UNIVERSIDAD DE LAS PALMAS
DE GRAN CANARIA

ESCUELA INGENIERÍA DE TELECOMUNICACIÓN Y ELECTRÓNICA

TRABAJO FIN DE GRADO

ANÁLISIS DE LA VARIACIÓN EN SEÑALES BIOMÉDICAS FRENTE A ESTÍMULOS EXTERNOS

Titulación: Grado en Ingeniería en Tecnologías de la Telecomunicación

Mención: Sistemas de Telecomunicación

Autor: Acaimo Tejera Fettmilch

Tutor: Carlos Manuel Travieso González

Fecha: Julio 2019

UNIVERSIDAD DE LAS PALMAS
DE GRAN CANARIA

ESCUELA INGENIERÍA DE TELECOMUNICACIÓN Y ELECTRÓNICA

TRABAJO FIN DE GRADO

ANÁLISIS DE LA VARIACIÓN EN SEÑALES BIOMÉDICAS FRENTE A ESTÍMULOS EXTERNOS

HOJA DE EVALUACIÓN

Calificación:_____

Presidente

Fdo.:

Vocal **Secretario/a**

Fdo.: Fdo.:

Fecha: Julio, 2019

Agradecimientos

Mil gracias a todos aquellos que me han ayudado, amigos, familiares y profesores. En especial, a mis padres y a mi abuela, cuya paciencia y confianza siempre admiraré. Sin olvidar a los que ya no están. Ha sido una experiencia digna de vivir gracias a ustedes.

Índice

Índice de figuras

Índice de códigos

Índice de tablas

Capítulo 1. Introducción

1.1 Motivación

La sanidad electrónica se abre camino en la era digital a pasos agigantados y ya está modificando sustancialmente el sistema médico vigente. Dentro de las siete áreas prioritarias de la Agenda Digital para Europa, e-Salud se apoya en la premisa de que pacientes y médicos aprovechen el potencial de las tecnologías de la información y la comunicación (TIC) para intentar solucionar un problema que empieza a ser serio y urgente, y que no es otro, que el de la saturación de la capacidad de atención sanitaria, y no poder por lo tanto, cubrir los estándares mínimos para una atención adecuada.

Según la Organización para la Cooperación y el Desarrollo Económico (O.C.D.E.), los países destinan actualmente a la sanidad un 6% del P.I.B. y se prevé que el gasto sanitario crecerá hasta el 9% en 2030 [1]. Un aspecto a tener muy en cuenta en este sentido, es el aumento de las enfermedades crónico-degenerativas. De hecho, según informes de la Organización Mundial de la Salud (OMS), se espera que en un futuro inmediato (2020) sean las responsables del 73% de las muertes a nivel mundial [2]. Se entiende por enfermedad crónico-degenerativa por aquella en la que tiene lugar un proceso continuo basado en cambios degenerativos en las células, en la cual la funcionalidad de los órganos empeora con el transcurso del tiempo. Ejemplos muy conocidos de enfermedades degenerativas son las del cáncer, diabetes tipo 2, esclerosis, osteoporosis, Parkinson o Alzheimer. Sobre todo en éstas últimas, debido a la necesidad de tratamientos en fases muy iniciales, la detección precoz se ha convertido en uno de los principales focos de investigación en el campo de las enfermedades degenerativas. Es en este punto dónde entran de lleno las posibles las nuevas técnicas de diagnóstico y tratamiento.

En este sentido, las TIC's están demostrando un gran potencial de impacto asistencial clínico y diagnóstico. Así, por ejemplo, la utilización de la telemedicina permite una mayor precocidad en la detección de enfermedades, lo que implica no sólo un mayor éxito clínico con su correspondiente reducción del gasto médico, sino que además, una solución óptima para el desbloqueo de un sistema sanitario que empieza a estar ya seriamente colapsado.

Es aquí, dónde este proyecto encuentra sus mayores motivaciones, enfocándose más a aplicaciones pertenecientes a la telemedicina en casa, y más en concreto a la detección precoz de enfermedades degenerativas de naturaleza neurológica y que puedan implicar una bajada de la respuesta a un estímulo externo.

Surge también una atracción inicial por este proyecto, por el carácter multidisciplinar que posee. El hecho de experimentar con la placa *e-Health* y con sus correspondientes sensores, analizar las posibilidades que ofrece en este ámbito una plataforma libre como es Arduino, como interfaz hardware para la transmisión de datos, y la posibilidad de profundizar con un entorno software tan potente como Matlab, ha sido determinante para decantarme por este proyecto.

Resulta interesante resaltar también la naturaleza *"low cost"* que caracteriza la tecnología usada, algo que dota al proyecto, a un enfoque más real de ser usado por un paciente desde su propio hogar, y ceñirse por lo tanto, a uno de los pilares más importantes dentro del ámbito de la telemedicina, la viabilidad económica.

Aparte del aspecto tecnológico mencionado, con sus atractivas características de auge en telemedicina, carácter multidisciplinar y viabilidad económica, surge también una motivación personal por el estudio del origen y la propagación de las señales corporales. Existe aquí aún mucho por descubrir, debido a la complejidad y fascinante funcionamiento del cuerpo humano. Observar como varían las señales de las personas ante impulsos emotivos de asco y/o miedo ha sido toda una experiencia. No sólo por la cantidad de señales que se pueden llevar a estudio sino también por cuan diferentes llegan a ser por el hecho de tratarse de una persona u otra.

Con el fin de proyectar una visión actual, de la importancia que ya tienen las TIC's en el ámbito de este TFG, y como dan solución a aspectos mencionados anteriormente, se mencionan a continuación una serie de noticias a nivel regional y nacional.

- **Un sistema de telemedicina controla a 100 pacientes de diálisis y hemodiálisis (14 Julio 2019).** De esta forma se monitoriza y realizan ajustes a tiempo real del tratamiento de cada uno de los usuarios, ya sean de Gran canaria, Lanzarote o Fuerteventura [3].

- **El Cabildo de Tenerife invierte casi 500.000 euros en impulsar la telemedicina y descentralización de pruebas para aligerar la TF-5 (9 Mayo 2018).** Según los cálculos del Cabildo, estas medidas eliminarán unos 100 coches

de la TF-5, a razón de unos 1.800 desplazamientos de ida y vuelta a la semana, una medida que contribuirá a mejorar la descongestión del tráfico [4].

- **Hablar con el médico desde el ordenador ya es posible en España (24 Septiembre 2018).** La atención telemática en el ámbito de la salud aumenta en el entorno doméstico pero también en centros escolares y residencias de mayores, donde permite ahorrar tiempos de espera a la sanidad [5].

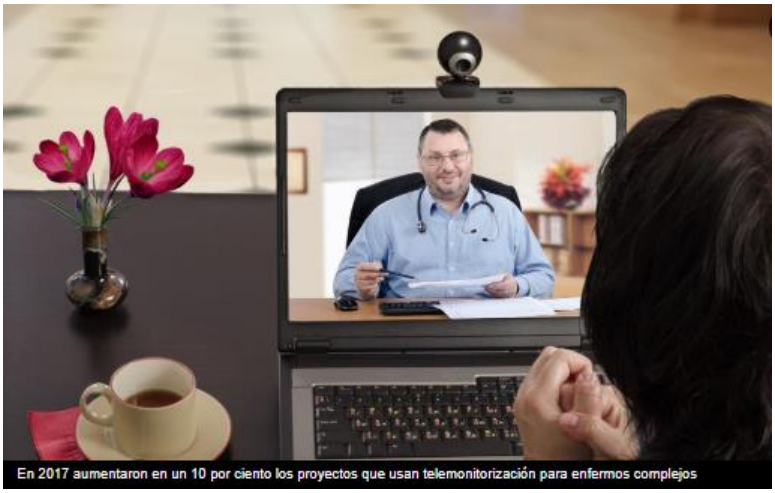


Figura 1. 1 Videoconferencia con el médico

1.2 Estado del arte

Históricamente, esta dinámica comenzó a crecer desde la década del cincuenta del siglo pasado en universidades de Estados Unidos y de Europa, en las que surgieron profesionales para aplicar los principios y el método de la ingeniería a los problemas médicos. No han parado de aparecer desde entonces sensores biomédicos de toda índole. Desde los más comunes como pueden ser los sensores de presión arterial y temperatura, hasta los más novedosos, como los implantados en dientes para obtener información estadística de lo que se come [6] o plantillas inteligentes que testean el comportamiento del pie para prevenir lesiones [7].

La determinación del estado de la salud siempre ha sido una cuestión de medir las funciones más básicas del cuerpo. Antes de la instrumentación, se buscaban indicadores que permitiesen saber el estado de la temperatura corporal, el pulso o la frecuencia respiratoria. Hoy en día, pocos ámbitos necesitan de una monitorización de datos tan

constante como la salud. Un sistema de instrumentación actual, está compuesto, en líneas generales, por:

- Equipos y sistemas de telecomunicación, normalmente de la rama tipo telemático y equipos informáticos.
- Servicios y aplicaciones telemáticas que apoyan la gestión sanitaria.
- Dispositivos terminales específicos para uso médico
- Dispositivos terminales orientados al intercambio de datos, captación de señales biomédicas y de control en el entorno del paciente.

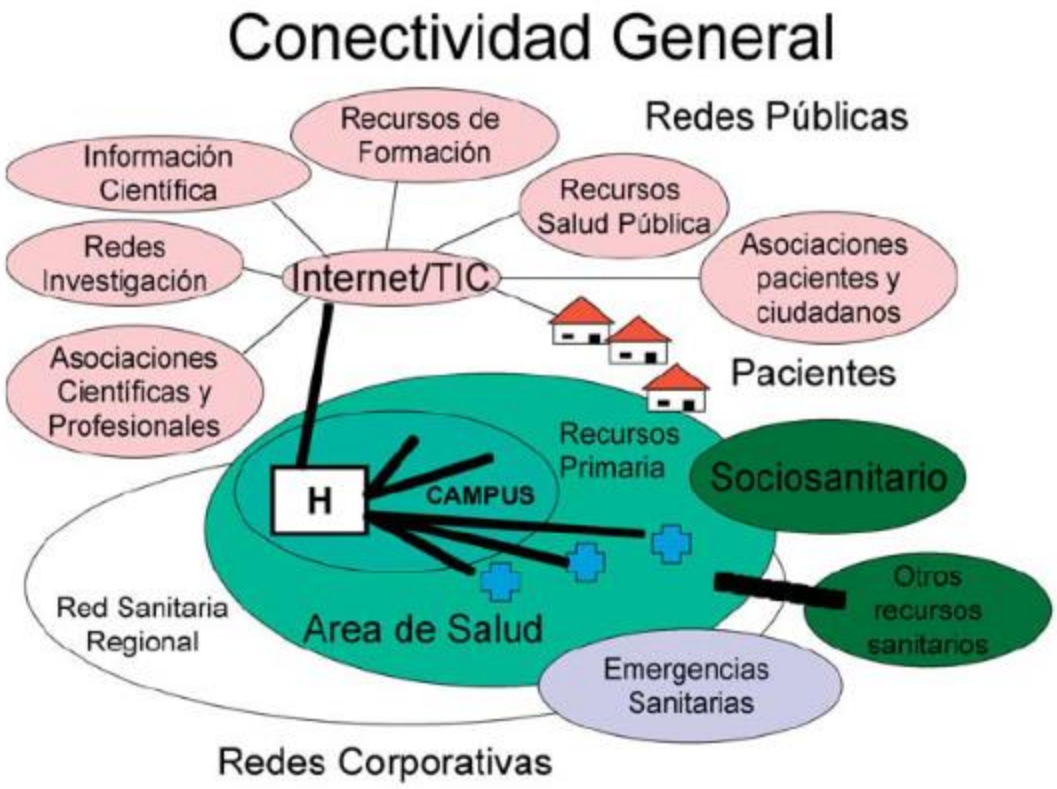


Figura 1. 2 Conectividad entre sectores médicos

Más allá de la instrumentación actual, las aplicaciones que está ofreciendo la telemedicina en el sector clínico actual, sirven de ayuda en muy diversos campos, como por ejemplo:

- **Procesos asistenciales.** A través de las redes de comunicaciones, los profesionales sanitarios, transmiten gran contenido de material médico, transmisión de imágenes, videoconferencia y comunicación de datos, es un factor esencial de la consultas a distancia, que por otro lado, son cada vez más

habituales. Ejemplos del alcance de este tipo de aplicaciones son, envíos de electrocardiogramas de un paciente con Infarto Agudo de Miocardio desde donde se produce la atención sanitaria inicial, o la vigilancia remota de un programa de rehabilitación desde el domicilio del paciente.

- **Servicios de información a ciudadanos.** La educación sanitaria es especialmente importante en el caso de enfermedades crónicas. En algunos casos, por ejemplos casos de Alzheimer, la educación sanitaria no sólo está orientada a los pacientes, sino también, a sus cuidadores. Serán, por lo tanto, todas aquellas aplicaciones que haciendo uso de infraestructuras y comunicaciones, de gran valor, para ofrecer contenidos informativos de calidad y fácilmente comprensibles por los usuarios.

- **Formación a profesionales.** No sólo el paciente y/o cuidador se ven beneficiados por la información médica transmitida por las infraestructuras de telecomunicación. La enseñanza a profesionales alcanza valores mayores, a través de videoconferencias, aplicaciones, revisiones cruzadas o sesiones clínicas virtuales.

Al ver el alcance que tienen todas estas aplicaciones médicas por parte de la telemedicina, no es de extrañar, que las TIC's tengan cada vez más una importancia creciente en el mundo sanitario. De esta manera, buena parte de los nuevos proyectos que acomete el sector están relacionados con estas tecnologías. De hecho, según los resultados de estudios recientes, las inversiones sanitarias muestran un porcentaje significativo en aspectos de nuevos sistemas digitales de radiología (72%), informatización de entradas médicas (64%) y sistemas informáticos centrales (61%) [8].

En relación a Europa, se estima que el mercado de las TIC's para el sector salud seguirá creciendo, por el interés inevitable de poder buscar una vía para la solución de desborde del sistema sanitario y de gasto desproporcionado comentado anteriormente en la sección de motivación. Por lo tanto, de entre las principales motivaciones para la aplicación de las tecnologías de la información en sanidad, se encuentra la mejora de la eficiencia en la gestión. Relación directa con ésta será la mejora de la calidad y viabilidad económica de una asistencia sanitaria tal como demanda la sociedad actual.

1.3 Propuesta

En la línea de trabajo mencionada en la introducción, se enmarcan los objetivos de este estudio, orientado a la monitorización de las señales fisiológicas apropiadas. Observar la viabilidad que tenga, el análisis y parametrización de las señales captadas será, en primera instancia, uno de los objetivos primordiales. Más allá de éstos, el uso de plataformas "*low-cost"*, ya no sólo por ser un estudio universitario, sino por, por la orientación que se le busca al proyecto, tipo monitorización en casa, se antoja también crucial.

El estudio se puede dividir en los siguientes grupos de objetivos:

- Configurar una correcta transferencia de datos capturados por los sensores vía puerto serie.
- Crear una interfaz de usuario que permita, no sólo, una correcta recogida de datos mientras se intenta forzar reacciones ante estímulos de asco y miedo mediante reproducciones de vídeo, sino también, una representación gráfica dinámica de las diferentes variables fisiológicas.
- Analizar y parametrizar dichas variables, para un posterior clasificado, mediante red neuronal que arroje conclusiones.
- Evaluar el sistema y dichas conclusiones

1.4 Estructura del documento

El presente documento se divide en 6 capítulos, además de las referencias bibliográficas.

- **Capítulo 1. Introducción.** Se plantean los motivos y objetivos del proyecto, surgidos de una temática que da pie a unas líneas de investigación para intentar dar solución o variantes a una problemática actual.

- **Capítulo 2. Tecnologías hardware y entornos software utilizados.** Se muestra una visión global, por un lado, de la placa Arduino y de la plataforma de sensores *e-Health*, y por el otro, del entorno Matlab que nos permitirá el análisis de los datos transferidos por los primeros.

- **Capítulo 3. Monitorización y creación de la base de datos.** Visión más detallada de las tareas llevadas a cabo en Matlab, para a través de una interfaz de usuario, crear una base de datos gracias a la recogida de datos de los sensores.

- **Capítulo 4. Base de datos.** En esta sección se da una idea, de qué manera se hace la monitorización a los pacientes, cuántos son, y de qué forma se reparten en los diferentes vídeos.

- **Capítulo 5. Análisis gráfico.** Se recoge aquí, todo lo correspondiente, al tratamiento que se le somete a la base de datos para estar en disposición de obtener conclusiones desde una perspectiva visual, es decir, analizando las gráficas de cada uno de los pacientes de cada uno de los vídeos. Estas acciones son las de, análisis gráfico, segmentación de éstos en estados, parametrización estadística y clasificación mediante una serie de experimentos con redes neuronales.

- **Capítulo 6. Segmentación.** En este apartado se procede a separar los ficheros de texto de la base de datos correspondiente a cada usuario, y separarlos en 5 estados, de tal manera que cada uno está asociado a diferentes momentos de la monitorización.

- **Capítulo 7. Parametrización.** Se usarán aquí diferentes tipos de parámetros con el fin de extraer una serie de propiedades estadísticas de las variables fisiológicas propias de cada parámetro.

- **Capítulo 8. Clasificación.** Mediante el uso de redes neuronales, se realizan una serie de experimentos, para saber, en términos de porcentaje de éxito, qué propiedades de los parámetros de los pacientes, son más adecuados para cumplir los objetivos buscados en este trabajo.

- **Capítulo 9. Conclusiones y líneas futuras.** Se recogen en qué medida se han podido cumplir los objetivos en forma de conclusiones, y abren posibles líneas futuras de investigación surgidas del desarrollo del TFG.

