.eje_x, self.eje_y_A5)  # Update the data.
353         self.linea_A6.setData(self.eje_x, self.eje_y_A6)  # Update the data.
354
355         self.timer_online = QTimer()
356         self.timer_online.setInterval(20) # Con 4 senales 15 es un buen numero.
357         self.timer_online.timeout.connect(self.update_plot_data)
358         self.timer_online.start()
359
360     def activar_ploteo_online(self, boton_a_cambiar_texto):
361         if self.ploteo_online_activado == True:
362             self.ploteo_online_activado = False
363             boton_a_cambiar_texto.setText("Activar ploteo online")
364             print ("Ploteo online activo: ", self.ploteo_online_activado)
365
366         else:
367             self.ploteo_online_activado = True
368             boton_a_cambiar_texto.setText("Desactivar ploteo online")
369             print ("Ploteo online activo: ", self.ploteo_online_activado)
370
371
```

```
    def update_plot_data(self):
        global DATOS_NUEVOS_A_PLOTEAR

        if (self.ploteo_online_activado == True) and (DATOS_NUEVOS_A_PLOTEAR == True):

    # Para saber la dimension de una matriz 'a' tenemos la funcion a.shape
    # Esta funcion devuelve asi: (num_filas,num_columas). Si hacemos a.shape[0]
    # obtenemos el numero de filas. Si hacemos a.shape[1] obtenemos el numero de col.
    # Si solo queremos el numero de filas tambien podemos usar len(a), ya que len da el
    # numero de filas (pero no el de columnas).

            if ((len(DATOS_UTILES) > ONLINE_WINDOW_LENGTH) and (self.buffer_lleno == False)):
    # La ventana no mostrara nada hasta que haya como minimo tantas muestras como para llenar la ventana.
                punto_inicio = len(DATOS_UTILES)-1-ONLINE_WINDOW_LENGTH
                punto_final = len(DATOS_UTILES)-1
                self.eje_x = DATOS_UTILES[punto_inicio:punto_final,0]
                self.eje_y_A1 = DATOS_UTILES[punto_inicio:punto_final,1]
                self.linea_A1 = self.graphA1.plot(self.eje_x, self.eje_y_A1)

                self.eje_y_A2 = DATOS_UTILES[punto_inicio:punto_final,2]
                self.linea_A2 = self.graphA2.plot(self.eje_x, self.eje_y_A2)

                self.eje_y_A3 = DATOS_UTILES[punto_inicio:punto_final,3]
                self.linea_A3 = self.graphA3.plot(self.eje_x, self.eje_y_A3)

                self.eje_y_A4 = DATOS_UTILES[punto_inicio:punto_final,4]
                self.linea_A4 = self.graphA4.plot(self.eje_x, self.eje_y_A4)

                self.eje_y_A5 = DATOS_UTILES[punto_inicio:punto_final,5]
                self.linea_A5 = self.graphA5.plot(self.eje_x, self.eje_y_A5)

                self.eje_y_A6 = DATOS_UTILES[punto_inicio:punto_final,6]
                self.linea_A6 = self.graphA6.plot(self.eje_x, self.eje_y_A6)
#                print "Entra en la opcion 0"
                self.buffer_lleno = True


            if self.buffer_lleno == True:

    # Filtramos y actualizamos los graficos con la informacion actualizada en el anterior IF.
    # Los coeficientes de los filtros de han dejado como variables constantes y estan al inicio del
    # codigo.

                datos_nuevos_filt_A1 = signal.filtfilt(B, A, DATOS_UTILES[len(DATOS_UTILES)-1-DATA_CHUNK_SIZE:len(DATOS_UTILES)-1,1])
                datos_nuevos_filt_A2 = signal.filtfilt(B, A, DATOS_UTILES[len(DATOS_UTILES)-1-DATA_CHUNK_SIZE:len(DATOS_UTILES)-1,2])
                datos_nuevos_filt_A3 = signal.filtfilt(B, A, DATOS_UTILES[len(DATOS_UTILES)-1-DATA_CHUNK_SIZE:len(DATOS_UTILES)-1,3])
                datos_nuevos_filt_A4 = signal.filtfilt(B, A, DATOS_UTILES[len(DATOS_UTILES)-1-DATA_CHUNK_SIZE:len(DATOS_UTILES)-1,4])
                datos_nuevos_filt_A5 = signal.filtfilt(B, A, DATOS_UTILES[len(DATOS_UTILES)-1-DATA_CHUNK_SIZE:len(DATOS_UTILES)-1,5])
                datos_nuevos_filt_A6 = signal.filtfilt(B, A, DATOS_UTILES[len(DATOS_UTILES)-1-DATA_CHUNK_SIZE:len(DATOS_UTILES)-1,6])

                datos_nuevos_filt_A1 = signal.filtfilt(D, C, datos_nuevos_filt_A1)
                datos_nuevos_filt_A2 = signal.filtfilt(D, C, datos_nuevos_filt_A2)
                datos_nuevos_filt_A3 = signal.filtfilt(D, C, datos_nuevos_filt_A3)
                datos_nuevos_filt_A4 = signal.filtfilt(D, C, datos_nuevos_filt_A4)
                datos_nuevos_filt_A5 = signal.filtfilt(D, C, datos_nuevos_filt_A5)
                datos_nuevos_filt_A6 = signal.filtfilt(D, C, datos_nuevos_filt_A6)

                self.eje_x = self.eje_x[DATA_CHUNK_SIZE:] # Elimina las primeras n filas
                self.eje_x = np.concatenate((self.eje_x,DATOS_UTILES[len(DATOS_UTILES)-1-DATA_CHUNK_SIZE:len(DATOS_UTILES)-1,0]),0)

                self.eje_y_A1 = self.eje_y_A1[DATA_CHUNK_SIZE:] # Elimina las primeras n filas
                self.eje_y_A1 = np.concatenate((self.eje_y_A1,datos_nuevos_filt_A1),0)
                self.eje_y_A2 = self.eje_y_A2[DATA_CHUNK_SIZE:] # Elimina las primeras n filas
                self.eje_y_A2 = np.concatenate((self.eje_y_A2,datos_nuevos_filt_A2),0)
                self.eje_y_A3 = self.eje_y_A3[DATA_CHUNK_SIZE:] # Elimina las primeras n filas
                self.eje_y_A3 = np.concatenate((self.eje_y_A3,datos_nuevos_filt_A3),0)
                self.eje_y_A4 = self.eje_y_A4[DATA_CHUNK_SIZE:] # Elimina las primeras n filas
                self.eje_y_A4 = np.concatenate((self.eje_y_A4,datos_nuevos_filt_A4),0)
                self.eje_y_A5 = self.eje_y_A5[DATA_CHUNK_SIZE:] # Elimina las primeras n filas
                self.eje_y_A5 = np.concatenate((self.eje_y_A5,datos_nuevos_filt_A5),0)
                self.eje_y_A6 = self.eje_y_A6[DATA_CHUNK_SIZE:] # Elimina las primeras n filas
                self.eje_y_A6 = np.concatenate((self.eje_y_A6,datos_nuevos_filt_A6),0)

            for i in range(1, len(ARR_CANALES_ESCOGIDOS)):
                    if ARR_CANALES_ESCOGIDOS[i] == True:
                        if i == 1: #estamos en el grafico 1
                            self.linea_A1.setData(self.eje_x, self.eje_y_A1)  # Update the data.
                        elif i == 2:
                            self.linea_A2.setData(self.eje_x, self.eje_y_A2)  # Update the data.
                        elif i == 3:
                            self.linea_A3.setData(self.eje_x, self.eje_y_A3)  # Update the data.
                        elif i == 4:
                            self.linea_A4.setData(self.eje_x, self.eje_y_A4)  # Update the data.
                        elif i == 5:
                            self.linea_A5.setData(self.eje_x, self.eje_y_A5)  # Update the data.
                        elif i == 6:
                            self.linea_A6.setData(self.eje_x, self.eje_y_A6)  # Update the data.
                    else:
                        pass

            DATOS_NUEVOS_A_PLOTEAR = False

        else:
            pass
```

```
class Ventana_Config(QMainWindow):
    # https://stackoverflow.com/questions/40622095/pyqt5-closeevent-method
    def __init__(self):
        super(Ventana_Config, self).__init__()

        screen = QDesktopWidget().screenGeometry()
        self.contenedor = QWidget()
        pos_ancho = int(screen.width()/8)
        pos_alto = int(screen.height()/8)
        self.setGeometry(pos_ancho,pos_alto,200,400) # Tamano y posicion de la ventana (coord_x, coord_y, ancho, alto)

        self.setWindowTitle("Configuracion") # Extraemos el texto de la etiqueta para nombrar asi la ventana.

        # casilla D1 activado
        self.layoutconfig_fila_1 = QHBoxLayout() # Layout para primera fila de objetos.
        self.tick_A1 = QCheckBox('Canal I1, lectura senal digital')
        self.tick_A1.setChecked(True) #en principio tenemos todos activados
        self.layoutconfig_fila_1.addWidget(self.tick_A1)

        # casilla A1 activado
        self.layoutconfig_fila_2 = QHBoxLayout() # Layout para primera fila de objetos.
        self.tick_A2 = QCheckBox('Canal A1, lectura EDA')
        self.tick_A2.setChecked(True) #en principio tenemos todos activados
        self.layoutconfig_fila_2.addWidget(self.tick_A2)

        # casilla A2 activado
        self.layoutconfig_fila_3 = QHBoxLayout() # Layout para primera fila de objetos.
        self.tick_A3 = QCheckBox('Canal A2, lectura ECG')
        self.tick_A3.setChecked(True) #en principio tenemos todos activados
        self.layoutconfig_fila_3.addWidget(self.tick_A3)

        # casilla A3 activado
        self.layoutconfig_fila_11 = QHBoxLayout() # Layout para primera fila de objetos.
        self.tick_A11 = QCheckBox('Canal A3, lectura RESP')
        self.tick_A11.setChecked(True) #en principio tenemos todos activados
        self.layoutconfig_fila_11.addWidget(self.tick_A11)

        # casilla A4 desactivado
        self.layoutconfig_fila_22 = QHBoxLayout() # Layout para primera fila de objetos.
        self.tick_A22 = QCheckBox('Canal A4')
        self.tick_A22.setChecked(False) #en principio tenemos todos activados
        self.layoutconfig_fila_22.addWidget(self.tick_A22)

        # casilla A5 desactivado
        self.layoutconfig_fila_33 = QHBoxLayout() # Layout para primera fila de objetos.
        self.tick_A33 = QCheckBox('Canal A5')
        self.tick_A33.setChecked(False) #en principio tenemos todos activados
        self.layoutconfig_fila_33.addWidget(self.tick_A33)

        # agregar accion a los checkbox
        self.tick_A1.stateChanged.connect(lambda:self.btnstate(self.tick_A1)) # Digital
        self.tick_A2.stateChanged.connect(lambda:self.btnstate(self.tick_A2)) # EDA
        self.tick_A3.stateChanged.connect(lambda:self.btnstate(self.tick_A3)) # ECG
        self.tick_A11.stateChanged.connect(lambda:self.btnstate(self.tick_A11)) # Resp
        self.tick_A22.stateChanged.connect(lambda:self.btnstate(self.tick_A22)) # --
        self.tick_A33.stateChanged.connect(lambda:self.btnstate(self.tick_A33)) # --

        self.layout_fila_7 = QHBoxLayout() # Layout para primera fila de objetos.
        self.dir_MAC = QLabel("Direccion MAC Bitalino:")
        self.text = QLineEdit()
        self.text.setMaxLength(17)
        self.text.setMinimumWidth(150)
        self.text.setMaximumWidth(150)
        self.text.setReadOnly(False)
        self.text.setText(MAC_ADDR)

        self.layout_fila_7.addWidget(self.dir_MAC)
        self.layout_fila_7.addWidget(self.text)

        self.layout_fila_8 = QHBoxLayout() # Layout para primera fila de objetos.
        self.boton_actualizar_config = QPushButton("Actualizar configuracion")
        self.boton_actualizar_config.clicked.connect(self.actualizar_config)
        self.layout_fila_8.addWidget(self.boton_actualizar_config)

        l = QVBoxLayout() # Layout para las opciones de configuracion.

        l.addLayout(self.layoutconfig_fila_1)
        l.addLayout(self.layoutconfig_fila_2)
        l.addLayout(self.layoutconfig_fila_3)
        l.addLayout(self.layoutconfig_fila_11)
        l.addLayout(self.layoutconfig_fila_22)
        l.addLayout(self.layoutconfig_fila_33)
        l.addLayout(self.layout_fila_7)
        l.addLayout(self.layout_fila_8)

        w = QWidget()
        w.setLayout(l)
        self.setCentralWidget(w)
```

```
    def btnstate(self, A):
      global ARR_CANALES_ESCOGIDOS
      if A.text() == "Canal I1, lectura senal digital":
         if A.isChecked() == True:
            print (A.text()+" is selected")
            ARR_CANALES_ESCOGIDOS[1] = True
         else:
            print (A.text()+" is deselected")
            ARR_CANALES_ESCOGIDOS[1] = False
      if A.text() == "Canal A1, lectura EDA":
         if A.isChecked() == True:
            print (A.text()+" is selected")
            ARR_CANALES_ESCOGIDOS[2] = True
         else:
            print (A.text()+" is deselected")
            ARR_CANALES_ESCOGIDOS[2] = False

      if A.text() == "Canal A2, lectura ECG":
         if A.isChecked() == True:
            print (A.text()+" is selected")
            ARR_CANALES_ESCOGIDOS[3] = True
         else:
            print (A.text()+" is deselected")
            ARR_CANALES_ESCOGIDOS[3] = False

      if A.text() == "Canal A3, lectura RESP":
         if A.isChecked() == True:
            print (A.text()+" is selected")
            ARR_CANALES_ESCOGIDOS[4] = True
         else:
            print (A.text()+" is deselected")
            ARR_CANALES_ESCOGIDOS[4] = False
            # print("Caneles escogidos: ", ARR_CANALES_ESCOGIDOS)
      if A.text() == "Canal A4":
         if A.isChecked() == True:
            print (A.text()+" is selected")
            ARR_CANALES_ESCOGIDOS[5] = True
         else:
            print (A.text()+" is deselected")
            ARR_CANALES_ESCOGIDOS[5] = False
            # print("Caneles escogidos: ", ARR_CANALES_ESCOGIDOS)
      if A.text() == "Canal A5":
         if A.isChecked() == True:
            print (A.text()+" is selected")
            ARR_CANALES_ESCOGIDOS[6] = True
         else:
            print (A.text()+" is deselected")
            ARR_CANALES_ESCOGIDOS[6] = False
            # print("Caneles escogidos: ", ARR_CANALES_ESCOGIDOS)

    def actualizar_config(self):
        global MAC_ADDR
        MAC_ADDR = self.text.text()
        print("Direccion MAC actualizada a: ", MAC_ADDR)


class MainWindow(QMainWindow):

    def __init__(self):
        super().__init__()

        global TIEMPOS RECOGIDA

        screen = QDesktopWidget().screenGeometry()
        pos_ancho = int(screen.width()/3)
        pos_alto = int(screen.height()/3)
        self.setGeometry(pos_ancho,pos_alto,450,500) # Tamaño y posicion de la ventana (coord_x, coord_y, ancho, alto)


        self.bitalino_activo = False
        self.flag_mensaje_capturando = True

        self.LECTURA = np.zeros((DATA_CHUNK_SIZE,5), dtype='float64')

        # CREAMOS LAS VENTANAS DE PLOTEO:
        self.ventana_nueva_1 = Ventana_Ploteo()
        self.ventana_nueva_2 = Ventana_Ploteo_Online()

        # CREAMOS LA VENTANA DE CONFIGURACION:
        self.ventana_nueva_3 = Ventana_Config()

        self.boton_ploteo = QPushButton("Abrir ventana de ploteo")
        self.boton_ploteo.clicked.connect(self.abrir_ventana_ploteo)

        self.boton_ploteo_online = QPushButton("Abrir ventana de ploteo online")
        self.boton_ploteo_online.clicked.connect(self.abrir_ventana_ploteo_online)

        self.btn_sincro = QPushButton("Sincronizar Bitalino")
        self.btn_sincro.clicked.connect(lambda checked: self.sincronizar_bitalino(self.btn_sincro))

        self.btn_activar_desactivar = QPushButton("Activar/desactivar Bitalino")
        self.btn_activar_desactivar.clicked.connect(lambda checked: self.activar_desactivar_bitalino(self.btn_activar_desactivar))

        self.btn_captura = QPushButton("Lanzar captura datos")
        self.btn_captura.clicked.connect(lambda checked: self.boton_ejecutar_pulsado(self.btn_captura))

        self.btn_volcado = QPushButton("Volcar datos a txt")
        self.btn_volcado.clicked.connect(self.volcar_datos)

        self.btn_configuracion = QPushButton("Configuracion")
        self.btn_configuracion.clicked.connect(self.abrir_ventana_config)

        self.text = QPlainTextEdit()
        self.text.setReadOnly(True)
```

```
        h = QHBoxLayout()

        self.btn_clear_text = QPushButton("Limpiar texto")
        self.btn_clear_text.clicked.connect(self.limpiar_texto)

        self.btn_reset_vars = QPushButton("Resetear variables")
        self.btn_reset_vars.clicked.connect(self.resetear_variables)

        h.addWidget(self.btn_clear_text)
        h.addWidget(self.btn_reset_vars)

        l = QVBoxLayout()
        l.addWidget(self.btn_sincro)
        l.addWidget(self.btn_activar_desactivar)
        l.addWidget(self.btn_captura)
        l.addWidget(self.btn_volcado)
        l.addWidget(self.boton_ploteo)
        l.addWidget(self.boton_ploteo_online)
        l.addWidget(self.btn_configuracion)
        l.addWidget(self.text)
        l.addLayout(h)

        w = QWidget()
        w.setLayout(l)

        self.sincronizado = False
        self.captura_activa = False

        self.setCentralWidget(w)

        self.timer = QTimer()
        self.timer.setInterval(20)
        self.timer.timeout.connect(self.inicio_lectura_datos)
        self.timer.start()

    def limpiar_texto(self):
        self.text.clear()

    def resetear_variables(self):
        global DATOS_UTILES, TIEMPOS_RECOGIDA
        DATOS_UTILES = np.zeros((2,NUMERO_CANALES+2), dtype='float64')
        TIEMPOS_RECOGIDA = timer()
        texto = 'Variables reseteadas'
        self.text.appendPlainText(texto)


    def sincronizar_bitalino(self, boton_a_cambiar_texto):
        global CH_MASK
        if self.sincronizado == False:
            # Starting PC - BITalino communication
            CH_MASK = [1, 2, 3, 4, 5, 6]
            CH_MASK = np.array(CH_MASK) - 1 # Resta 1 para que encaje con la cuenta
            self.DEVICE = BITalino(MAC_ADDR)     # desde 0 de bitalino. 0 = Analog 1.
            DATOS = self.DEVICE.state()
            while bool(DATOS) == False: # Signifca que mientras DATOS este vacío.
                DATOS = self.DEVICE.state()
                time.sleep(100)
#            print("DATOS: ",DATOS)
            boton_a_cambiar_texto.setText("Desincronizar bluetooth.")
            self.text.clear()
            texto = 'The MAC address is: ' + str(MAC_ADDR)
            self.text.appendPlainText(texto)
            texto = 'The sampling rate is: ' + str(SRATE) + 'Hz'
            self.text.appendPlainText(texto)
            texto = 'The battery is at ' + str(round(2*DATOS['battery']*3.3/(pow(2, 10)-1), 3)) + ' Volts within the range [3.4-3.8].'
            self.text.appendPlainText(texto)
            del DATOS
            self.sincronizado = True
            self.text.appendPlainText("Dispositivo sincronizado: True")
        else:
            boton_a_cambiar_texto.setText("Sincronizar bluetooth.")
            self.DEVICE.close()
            self.sincronizado = False
            self.text.clear()
            self.text.appendPlainText("Conexión bluetooth cortada.")


    def activar_desactivar_bitalino(self, boton_a_cambiar_texto):
        if (self.sincronizado == True):
            if self.bitalino_activo == True:
                self.bitalino_activo = False
                self.DEVICE.stop()
                boton_a_cambiar_texto.setText("Activar bitalino")
                self.text.appendPlainText("BITalino desactivado")

            else:
                self.bitalino_activo = True
                self.DEVICE.start(SRATE, CH_MASK)
                boton_a_cambiar_texto.setText("Desactivar bitalino")
                self.text.appendPlainText("BITalino activado")
        else:
            self.text.appendPlainText("Error: el bitalino no está sincronizado.")
```

```
753
754     def boton_ejecutar_pulsado(self, boton_a_cambiar_texto):
755         if (self.sincronizado == True) and (self.bitalino_activo == True):
756             if self.captura_activa == False:
757                 self.captura_activa = True
758                 boton_a_cambiar_texto.setText("Parar captura de datos")
759                 self.text.appendPlainText("Inicio de la captura de señales.")
760             else:
761                 self.captura_activa = False
762                 boton_a_cambiar_texto.setText("Lanzar captura datos")
763                 self.text.appendPlainText("Fin de la captura de señales.")
764                 self.flag_mensaje_capturando = False
765         else:
766             if self.sincronizado == False:
767                 self.text.appendPlainText("Error: el bitalino no está sincronizado.")
768             if self.captura_activa == False:
769                 self.text.appendPlainText("Error: el bitalino no está activado.")
770
771
772     def inicio_lectura_datos(self):
773         global PERMISO_RELECTURA, TIEMPOS_RECOGIDA, DATOS_UTILES, DATOS_NUEVOS_A_PLOTEAR
774
775         if (self.sincronizado == True) and (self.captura_activa == True) and (PERMISO_RELECTURA == True):
776             if self.bitalino_activo == True:
777
778                 if self.flag_mensaje_capturando == False:
779                     self.text.clear()
780                     self.text.appendPlainText("Capturando senales.")
781                     self.flag_mensaje_capturando = True
782
783                 DATOS_RECOGIDOS = np.zeros((DATA_CHUNK_SIZE,NUMERO_CANALES+2), dtype='float64')
784
785                 now = timer()
786                 TIEMPOS_RECOGIDA = np.append(TIEMPOS_RECOGIDA, now)
787                 tmp_ini_lectura = now-TIEMPOS_RECOGIDA[1]
788
789                 if len(TIEMPOS_RECOGIDA)>4:
790                     print ("T perd: ", TIEMPOS_RECOGIDA[len(TIEMPOS_RECOGIDA)-1]-TIEMPOS_RECOGIDA[len(TIEMPOS_RECOGIDA)-2])
791
792                 self.LecturaDatos(self.DEVICE, DATOS_RECOGIDOS)
793
794                 now = timer()
795                 TIEMPOS_RECOGIDA = np.append(TIEMPOS_RECOGIDA, now)
796                 tmp_fin_lectura = now-TIEMPOS_RECOGIDA[1]
797                 vector_tiempo = np.linspace(tmp_ini_lectura, tmp_fin_lectura, DATA_CHUNK_SIZE)
798                 PERMISO_RELECTURA = True
799
800                 DATOS_RECOGIDOS[:,0] = vector_tiempo
801                 DATOS_UTILES = np.concatenate((DATOS_UTILES, DATOS_RECOGIDOS),axis=0)
802                 DATOS_NUEVOS_A_PLOTEAR = True
803
804         else:
805             pass
806
807     def LecturaDatos(self, DEVICE, DATOS_RECOGIDOS):
808         # Start acquisition and reading 300 samples:
809         global PERMISO_RELECTURA
810
811         PERMISO_RELECTURA = False
812
813         DATA = DEVICE.read(DATA_CHUNK_SIZE)
814         DATOS_RECOGIDOS[:,2:NUMERO_CANALES+2] = DATA[:,5:5+NUMERO_CANALES]  ## IMPORTANTISIMO QUE SE COPIE ASI
815         DATOS_RECOGIDOS[:,1] = DATA[:,1]
816
817
818     def abrir_ventana_ploteo(self):
819         if self.ventana_nueva_1.isVisible():
820             self.ventana_nueva_1.hide()
821         else:
822             self.ventana_nueva_1.show()
823
824     def abrir_ventana_ploteo_online(self):
825         if self.ventana_nueva_2.isVisible():
826             self.ventana_nueva_2.hide()
827         else:
828             self.ventana_nueva_2.show()
829
830     def abrir_ventana_config(self):
831         if self.ventana_nueva_3.isVisible():
832             self.ventana_nueva_3.hide()
833         else:
834             self.ventana_nueva_3.show()
835
836     def volcar_datos(self):
837         global DATOS_UTILES, NOMBRE_ARCHIVO
838
839         # Borramos las primeras dos filas que estan vacias y era para inicializar.
840         DATOS_UTILES = np.delete(DATOS_UTILES,0,0)
841         DATOS_UTILES = np.delete(DATOS_UTILES,0,0)
842
```