Capítulo 2. Tecnologías hardware y entornos software utilizados

2.1 Arduino

El proyecto Arduino nació en 2003, en el Instituto de Diseño Interactivo de Ivrea, Italia. El proyecto se diseñó con la idea de que los estudiantes de electrónica tuviesen una alternativa más económica a las placas existentes en ese entonces. Resultando una placa con todos los elementos necesarios para conectar periféricos a las entradas y salidas de un microcontrolador, pudiendo ser programada tanto en Windows, macOS como en GNU/Linux.

La característica más destacable de esta plataforma y por la que se ha convertido en uno de los tipos de placas más populares del mundo, es que a diferencia de otras plataformas, no cuenta con un único modelo, sino que ofrece unas bases de hardware abierto para que otras compañías puedan crear sus propias placas. Sin ir más lejos, es lo que hace Libelium, con la placa e-*Health* de sensores que se utiliza en este trabajo.

A groso modo, Arduino es una plataforma hardware de código abierto y licencia libre, dónde su entorno de desarrollo se apoya en un lenguaje de programación *Processing y Wiring* [9]Esto se traduce en un IDE de Arduino que resulta muy familiar y considerablemente sencillo, ya que está escrito en Java pero con un compilador GCC para C/C++ simplificado, con el que se puede controlar multitud de dispositivos conectados a un microcontrolador, pudiéndose heredar todas las funcionalidades Java [10]

Lógicamente, el tipo de periféricos que se quieran usar para mandar datos al microcontrolador dependerá del uso que se desee dar al montaje. Desde teclados, a cámaras para obtener imágenes, como diferentes tipos de sensores. Por el otro lado, la interfaz de salida llevará la información captada a periféricos diversos, que pueden ser típicamente pantallas o altavoces, o también a otras placas o controladores, como es en nuestro caso.

Todo esto hace de Arduino una poderosa herramienta que permite realizar diversos proyectos. Es una tecnología que tiene una rápida curva de aprendizaje con

básicos conocimientos de programación y electrónica, que permite desarrollar proyectos en el ámbito de las Smart Cities, Internet de las cosas, salud, ocio, educación, robótica, etc.

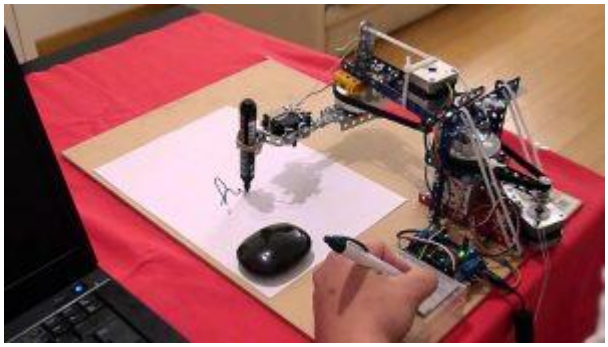


Figura 2. 1 La robótica con Arduino [3]

El hecho de que Arduino sea un proyecto, y no un modelo concreto de placa, se pueden encontrar diferentes tipos de placas. Las hay sencillas o con características mejoradas. La usada en este proyecto, es la placa Arduino Uno R3, y sus características se detallan a continuación.

2.1.1 Arduino Uno R3

La Arduino UNO es la opción más robusta e ideal para iniciarse en la plataforma Arduino, si bien es cierto que no cuenta con una capacidad de procesamiento elevada, se considera más que suficiente para el requerido en este TFG. Es una placa basada en el microcontrolador ATmega328P [11]. Tiene 14 pines de entrada/salida digital, 6 entradas analógicas, un cristal de 16Mhz, conexión USB, conector jack de alimentación, terminales para conexión ICSP y un botón de reseteo [12].

Figura 2. 2 Arduino UNO

Sus especificaciones son las siguientes:

microcontrolador	ATmega328
voltaje operativo	5V
voltaje entrada recomendado	7-12V
voltaje entrada límite	6-20V
pines E/S digitales	14
pines entrada analógica	6
corriente por pin E/S	40 mA
corriente pin 3.3V	50 mA
memoria Flash	32 KB
SRAM	2 KB
EEPROM	1 KB
velocidad reloj	16 MHz
peso	25 gramos

Tabla 2. 1 Especificaciones Arduino UNO R3 [6]

2.2 Plataforma de sensores e-Health

Antes de introducirnos en las características y especificaciones de esta plataforma, se entiende interesante un leve inciso teórico sobre los sensores biomédicos y las señales fisiológicas que se miden.

Los sensores biomédicos son usados de forma rutinaria en la medicina, con la finalidad de medir una gran variedad de variables fisiológicas y en líneas generales se entienden por aquellos dispositivos que transforman una determinada magnitud física en una magnitud eléctrica.

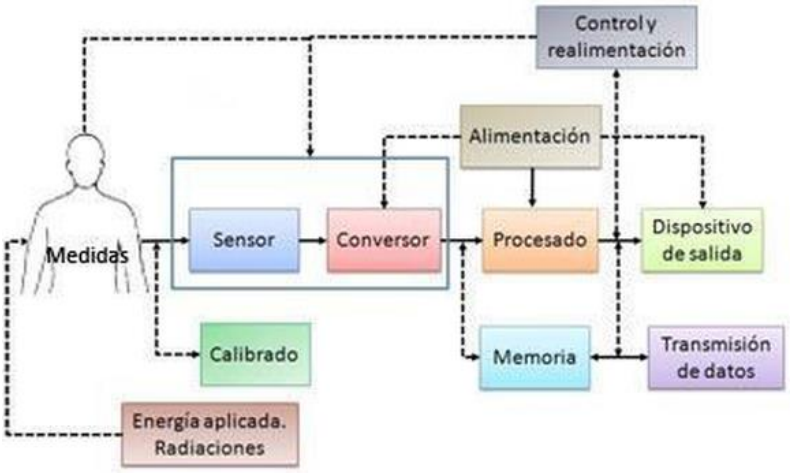


Figura 2. 3Diagrama sistema instrumentación biomédica [7]

El sensor, idealmente, sólo debería responder a la energía presente en la medida que se desea realizar y excluir las demás. Además, debe poseer una interfaz con el tejido o sistema vivo de forma que no interfiera en éste, debe de minimizar la energía extraída y ser lo menos invasivo posible. Por lo general son clasificados en relación a la cantidad a ser medida y típicamente son categorizados como físicos, eléctricos y químicos dependiendo específicamente de su aplicación. Los biosensores, poseen dos componentes distintivos, un elemento de reconocimiento biológico y una estructura de soporte, que a su vez actúa como transductor.

Como se había comentado, en este trabajo la plataforma usada para medir ciertas variables fisiológicas, es la *e-Health* Sensor Platform diseñada por CookingHacks. Ésta permite a los usuarios de Arduino y Raspberry Pi realizar aplicaciones biométricas y médicas donde se necesite monitorizar el cuerpo a través de 9 sensores diferentes. Puede ser alimentada a través del Arduino por una fuente de alimentación externa o por el puerto USB.

Figura 2. 4 *e-Health* en Arduino [8]

La plataforma de sensores nos permitirán medir las siguientes variables fisiológicas.

2.2.1 Pulso y oxígeno en sangre (SPO2)

La oximetría de pulso es un método no invasivo para indicar la saturación arterial de oxígeno de la hemoglobina funcional. La saturación de oxígeno es la cantidad de oxígeno disuelta en la sangre. El flujo sanguíneo se ve afectado por la concentración de la hemoglobina desoxigenada y oxigenada [13], y sus coeficientes de absorción se miden utilizando dos longitudes de onda correspondientes al espectro de luz roja y al de la luz infrarroja. Debido a que estas hemoglobinas tienen diferentes absorciones a estas longitudes de onda, un fotodetector percibe la luz no absorbida de los led's para calcular la saturación arterial de oxígeno.

Los rangos normales aceptables para los pacientes son del 95 al 99 por ciento, indicando problemas hipóxicos los pacientes que dan valores entre el 88 y 94 por ciento, y en el otro extremo, los valores del 100 por ciento pueden indicar envenenamiento por monóxido de carbono.

El sensor debe estar conectado a Arduino o Raspberry Pi, y no usar batería externa/interna.

Figura 2. 5 Pulsioxímetro

Debido a que una de las causas de hipoxemia son las emociones fuertes, que aceleran el corazón y las respiración [14] será el oxígeno en sangre una de las variables fisiológicas a analizar en este TFG.

2.2.2 Electrocardiograma (ECG)

Es uno de los exámenes médicos más utilizados en la actualidad. Es muy útil en gran variedad de patologías cardíacas que han sido invaluables por los médicos durante décadas. Tiene la gran ventaja que sirve para valorar el estado cardiaco de una forma no invasiva.

Es importante estar atento a que cada uno de los sensores o parches tiene un polo, por lo que la colocación de éstos debe de ser como indica la figura.

Figura 2. 6 Colocación parches ECG

De tal manera que el polo positivo (rojo) debe ir colocado en el pectoral derecho, el negativo (negro) en la parte izquierda del abdomen y finalmente, el neutro (blanco) en el pectoral izquierdo.

Aunque recientes estudios demostraron que nuestros pensamientos, percepciones y reacciones emocionales son transmitidas del cerebro al corazón mediante las ramas conocidas como simpática y parasimpática [15], pudiendo perfectamente lanzar información valiosa para los fines de este TFG, debido a no encontrarnos en consultas médicas en el momento de las medidas y por consiguiente existir la posibilidad de incomodar al paciente por el hecho de tener que desnudarse el tronco superior, no se usó este sensor.

2.2.3 Flujo de aire

Un indicador amplio de inestabilidad fisiológica importante suelen ser las tasas respiratorias anormales de tal manera que la frecuencia respiratoria es uno de los primeros indicadores de alguna afección, pudiendo este sensor proporcionar una alerta precoz de hipoxemia o apnea.

Este dispositivo consiste en un hilo flexible que encaja detrás de las orejas, y un conjunto de dos dientes que se colocan en las fosas nasales. La respiración se mide por estas púas.

Un ser humano adulto normal tiene una frecuencia respiratoria de 15-30 respiraciones por minuto.

Ante un estímulo estresante, el cuerpo se prepara de manera automática, tanto para una reacción como para escapar del estímulo estresante. Uno de los típicos síntomas son, pupilas dilatadas, sudoración (influirá en la conductividad de la piel), tensión muscular, taquicardia, aumento de la tensión arterial y también aumento de la frecuencia respiratoria [16]. Por la tanto, aunque fuese un poco complicada la colocación óptima del sensor, debido a la posible viabilidad de éxito de esta señal fisiológica, fue una de las elegidas para este estudio.

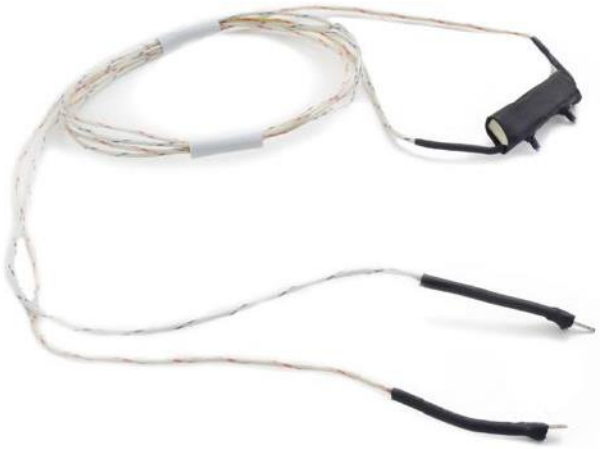


Figura 2. 7 Sensor de flujo de aire

2.2.4 Temperatura corporal

Aunque el parámetro de la temperatura corporal puede parecer en un principio trivial y poco innovador, lo cierto es, que la regulación de la temperatura corporal figura entre las funciones más importantes de cualquier organismo.

El centro termorregulador se haya en el hipotálamo. Es aquí donde se analiza la información que proviene de otras regiones del cerebro, médula espinal, tejidos y sensores térmicos periféricos de la piel. Los mecanismos generados por éstos incluyen la transpiración, la vasodilatación para controlar el sobrecalentamiento y, por el contrario, la vasoconstricción para evitar temperaturas demasiado bajas. El cuerpo humano sano se autorregula con una baremo de +/- 0.2º C con respecto al valor normal, el cual, es aceptado comúnmente como entre 6.5º C y 37.5 º C [17].

Figura 2. 8 Sensor de temperatura

Otro aspecto a tener en cuenta a la hora de medir y no sacar conclusiones precipitadas, es que el organismo varía dependiendo de la hora del día y del nivel de actividad de la persona, con temperaturas más bajas por la mañana y temperaturas más altas por la tarde y por la noche, a medida que cambian las necesidades y actividades del cuerpo.

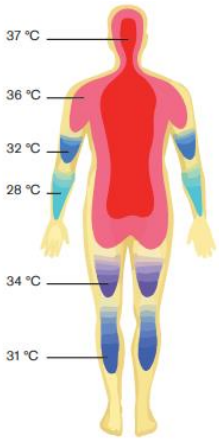


Figura 2. 9 Temperaturas normales en diferentes zonas del cuerpo [14]

Se demuestra en ciertas investigaciones [18], que dependiendo de diferentes estímulos emocionales de distinta naturaleza, ciertas zonas del cuerpo se activan o desactivan en consecuencia, aspecto que se puede poner de manifiesto a nivel de temperatura corporal. Esta investigación se apoya en el hecho científico que se explicaba en la sección del sensor del flujo de aire, y es el de que, nuestro sistema emocional en el cerebro envía señales al cuerpo para que podamos lidiar con nuestra situación.

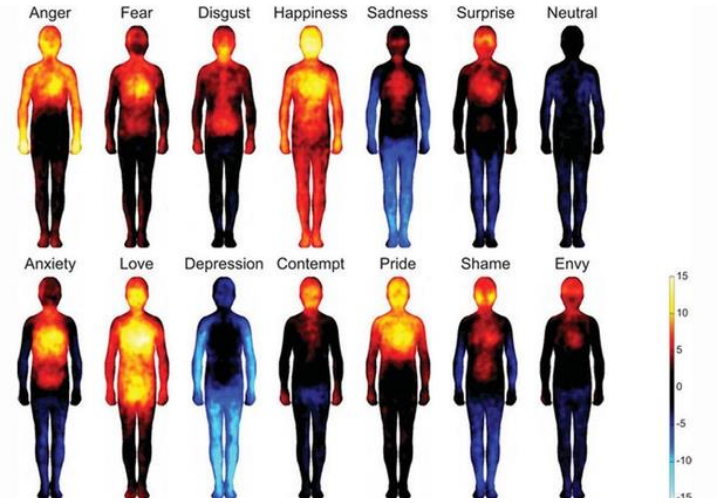


Figura 2. 10 Temperatura según la emoción [16]

En la web oficial del fabricante se menciona un aspecto que al final influyó en las medidas de este trabajo en términos de valor absoluto, y es el de la precisión de este sensor. Se entiende que la información de importancia para este estudio radica en la variabilidad de la señal fisiológica, no siendo tan importante el valor absoluto. Aun así, se intentó modificar la librería de este sensor, para que tomara los datos de una manera empírica, es decir, pasándole los valores de una tabla del fabricante, ya que, de la librería original, los valores arrojados eran asociados a estados febriles. Los cambios introducidos se explican con más detalle en este capítulo en la sección de librerías, pero adelantamos que los nuevos valores fueron similares a la librería original. Lo ideal hubiera sido llevarse a cabo la calibración del sensor midiendo con un multímetro los valores reales de las resistencias implicadas en la resolución del valor de temperatura correspondiente e introducirlas en la librería original. Debido, a lo ya comentado sobre que nuestro caso lo que interesa es la variabilidad de la señal, se deja esta calibración para el apartado de líneas futuras.

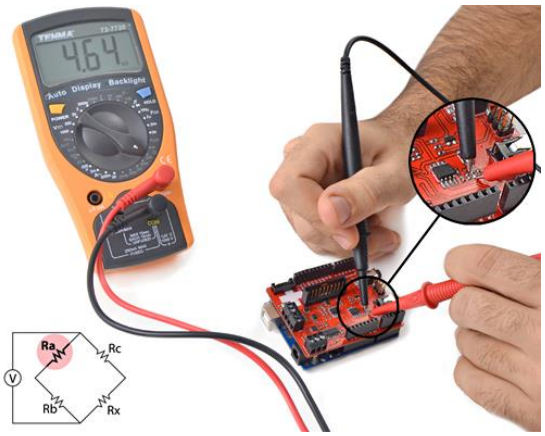


Figura 2. 11 Proceso de calibración [17]

2.2.5 Presión sanguínea

Se define como la presión de la sangre en las arterias a medida que es bombeada por la red sanguínea. Se registra con dos números, la presión sistólica (cuando el corazón late) y la diastólica (entre latidos). Las medidas se ven alteradas por diversos factores, como la posición del cuerpo, la respiración, el estado emocional, el ejercicio y el sueño. A pesar de su relación con el estado emocional, algo que a priori resulta interesante en este TFG, por su dependencia de tantos factores y por el hecho de entorpecer mucho su colocación con el resto de sensores, por su aparatosidad e incomodidad para el paciente, se decidió finalmente no decantarnos por este sensor. Además parece ser, que más que actuar en consecuencia con los estímulos emocionales, lo hace con el estrés generado por éstos, algo que no resulta evidente que se pueda medir sobre la marcha **[19]**.

Figura 2. 12 Sensor de presión arterial

2.2.6 Posición del paciente

El acelerómetro controla cinco posiciones diferentes del paciente (de pie/sentado, posición supina, boca abajo, izquierda y derecha).

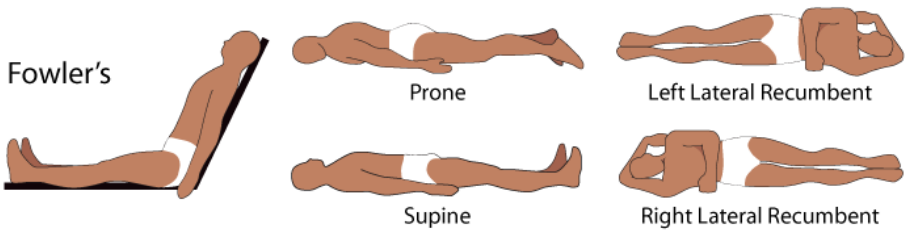


Figura 2. 13 Posiciones detectadas por el sensor

Este sensor está más orientado a enfermedades del sueño y poder analizar los movimientos durante éste y detectar patrones irregulares. Es muy usado también en temas de alerta para emergencias al ayudar a detectar desmayos o caídas de personas mayores o con discapacidades. Difícilmente se le puede aprovechar alguna utilidad para los propósitos de nuestro estudio, así que se descartó su uso.

Figura 2. 14 Sensor de posición del paciente

2.2.7 Respuesta galvánica de la piel (GSR)

También conocida como conductancia de la piel, es un método para medir la conductancia eléctrica de la piel, que varía con el nivel de humedad que se presente en el momento de la medición. Esto es de sumo interés para esta investigación, ya que las glándulas sudoríparas son controladas por el sistema nervioso simpático, por lo que momentos de fuerte emoción, cambian la resistencia eléctrica de la piel, por lo que es muy usado como indicador de la excitación psicológica o fisiológica.

El sensor de respuesta de la piel galvánica mide la conductancia eléctrica entre 2 puntos, y es esencialmente un tipo de ohmímetro. Se medirá en los dedos de la palma aprovechando la respuesta galvánica basada en el sudor propio del cuerpo. Cuando se produce un nivel elevado de sudoración, la resistencia eléctrica de la piel disminuye. Una piel seca registra mucha mayor resistencia. Emociones tales como la excitación, estrés, choque, etc. puede hacer fluctuar la conductividad de la piel. Es tal la conexión de estas medidas con las ramas neurológicas del cuerpo, que se están desarrollando nuevos dispositivos médicos comerciales cada vez más portátiles (brazaletes, relojes) para monitorizar actividades de investigación en el campo de la neurociencia en entornos fuera del laboratorio [20].

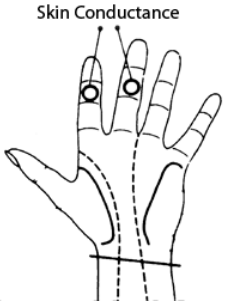


Figura 2. 15 Puntos de medida de conductancia

Al ser un sensor de tipo ohmímetro, al igual que el de la temperatura pude requerir una calibración para mejorar la precisión de las medidas. El proceso será idéntico al explicado en la parte del sensor de temperatura corporal. Se medirán los valores reales de voltaje y se introducirá en la librería de este sensor.

Figura 2. 16 Medida voltaje correspondiente al sensor GSR

```
eHealth.cpp
347  //|*****************************************************************************
348  //| Name: getSkinResistance()                                                  *
349  //| Description: Returns the value of skin resistance.                         *
350  //| Param : void                                                               *
351  //| Returns: float with the value of skin resistance                           *
352  //| Example: float resistance = eHealth.getSkinResistance();                   *
353  //|*****************************************************************************
354
355  float eHealthClass::getSkinResistance(void)
356  {
357    // Local variable declaration.
358    float resistance;
359    float conductance;
360
361    // Read an analogic value from analogic2 pin.
362    float sensorValue = analogRead(A2);
363    float voltage = (sensorValue * 5.0) / 1023;
364
365    delay(2);
366    conductance = 2*((voltage - 0.5) / 100000);
367
368    //Conductance calcultacion
369    resistance = 1 / conductance;
370    delay(2);
371
372    return resistance;
373  }
```

Código 2. 1 Valor por defecto del voltaje en librería del GSR

2.2.8 Glucosa en sangre

Con este sensor (glucómetro) se determina la concentración aproximada de glucosa en la sangre. Se coloca una gota de sangre en una tira de prueba desechable que le medidor lee. Se muestra normalmente en niveles de mg/dl. Al igual que en el caso de la presión arterial su fluctuación por motivos emocionales se ve más vinculada a estados mentales tipo estrés, tras la movilización de las hormonas contrarreguladoras u “hormonas del estrés” (adrenalina, cortisol, etc.) las cuales tienen un efecto directo en los niveles de glucemia [21]. Con la velocidad con la que se pueda medir una fluctuación por un estímulo emocional, debido a que es un efecto indirecto, se antoja por lo tanto complicado. Esto unido a lo invasivo que resulta la medición (extracción de sangre), este sensor también es desestimado para este estudio.

Figura 2. 17 Glucómetro

2.2.9 Electromiograma (EMG)

Mide la actividad eléctrica de los músculos en reposo y durante la contracción. Las señales pueden analizarse para medir niveles de activación, biomecánica del movimiento, etc. Aunque tiene una fuerte relación neurológica, este sensor va más orientado a control para dispositivos protésicos o trastornos del control motor, algo que podría ayudar a relacionarse con una reacción a un estímulo emocional, pero ya de tipo avanzado, por lo que no serviría para un diagnóstico precoz.

Al igual que en el electrocardiograma es de vital importancia colocar los sensores en las zonas apropiadas.

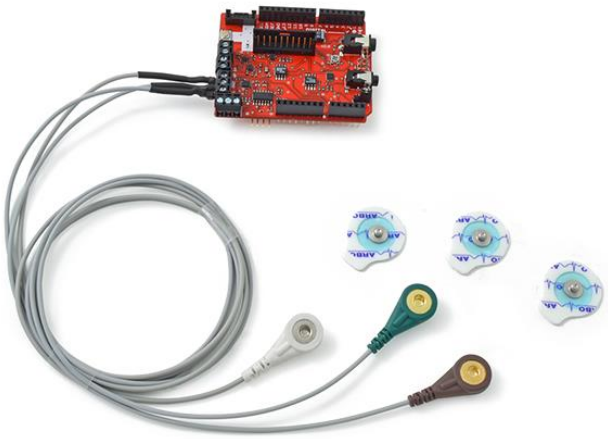


Figura 2. 18 Sensores EMG

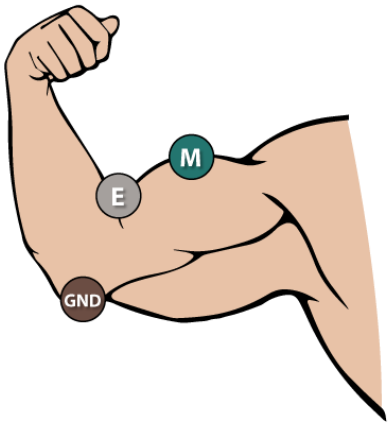


Figura 2. 19 Colocación correcta EMG

2.3 Entornos software

2.3.1 Arduino

Otro factor importante en el éxito de Arduino, es la comunidad que apoya todo este desarrollo, comparte conocimiento, elabora librerías para facilitar el uso de Arduino y publica sus proyectos para que puedan ser replicados, mejorados o ser base para otro proyecto relacionado. Como se comentó al comienzo de este capítulo el entorno de desarrollo integrado (IDE) de Arduino resulta bastante intuitiva y el lenguaje C++ simplificado facilita bastante la programación, que en nuestro caso, es la del puerto serie para la correcta transmisión de datos desde los sensores al ordenador. Es de software libre y se puede utilizar en cualquier placa de Arduino.

En el siguiente Capítulo, se explicará con más detalle de qué manera se lleva a cabo la transmisión de datos por puerto serie y qué facilidades nos ofrece esta IDE.

2.3.2 Librerías e-Health

La plataforma de sensores *e-Health* cuenta con una librería de C++ que permite leer fácilmente todos los sensores y enviar información mediante cualquiera de las interfaces disponibles. Los archivos de las librerías vienen en dos carpetas separadas, *"eHealth"* y "*PinChangeInt".* Ésta última es necesaria solo cuando se utiliza el sensor pulsioxímetro (pulso y saturación de oxígeno). Para la correcta lectura y transmisión de datos de los sensores, como para el reconocimiento en el puerto USB del montaje e-Health/Arduino es muy recomendable asegurarse de instalar las librerías correspondientes a la versión exacta de la placa que se va a utilizar. Ya que si se instala una versión más básica, es posible que debas añadir más librerías a posteriori, para poder usar cierto tipo de datos. Este asunto se explicará también con más detalle en el siguiente capítulo. Así como los cambios en la librería del sensor de temperatura. En esta sección nos limitaremos a introducir un breve inciso de cuáles son los comandos en librerías que manejan la recogida de datos de los sensores.

- Pulso y oxígeno en sangre (SPO2):

initPulsioximeter ()	Inicializa el sensor de pulsioximeter
readPulsioximeter ()	Lee un valor del sensor de pulsioximeter
getBPM ()	Devuelve los latidos del corazón por minuto

getOxygenSaturation ()	Devuelve la saturación de oxígeno en la sangre en porcentaje

Tabla 2. 2 Funciones SPO2

-Electrocardiograma (ECG):

getECG ()	Devuelve un valor analógico para representar la Electrocardiografía

Tabla 2. 3 Función ECG

-Flujo de aire:

getAirFlow ()	Devuelve un valor analógico para representar el flujo de aire
airFlowWave ()	Imprime la forma de onda del flujo de aire en el monitor en serie

Tabla 2. 4 Funciones flujo de aire

-Temperatura corporal:

getTemperature ()	Devuelve la temperatura corporal

Tabla 2. 5 Función temperatura

-Presión sanguínea:

initBloodPressureSensor ()	Inicializa y mide el sensor de presión arterial
getBloodPressureSensor ()	Devuelve el número de datos almacenados en el sensor de presión arterial
getSystolicPressure (i)	Devuelve el valor del número de presión sistólica i
getDiastolicPressure (i)	Devuelve el valor del número de presión diastólica i

Tabla 2. 6 Funciones presión sanguínea

-Posición del paciente:

initPositionSensor ()	Inicializa el sensor de posición
getBodyPosition ()	Devuelve la posición del cuerpo
printPosition ()	Imprime la posición actual del cuerpo

Tabla 2. 7 Funciones acelerómetro

- Respuesta galvánica de la piel (GSR):

getSkinConductance ()	Devuelve el valor de la conductancia de la piel
getSkinResistance ()	Devuelve el valor de la resistencia de la piel
getSkinConductanceVoltage ()	Devuelve el valor de la conductancia de la piel en voltaje

Tabla 2. 8 Funciones GSR

-Glucosa en sangre:

readGlucometer ()	Lea los valores almacenados en el glucómetro
getGlucometerLength ()	Devuelve el número de datos almacenados en el glucómetro
numberToMonth ()	Convierte la variable del mes de numérico a carácter

Tabla 2. 9 Funciones glucómetro

-Electromiograma (EMG):

getEMG ()	Devuelve un valor analógico para representar la Electromiografía

Tabla 2. 10 Funciones EMG

2.3.3 Interfaz de usuario en Matlab

Se parte de un código inicial [22], que consiste en un programa que recoge datos por puerto serie del sensor ECG, para, mediante una interfaz de usuario, guardar datos personales de un paciente y graficar los parámetros ECG, con el fin de identificar personas mediante identificación biomédica. Se detallará en profundidad en el capítulo 3, cuál es su aspecto inicial y qué cambios se le introduce para los objetivos de este trabajo.

Capítulo 3. Monitorización y creación de la base de datos

En este capítulo se explicarán las tareas llevadas a cabo tanto en Arduino, para la correcta transmisión de datos desde los sensores por el puerto serie, como en el entorno Matlab, para la recogida de datos, representación de éstos e idónea conversión a formatos que se presten a un posterior análisis. Todo ello en una interfaz de usuario que permita el correcto sincronismo entre toma de datos y reproducción de vídeos, de tal manera, que permita la captura de la reacción ante estímulos emocionales presentes en éstos. El resultado consistirá en la generación de una serie de archivos que conformarán nuestra base de datos.

3.1 Plataforma Arduino

3.1.1 Transmisión por puerto serie con Arduino

Las señales fisiológicas recogidas por los diferentes sensores serán enviados por puerto serie cada cierto tiempo y orden dentro de una trama, que se especificará a través de la IDE de Arduino y cuyo microcontrolador llevará a cabo de la siguiente manera.

```
#include <eHealthDisplay.h>

#include <QueueList.h>
#include <Event.h>
#include <Timer.h>
#include <eHealth.h>
#include <PinChangeInt.h>

Timer t;
//Timer t2;
int cont =0;
char delimitador = '*';
char finPaquete = '\n';
QueueList<float> cola;
```

Código 3. 1 Inclusión librerías *eHealth*

Las librerías básicas instaladas son las *eHealth* y *pinChangeInt*. Ésta última es necesaria para el manejo de las interrupciones, lo cual es necesario en la lectura del pulsioxímetro [23]. Para poder manejar listas, *timers* y eventos, hubo que añadir las otras librerías.

```
void setup()
{
  Serial.begin(9600);
  eHealth.initPulsioximeter();
  t.every(250, enviarVariables);
  PCintPort::attachInterrupt( 6, readPulsioximeter, RISING);
}

void loop()
{
  t.update();
```

Código 3. 2 Timer *enviarVariables*

Se configura el puerto serie a 9600 baudios y se inicia el pulsioxímetro. El *timer* llamará a la función *enviarVariables* cada 250 ms. En esta parte, se eliminó un segundo *timer que invocaba la función recogerAirflow,* esta función se limitaba a meter en cola el valor de *airflow.* Como se comentó en el capítulo anterior, no se usaron diversos sensores, por lo que un control extra sobre el envío de una señal no se creyó necesario. Las interrupciones se manejan mediante *attachInterrupt*, a la que se pasa por parámetro un 6 por el tipo de placa usada, la variable a controlar y por último el parámetro *Rising* que indica que se dispara la interrupción cuando el pin pasa de nivel bajo a alto [24].

```
void enviarVariables()
{
  Serial.print("Cabecera");
  Serial.print(delimitador);
  Serial.print(eHealth.getBPM());
  Serial.print(delimitador);
  Serial.print(eHealth.getOxygenSaturation());
  Serial.print(delimitador);
  Serial.print(eHealth.getTemperature());
  Serial.print(delimitador);
  Serial.print(eHealth.getSkinConductance());
  //ACAYMO
  Serial.print(delimitador);
  Serial.print(eHealth.getAirFlow());


  while (!cola.isEmpty ()){
    Serial.print(delimitador);
    Serial.print(cola.pop());

  }
  Serial.print(finPaquete);
}
```

Código 3. 3 Función *enviarVariables*

La función *enviarVariables* se limita simplemente a crear la trama recogiendo datos de los sensores y separándolos por el delimitador *. Mientras la cola no esté vacía se sacan valores de ésta.

```
void readPulsioximeter() {
  cont ++;
  if (cont == 100) { //Get only one of  100 measures to reduce the latency
    eHealth.readPulsioximeter();
    cont = 0;
  }
}
```

Código 3. 4 Lectura pulsioxímetro

Se implementa por último la lectura del pulsioxímetro.

A modo de resumen, los datos se almacenarán en una lista que contará con una cabecera para identificar el inicio del paquete, y ésta irá seguida de los diferentes datos de los sensores separados por el identificador *. Con el mismo propósito de la cabecera se envía también un fin de paquete (salto de línea). La trama que se enviará por el puerto serie tendrá el siguiente aspecto:

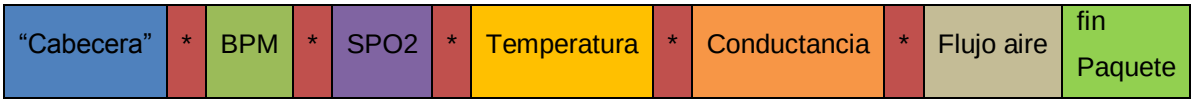


Figura 3. 1 Trama de datos

El correcto funcionamiento del envío por el puerto serie se puede monitorear por el Monitor Serial que se incluyen en las herramientas de la IDE.