```
843         text, ok = QInputDialog.getText(self, 'Seleccion nombre archivo', '¿Con qué nombre guardarás el archivo?')
844 
845         if bool(text) == True and ok == True:
846             NOMBRE_ARCHIVO = text + ".txt"
847             f = open(NOMBRE_ARCHIVO, "a")
848             np.savetxt(f, DATOS_UTILES)
849             f.close()
850             self.text.appendPlainText("Datos volcados a txt correctamente.")
851         else:
852             if ok == False:
853                 self.text.appendPlainText("Guardado cancelado por el usuario.")
854             else:
855                 self.text.appendPlainText("Error: es necesario elegir un nombre para el archivo.")
856 
857 
858 if __name__ == '__main__':
859 
860     TIEMPOS_RECOGIDA = timer()
861     app = QApplication(sys.argv)
862     w = MainWindow()
863     w.show()
864 
865     app.exec_()
866     sys.exit("Programa finalizado")
```

ANEXO IV.

DOCUMENTO DE CONSENTIMIENTO PARA LAS PRUEBAS DEL PROYECTO

ANEXO IV. DOCUMENTO DE CONSENTIMIENTO PARA LAS PRUEBAS DEL PROYECTO

DOCUMENTO CONSENTIMIENTO_ BioBit001

Investigador Principal: Javier Muguerza Rivero

Contacto: j.muguerza@ehu.eus, Tlfn:943018033 Facultad de Informática, UPV/EHU

CÓDIGO: M10/2017/131 (NoRefCEID)

El proyecto para el que solicito tu permiso se titula "Comparativa de las bioseñales registradas por los dispositivos Biopac y Bitalino" y su objetivo es el de diseñar un dispositivo portable con la placa Bitalino que permita evaluación el estado del Sistema Nervioso Autónomo en relación al estrés y que pueda ser utilizado en diferentes escenario como es el caso de: personas de la tercera edad, enfermos crónicos, personas con determinada discapacidad intelectual o motora, etc. Para ello se pretende desarrollar un sistema capaz de interpretar las señales fisiológicas con el objeto de mejorar la calidad de vida los colectivos anteriormente nombrados en varios frentes: permitir el uso de dispositivos mediante el pensamiento, guiarles en sus tareas cotidianas, mejorar la comunicación de los usuarios con las personas de su entorno, etc.

En la presente intervención se investiga la generación de la señal electrodérmica de la piel. Con tu participación queremos ver si la señal registrada con el dispositivo Bitalino tiene una correlación elevada con la señal recogida con el Biopac (dispositivo validado científicamente)