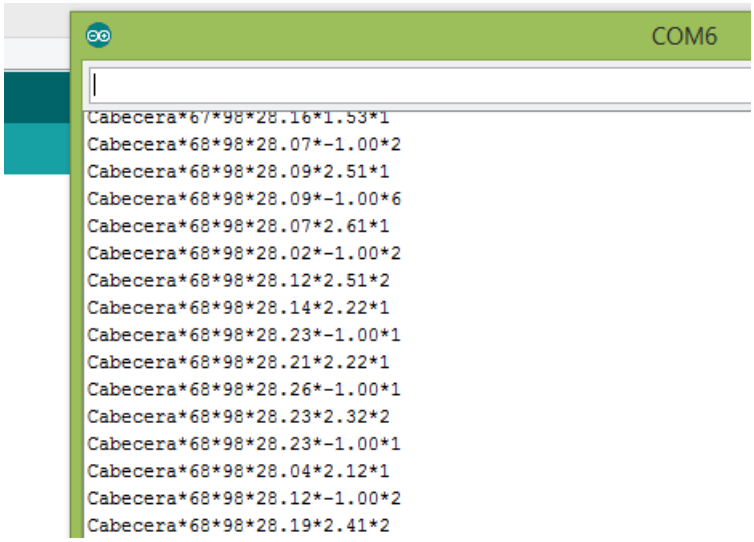


Figura 3. 2 Monitor Serial del IDE

De esta manera se ve la trasferencia de datos por puerto serie 6 por el Monitor Serial

3.1.2 Cambios en librería

Aparte de programar la transferencia de datos por puerto serie, se introdujeron cambios en la librería *eHealth*. Concretamente a la correspondiente de la recogida de datos del sensor de temperatura. Esto se hizo debido a que el sensor devolvía valores demasiado altos, aquellos que normalmente indican estados febriles en el paciente. Como se aprecia en la figura siguiente, en la librería de *eHealth* se hace la correlación resistencia-temperatura con unos datos que no se sabe muy bien de dónde salen. Por lo tanto, se intentó solucionar eliminando esta programación por aproximación y hacerlo de una forma empírica, introduciendo directamente los valores de la curva de comportamiento del sensor de temperatura [25].

```
float eHealthClass::getTemperature(void)
{
    //Local variables
    float Temperature; //Corporal Temperature
    float Resistance;  //Resistance of sensor.
    float ganancia=5.0;
    float Vcc=3.3;
    float RefTension=3.0; // Voltage Reference of Wheatstone bridge.
    float Ra=4700.0; //Wheatstone bridge resistance.
    float Rc=4700.0; //Wheatstone bridge resistance.
    float Rb=821.0; //Wheatstone bridge resistance.
    int sensorValue = analogRead(A3);

    float voltage2=((float)sensorValue*Vcc)/1023; // binary to voltage conversion

    // Wheatstone bridge output voltage.
    voltage2=voltage2/ganancia;
    // Resistance sensor calculate
    float aux=(voltage2/RefTension)+Rb/(Rb+Ra);
    Resistance=Rc*aux/(1-aux);
    if (Resistance >=1822.8) {
        // if temperature between 25°C and 29.9°C. R(t°)=6638.20457*(0.95768)^t
        Temperature=log(Resistance/6638.20457)/log(0.95768);
    } else {
        if (Resistance >=1477.1){
            // if temperature between 30°C and 34.9°C. R(t°)=6403.49306*(0.95883)^t
                Temperature=log(Resistance/6403.49306)/log(0.95883);
        } else {
            if (Resistance >=1204.8){
            // if temperature between 35°C and 39.9°C. R(t°)=6118.01620*(0.96008)^t
                Temperature=log(Resistance/6118.01620)/log(0.96008);
            }
            •
            •
            •
```

Código 3. 5 *getTemperature* de *eHealth*

En la siguiente figura se aprecia los cambios introducidos.

prueba_getTemperature

```
#include <stdio.h>
#include <stdlib.h>

float lectura;
float coeficienteResistivo;
float ganancia=4.65;
float Vcc=3.3;
float Ra = 4700.0;
float Rb = 820.0;
float Rc = 4700.0;
float vRef = 3.0;
float aux=0.0;
  float resistencia=0.0;
  float convertido = 0.0;
  float conversion = 0.0;
  float laTemperatura=0.0;
  float voltage2=0.0;

float temperatura[] = { 7.3554,6.9894,6.6444,6.3194,6.0114,5.7194,5.4444,5.1834,4.9374,4.7034,
                        4.4824,4.2734,4.0744,3.8864,3.7084,3.5394,3.3784,3.2264,3.0814,2.9444,
                        2.8144,2.6904,2.5724,2.4604,2.3544,2.2524,2.1564,2.0644,1.9774,1.8944,
                        1.8154,1.7394,1.6674,1.5994,1.5334,1.4714,1.4124,1.3554,1.3014,1.2494,
                        1.2004,1.1524,1.1074,1.0644,1.0234,0.9842,0.9466,0.9106,0.8762,0.8432,
                        0.8117,0.7815,0.7526,0.7249,0.6983,0.6729,0.6485,0.6252,0.6028,0.5813,
                        0.5607,0.5409,0.5219,0.5037,0.4862,0.4694,0.4533,0.4378,0.4229,0.4086,
                        0.3949 };
```

Figura 3. 3 Valores de la curva del sensor de temperatura

La curva de valores se extrajo del *datasheet* del fabricante [26]. Así como el esquemático para saber en qué pin se lleva a cabo la lectura del valor analógico (A3), esto se encuentra en la web oficial de *e-Health* [27].

Lamentablemente el resultado no fue el deseado. Daba la sensación de que había bajado un poco pero seguían estando por encima de los valores normales para un paciente sano. Finalmente se decidió usar el sensor en este estado, ya que cómo se explicaba en el capítulo 2, la información interesante para este TFG no son los valores absolutos, sino de qué manera cambian cuando se produce una reacción a un estímulo emocional (miedo/asco en vídeos). En el capítulo 6, se propone como líneas futuras, hacer un calibrado al sensor.

3.2 Interfaz de usuario en entorno Matlab

El hacer que la interfaz de usuario estuviera bien sincronizada, en cuestión de timers y velocidad de transferencia de datos para lograr que las diferentes señales se representaran dinámicamente a medida que se iban transfiriendo, ha sido, una de los aspectos más complicados que han surgido a lo largo de este TFG. La cantidad de timers que se estaban manejando en el software inicial complicaban de sobremanera el sincronismo, y en ocasiones la representación en la gráfica se bloqueaba. La orientación a objetos de la parte de representación de datos, nos permitió, por un lado, eliminar el timer que se encargaba de representar, y por otro lado, esta parte del código, bajo mi punto de vista, quedó más estructurada e intuitiva.. Por lo tanto, los primeros cambios hechos con respecto a la interfaz de usuario fue reducir el número de timers y terminar de orientarlo todo a objetos. Una vez, logrado esto, los siguientes pasos fueron programar la representación dinámica de todas las variables fisiológicas (la versión original representaba exclusivamente ECG) y añadir diversas funciones nuevas con el propósito de orientar la interfaz al objetivo más inmediato en ese momento del TFG, que no era otro que el de crear una base de datos, en la cual se pudiera analizar como reaccionaban los pacientes a un estímulo emocional de asco y miedo que se hacía sentir mediante la reproducción de vídeos. Es en este punto dónde se desvían los objetivos del software inicial con el desarrollado en este trabajo, ya que el de partida era el de la identificación biomédica. A continuación se detallan con más profundidad los cambios que se acaban de mencionar.

3.2.1 Orientación a objetos

Para un acceso a datos más lógico y obtener un código más estructurado e intuitivo, se añadió a las clases ya existentes una nueva clase para la representación dinámica de las señales fisiológicas. Las clases definidas son las siguientes.

3.2.1.1 Clase Serial

```
classdef Serial < handle

    properties
        COM = 'COM6';
        puerto;
        conectado = 0;
        baudRate = 9600;
        stopBits = 1;
        dataBits = 8;
        paridad = 'none';
    end
```

Código 3. 6 Propiedades clase *Serial*

En las propiedades de Serial se crean todas aquellas variables que almacenarán el estado del objeto que se crea para manejar la configuración del puerto serial. Como se aprecia, la transmisión serie se llevará a cabo por el puerto COM 6. Es la versión de Windows la que asigna el puerto COM al terminal USB donde se conecta el montaje *e-Health*, por lo que, es posible que esta variable se deba de cambiar si se conecta a otro ordenador. En el caso de Windows, es el administrador de dispositivos el que indica a qué puerto COM está conectado. El resto de propiedades son las encargadas de la configuración más típica para un puerto serial [28], es decir, velocidad de transferencia de datos a 9600 baudios, 1 bit de parada, 8 bits de datos y sin bit de paridad. La variable *conectado* se utilizará para conocer el estado del puerto serial.

Por otro lado, los métodos de la clase son aquellas únicas funciones que están autorizadas a manejar las propiedades de un objeto Serial. Intuitivamente es fácil de imaginar, al tratarse del manejo de un puerto, que solamente necesitaremos dos, conectar y desconectar.

```
function conectar(obj)
    %borrar previos
    delete(instrfind({'Port'},{'COM6'}));

    %crear objeto serie
    obj.puerto = serial('COM6');
    set(obj.puerto, 'Baudrate', obj.baudRate);
    set(obj.puerto, 'StopBits', obj.stopBits);
    set(obj.puerto, 'DataBits', obj.dataBits);
    set(obj.puerto, 'Parity', obj.paridad);
    warning('off','MATLAB:serial:fscanf:unsuccessfulRead');

    %cerrar el puerto por si queda abierto para siguiente sesión
    fclose(obj.puerto);

    %abrir puerto serie
    try
        fopen(obj.puerto);
        obj.conectado = 1;

    catch
        fclose(obj.puerto);
        obj.conectado = 0;
        errordlg('Error en la conexión, inténtelo de nuevo','WARNING');
        return;
    end
end
```

Código 3. 7 Método conectar de clase Serial

Se crea el objeto, al que se le asignan los atributos, se cierra el puerto como prevención para una siguiente sesión, se abre para la actual por lo que la variable *conectado* a de estar a 1. Se maneja excepción con catch.

```
%cerrar puerto serie
function desconectar(obj)
    fclose(obj.puerto);
    obj.conectado = 0;

end
```

Código 3. 8 Método desconectar de clase Serial

La función desconectar será tan sencillo como cerrar el puerto y actualizar *conectado* a 0.

3.2.1.2 Clase Variables

```
classdef Variables < handle

    properties
        pulso;
        oxigeno;
        temperatura;
        conductancia;
        flujoAire;
        video;
        matriz;
        contador;
        tiempoGrafica;
        datos;
        vuelta;
        interfazIniciada;
    end

    methods
        function obj = Variables
            obj.pulso = 0;
            obj.oxigeno = 0;
            obj.temperatura = 0;
            obj.conductancia = 0;
            obj.flujoAire = 0;
            obj.video = 0;
            obj.matriz = zeros(6,10000);
            obj.contador = 1;
            obj.tiempoGrafica = 1;
            obj.datos=zeros(1,100);
            obj.vuelta=0;
            obj.interfazIniciada=0;
        end
```

Código 3. 9 Métodos y propiedades clase *Variables*

Como se observa, simplemente se crean todas aquellas propiedades que se crean necesarias para el manejo de las variables fisiológicas, pulso, oxígeno, temperatura, conductancia y flujo de aire. Los métodos inicializan las propiedades creadas para el objeto Variables.

3.2.1.3 Clase DatosPaciente

```
classdef DatosPaciente < handle

    properties
        nombre;
        apellido1;
        apellido2;
        edad;
        sexo;
        altura;
        peso;
    end

end
```

Código 3. 10 Propiedades clase *DatosPaciente*

Esta clase define aquellos datos que se recogerán del paciente a monitorizar. Se explicará en la sección de interfaz de usuario de este capítulo de qué manera se recogen.

3.2.1.4 Clase *Interfaz*

```
properties
    estadoDispositivo;
    escaneoTimer;
    variablesTimer;
    titulo_videoMenu;
    videoMenu;
    nombre_video;
    criticos_videos;
    conectarButton;
    desconectarButton;
    inicioButton;
    pararButton;
    nuevoButton;
    variablesLista;
    titlesVariablesLista;
    datosPacienteLista;
    fichero;
    ejexGrafica;
    grafica_sup_izq;
    grafica_sup_medio;
    grafica_sup_der;
    grafica_inf_izq;
    grafica_inf_der;
    estadoVideo;
    videoOn;
    videoOff;

methods
    function obj = InterfazClass
    obj.estadoDispositivo = 0;
    obj.escaneoTimer = 0;
    obj.variablesTimer = 0;
    obj.titulo_videoMenu = 0;
    obj.videoMenu = 0;
    obj.nombre_video = 0;
    obj.criticos_videos = 0;
    obj.conectarButton = 0;
    obj.desconectarButton = 0;
    obj.inicioButton = 0;
    obj.pararButton = 0;
    obj.variablesLista = 0;
    obj.titlesVariablesLista = 0;
    obj.datosPacienteLista = 0;
    obj.fichero = 0;
    obj.ejexGrafica = 0;
    obj.estadoVideo = 0;
    obj.videoOn = 0;
    obj.videoOff = 0;
```

Código 3. 11 Propiedades y métodos clase *Interfaz*

Al igual que en la clase Variables, se definen todas aquellas propiedades que se crean útiles para la configuración de la interfaz de usuario. Desde variables de estado a botones, colocación de las gráficas y creación del fichero .txt, que salva los datos y puntos interesantes de los vídeos.

3.2.1.5 Clase *GraficaDatos*

Esta clase es la que se añade respecto al software inicial para el manejo de la representación de las gráficas de manera dinámica a medida que van entrando datos por el puerto serial.

```
properties
    grafica;
    tituloGrafica;
    datos;
    ejes;
    posicionVector;
    indice;
    creado;
end

methods
    function obj = GraficaDatos
        obj.posicionVector = 0.0;
        obj.datos = zeros(1,10000);
        obj.indice = 1;
        obj.creado = 0;
    end
```

Código 3. 12 Propiedades y métodos clase *GraficaDatos*

El método *GraficaDatos* inicializa ciertas propiedades.

```
function nuevaGraficaDatos( obj, posicionVector, titulo )
    if ( obj.creado == 0 )
          disp( 'Creando gráfica' );
        obj.creado = 1;
        obj.posicionVector = posicionVector;
        losEjes = axes('Position', posicionVector, 'xlimmode', 'manual');
        obj.ejes = losEjes;
        obj.tituloGrafica = titulo;
        obj.grafica = plot(0,NaN,'-r', 'parent', losEjes, 'tag', titulo);
    end
    obj.datos = zeros(1,10000);
    obj.indice = 1;
    grid on;
end
```

Código 3. 13 Función *nuevaGraficaDatos*

El método *nuevaGraficaDatos* crea una nueva gráfica, con su respectiva configuración de posición, ejes y título que serán insertados como parámetros de entrada de un *plot*.

```
function ponerTitulo(obj, titulo)
    obj.tituloGrafica = titulo;
end

function aplicarEjes(obj, posicion)
    obj.ejes = axes('Position',posicion, 'xlimmode', 'manual');
end
```

Código 3. 14 Función *ponerTitulo*

Aquí se crean métodos para insertar título y aplicar los ejes para el plot.

```
function representarDatos(obj, dato, tiempo)
    x = get (obj.grafica, 'xdata');
    y = get (obj.grafica, 'ydata');
    set(obj.grafica,'ydata',[y dato],'xdata',[x tiempo]);
    obj.datos( obj.indice ) = dato;
    obj.indice = obj.indice+1;
end
```

Código 3. 15 Función *representarDatos*

Se recogen los datos en los vectores x e y, se insertan en la gráfica mediante incremento de índice.

```
function resetGrafica( obj )
delete( obj.ejes );
losEjes = axes('Position', obj.posicionVector, 'xlimmode', 'manual');
obj.ejes = losEjes;
obj.grafica = plot(0,NaN,'-r', 'parent', obj.ejes, 'tag', obj.tituloGrafica);
obj.datos = zeros(1,10000);
obj.indice = 1;
end
```

Código 3. 16 Función *resetGrafica*

Queda tan sólo resetear los valores para una futura creación de un objeto Gráfica.

Se cree oportuno resaltar en este punto, aprovechando lo tediosa que pueda llegar a resultar la creación de esta clase, que ésta nos permite ahora una creación de gráficas en la interfaz de usuario mucho menos engorrosa y conceptualmente mucha más sencilla, ya que resulta muy intuitivo después entender los pasos que se dan al llamar a estos métodos sucesivamente.

3.2.2 Interfaz de usuario

La interfaz de usuario es el entorno gráfico que se mostrará a éste. En este entorno se visualizará inicialmente un cuadro de diálogo dónde el paciente puede introducir sus datos. Será aquí donde se recojan éstos datos para encabezar el fichero .txt que servirá como base para la formación de la base de datos. Con el cuadro de diálogo se abre automáticamente una ventana que contiene la interfaz de usuario en sí. Todo esto se muestra en las siguientes figuras.

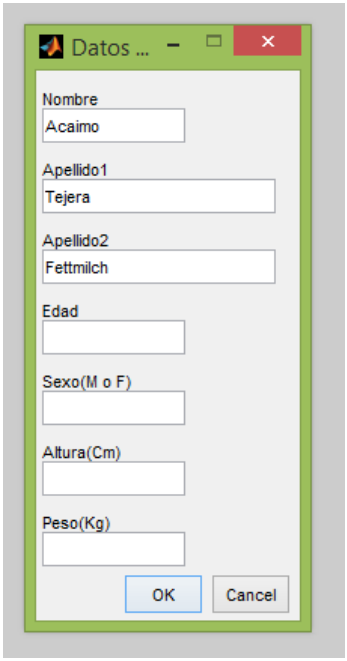


Figura 3. 4 Cuadro de diálogo

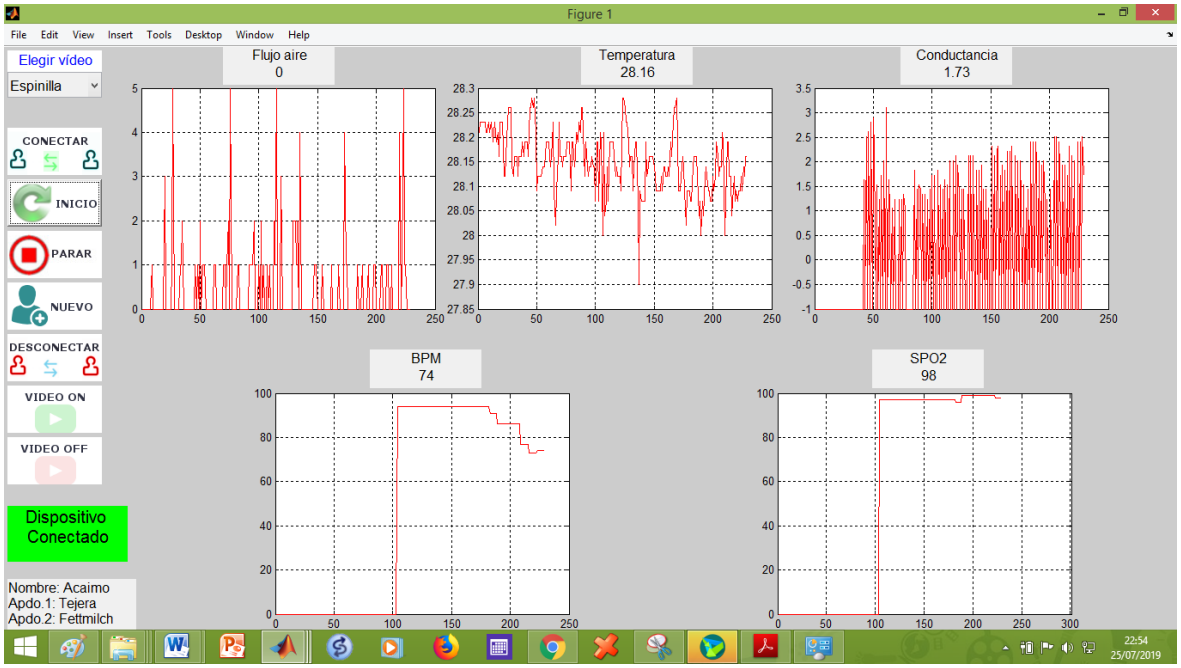


Figura 3. 5 Interfaz de usuario

Como se puede ver, la interfaz de usuario consta de 5 gráficas, una por cada una de las variables fisiológicas que se recogen por el terminal serial. Las gráficas se van formando dinámicamente a medida que se obtienen los datos. Por la parte izquierda, de arriba abajo, se encuentra un desplegable para elegir el vídeo a reproducir, una serie de botones con diferentes funcionalidades, un cuadro de estado del dispositivo y por último un cuadro con el nombre completo del paciente. Cada uno de estos elementos se

explicarán a continuación en el orden de ejecución cuando se acomete una monitorización cualquiera.

```
function interfaz()
clc;
clear all;
    %objetos
    interfaz = InterfazClass;
    paciente = DatosPaciente;
    % El objeto de la clase Serial
    serial = Serial;
    variables = Variables;
    variables.tiempoGrafica=1;
    variables.vuelta = 1;

    %vector de objetos gráficos
    misGraficas = [ GraficaDatos, GraficaDatos,
      GraficaDatos, GraficaDatos, GraficaDatos ];
    anchoBoton = 110;
    altoBoton = 55;
    separacionBotones = 5;
    primerBotonEjeY = 650;
    primerBotonEjeX = 5;
    indiceBoton = 0;
```