¿En qué consiste tu participación?:

Te sentarás en una silla, verás un video de relajación y posteriormente se evaluará tu capacidad de cálculo numérico, todo ello en un tiempo máximo de 20 minutos. Durante la prueba, que no conlleva ningún riesgo, se recogerán la respuesta electrodérmica de la piel relacionada con el sistema nervioso autónomo.,

Tendrás que venir al laboratorio de investigación del grupo Aldapa, situado en la tercera planta de la Facultad de Informática de la UPV/EHU en San Sebastián, y serán los investigadores Javier Muguerza quienes te guiarán en el transcurso de la prueba.

Tu participación es voluntaria y no recibirás remuneración por ella, si no quieres participar no va a haber ningún tipo de perjuicio para ti. Los datos que se registren podrán ser almacenados de forma anónima en una colección de muestras para investigaciones relacionadas con la inicialmente propuesta.

Las pruebas que se van a realizar no son diagnósticas, además, ningún miembro del equipo tiene formación médica, por lo que, si el participante tuviera alguna dolencia cardíaca, no se podrá informar sobre ello.

Te podrás retirar de la investigación cuando lo creas oportuno, para ello solo deberás ponerte en contacto bien con Javier Muguerza (j.muguerza@ehu.eus) o con Ibai Gurrutxaga (i.gurrutxaga@ehu.eus).

Si quieres conocer los resultados obtenidos tras las investigaciones llevadas a cabo con los datos obtenidos en esta prueba, puedes ponerte en contacto en las direcciones anteriormente especificadas, teniendo en cuenta que el proyecto terminará en diciembre del 2023.

Los datos que se obtengan de tu participación, serán tratados con absoluta confidencialidad. De acuerdo con la ley de protección de datos, se incluirán en el fichero de la UPV/EHU, de referencia INB - BCI (INB0102). Puedes consultar en cualquier momento los datos que has facilitado, o solicitar que se rectifiquen, cancelen o simplemente que no se utilicen para algún fin concreto de esta investigación. La manera de hacerlo es dirigiéndose al Responsable de Seguridad LOPD de la UPV/EHU, Rectorado Barrio Sarriena, s/n 48940-Leioa (lopd@ehu.eus).

Para más información sobre protección de datos se recomienda consultar en internet la página:

www.ehu.es/babestu