Código 3. 17 Objetos gráficos y variables en *interfaz*

Inicialmente se declaran los objetos y variables necesarios.

```
scrsz = get(0, 'ScreenSize'); %función que me recoge la resolución de la
figure('Units','pixels','Position',[0 13 (scrsz(3)-100) (scrsz(4)-100)]);

solicitarDatosPaciente(interfaz,paciente);%función que me abre un cuadro
```

Código 3. 18 Llamada a *solicitarDatosPaciente*

Se recoge la resolución de la pantalla para que, al crearme la figura, me ocupe la totalidad de ésta. Acto seguido se llama a *solicitarDatosPaciente* que es la encargada de mostrar el cuadro de diálogo para la recogida de datos del usuario. Se logra como se muestra a continuación.

```
function solicitarDatosPaciente(paciente)

    prompt={'Nombre', 'Apellido1', 'Apellido2', 'Edad',
        'Sexo(M o F)', 'Altura(Cm)', 'Peso(Kg)' };
    fields = {'nombre', 'apellido1', 'apellido2', 'edad',
        'sexo', 'altura', 'peso'};
    tamCuadro=[1 15; 1 25; 1 25; 1 15; 1 15; 1 15; 1 15];
    pacienteDialog = inputdlg(prompt,'Datos del paciente',tamCuadro);

    if ~isempty(pacienteDialog)
       pacienteDialog = cell2struct(pacienteDialog,fields);
       paciente.nombre = pacienteDialog.nombre;
       paciente.apellido1 = pacienteDialog.apellido1;
       paciente.apellido2 = pacienteDialog.apellido2;
       paciente.edad = str2num(pacienteDialog.edad);
       paciente.sexo = pacienteDialog.sexo;
       paciente.altura = str2num(pacienteDialog.altura);
       paciente.peso = str2num(pacienteDialog.peso);
    end

end
```

Código 3. 19 Función *solicitarDatosPaciente*

Tras el cuadro de diálogo, se prosigue en *interfaz* con los cuadros que se sitúan justo encima de cada gráfica, y que muestran los datos capturados de las variables fisiológicas. Se detalla sólo el primero de ellos (correspondiente al pulso, BPM) ya que es un tanto largo y repetitivo. Por cada cuadro se crea primero el título y después el cuadro donde aparecerá el dato. En el resto de cuadros sólo varían las coordenadas y evidentemente los títulos.

```
if ( variables.interfazIniciada == 0 )

    variables.interfazIniciada = 1;
    interfaz.titlesVariablesLista = [0;0;0;0;0];
    interfaz.variablesLista = [0;0;0;0;0];
    editTexto = {'BPM', 'SPO2', 'Temperatura', 'Conductancia', 'Flujo aire'};
    %situación justo encima de las gráficas,de los cuadros con los valores de
    for i = 1:2
        interfaz.titlesVariablesLista(i) = uicontrol('Style', 'text',...
                                    'unit','pix',...
                                    'position',[427+585*(i-1) 307 130 20],...
                                    'fontsize',12,...
                                    'String', editTexto{i});
    end
    for i = 1:2
        interfaz.variablesLista(i) = uicontrol('Style', 'text',...
                                    'unit','pix',...
                                    'position',[427+585*(i-1) 282 130 25],...
                                    'fontsize',12);
    end
    for i = 3:4
    :
    :
    :
```

Código 3. 20 Cuadros de valores en gráficas

Lo siguiente en ejecutarse son los *timers*. Inicialmente eran tres pero se eliminó el *pintarVariables* ya que daba problemas. La idea de los *timers* es que se lleve a cabo la recogida de datos y la representación de éstos de manera secuencial sin que lleguen a colisionar. Cada uno de ellos se ejecutan con un cierto periodo e invocan a la función especificada cada vez que se cumple cierto tiempo. Al ser bloqueantes, nos aseguramos que no se intentará representar un dato sin haber sido previamente recogido. Eso sí, hay que especificar con precisión los periodos, para que, por un lado no dejar de recoger datos que se han enviado por el puerto serial (recordar los 250ms de lanzamiento con los que actuaba el microcontrolador de Arduino), y por el otro, no dejar de representar ningún valor recogido. También hay que tener en cuenta, que si estos tiempos los queremos muy pequeños (mayor precisión), la gráfica se nos llenará tal vez demasiado y por consiguiente también el tamaño que ocupe la base de datos.

```
%TIMERS
interfaz.escaneoTimer = timer ('Name', 'timerEscanear',...
  'Period', 0.25,...
  'ExecutionMode', 'fixedSpacing',...
  'BusyMode', 'drop',...
  'TimerFcn', @(o, e) escaneo(serial, variables, interfaz));

interfaz.variablesTimer = timer ('Name', 'mostrarVariables',...
   'Period',0.25,...
   'ExecutionMode', 'fixedSpacing',...
   'BusyMode', 'drop',...
   'TimerFcn', @(o,e) mostrarVariables(interfaz, variables, misGraficas) );
            'TimerFcn', @(o, e) pintarVariables(eHealth) );
```

Código 3. 21 Timers en *interfaz*

El primer *timer* llama a la función escaneo, que se encarga de leer las tramas del puerto serial y separar ésta en las diferentes variables fisiológicas que hemos separado con el carácter *. Esto lo hace la función descomponer.

```
function escaneo(serial, variables, interfaz)
  datos = fscanf(serial.puerto,'%s'); %lectura
  descomponer(datos, variables, interfaz);
  %pause(0.001);
end
```

```
function descomponer(datos,variables,interfaz)

datos = regexp(datos, '\*', 'split'); %separar datos en campos

variables.pulso = 0;
variables.oxigeno = 0;
variables.temperatura = 0;
variables.conductancia = 0;
variables.flujoAire = 0;

 if(strcmp(datos(1),'Cabecera'))
     variables.pulso = datos(2);
     variables.oxigeno = datos(3);
     variables.temperatura = datos(4);
     variables.conductancia = datos(5);
     variables.flujoAire = datos(6);

     variables.matriz(1,variables.contador) = str2double(datos(2));
     variables.matriz(2,variables.contador) = str2double(datos(3));
     variables.matriz(3,variables.contador) = str2double(datos(4));
     variables.matriz(4,variables.contador) = str2double(datos(5));
     variables.matriz(5,variables.contador) = str2double(datos(6));%
     variables.matriz(6,variables.contador) = interfaz.estadoVideo;
     variables.contador = variables.contador + 1;
 end
end
```

Código 3. 22 Funciones *escaneo* y *descomponer*

El segundo *timer* invoca a la función *mostrarVariables*.

```
function mostrarVariables(interfaz, variables, misGraficas)

   set (interfaz.variablesLista(1), 'String', variables.pulso);
   set (interfaz.variablesLista(2), 'String', variables.oxigeno);
   set (interfaz.variablesLista(3), 'String', variables.temperatura);
   set (interfaz.variablesLista(4), 'String', variables.conductancia);
   set (interfaz.variablesLista(5), 'String', variables.flujoAire);
```

Código 3. 23 Función *mostrarVariables*

Que por un lado introduce los datos en los cuadros que se crearon al comienzo de interfaz.

```
flujoAire=str2double(variables.flujoAire);
misGraficas(1).representarDatos(flujoAire,variables.tiempoGrafica);

temperatura=str2double(variables.temperatura);
misGraficas(2).representarDatos(temperatura,variables.tiempoGrafica);

conductancia=str2double(variables.conductancia);
misGraficas(3).representarDatos(conductancia,variables.tiempoGrafica);

pulso=str2double(variables.pulso);
misGraficas(4).representarDatos(pulso,variables.tiempoGrafica);

oxigeno=str2double(variables.oxigeno);
misGraficas(5).representarDatos(oxigeno,variables.tiempoGrafica);
```

Código 3. 24 Variables a objetos gráficos

Y por el otro, los introduce en los objetos *misGraficas*.

Tras la ejecución de los *timers*, entran en juego los botones que se encuentran en la zona izquierda de la interfaz. Se irán detallando por el orden lógico de ejecución cuando se quiere testear a un paciente. El primero que nos encontramos es el botón desplegable *videoMenu*, y nos permitirá seleccionar el vídeo a reproducir.

```
interfaz.titulo_videoMenu = uicontrol('Style','text',...
'Position', [primerBotonEjeX
primerBotonEjeY anchoBoton (altoBoton-30)],...
'String', {'Elegir vídeo'},...
'BackgroundColor', 'white',...
'ForegroundColor', 'blue',...
'fontsize',13);
indiceBoton=indiceBoton+1;


interfaz.videoMenu = uicontrol('Style','popup',...
'Position', [primerBotonEjeX
(primerBotonEjeY-(indiceBoton*(altoBoton+separacionBotones-5)))
anchoBoton altoBoton],...
'String', {'Espinilla','Jackass','Moco','Verruga','Grano',
'Mix asco','Carretera','Luces fuera','Exorcismo','Mix miedo'},...
'fontsize',13,...
'Callback',{@popupmenu_videoMenu_callback, interfaz});
indiceBoton=indiceBoton+1;
```

Código 3. 25 Botón desplegable *videoMenu*

La primera parte es la correspondiente al título del desplegable y la segunda a las opciones a elegir que ofrece. La función a la que llama es la que se encuentra en la parte

del *Callback*. Ésta simplemente asigna al vídeo elegido los correspondientes momentos que se entienden como de mayor susto o asco, es decir, los momentos más intensos.

```
function [ interfaz ] = popupmenu_videoMenu_callback( ~, ~, interfaz )
% Función que se encarga de permitir al usuario elegir el vídeo a repro
% unos momentos críticos(muestras) qué se corresponden a los momentos d


interfaz.criticos_videos=zeros(1,10);  %vector que me recoge según el v
interfaz.nombre_video='';
switch get(interfaz.videoMenu,'Value')    %switch para lo que se recoge
     case 1     %video asco/espinilla
         interfaz.criticos_videos=[44 188];     %por ej, un momento imp
         interfaz.nombre_video='Espinilla';
         %disp(interfaz.criticos_videos);

     case 2     %vídeo asco/jackass
     :
     :
```

Código 3. 26 *Callback videoMenu*

En los siguientes botones no se detallará la parte correspondiente a su colocación en la interfaz sino sólo a la función que invoca, ya que lo único que cambia son sus coordenadas.

Después del desplegable se encuentra el botón de conectar. La función a la que invoca se limita a conectar el puerto serial y a mostrar el cuadro de estado del dispositivo (penúltimo botón) en modo activo (en verde).

```
function conectar_pushbutton_callback(hObject, ~, serial, interfaz)
serial.conectar();
if(serial.conectado == 1)  %si está conectado cambia el text edit a Disp
    set (interfaz.estadoDispositivo, 'String', 'Dispositivo Conectado');
    set (interfaz.estadoDispositivo, 'BackgroundColor', 'g' );

    [x_conectar,map]= imread('boton conectar activo.png');
    set(interfaz.conectarButton,'CData',x_conectar);
end
end
```

Código 3. 27 *Callback conectar*

.En el momento que se sabe que el cuadro de estado nos indique que efectivamente está correctamente conectado, podremos iniciar la grabación de toma de datos mediante el botón *iniciar*.

```
misGraficas(1).nuevaGraficaDatos(pos_sup_izq_norm,'grafica_sup_izq');
misGraficas(2).nuevaGraficaDatos(pos_sup_medio_norm,'grafica_sup_medio');
misGraficas(3).nuevaGraficaDatos(pos_sup_der_norm,'grafica_sup_der');
misGraficas(4).nuevaGraficaDatos(pos_inf_izq_norm,'grafica_inf_izq');
misGraficas(5).nuevaGraficaDatos(pos_inf_der_norm,'grafica_inf_der');

%arranque de los timer de escaneo y muestra de variables
start(interfaz.escaneoTimer);
start(interfaz.variablesTimer);
```

Código 3. 28 *Calback iniciar*

Este botón, se encarga de colocar cada objeto *misGraficas* en su sitio correspondiente de la interfaz mediante las coordenadas que se le pasan como primer parámetro de entrada. Una vez situados, se activan los *timers* para que comiencen a escanear datos y a representar éstos.

Acto seguido de comenzar la grabación y pasado un tiempo, se procederá a la reproducción del vídeo, tramo que como veremos en el capítulo 5, se llamará estado1. La reproducción del vídeo elegido en el desplegable se activará con el botón *VideoOn*.

```
ruta_windows_media_player='C:\Program Files\Windows Media Player\wmplayer.exe';
interfaz.estadoVideo = 1;

switch get(interfaz.videoMenu,'Value')
 case 1     %video asco/espinilla
  ruta_video ='C:\Users\acaymo\Desktop\TFG\VIDEO ESTÍMULOS\ASCO\Espinilla.avi';

 case 2     %vídeo asco/jackass
  ruta_video ='C:\Users\acaymo\Desktop\TFG\VIDEO ESTÍMULOS\ASCO\Jackass.mp4';

 case 3     %vídeo asco/moco
 •
 •
 •
```

Código 3. 29 Rutas de los vídeos

La función *VideoOn* permitirá lanzar un comando al sistema para que ejecute, mediante un reproductor especificado, el vídeo que se eligió en el video menú a través de su ruta. Por último, se hará notar que el vídeo está activo, realzando el botón mediante la lectura de una imagen creada que se caracteriza por su brillo. Por el contrario, la imagen del botón *videoOff* perderá protagonismo con una imagen más borrosa.

```
[x_videoOn]= imread('boton video on activo.png');    %
set(interfaz.videoOn,'CData',x_videoOn);

[x_videoOff]= imread('boton video off inactivo.png');
set(interfaz.videoOff,'CData',x_videoOff);
```

```
system(sprintf('"%s" -f wmplayer://quit "%s"',ruta_windows_media_player,ruta_video));

        videoOff_pushbutton_callback( interfaz.videoOff,   '',   interfaz);
```

Código 3. 30 Realce botones de vídeo

Una vez termina la reproducción del vídeo y se cierra la ventana, automáticamente el botón inicio pasará a inactivo y el video off a activo.

```
function videoOff_pushbutton_callback(~,~, interfaz)
% Función correspondiente al manejo del botón video off.
% botones video off y video on (activo y desactivo corres

   interfaz.estadoVideo = 0;
   set (interfaz.videoOn, 'BackgroundColor', 'default' );

    [x_videoOn]= imread('boton video on inactivo.png');
    set(interfaz.videoOn,'CData',x_videoOn);

    [x_videoOff]= imread('boton video off activo.png');
    set(interfaz.videoOff,'CData',x_videoOff);
end
```

Código 3. 31 *Callback videoOff*

Con el vídeo ya terminado, se esperará un tiempo a darle al botón *Parar* ya que el paciente ha de relajarse y volver a su estado habitual, a este intervalo se le llamará estado5.

La función *Parar* inicialmente detiene los *timers* de escaneo y muestra de variables. Una vez terminada la lectura, para la creación de la base de datos, se genera un fichero .txt con una cabecera, y los datos que se han ido guardando. Esta parte se hizo de dos formas diferentes. La primera guardando cada valor con el nombre de su variable fisiológica delante, lo cual a lo mejor es más vistoso, pero como se verá en la segunda opción, menos óptima.

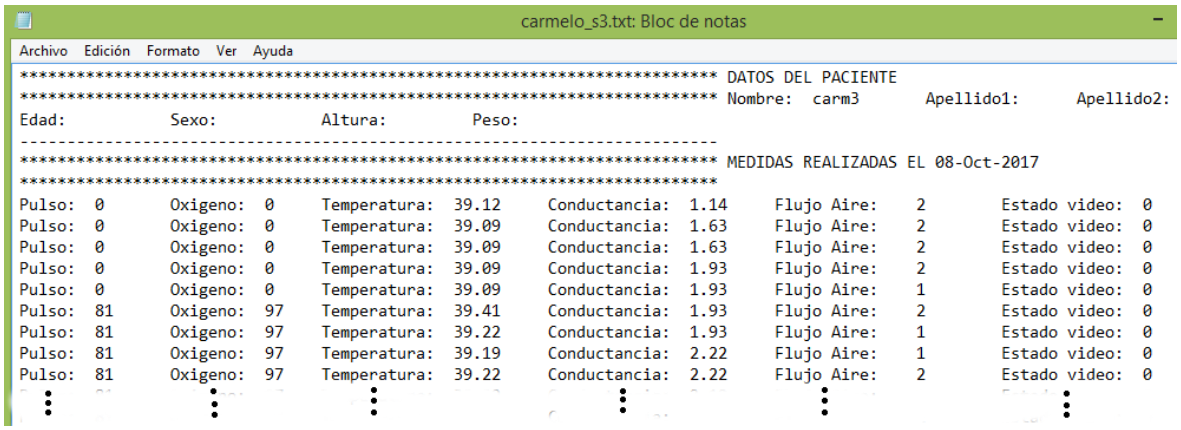


Figura 3. 6 Forma 1 de generación txt

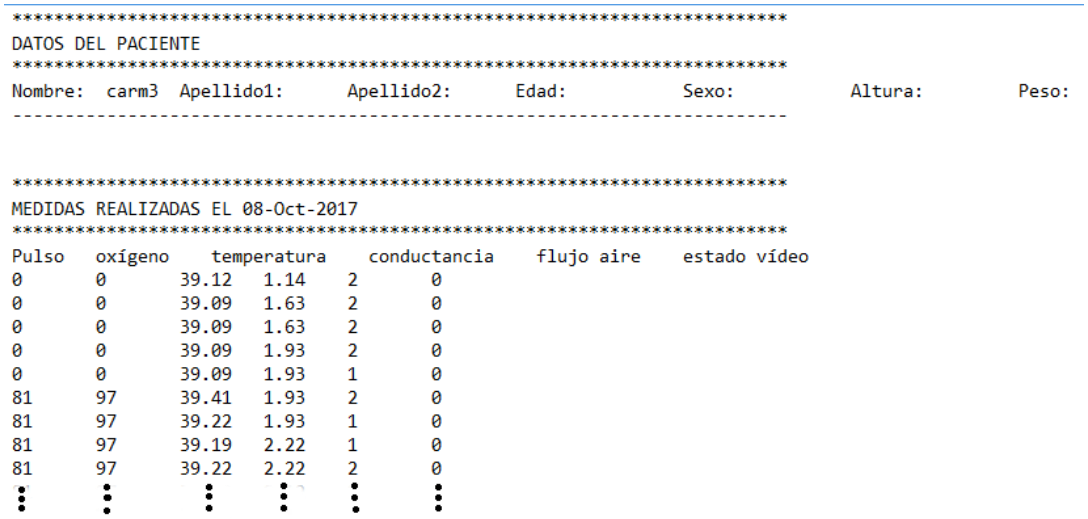


Figura 3. 7 Forma 2 de generación txt

La segunda opción separa los valores por tabulaciones, eliminando el nombre de la variable que estaba delante. Esto facilita mucho, como se verá en el capítulo 5, la segmentación del fichero de texto, es decir, la separación de las variables fisiológicas en vectores. Ya que Matlab tiene funciones para separar columnas e introducirlas en vectores en el caso de que estén separadas por ciertos caracteres, por ejemplo, la tabulación. Esto desgraciadamente no se descubrió una vez hecha esta segmentación con un código bastante más tedioso. Aunque sea menos óptimo, esto también se detallará en el capítulo 5, donde se agrupa todo lo referente al análisis de la base de datos generada.

También se explican aquí la dos funciones restantes que quedan en interfaz, que son *txt_a_graficas* y *txt_a_parametros*, ya que van orientadas al análisis mencionado.

En el caso de que se deseara testear a varios pacientes en la misma sesión, para poder ahorrarnos el tener que desconectar y volver a conectar el dispositivo por

cada uno de los pacientes de la sesión, se creó el botón nuevo. Esta función lógicamente agrupará todas las acciones de desconectar y volver a conectar, es decir, desconectar el puerto serial, resetear los botones, las variables y las gráficas, volver a solicitar los datos del paciente y por último volver a conectar el puerto serial.

Si no se quisiera monitorizar más clientes, para terminar la sesión sólo quedaría desconectar el dispositivo.

```
function desconectar_pushbutton_callback(hObject, ~, serial, interfaz)
%     fclose(serial.puerto);
    serial.desconectar();
    set (interfaz.estadoDispositivo, 'String', 'Dispositivo Desconectad
    set (interfaz.estadoDispositivo, 'BackgroundColor', 'r' );

    fclose(interfaz.fichero);
end
```

Código 3. 32 *Callback desconectar*

Hecho esto, y para cada uno de los pacientes, se va generando una serie de ficheros, que conformarían nuestra base de datos. De qué manera se hace la grabación, como se organizan la serie de grabaciones y cómo termina siendo nuestra base de datos, compete al siguiente capítulo.

Capítulo 4. Base de datos

Una vez creado el código que permite la generación de un fichero de texto al monitorizar a un paciente, se está en disposición de configurar la base de datos.

Se entiende como base de datos a un conjunto de datos vinculados a un mismo contexto y almacenados sistemáticamente para su posterior análisis o manipulación. En este trabajo, no se realiza ningún tipo de programa para gestionar la base de datos (SGBD) [29]. En el capítulo 6, se añadirá este aspecto como una cuestión a añadir en el futuro, ya que, como base de datos posiblemente relacionada a aspectos clínicos, requeriría sin duda, de alguna gestión para el acceso rápido y estructurado de datos, al verse sustancialmente aumentada en tamaño.

Aparte de la creación de los ficheros .txt, al final del código del botón *parar,* se genera también una captura de pantalla de la interfaz, para que el paciente pueda hacerse una idea visual del aspecto que tiene la grabación de sus variables fisiológicas.

```
F_graficas = getframe(interfaz.grafica_sup_izq, [-770 ,-15, 1366, 700]); %se obtie
nombre_paciente_graficas = strcat(paciente.nombre,'_',paciente.apellido1, '.bmp');
imwrite(F_graficas.cdata,  nombre_paciente_graficas);%guarda la imagen(.cdata) a f
```

Código 4. 1 Captura del conjunto de gráficas

Como se observa, a la función que maneja la captura (*getframe*), se le pasa como parámetros de entrada, un manejador gráfico (de tipo *gca* en Matlab) y unas coordenadas tales que se visualicen las cinco gráficas.

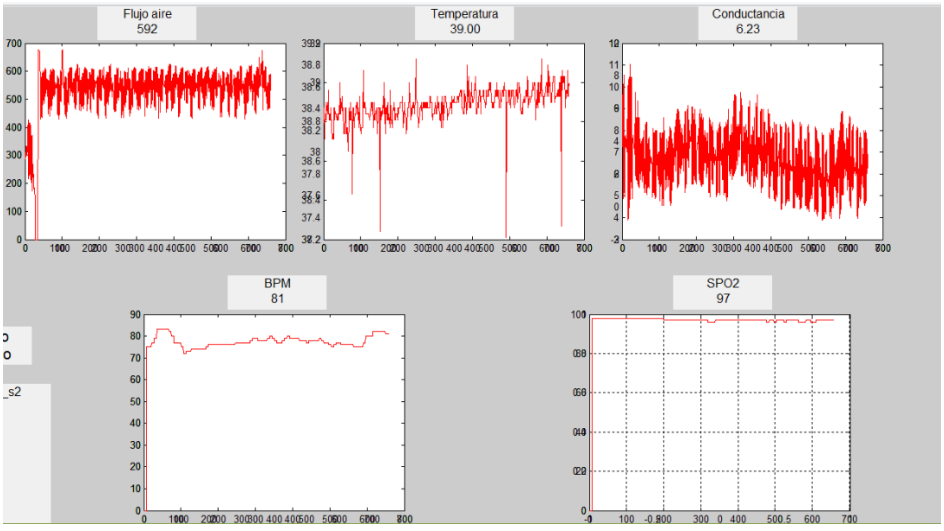


Figura 4. 1 Ejemplo de captura de la interfaz

Aunque si es cierto que esta imagen no se usa para el análisis de las señales generadas, al ser parte de la información creada a raíz de la monitorización del usuario, se considerará parte de la base de datos que se discute en este capítulo.

Se considerará parte también de este capítulo, no sólo, qué se tiene en la base de datos, sino, cómo se hace la grabación al paciente para poder crear la base de datos. No nos referimos aquí, al código que se explicó en el capítulo anterior, sino de qué manera se graba al paciente in situ, es decir, qué estrategia se lleva a cabo para que en el momento de la grabación se pueda facilitar la obtención de los objetivos planteados en este TFG.

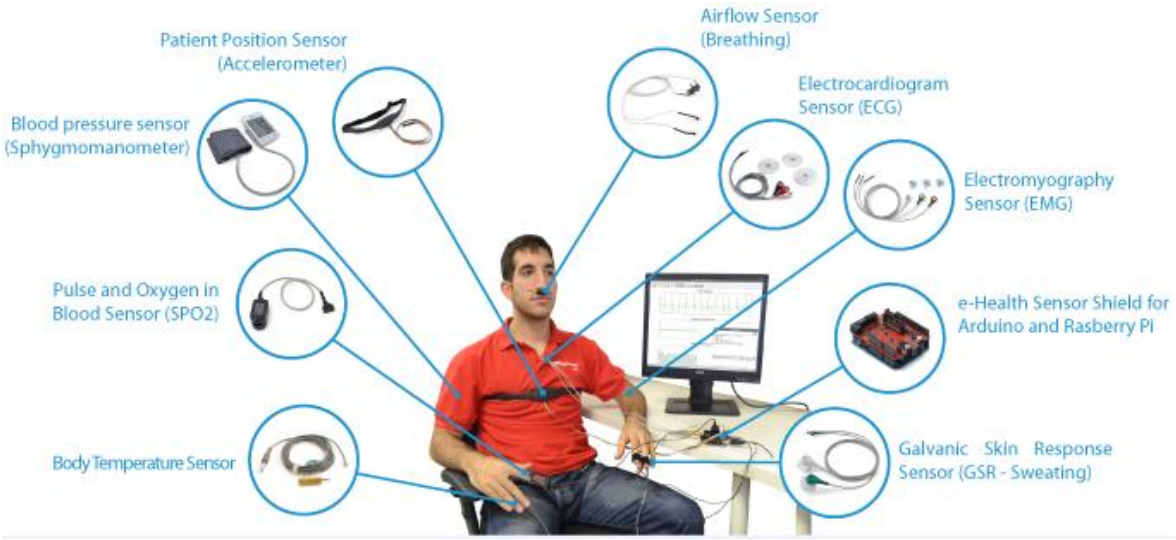


Figura 4. 2 Monitorización de un paciente

4.1 Grabación in situ

Una vez conectado el montaje e_Health al ordenador, comprobado que éste ha sido reconocido en el puerto COM correspondiente, y suponiendo que la placa Arduino está cargada con el programa que se creó para la transferencia de tramas de datos, nos encontraremos en disposición de abrir la interfaz de usuario. Se hará hincapié en este punto, lo importante que es para el éxito de la grabación, la correcta colocación de los sensores. Esto implica, para la medición de la conductancia de la piel por ejemplo, que el velcro se ajuste de tal manera a los dedos, que el sensor no pierda contacto con la piel en ningún instante. Ayuda también para lograr esta correcta medición, que el paciente apoye la palma en la mesa y no la mueva, de esta manera, el sensor no tendrá oportunidad de desplazarse. Por otro lado, tanto para el sensor de temperatura como para el pulsioxímetro, es importante, colocárselos y esperar un tiempo prudencial para

comenzar la grabación. Ya que, el sensor de temperatura necesita un tiempo para reflejar la temperatura real del paciente, y el pulsioxímetro para activarse y comenzar a arrojar medidas de pulso y de saturación de oxígeno en sangre. El cuarto sensor en concordia, el del flujo de aire, es el más complicado de colocar de todos los mencionados. No sólo ya, por sus largos hilos para poder llegar a colocarse tras las orejas de los pacientes, sino que, es de vital importancia que el juego de dos dientes de los que dispone el sensor, se coloquen los suficientemente cerca de las fosas nasales para que realmente se tomen los valores deseados.

Cuando el paciente se ha colocado todos los sensores, llega el momento no menos importante, de indicarle cómo ha de actuar durante la grabación. Debido a que, la futura segmentación del conjunto de estados separará éstos en 5 estados, y, es de crucial interés que el paciente adopte una cierta actitud en algunos de ellos. Especialmente en el estado 1 y en el estado 5. Tanto en el primero (estado relax) como en el último (vuelta al relax) el paciente ha de, como se intuye por la denominación de los estados, relajarse.

Por lo tanto, se le indicará que desde el momento en que se active el botón *inicio* (comienzo de la monitorización), cierre los ojos y trate de mantenerse en estado de tranquilidad. Una vez transcurrido un cierto tiempo, suficiente para que el estado cuente con un número de muestras considerable, se proseguirá con la reproducción del vídeo previamente seleccionado en el botón desplegable. Esta reproducción se inicia con el botón *videoOn.* En el momento de que el vídeo concluya, es de valor también, alargar lo suficiente la grabación para que el estado 5 se complete de forma óptima. Éste será concluido con el botón *parar*. Por último, o se desconecta el dispositivo (botón desconectar), o si se trata de una sesión de monitorización de varios pacientes, se le coloca los sensores a la siguiente persona y se activa el botón *nuevo* para una nueva grabación.

4.2 Usuarios y vídeos

Un apartado directamente proporcional al tamaño de la base de datos es lógicamente el número de vídeos a emplear y la cantidad de usuarios monitorizados para cada uno de ellos. Debido a lo que se busca provocar en los pacientes son sensaciones de asco y miedo, los vídeos son de un contenido en éstos sentidos, considerablemente impactantes. Ya que normalmente la sensación de asco tal vez no es tan espontánea como la del miedo, los vídeos tipo asco son más extensos, buscando que la sensación de

asco se vaya incrementando lo suficiente en el tiempo. Además, los vídeos de ambos tipos, son variopintos dentro de su temática, teniendo en cuenta que, a no todo el mundo le da miedo lo mismo, o, lo de asco lo mismo. Más allá de cómo son los vídeos y qué se buscan con ellos, es de especial interés obtener resultados con pacientes de todo tipo de género y edad, ya que, como se demostrará en el capítulo siguiente, cada persona es un mundo, y reacciona por lo tanto, de una manera diferente a otra.

A modo resumen, y para finalizar, el siguiente gráfico muestra cómo está conformada nuestra base de datos.

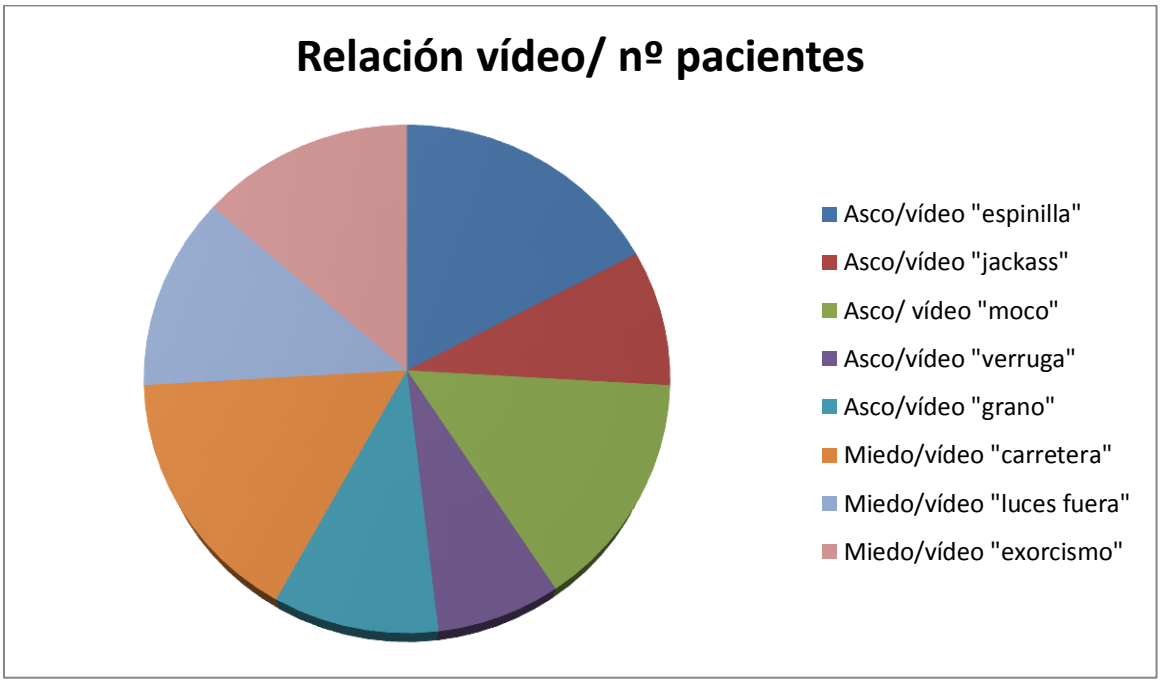


Figura 4. 3 Base de datos

Capítulo 5. Análisis gráfico

Una vez generada la base de datos, y antes de realizar ninguna modificación de tipo estadístico, para hacernos una idea más visual de cómo son las señales fisiológicas capturadas, se hace un análisis de tipo gráfico, teniendo siempre en mente, la búsqueda de un patrón que verifique la existencia de una reacción ante un estímulo de tipo miedo o de tipo asco.