Por último, te solicito permiso para poder guardar tus datos de manera anónima y utilizarlos en proyectos de investigación sobre este mismo tema.

He leído, he entendido y he podido hacer preguntas y por tanto consiento que se guarden y se utilicen mis datos	**No acepto que se guarden mis datos**

Lugar y fecha:

Firma de quien informa **Firma de quien consiente**

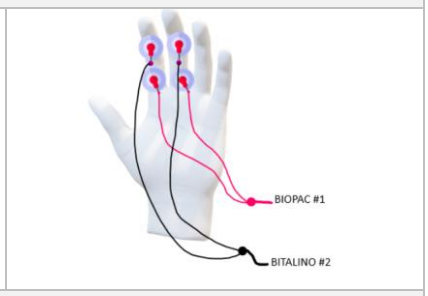
_____ _____

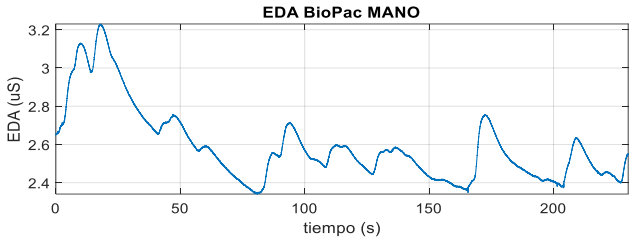
ANEXO V.

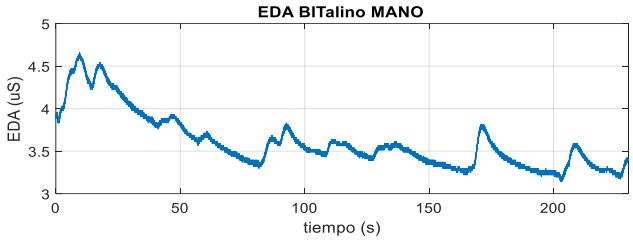
GRÁFICAS Y RESULTADOS COMPLETOS DE LAS PRUEBAS REALIZADAS MEDIANTE BITALINO

ANEXO V. GRÁFICAS Y RESULTADOS COMPLETOS DE LAS PRUEBAS REALIZADAS MEDIANTE BITALINO

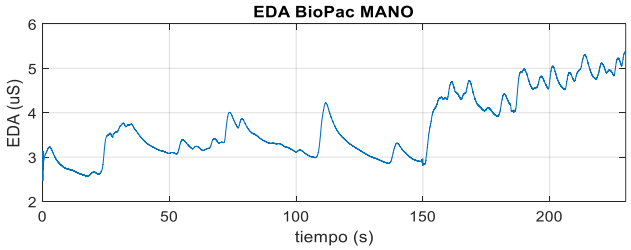
RESULTADOS PRUEBA 1.

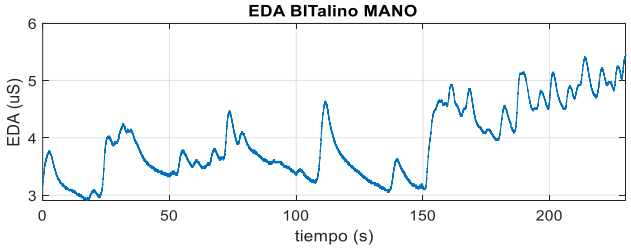
PARTICIPANTE	**001**
PRUEBA	Lectura EDA
ELECTRODOS	Mano no-dominante (izquierda) 
CORRELACIÓN EDA	0.9427



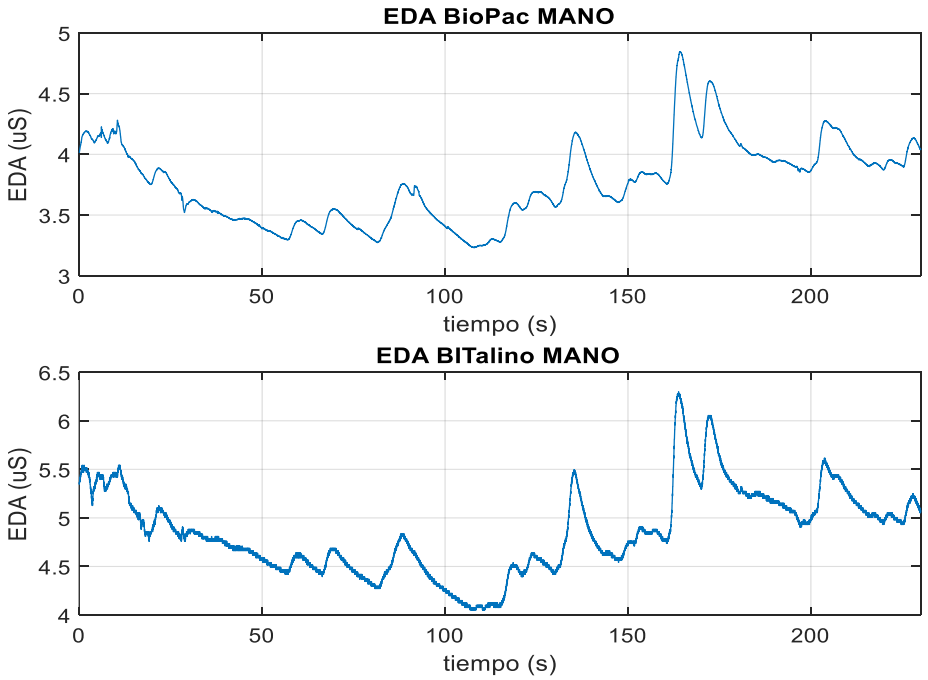


PARTICIPANTE	002
PRUEBA	Lectura EDA
ELECTRODOS	Mano no-dominante (izquierda) Biopac: índice (negro) + corazón (rojo) *posición #1 BITalino: índice (negro) + corazón (rojo) *posición #2
CORRELACIÓN EDA	0.9930

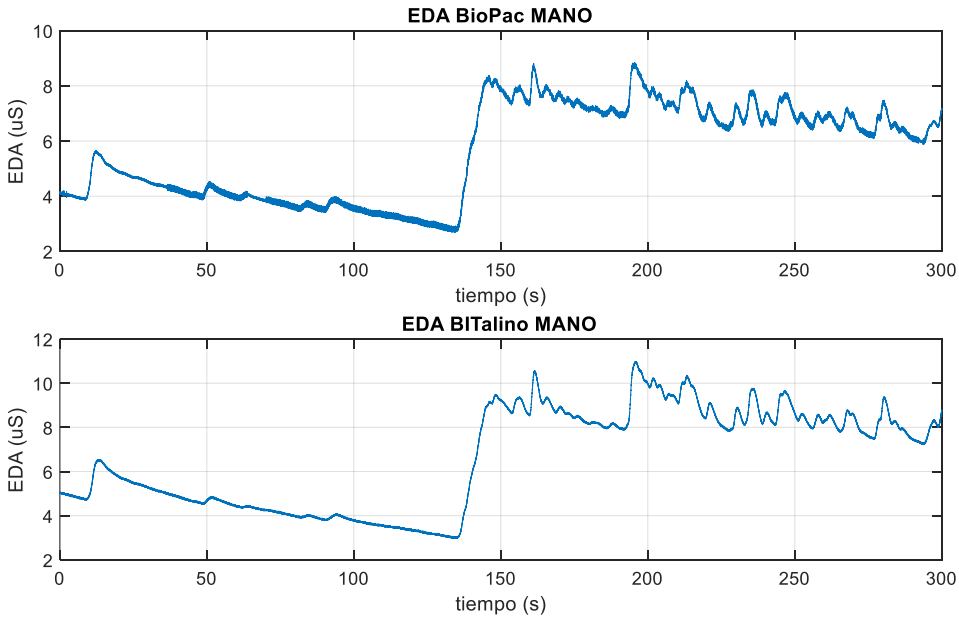


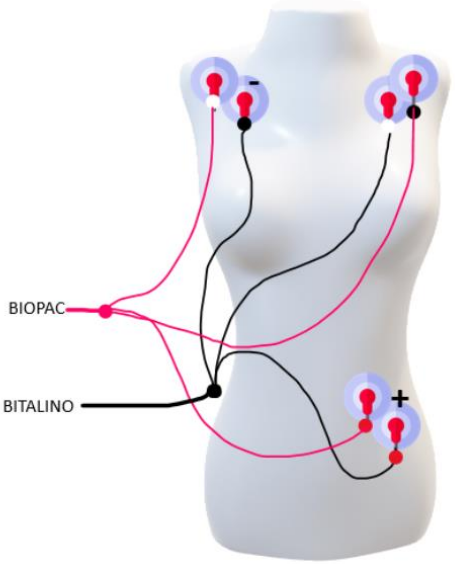


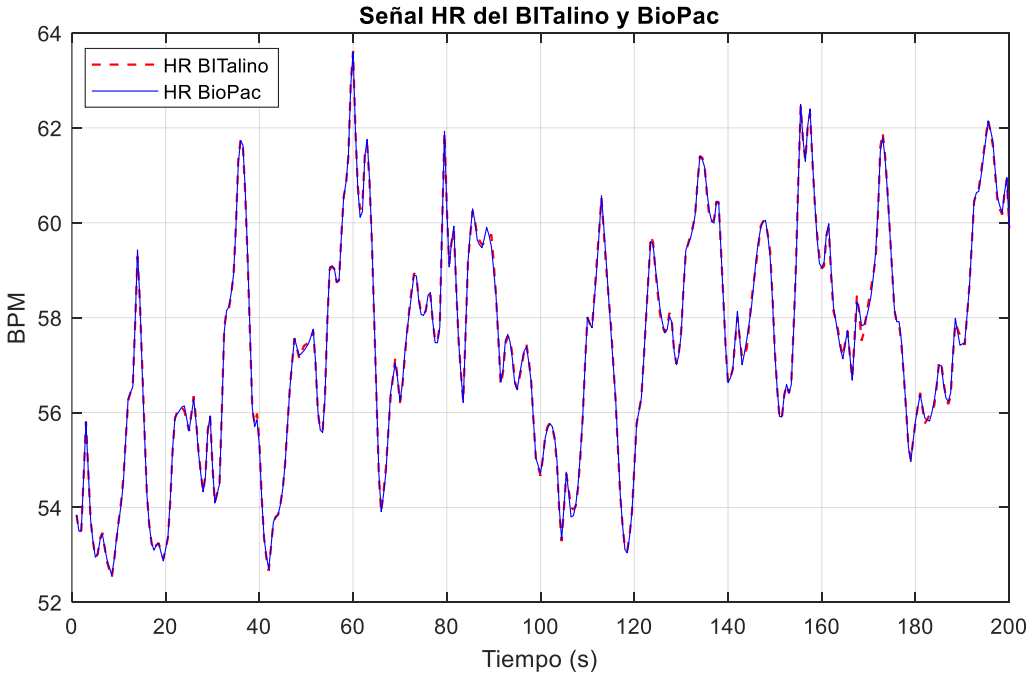
PARTICIPANTE	003
PRUEBA	Lectura EDA
ELECTRODOS	Mano no-dominante (izquierda) Biopac: índice (negro) + corazón (rojo) *posición #1 BITalino: índice (negro) + corazón (rojo) *posición #2
CORRELACIÓN EDA	0.9640



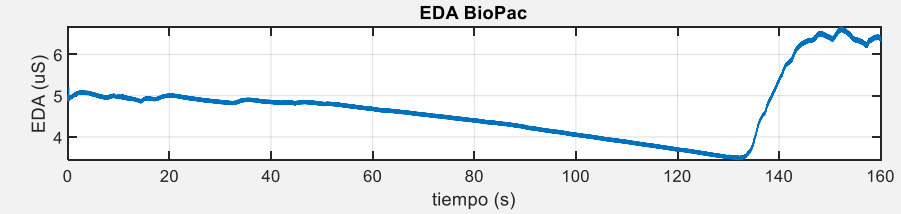
PARTICIPANTE	004
PRUEBA	Lectura EDA
ELECTRODOS	Mano no-dominante (izquierda) Biopac: índice (negro) + corazón (rojo) *posición #1 BITalino: índice (negro) + corazón (rojo) *posición #2
CORRELACIÓN EDA	0.9890

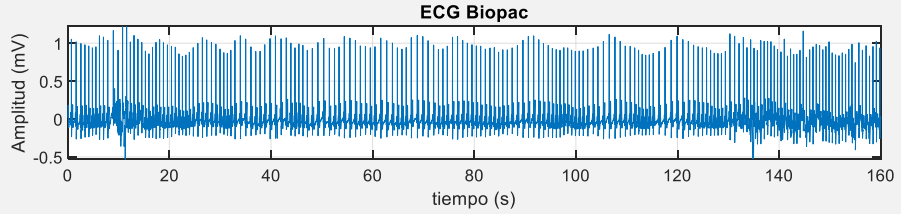


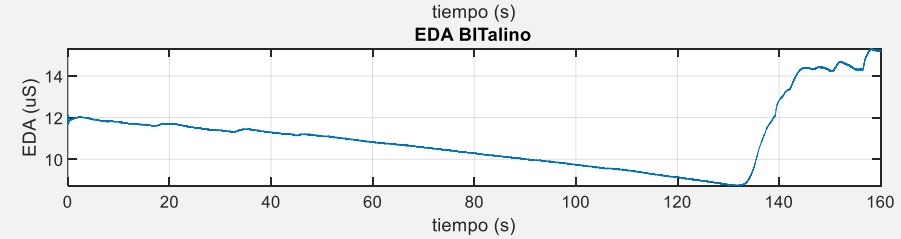
PARTICIPANTE	005	
PRUEBA	Lectura ECG	
ELECTRODOS	Biopac: Rojo: Positivo Negro: GND Blanco: Negativo BITalino: Rojo: Positivo Negro: Negativo Blanco: GND	
CORRELACIÓN HR	0.9997	

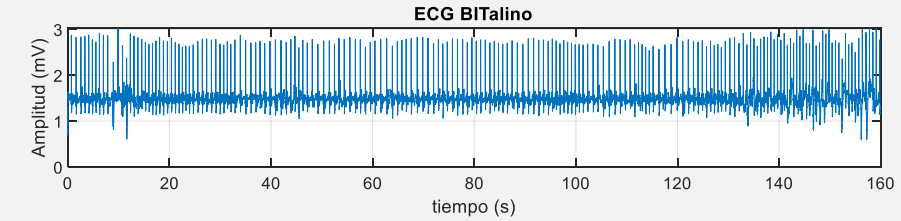


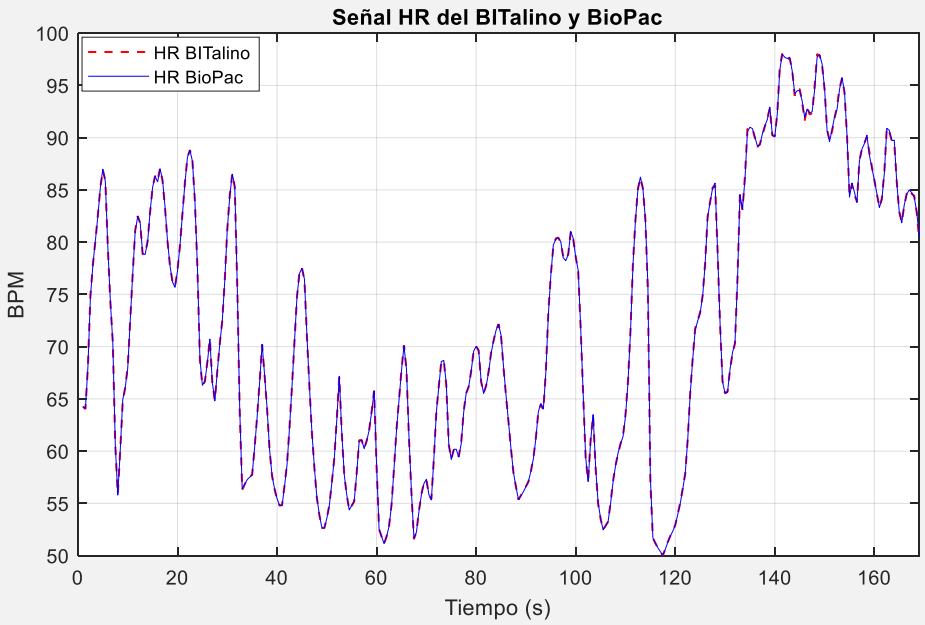
PARTICIPANTE	006
PRUEBA	Lectura EDA + ECG
ELECTRODOS	EDA: Mano no-dominante (izquierda) Biopac: índice (negro) + corazón (rojo) *posición #1 BITalino: índice (negro) + corazón (rojo) *posición #2 ECG: 3 electrodos por cada dispositivo
CORRELACIÓN EDA	0.5581
CORRELACIÓN HR	1



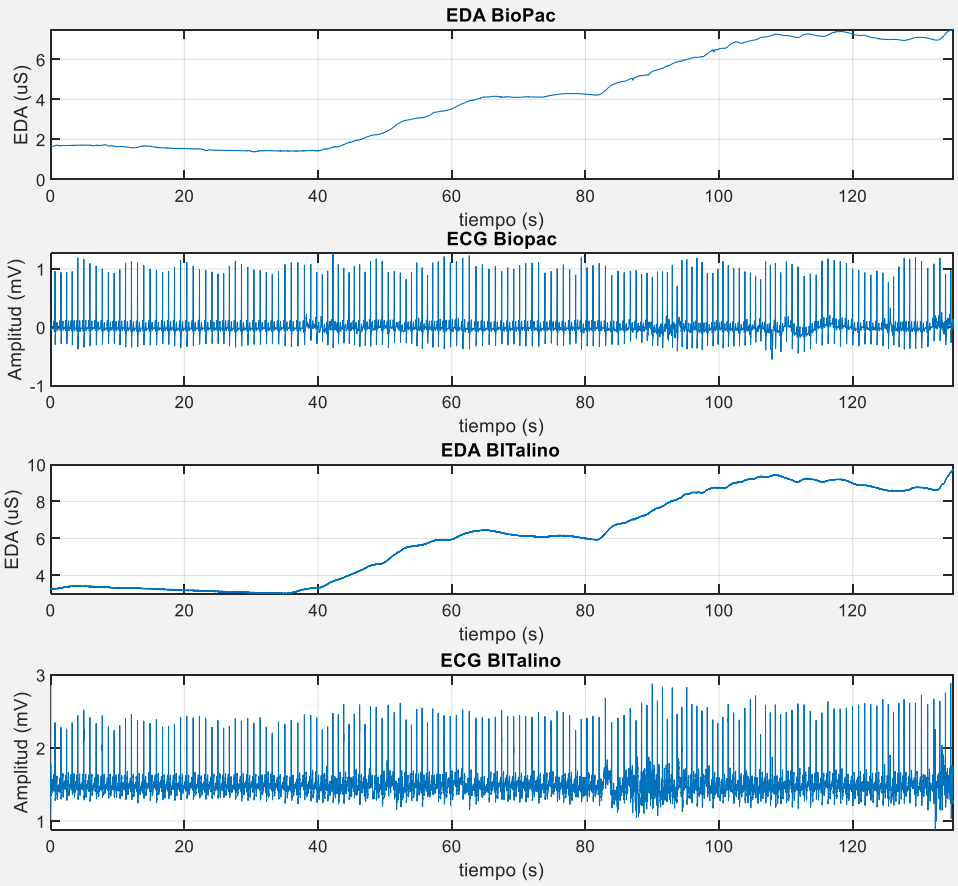