Para obtener las gráficas de cada variable fisiológica y paciente, se invoca desde *interfaz* y una vez generado el fichero de texto, a la función *txt_a_graficas.* Esta función inicialmente lo que busca es separar cada una de las señales captadas por los sensores y separarlas en vectores. Como se mencionó en el capítulo 3, en el que se explicaba detalladamente la interfaz de usuario, esta separación de variables se hizo de dos formas. Uno partiendo de un fichero de texto, donde cada señal fisiológica le precedía su correspondiente nombre, y el otro, donde se separaban por tabulaciones.

```
Conductancia:  2.02    Flujo Aire:   38      Estado video:  0      Pulso:  0      Oxigeno:  0     Temperatura:  38.15
Conductancia:  2.02    Flujo Aire:   23      Estado video:  0      Pulso:  0      Oxigeno:  0     Temperatura:  38.33
Conductancia:  1.83    Flujo Aire:   16      Estado video:  0      Pulso:  0      Oxigeno:  0     Temperatura:  38.15
Conductancia:  2.12    Flujo Aire:   35      Estado video:  0      Pulso:  0      Oxigeno:  0     Temperatura:  38.15
Conductancia:  1.73    Flujo Aire:    0      Estado video:  0      Pulso:  90     Oxigeno:  98    Temperatura:  38.24
Conductancia:  2.81    Flujo Aire:    0      Estado video:  0      Pulso:  90     Oxigeno:  98    Temperatura:  38.08
Conductancia:  1.63    Flujo Aire:    0      Estado video:  0      Pulso:  90     Oxigeno:  98    Temperatura:  38.08
```

Figura 5. 1Txt modo 1

```
Pulso   oxígeno    temperatura    conductancia    flujo aire    estado vídeo
0       0        38.15   2.02   38      0
0       0        38.15   2.02   23      0
0       0        38.33   1.83   16      0
0       0        38.15   2.12   35      0
0       0        38.15   1.73    0      0
90      98       38.24   2.81    0      0
90      98       38.08   1.63    0      0
90      98       38.08   2.81    0      0
90      98       38.05   1.05    0      0
90      98       38.08   2.51    1      0
90      98       38.08   1.14    2      0
```

Figura 5. 2 Txt modo 2

Para la separación de variables del modo 1, es necesario, ir buscando línea por línea, la palabra correspondiente a la señal fisiológica que se quiere introducir en el vector de su mismo nombre. En cambio, si el fichero tiene la forma del modo 2 con el código siguiente Matlab automáticamente nos importa los diferentes datos en vectores.

```
datos=importdata( nombreFichero);
Pulso=datos.data( : , 1 );
Oxigeno=datos.data( : , 2 );
Temperatura=datos.data( : , 3 );
Conductancia=datos.data( : , 4 );
Flujo_aire=datos.data( : , 5 );
Estado_video=datos.data( : , 6 );
```

Código 5. 1 *importdata* de Matlab

Una vez obtengo todo los vectores, el resto de la función está orientada a representarlos junto al estado de vídeo (señal cuadrada que indica cuando se encuentra el vídeo en reproducción o no) y junto a los críticos del vídeo, que son los momentos que se consideran más intensos. Esto se hace, tanto para el gradiente de la señal, como para su envolvente. Los resultados se salvan tanto en .jpg (imagen) como en .fig (matriz de datos). El análisis gráfico se hizo en profundidad para un vídeo de asco y otro de medio. A continuación se muestra un resumen de dicho análisis.

5.1 Análisis gráfico vídeo asco

Tras analizar cada uno de las gráficas de todos los pacientes del video de asco "espinilla", se llegaron a una serie de conclusiones para cada una de las variables fisiológicas que se detallan.

5.1.1 Temperatura

Se observa que existe un patrón, y que se repite a lo largo del todo el vídeo sin tener relación aparente con los momentos críticos de éste, pero sin embargo, sí que tienen mayor frecuencia de aparición durante la reproducción del vídeo, por lo que se descarta que sea un comportamiento asociado al sensor. Este patrón tiene forma de decrementos e incrementos abruptos y espontáneos.

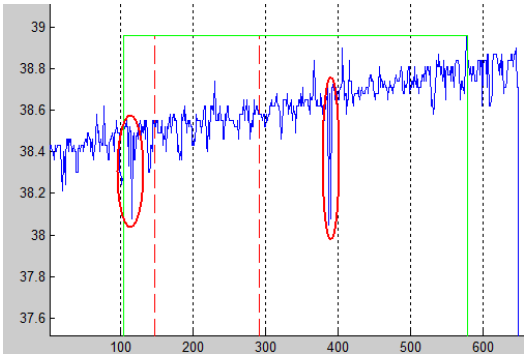


Figura 5. 3 Decrementos intensos en *video on*

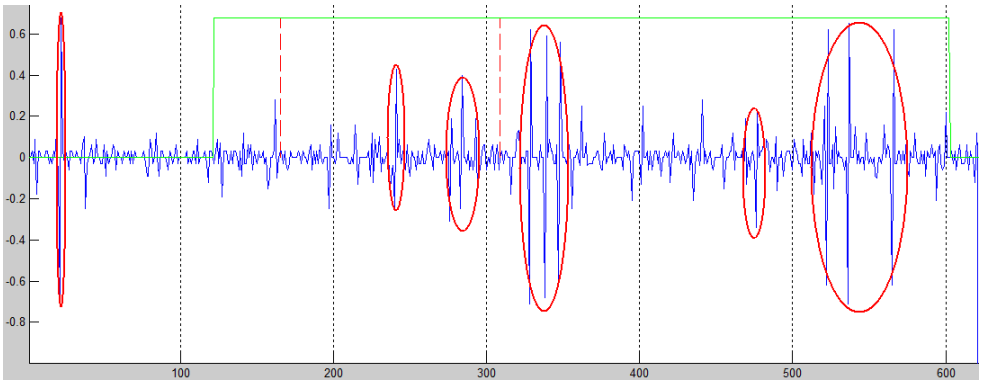


Figura 5. 4 Concentración patrón alrededor del crítico 2

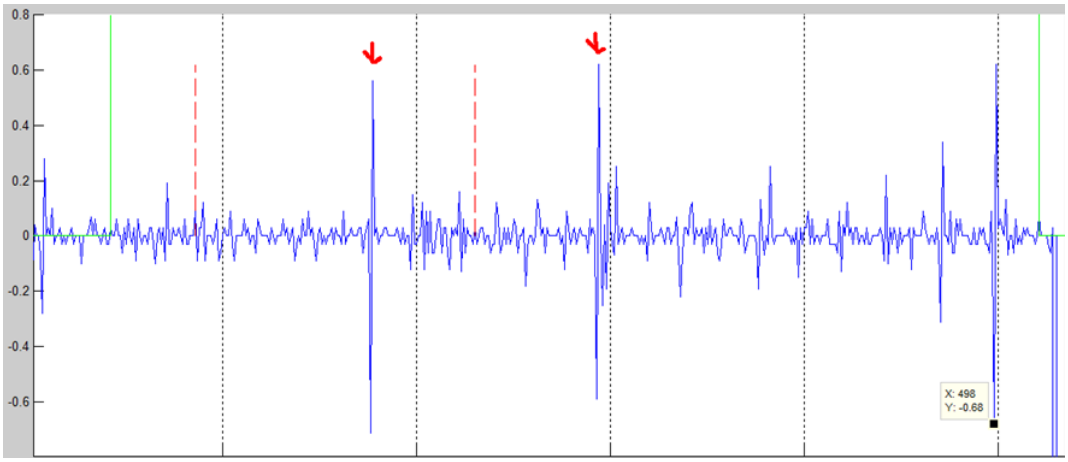


Figura 5. 5 Patrones intensos en *video on*

En cuanto a tendencia de la señal se refiere, existen coincidencias en algunos pacientes, en la que la tendencia cambia de positiva a negativa una vez pasado el 2º crítico.

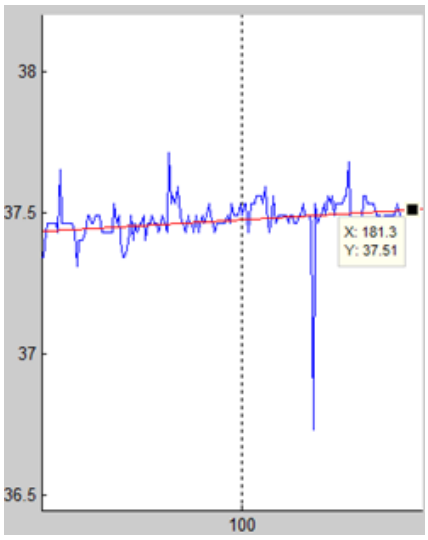


Figura 5. 6 Cambio de tendencia tras crítico 2

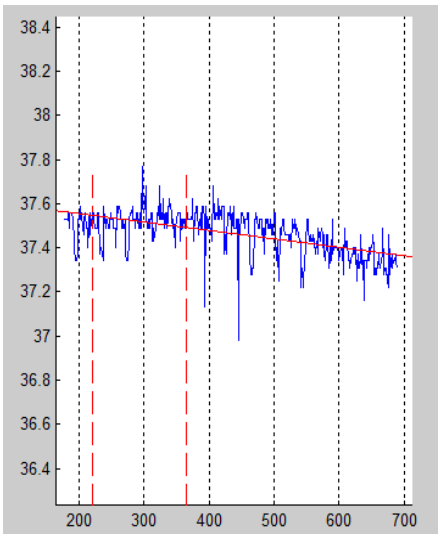


Figura 5. 7 Tendencia positiva al comienzo del vídeo

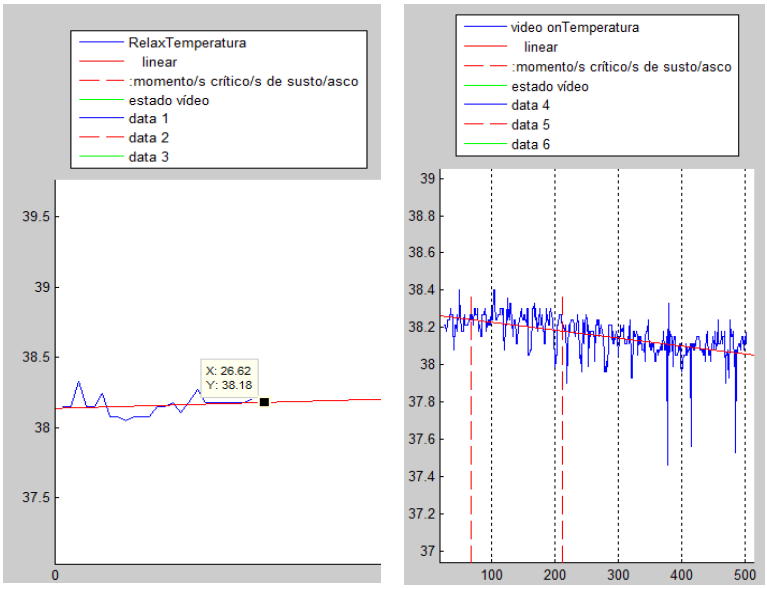


Figura 5. 8 Mismo comportamiento en otro paciente

En este sentido, parece haber un patrón, pero en otros pacientes, se observa que el aligeramiento de la tendencia positiva es más lento, y cabe la posibilidad, de que el cambio de tendencia no se termine de producir por el hecho de que no se ha dado el suficiente tiempo al sensor a alcanzar el valor de la temperatura corporal del paciente. Por lo que se reserva en este aspecto, un inciso para el capítulo 6, como asunto a tener muy en cuenta en monitorizaciones futuras.

Otro matiz interesante, que se puede observar bien en el gradiente, es, en términos de valor absoluto, cómo de intensos son los patrones de decrementos e incrementos abruptos que se producen. Es curioso como los mayores valores de dos

pacientes apenas llegan a los ± 0.4 grados, mientras que entre el resto de pacientes el máximo más bajos que se puede encontrar gira en torno al ± 0.7 grados.

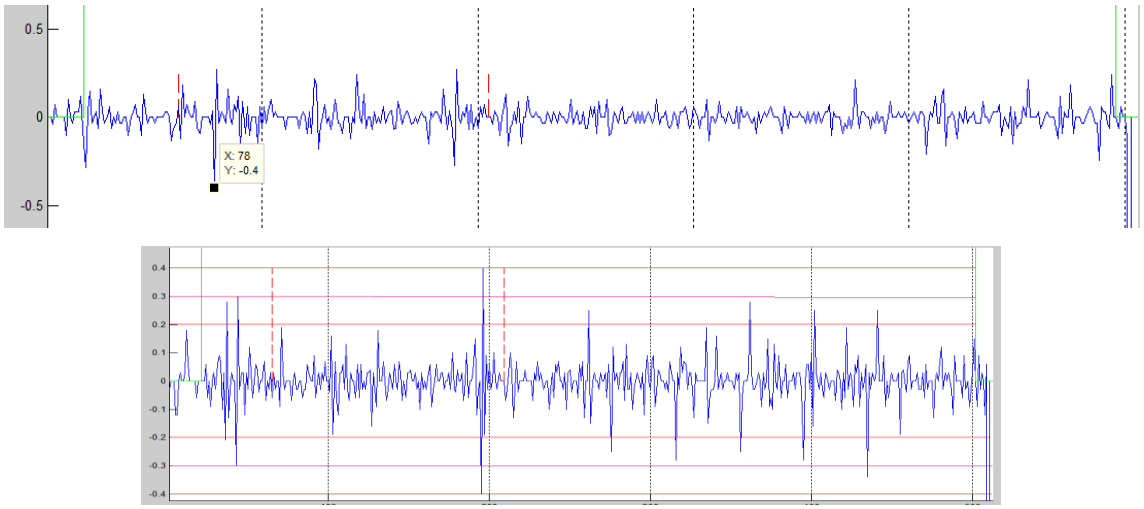


Figura 5. 9 Pacientes con baja intensidad en los patrones

5.1.2 Pulso

En la gran mayoría de los pacientes se observa cómo una vez comienza el vídeo existe una tendencia del pulso a descender para, pasado un tiempo, ascender, en muchos de los casos, a máximos superiores a las zonas en el que el vídeo no se reproduce.

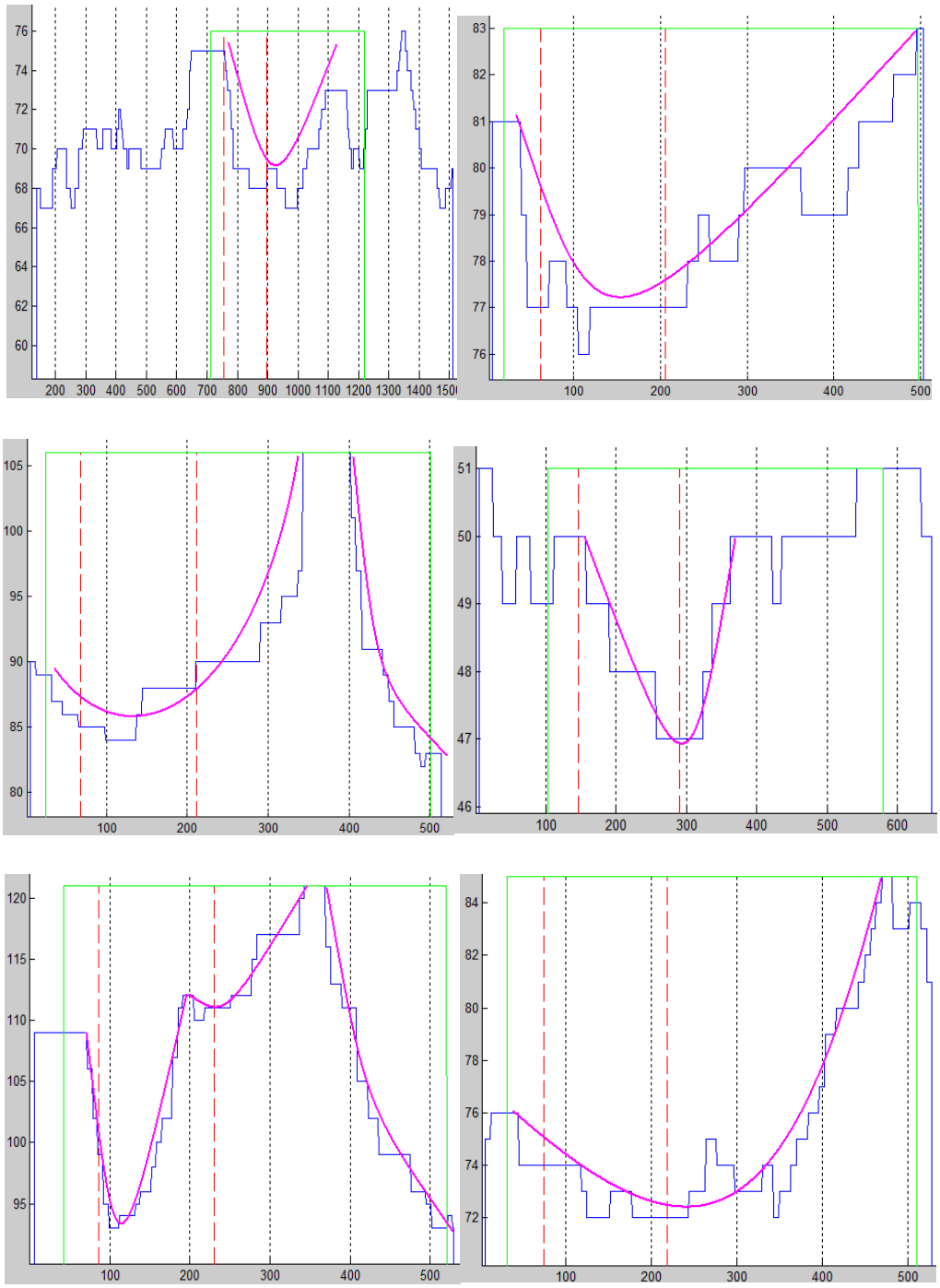


Figura 5. 10 Tendencias similares en pulso

En cuanto a los críticos parece no existir una reacción clara a ellos. Sin embargo, en cuanto a reacción a la emoción en general (*video on*), sí que podría ser interesante la monitorización de esta variable corporal.

5.1.3 Oxígeno

Es esta variable una de las más llamativas de las que se han analizado, debido a que tiene una claro cambio de comportamiento una vez se inicia el vídeo. Valor añadido, ya que suele ser un parámetro biológico bastante constante.