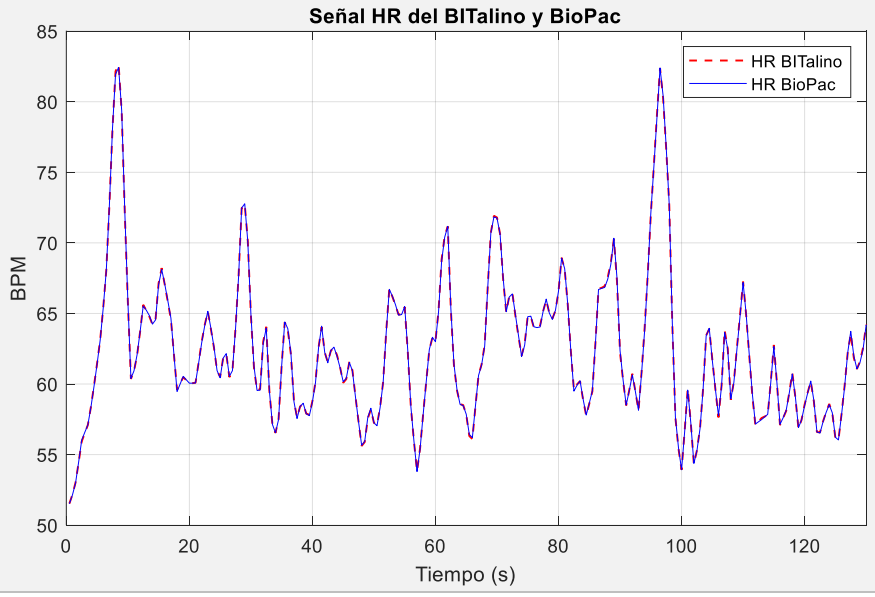




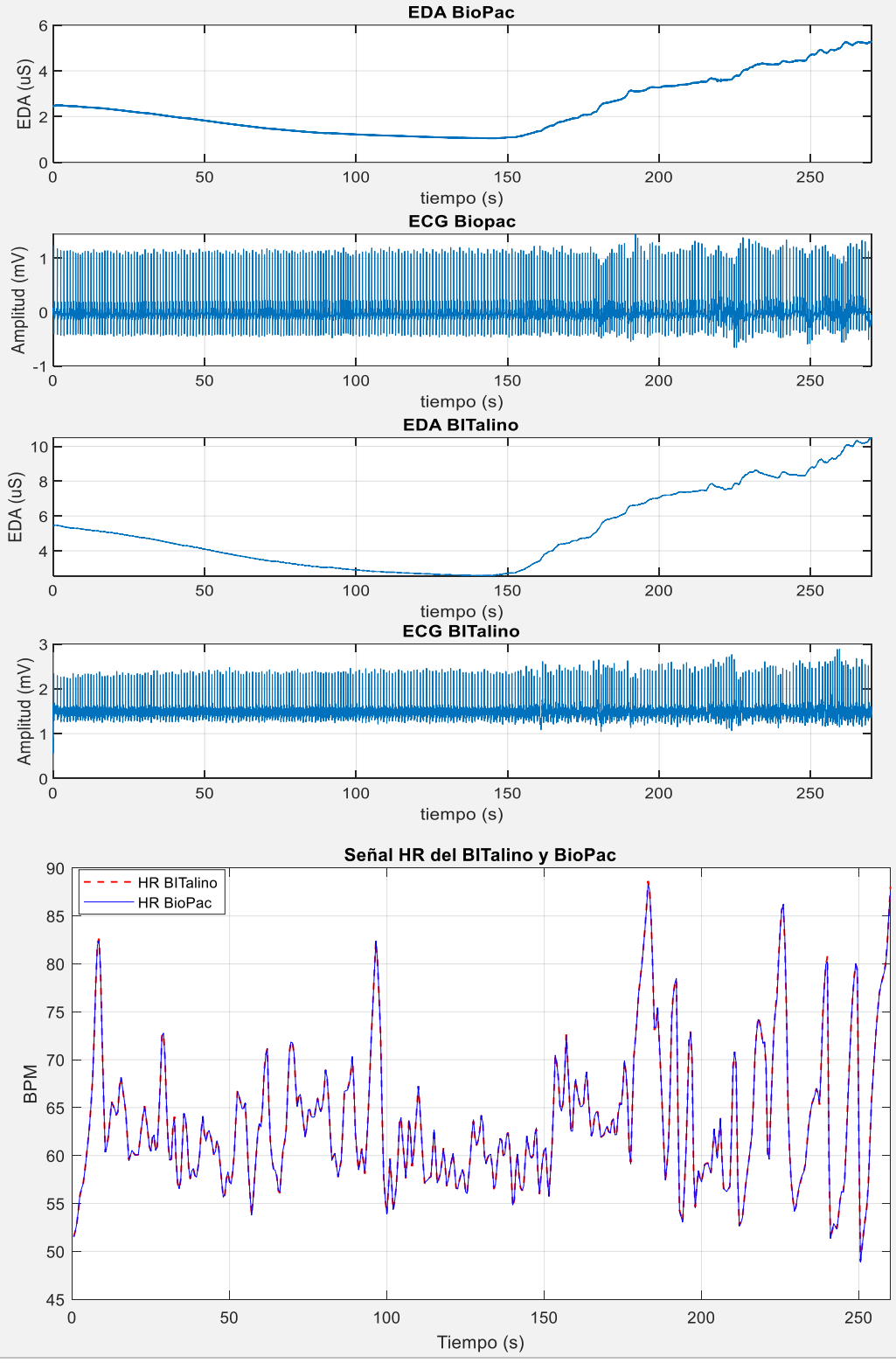


PARTICIPANTE	007
PRUEBA	Lectura EDA + ECG
ELECTRODOS	EDA: Mano no-dominante (izquierda) Biopac: índice (negro) + corazón (rojo) *posición #1 BITalino: índice (negro) + corazón (rojo) *posición #2 ECG: 3 electrodos por cada dispositivo (apartado tal)
CORRELACIÓN EDA	0.9923
CORRELACIÓN HR	07276

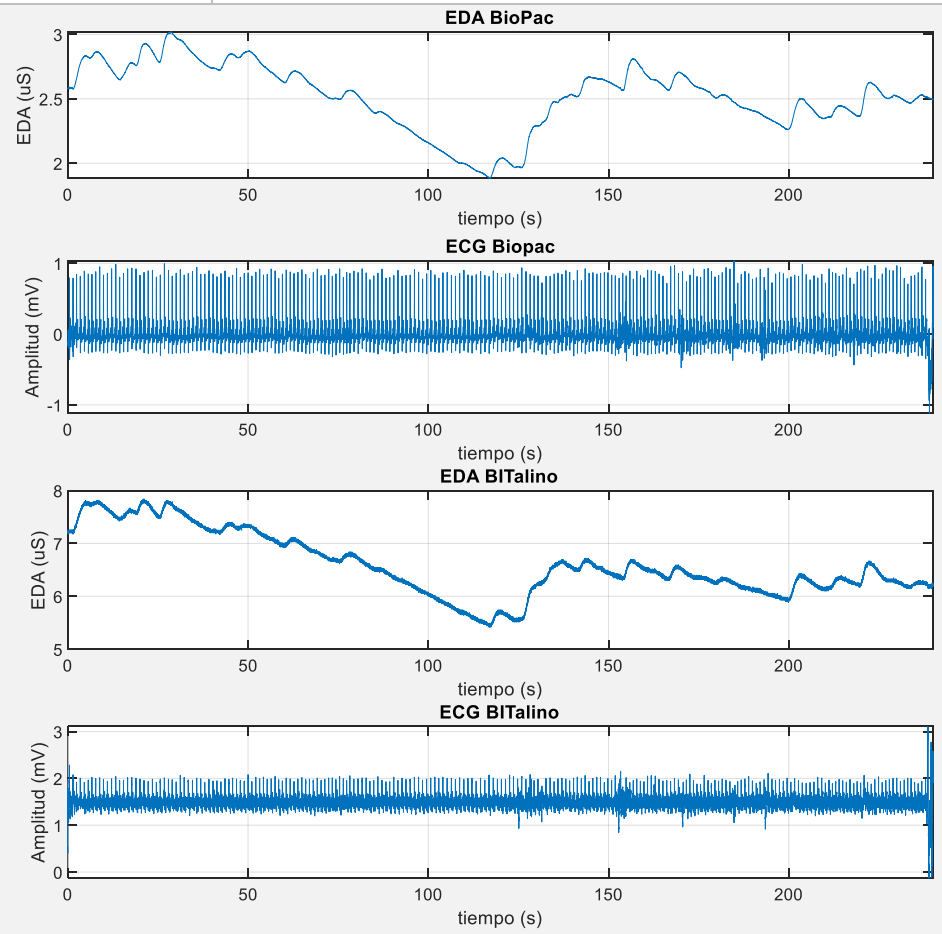


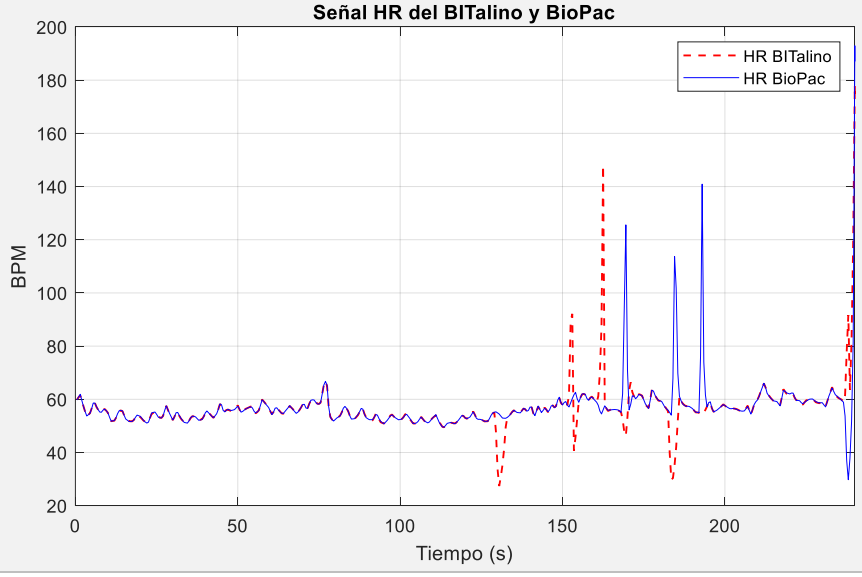


PARTICIPANTE	008
PRUEBA	Lectura EDA + ECG
ELECTRODOS	EDA: Mano no-dominante (izquierda) Biopac: índice (negro) + corazón (rojo) *posición #1 BITalino: índice (negro) + corazón (rojo) *posición #2 ECG: 3 electrodos por cada dispositivo (apartado tal)
CORRELACIÓN EDA	0.9957
CORRELACIÓN HR	0.8724

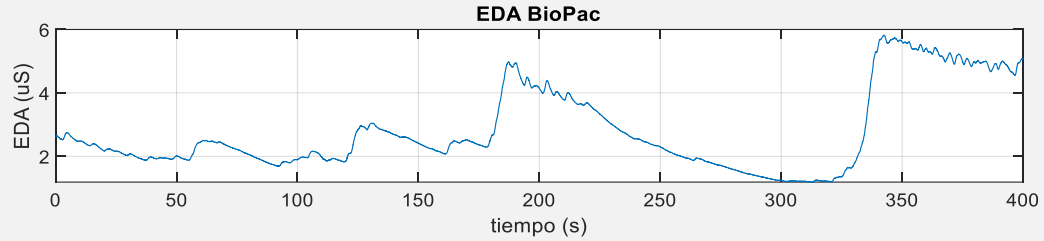


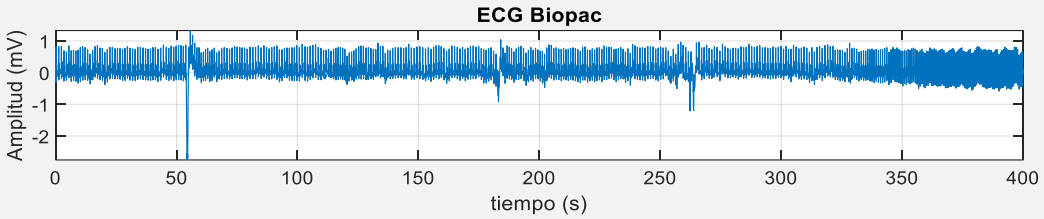
PARTICIPANTE	009
PRUEBA	Lectura EDA + ECG
ELECTRODOS	EDA: Mano no-dominante (izquierda) Biopac: índice (negro) + corazón (rojo) *posición #1 BITalino: índice (negro) + corazón (rojo) *posición #2 ECG: 3 electrodos por cada dispositivo (apartado tal)
CORRELACIÓN EDA	0.8922
CORRELACIÓN HR	0.9991

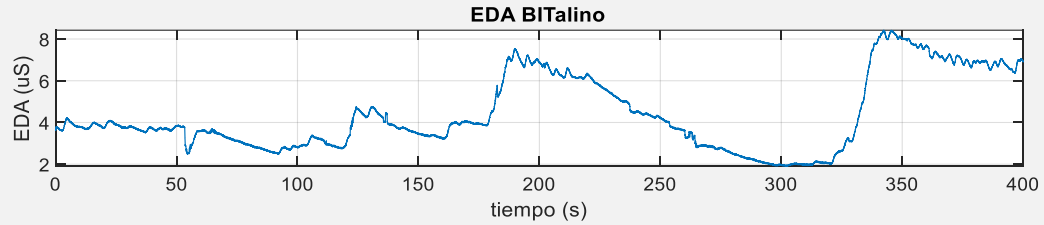


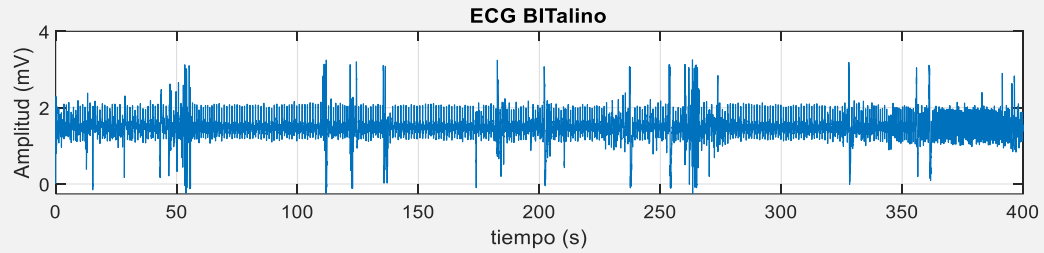


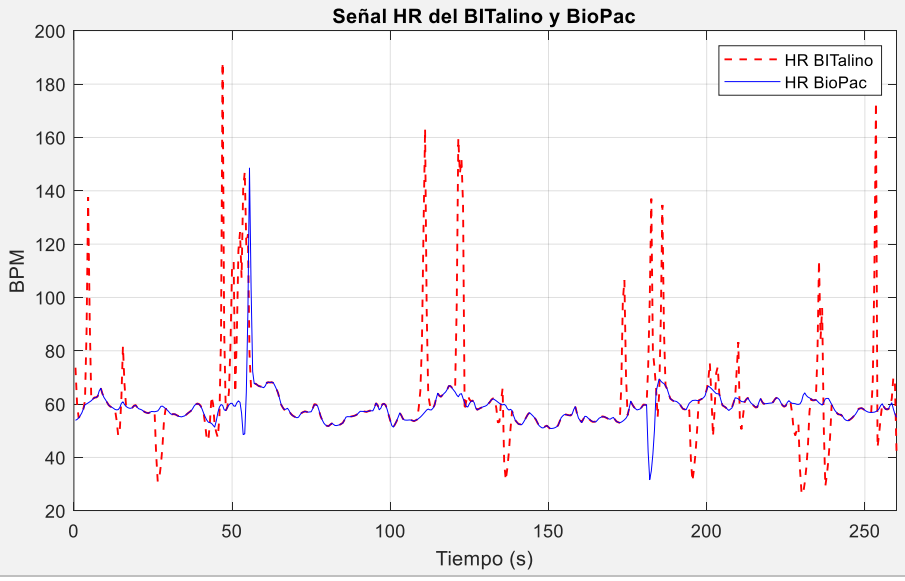
PARTICIPANTE	010
PRUEBA	Lectura EDA + ECG
ELECTRODOS	EDA: Mano no-dominante (izquierda) Biopac: índice (negro) + corazón (rojo) *posición #1 BITalino: índice (negro) + corazón (rojo) *posición #2 ECG: 3 electrodos por cada dispositivo (apartado tal)
CORRELACIÓN EDA	0.9763
CORRELACIÓN HR	0.1571



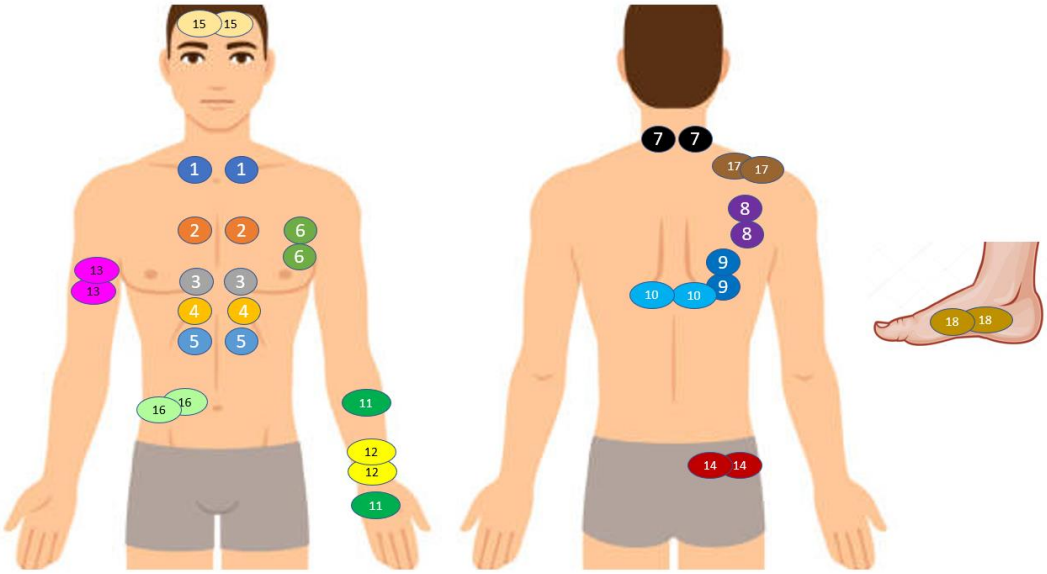






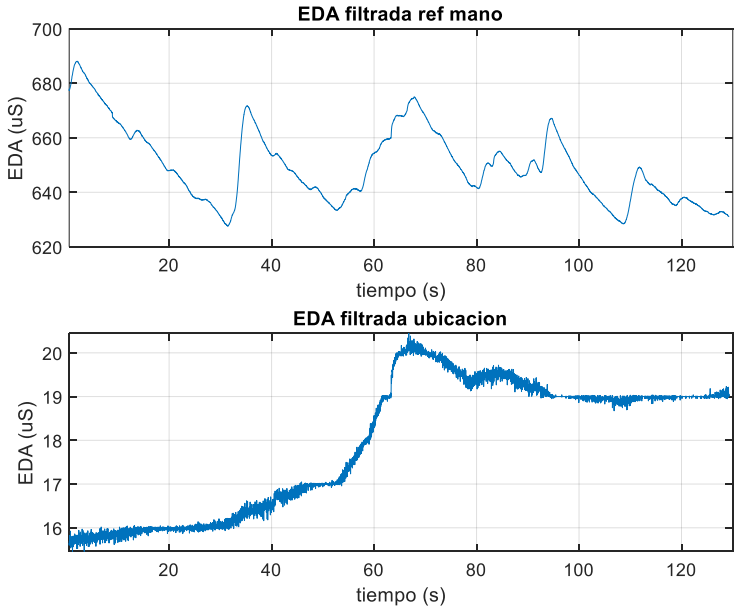


RESULTADOS PRUEBA 3.

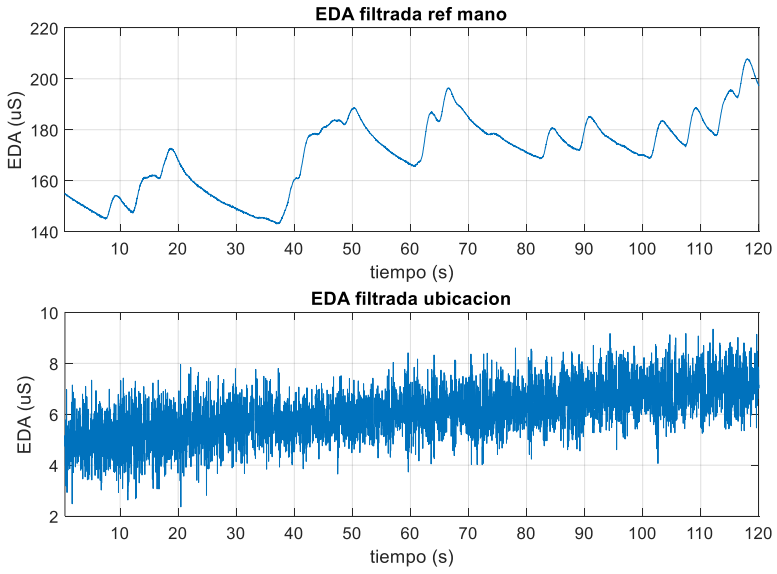


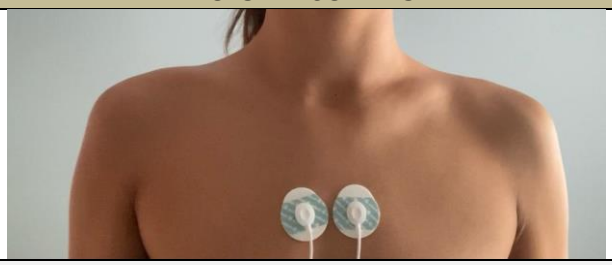
1	Cuello delantero	10	Espina dorsal
2	Pectoral superior	11	Antebrazo
3	Pectoral inferior	12	Muñeca
4	Abdomen superior	13	Bíceps
5	Recto abdominal	14	Glúteo superior
6	Axila	15	Frente
7	Cuello trasero	16	Apéndice
8	Escapula	17	Trapecio
9	Dorsal	18	Pie

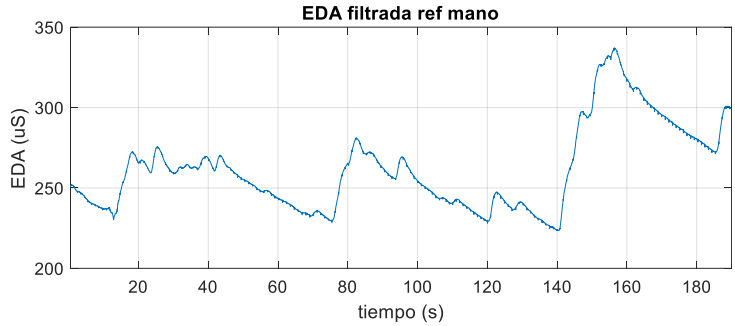
1	CUELLO DELANTERO
ELECTRODOS	
CORRELACIÓN PARTICIPANTE 011	0.3551

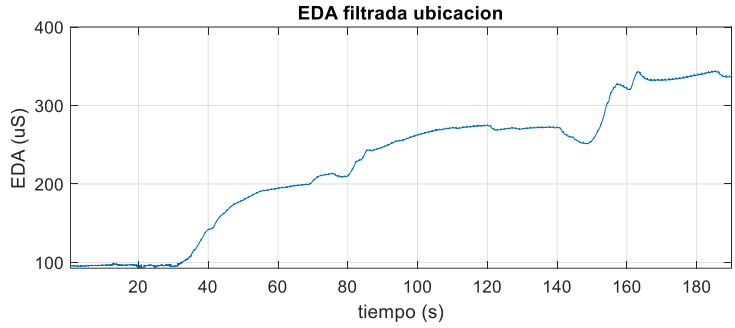


CORRELACIÓN PARTICIPANTE 012	0.5997

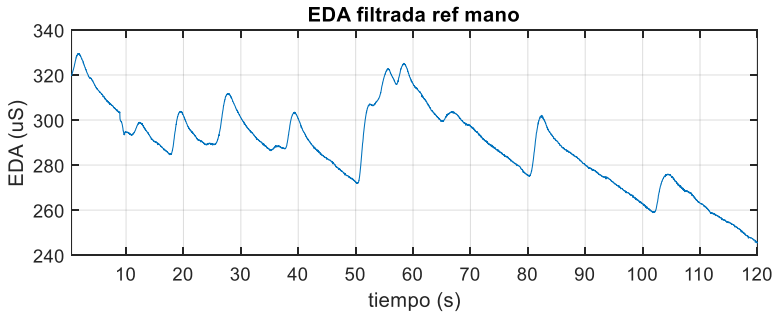


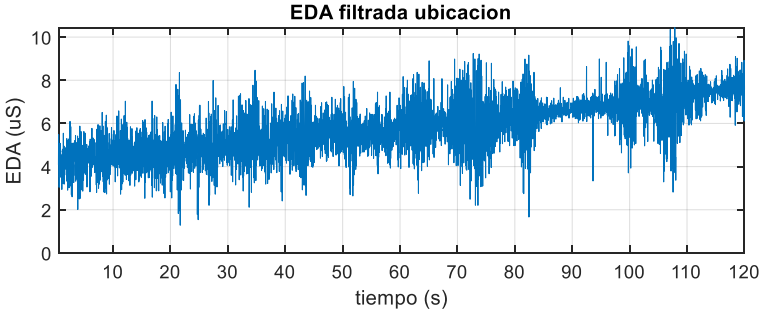
2	PECTORAL SUPERIOR
ELECTRODOS	
CORRELACIÓN PARTICIPANTE 011	0.4094



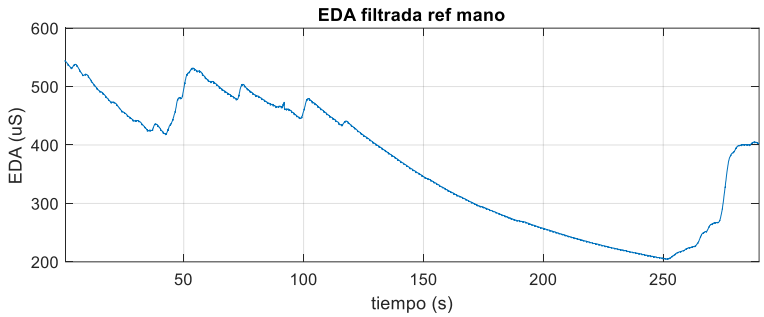


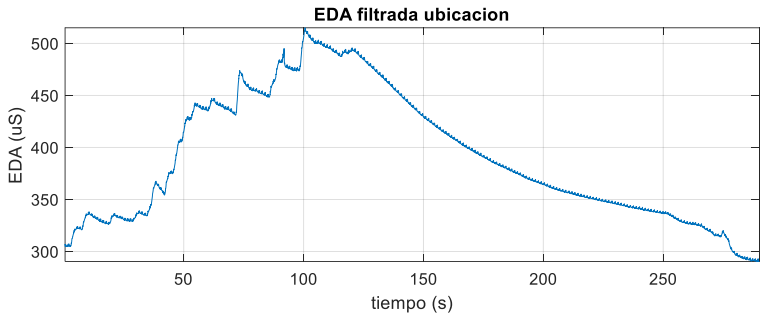
CORRELACIÓN PARTICIPANTE 012	-0.4857



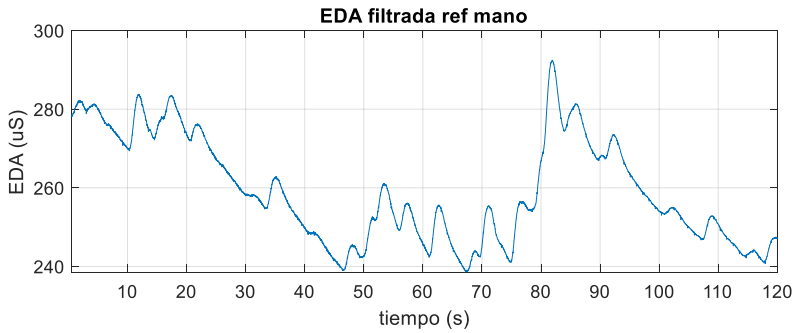


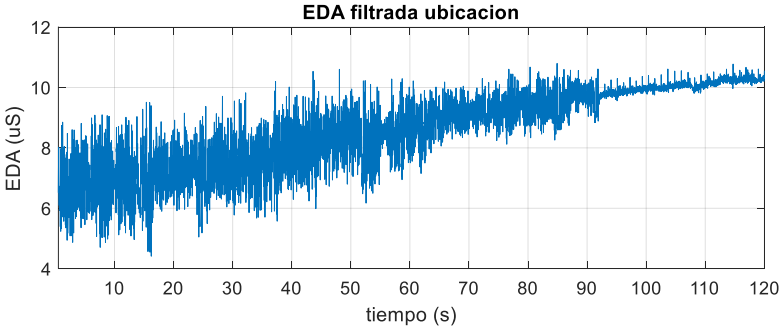
3	PECTORAL INFERIOR
ELECTRODOS	
CORRELACIÓN PARTICIPANTE 011	0.4246

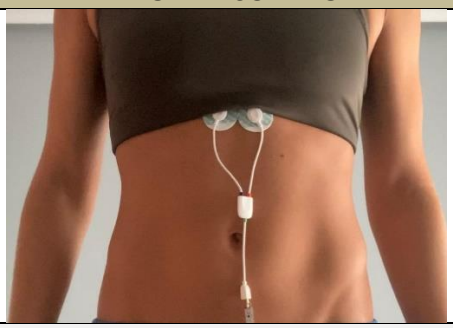


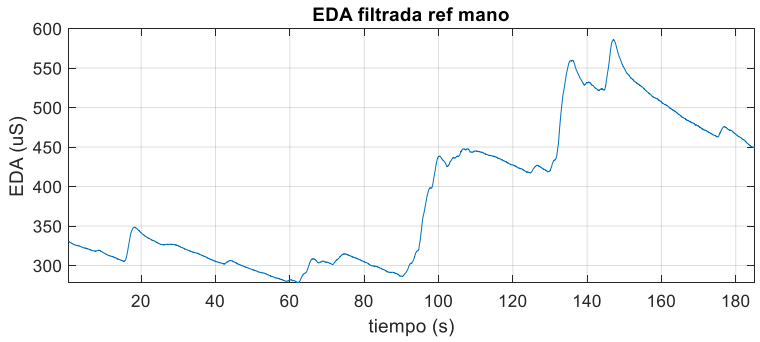


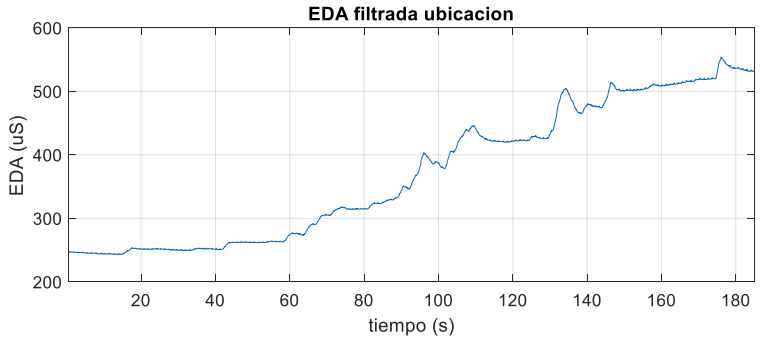
CORRELACIÓN PARTICIPANTE 012	-0.1403



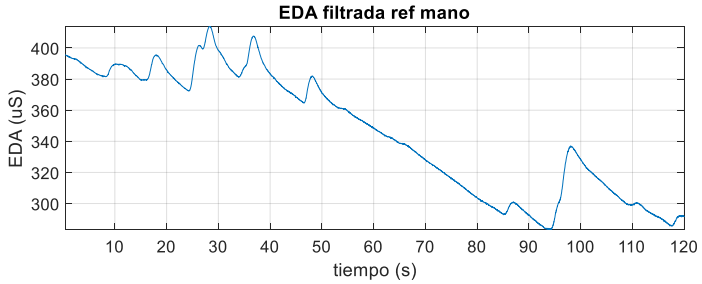


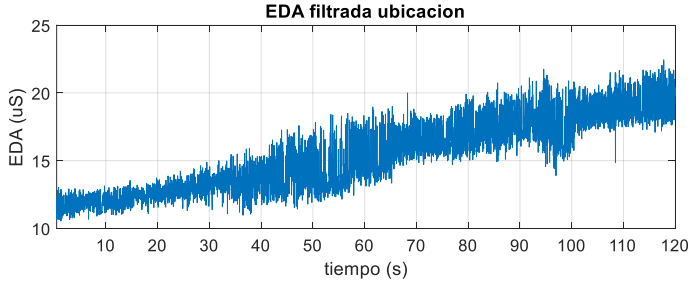
4	ABDOMEN SUPERIOR
ELECTRODOS	
CORRELACIÓN PARTICIPANTE 011	0.9028

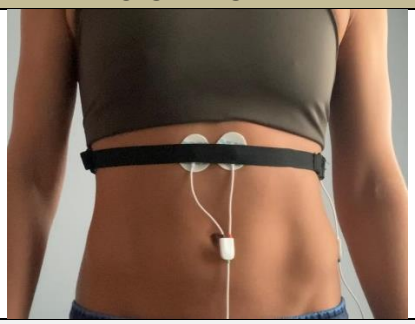


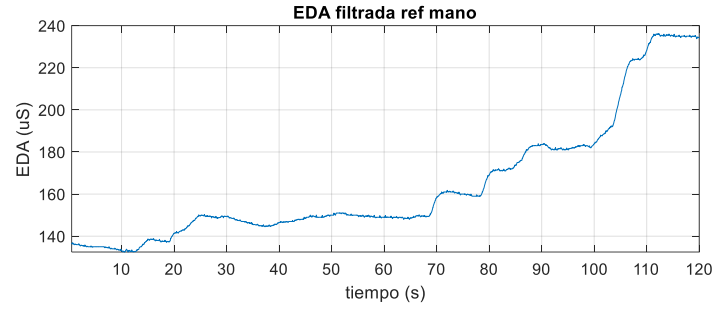


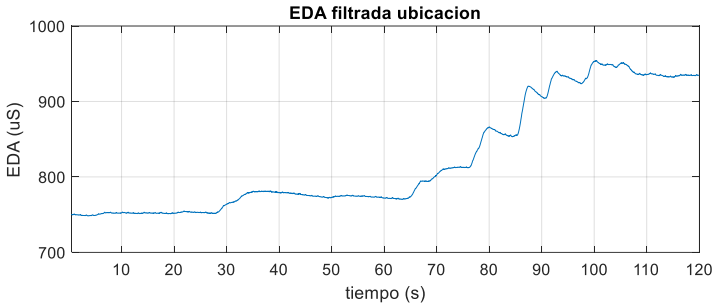
CORRELACIÓN PARTICIPANTE 012	-0.7368



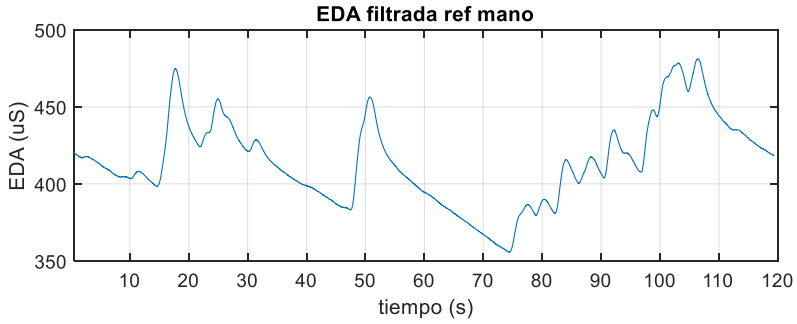


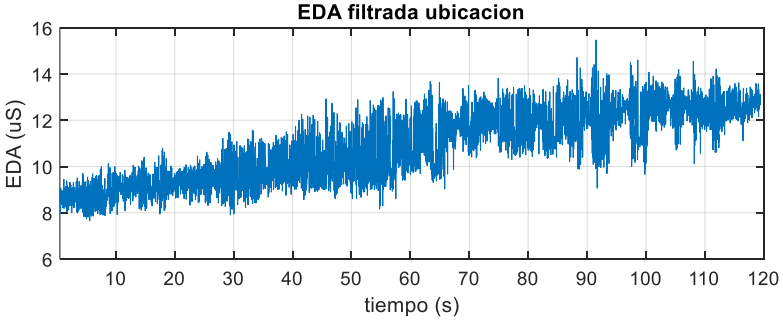
5	RECTO ABDOMINAL
ELECTRODOS	
CORRELACIÓN PARTICIPANTE 011	0.8921

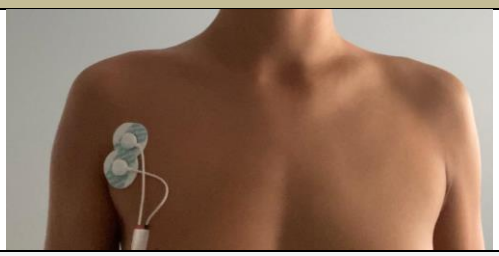


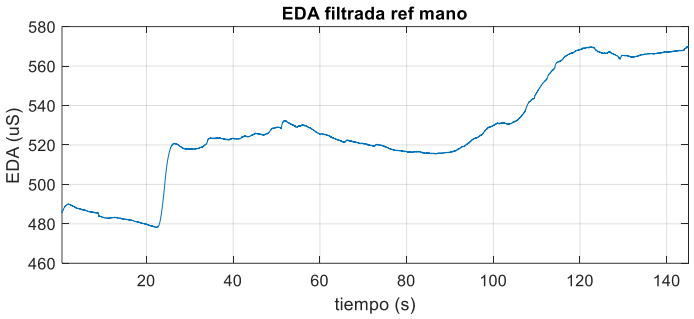


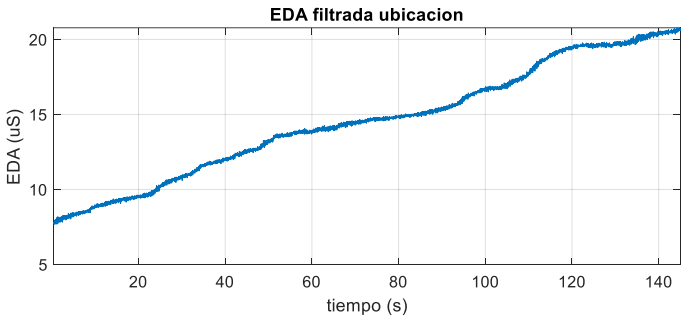
CORRELACIÓN PARTICIPANTE 012	0.1909



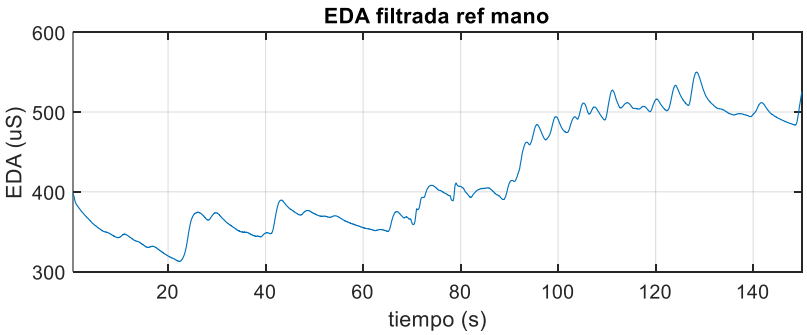


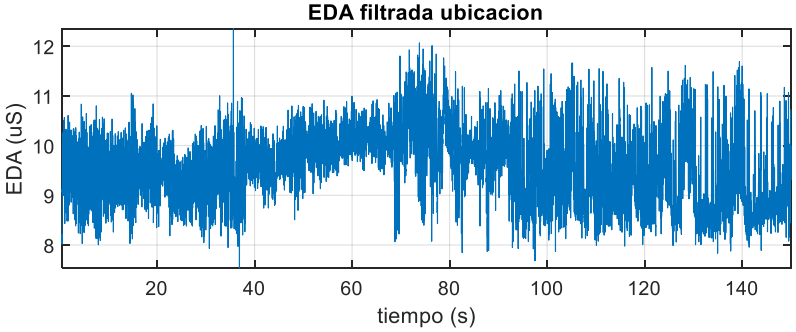
6	AXILA
ELECTRODOS	
CORRELACIÓN PARTICIPANTE 011	0.9520

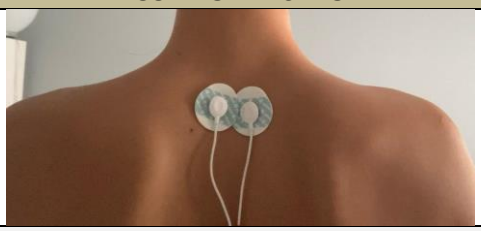


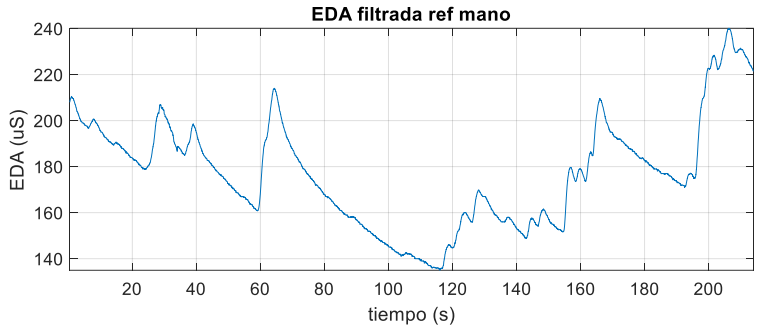


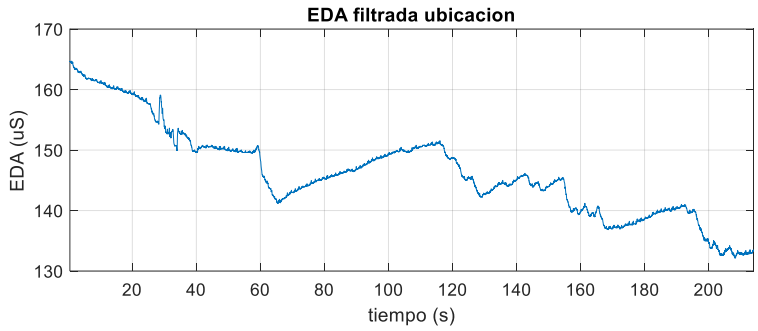
CORRELACIÓN PARTICIPANTE 012	0.1393



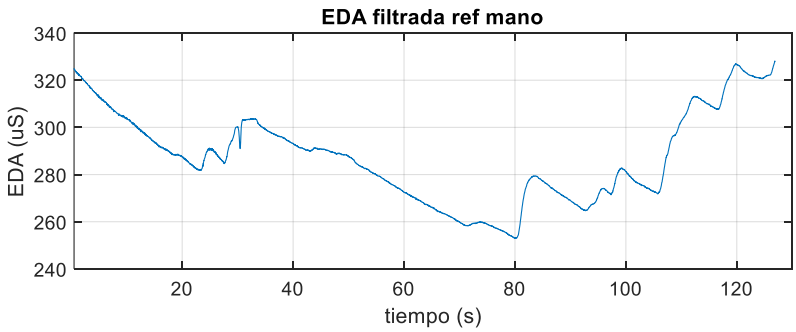


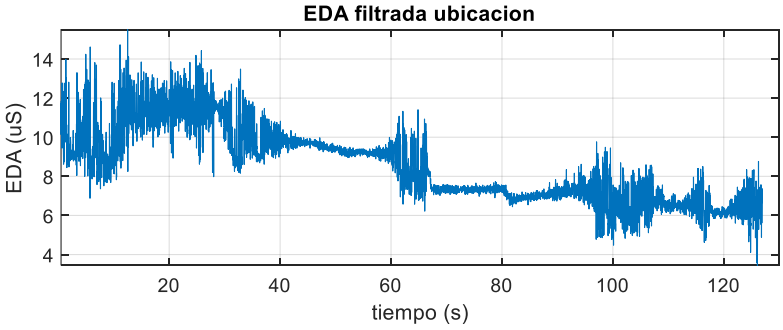
7	CUELLO TRASERO
ELECTRODOS	
CORRELACIÓN PARTICIPANTE 011	-0.0542

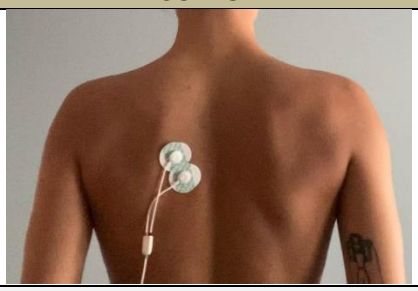


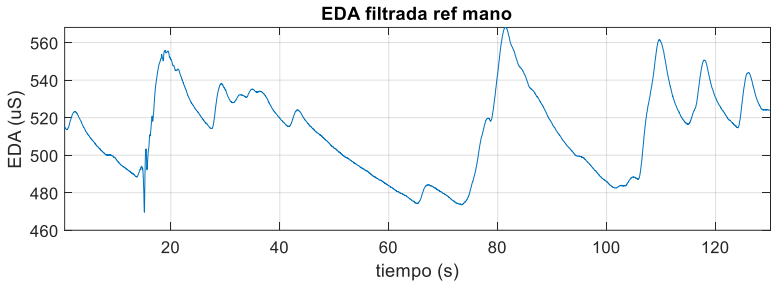


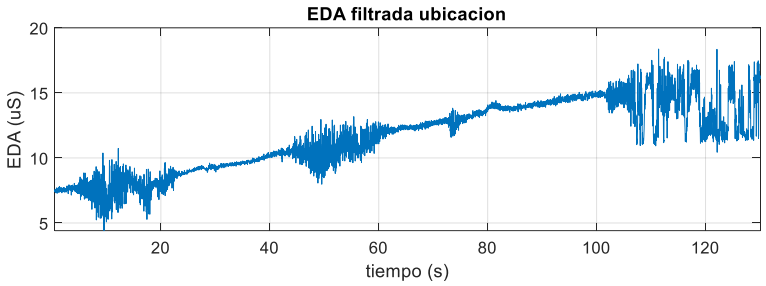
CORRELACIÓN PARTICIPANTE 012	0.2181



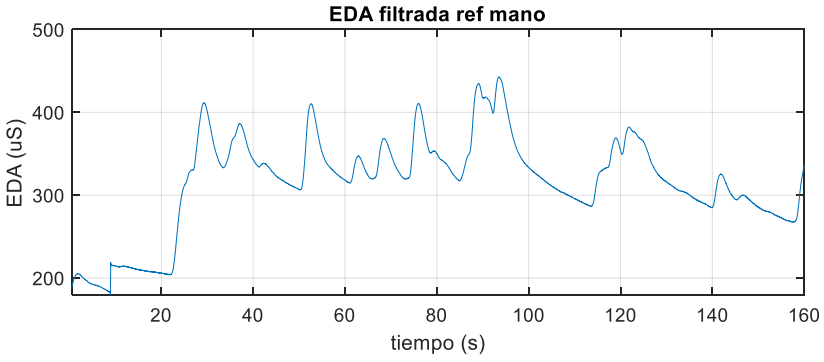


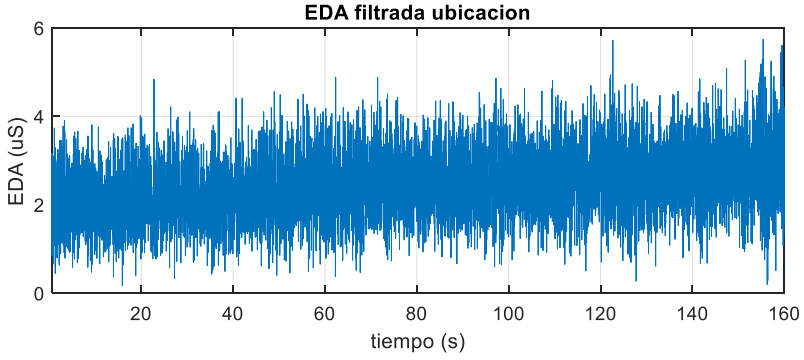
8	ESCAPULA
ELECTRODOS	
CORRELACIÓN PARTICIPANTE 011	0.1207

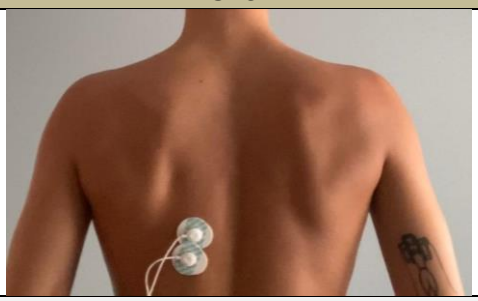


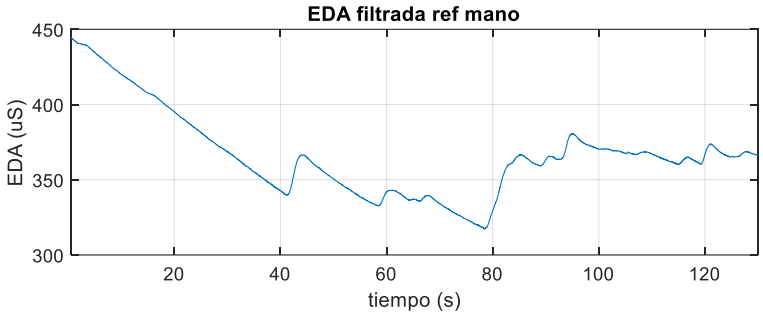


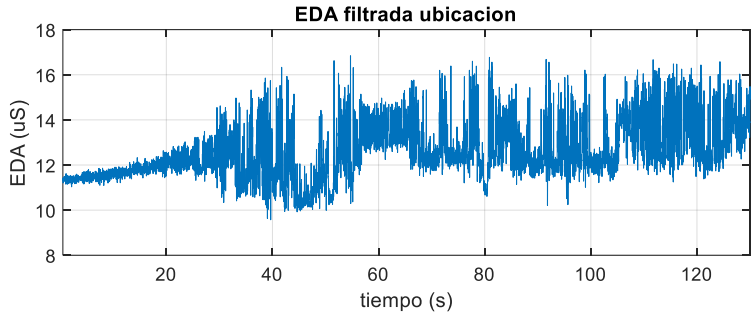
CORRELACIÓN PARTICIPANTE 012	0.0523



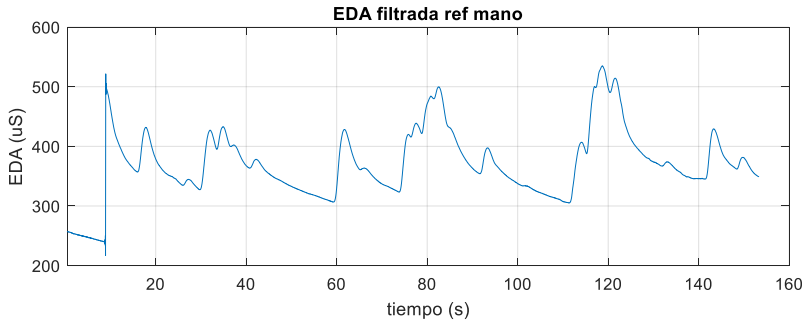


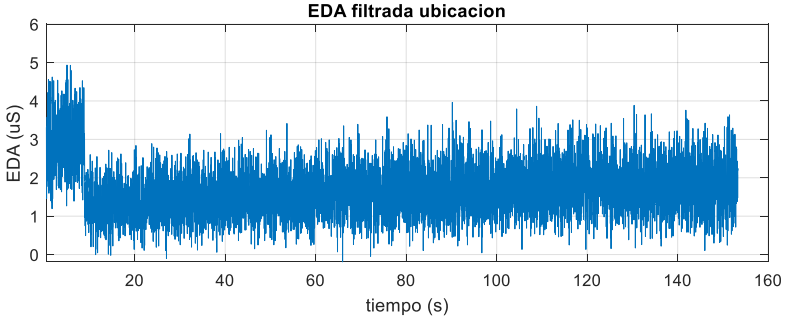
9	DORSAL
ELECTRODOS	
CORRELACIÓN PARTICIPANTE 011	-0.0430



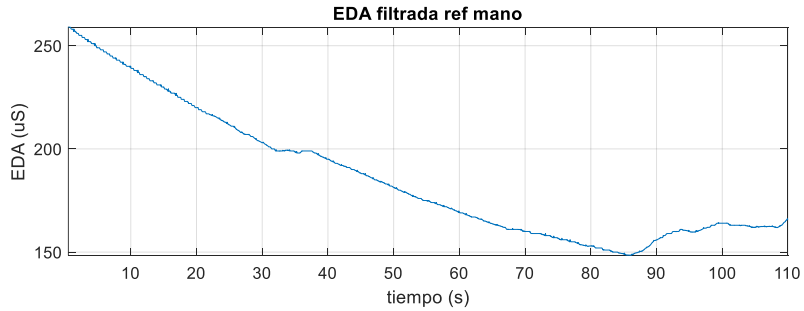


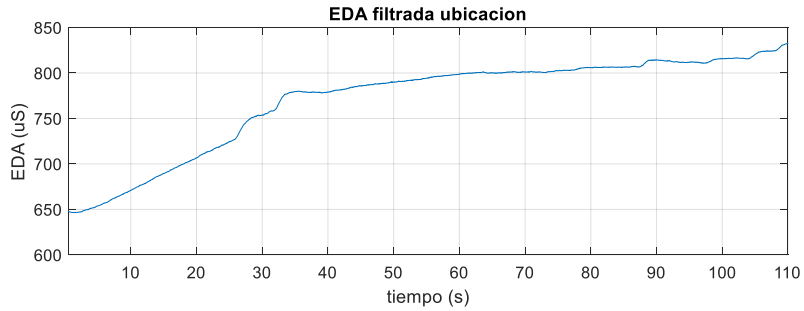
CORRELACIÓN PARTICIPANTE 012	-0.1791



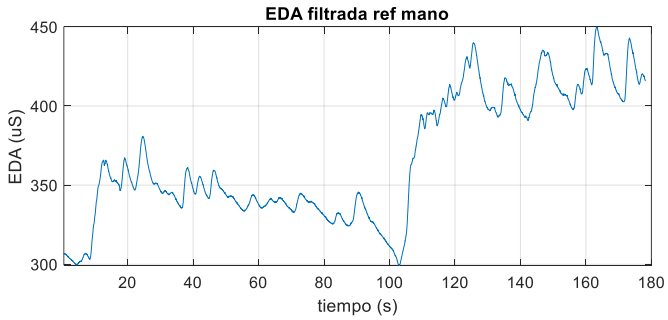


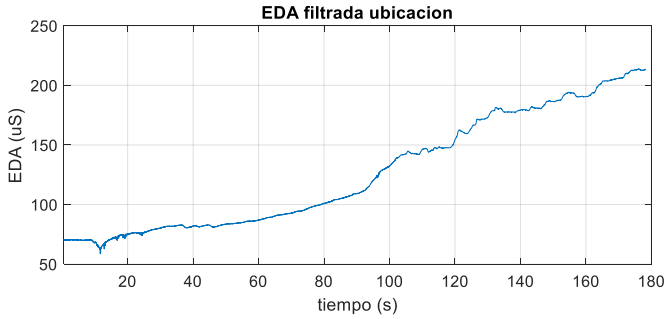
10	ESPINA DORSAL
ELECTRODOS	
CORRELACIÓN PARTICIPANTE 011	-0.7055

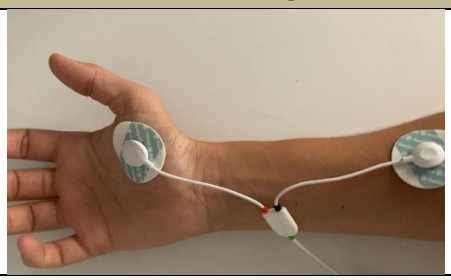


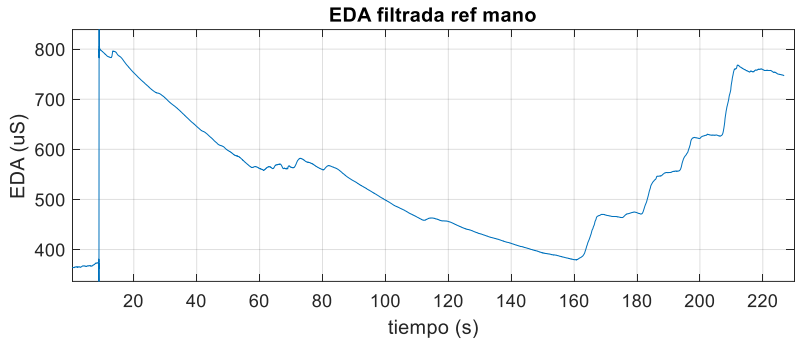


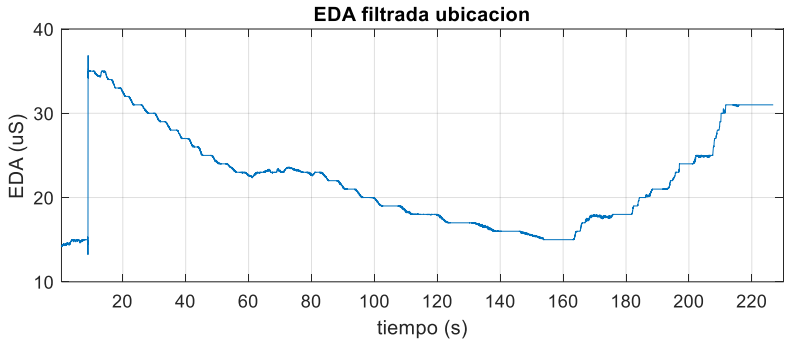
CORRELACIÓN PARTICIPANTE 012	0.8033



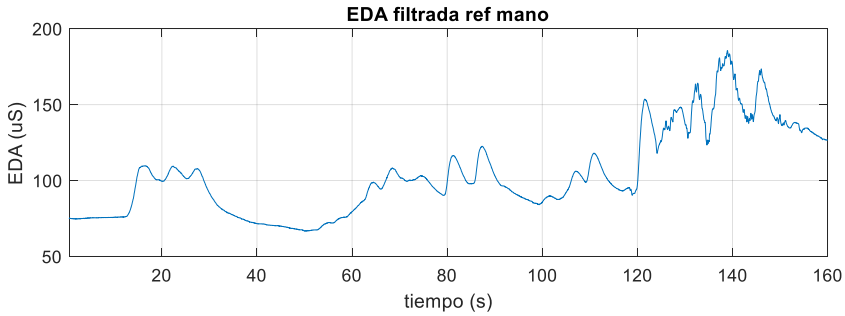


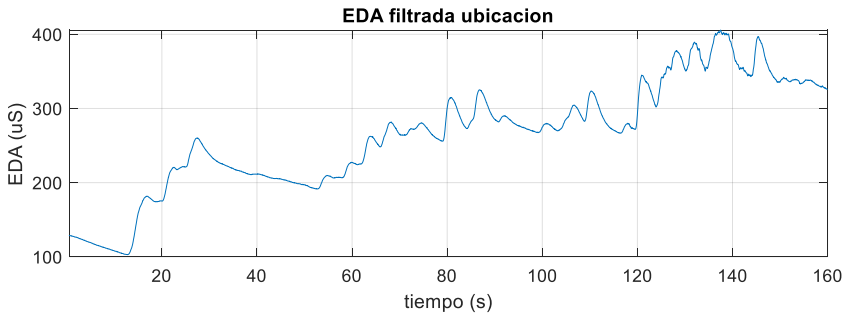
11	ANTEBRAZO
ELECTRODOS	
CORRELACIÓN PARTICIPANTE 011	0.9921



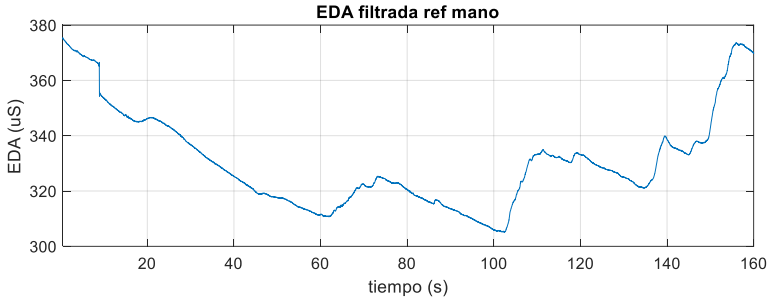


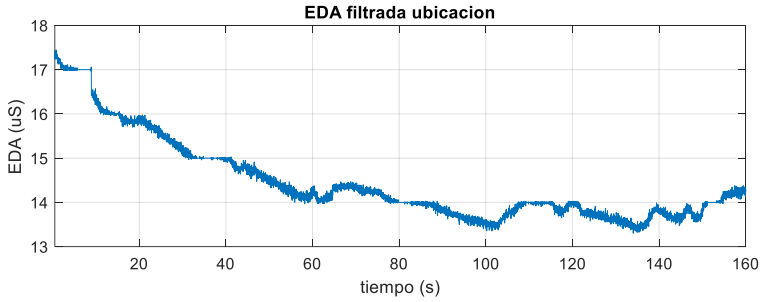
CORRELACIÓN PARTICIPANTE 012	0.8426



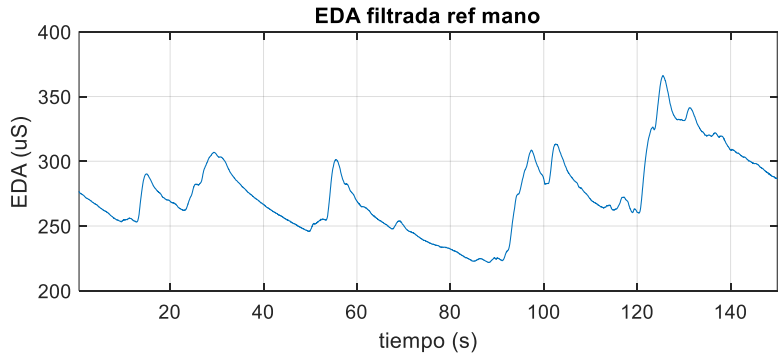


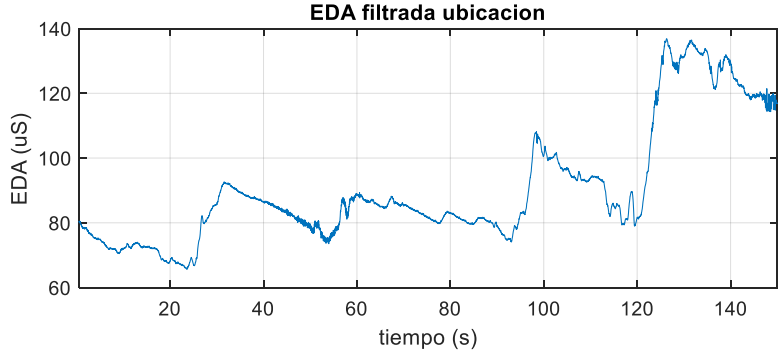
12	MUÑECA
ELECTRODOS	
CORRELACIÓN PARTICIPANTE 011	0.6266

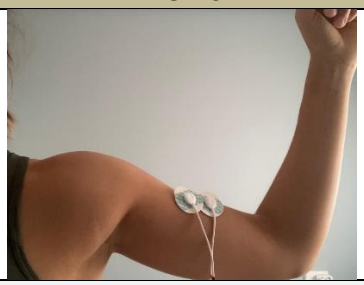


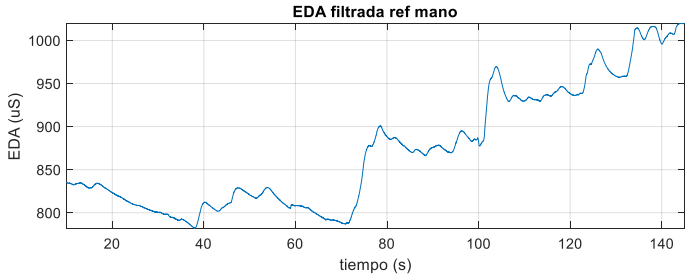


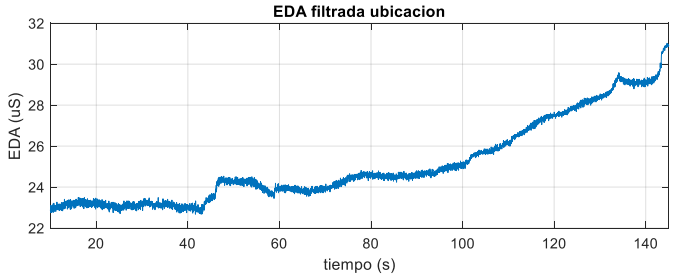
CORRELACIÓN PARTICIPANTE 012	0.7590



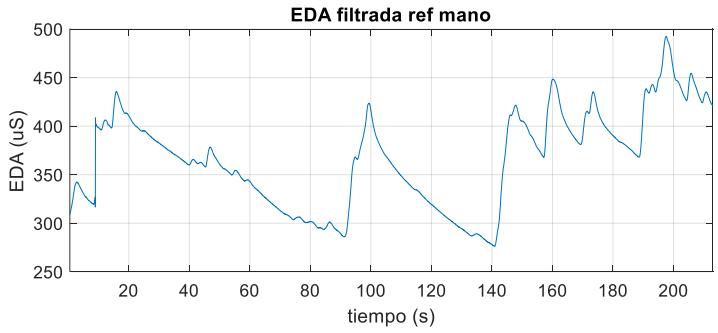


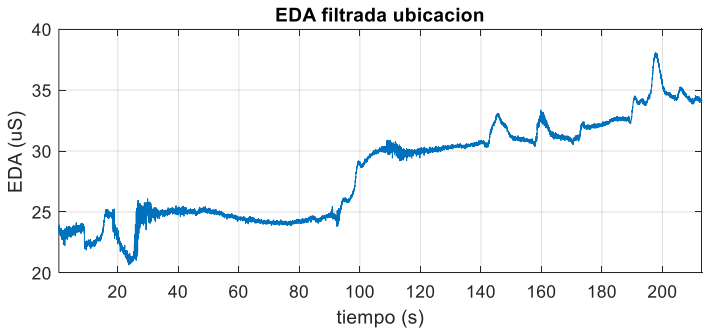
13	BÍCEPS
ELECTRODOS	
CORRELACIÓN PARTICIPANTE 011	0.7444

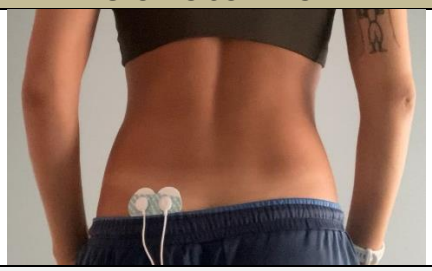


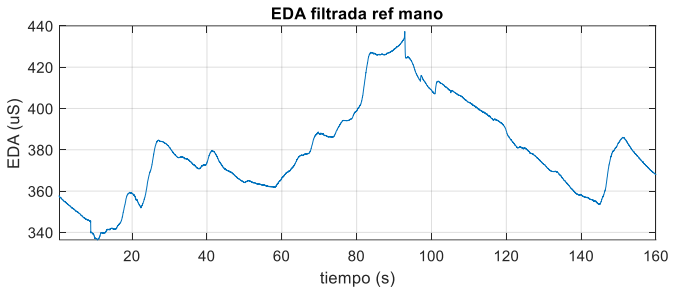


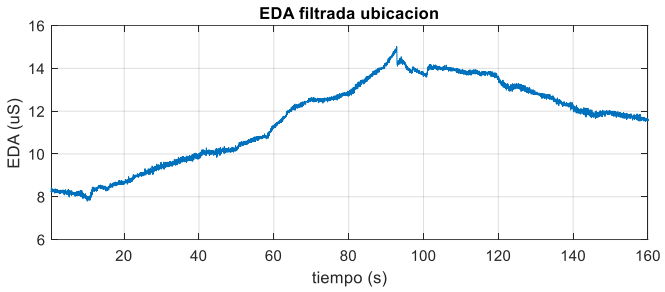
CORRELACIÓN PARTICIPANTE 012	0.4881



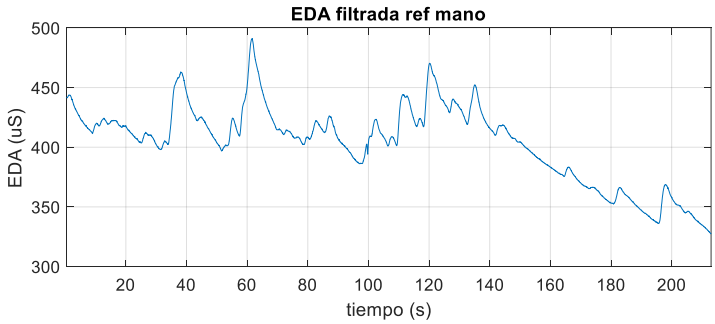


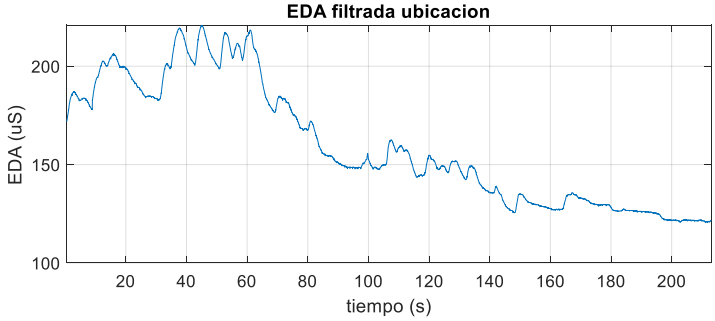
14	GLÚTEO SUPERIOR
ELECTRODOS	
CORRELACIÓN PARTICIPANTE 011	0.7735

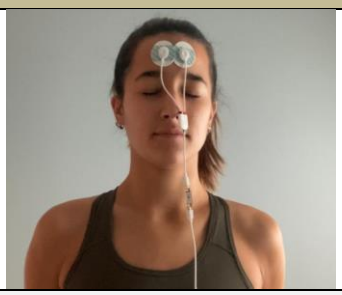


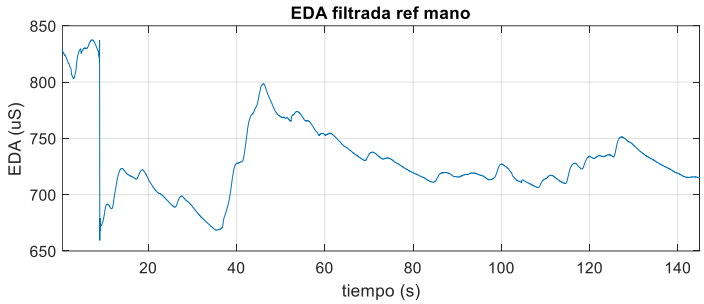


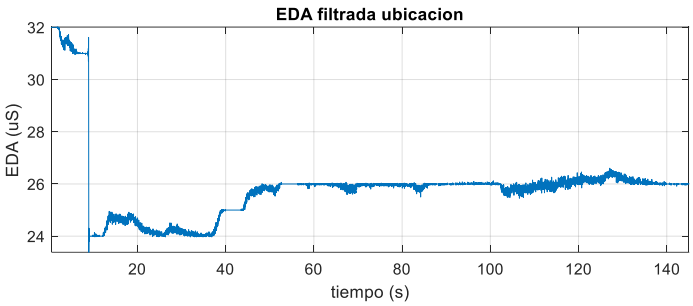
CORRELACIÓN PARTICIPANTE 012	0.6484



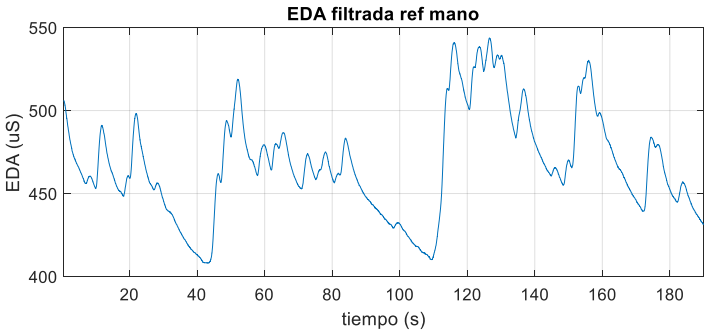


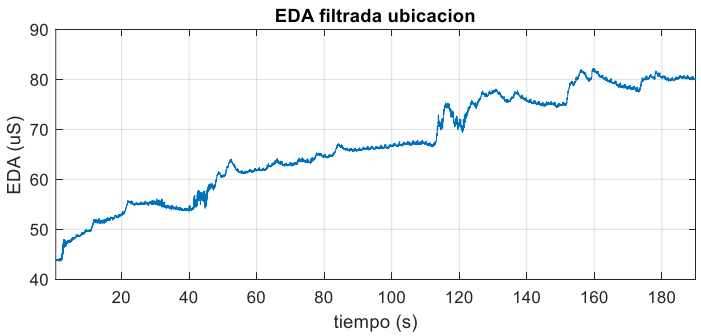
15	FRENTE
ELECTRODOS	
CORRELACIÓN PARTICIPANTE 011	0.8337

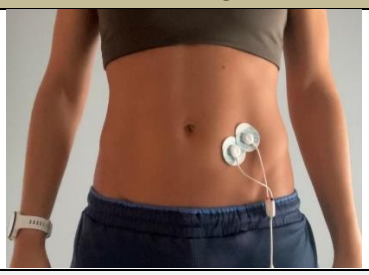


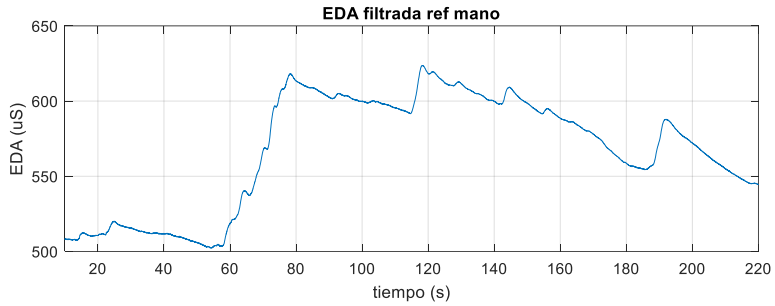


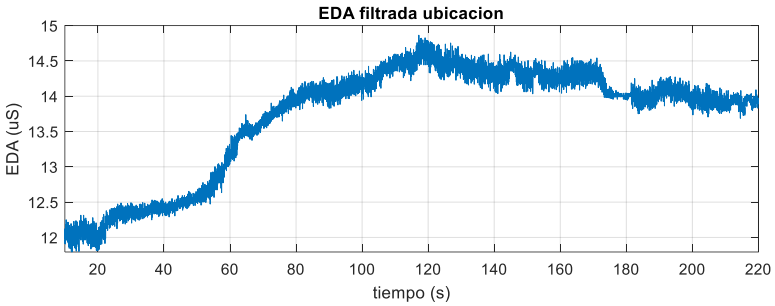
CORRELACIÓN PARTICIPANTE 012	0.3690



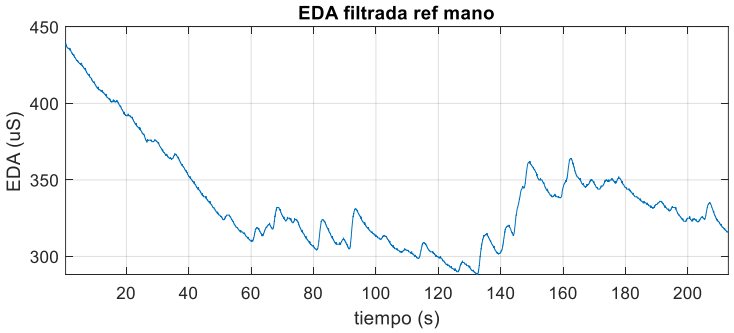


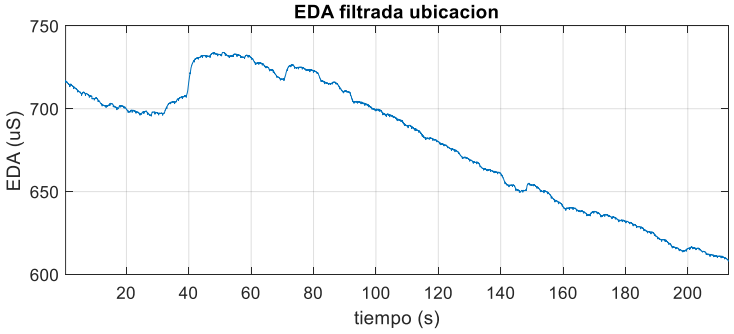
16	APÉNDICE
ELECTRODOS	
CORRELACIÓN PARTICIPANTE 011	0.6594



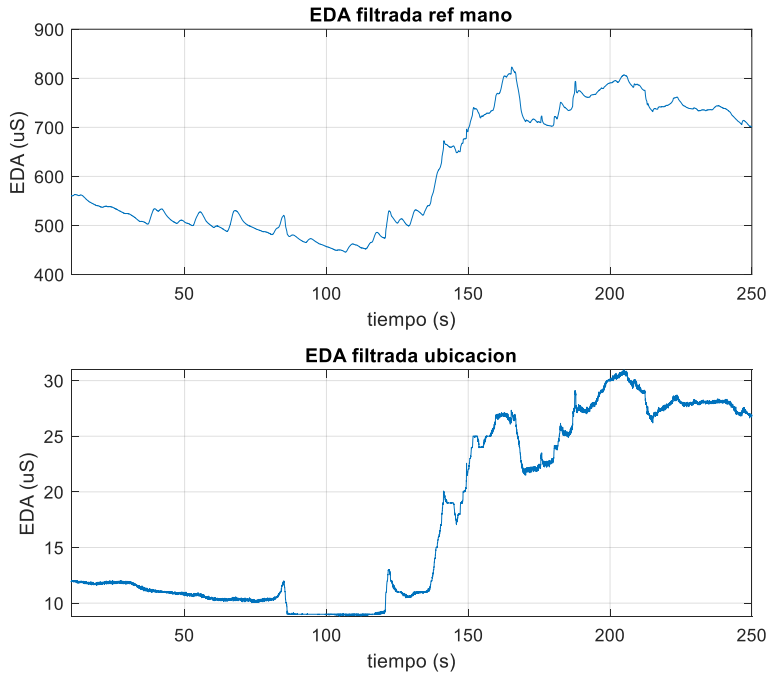


CORRELACIÓN PARTICIPANTE 012	-0.0516

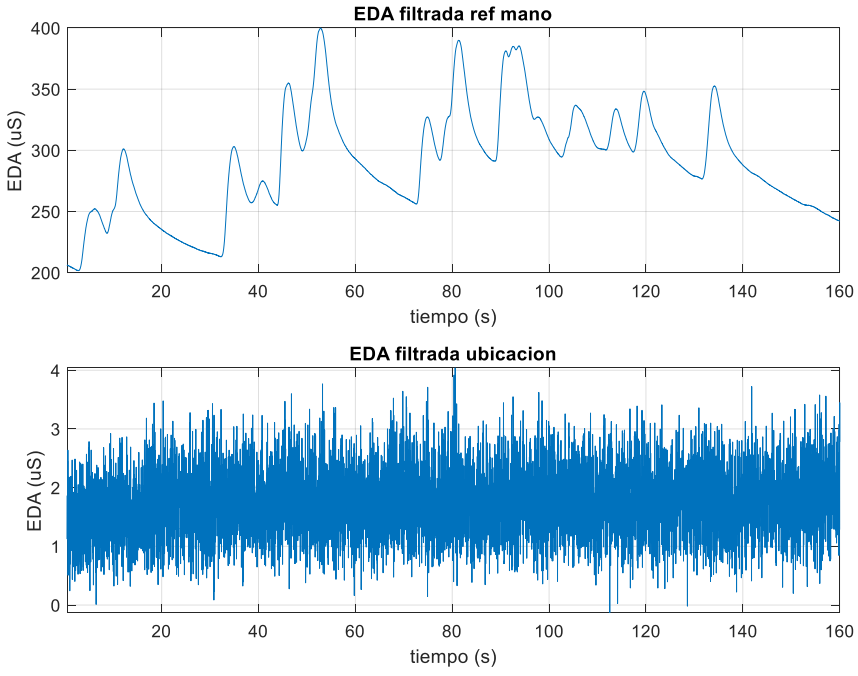


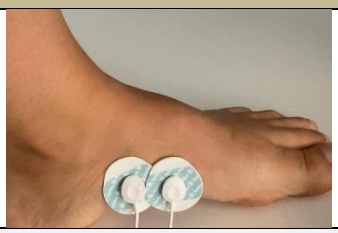


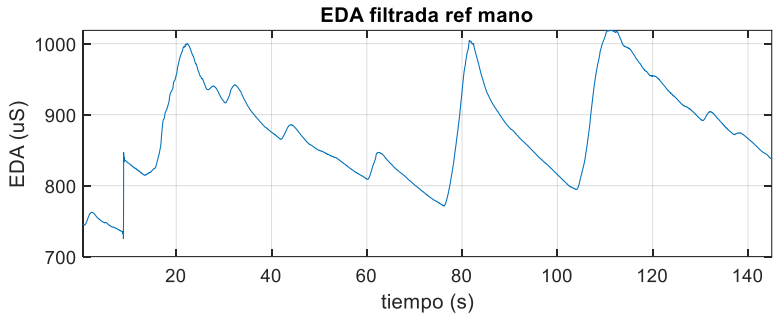
17	TRAPECIO
ELECTRODOS	
CORRELACIÓN PARTICIPANTE 011	0.9598

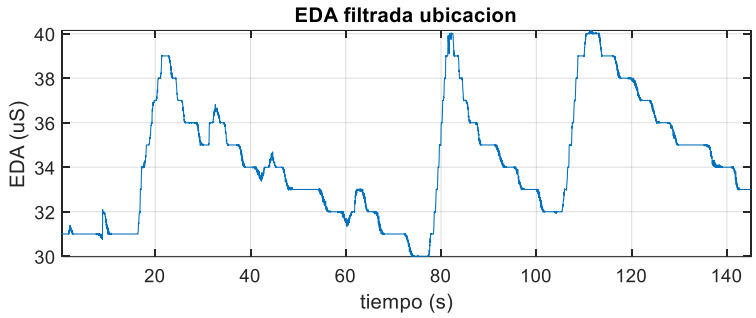


CORRELACIÓN PARTICIPANTE 012	-0.1017



18	PIE
ELECTRODOS	
CORRELACIÓN PARTICIPANTE 001	0.9729





CORRELACIÓN PARTICIPANTE 012	0.9660