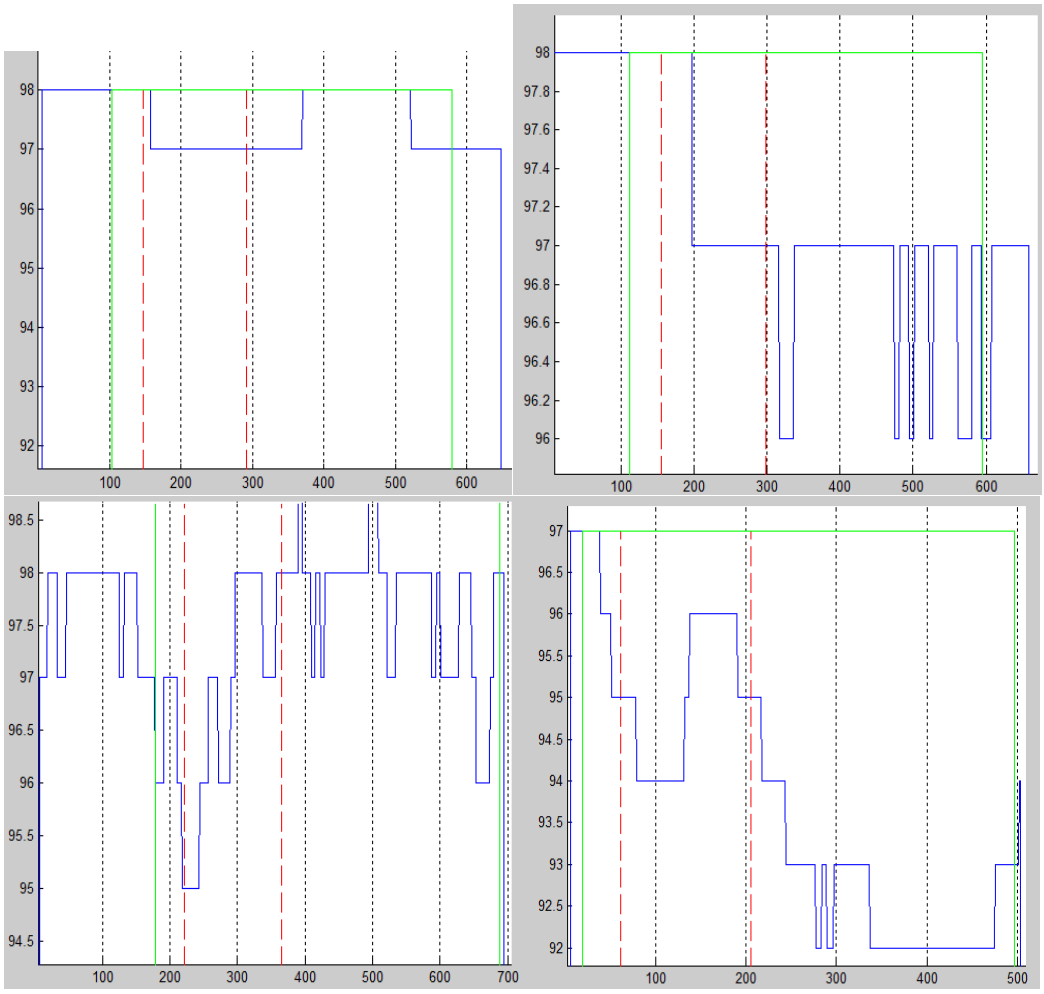


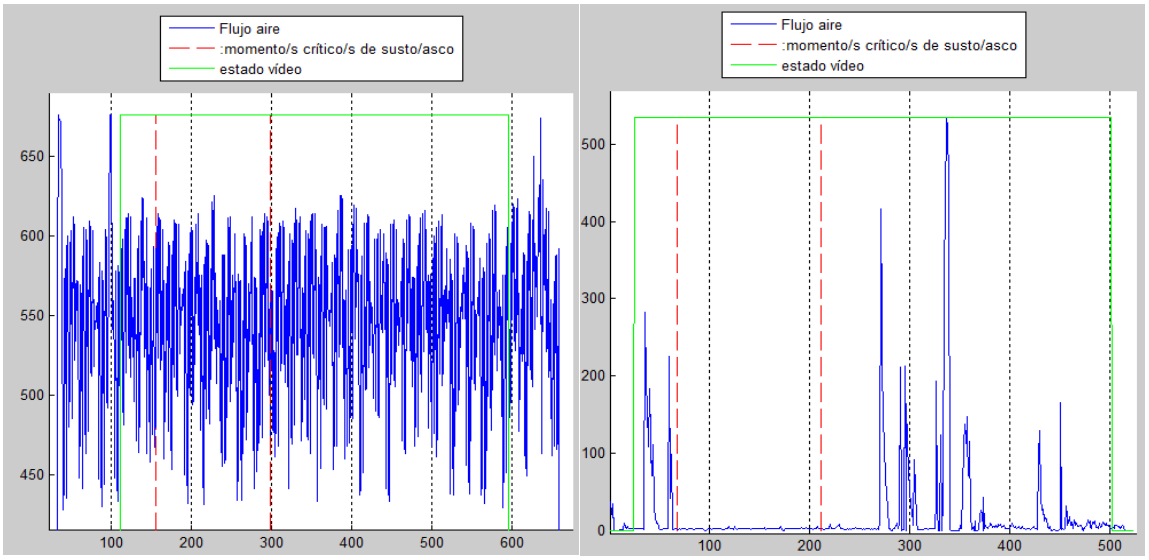
Figura 5. 11 Oxígeno en asco

Como se aprecia la tendencia generalizada es una constante inicial, que una vez comenzado el vídeo disminuye. En algún paciente (como el de la última gráfica), llega incluso a disminuir a valores por debajo de los normales (entre 99% y 95%).

Aunque después del primer crítico, el pulso ya suele estar bajando en valor, es muy difícil asociar alguna reacción en particular a los críticos. Hay que recordar que la situación de los críticos en el vídeo, es una acción un tanto subjetiva, ya que dependiendo de la persona que lo haga, le repugnará tal vez más una situación determinada que otra.

5.1.4 Flujo de aire

Si la variable biológica anterior se puede considerar muy positiva para los objetivos de este TFG, el flujo de aire, a simple vista, parece todo lo contrario. Si es verdad, que tal vez esto sea adelantarse a los acontecimientos, ya que como mencionábamos, en el capítulo 2, este sensor es particularmente complicado de colocar idóneamente al paciente. Esto se aprecia claramente en las gráficas, donde algunas arrojan valores muy altos, y en cambio, otras más bien bajos.



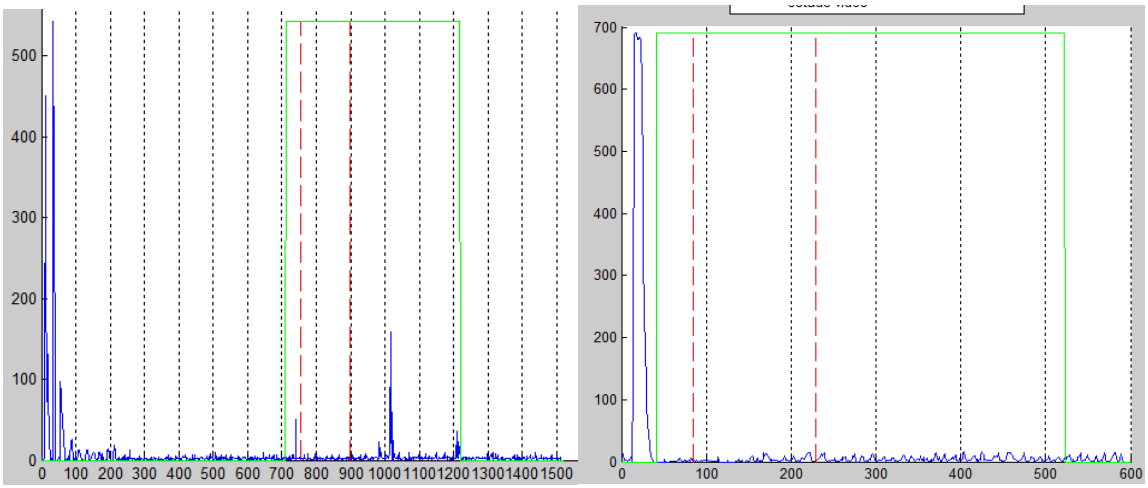
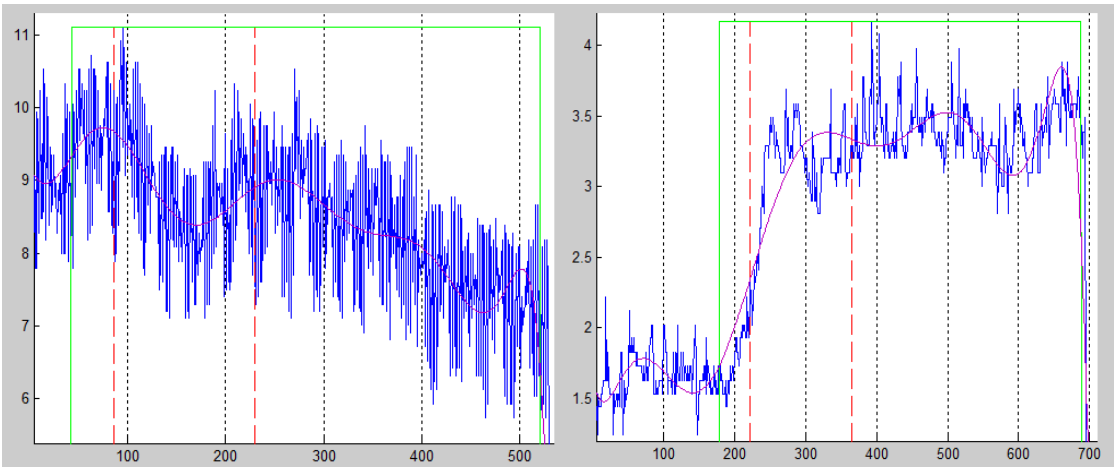


Figura 5. 12 Flujo de aire en asco

5.1.5 Conductancia

Al igual que con el flujo de aire, cada paciente parece tener un comportamiento diferente.



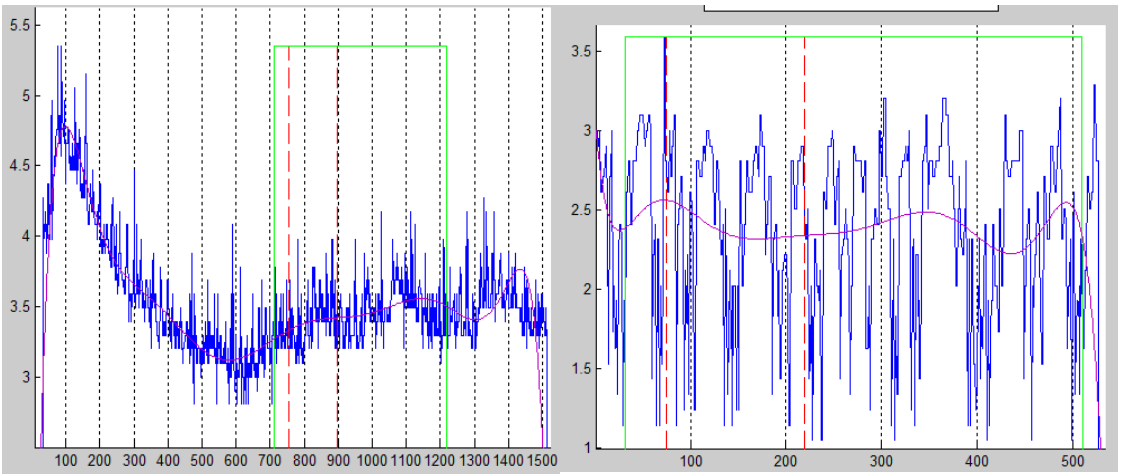


Figura 5. 13 Conductancia en asco

5.2 Análisis gráfico vídeo miedo

Para evitar que la sección de análisis gráfica se torne demasiado repetitiva, se opta en este tipo de vídeo a comparar las reacciones de ciertos pacientes con las que tuvo en la parte del asco.

5.2.1 Temperatura

En las siguientes figuras, se puede apreciar que los patrones entre tipos de vídeo tienen parentescos.

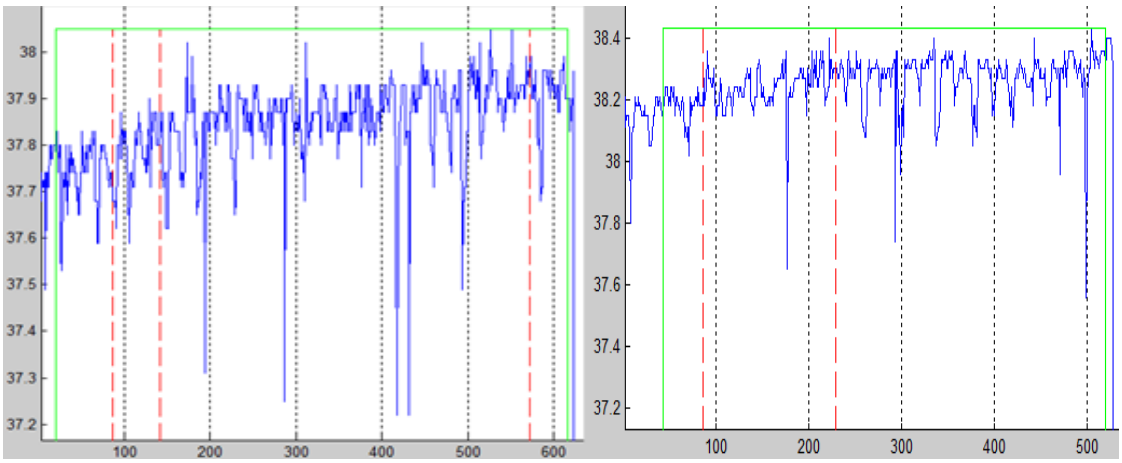


Figura 5. 14 Para paciente 1: izquierda temperatura asco, derecha temperatura miedo

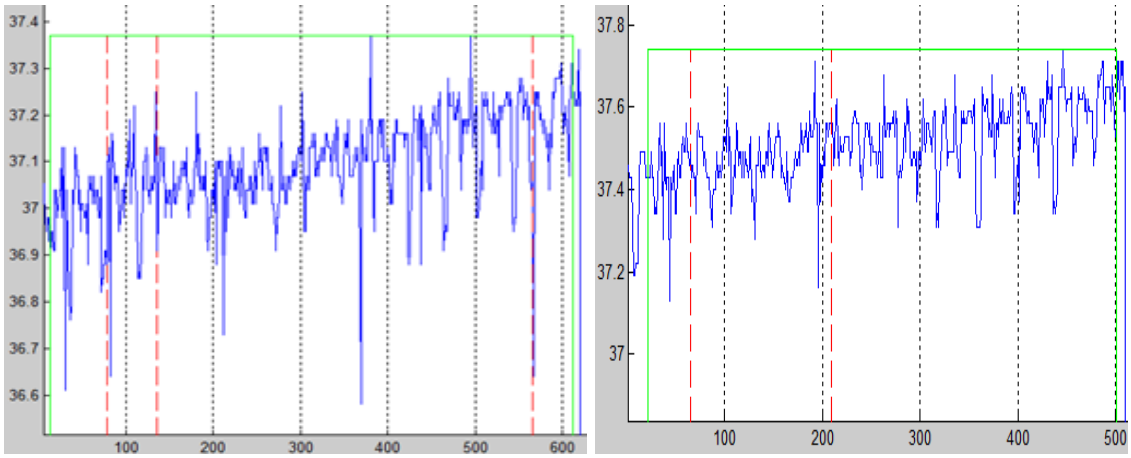


Figura 5. 15 Para paciente 1: izquierda temperatura asco, derecha temperatura miedo

Las dos comparaciones anteriores, la gráficas correspondientes a los vídeo tipo miedo, parte de una temperatura menor que en las de tipo asco. Esto no tiene por qué tiene que ser indicativo de cambio de comportamiento ante diferentes estímulos, sino que tal vez, o no se dio el suficiente tiempo a que el sensor se estabilizara, o el paciente no le tenía con el suficiente contacto.

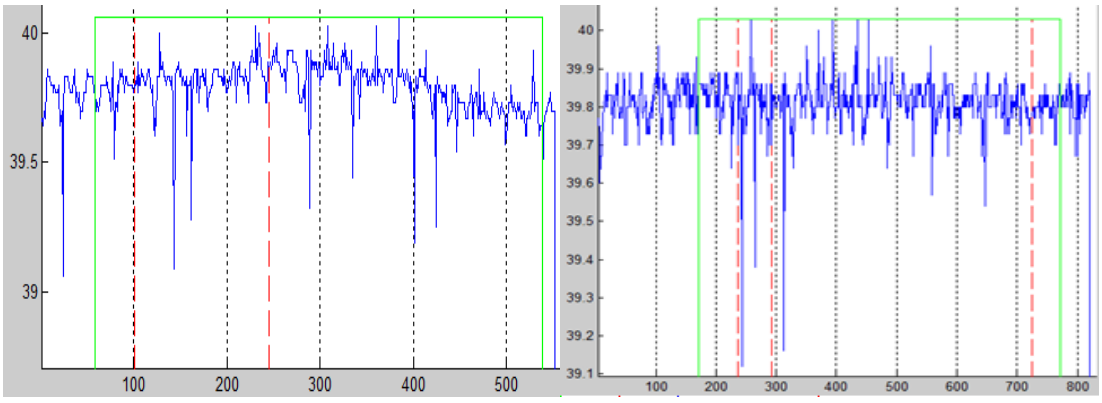


Figura 5. 16 Para paciente 3: izquierda temperatura asco, derecha temperatura miedo

En la última comparación, los valores son similares. Se puede decir también, que en general no hay diferencias llamativas entre vídeos.

5.2.2 Pulso

A diferencia de la temperatura, el comportamiento de los pacientes ante vídeos de distinto tipo, cambia totalmente de un vídeo a otro. Curiosamente, en el segundo ejemplo, ni siquiera en cuanto a valor medio se refiere.

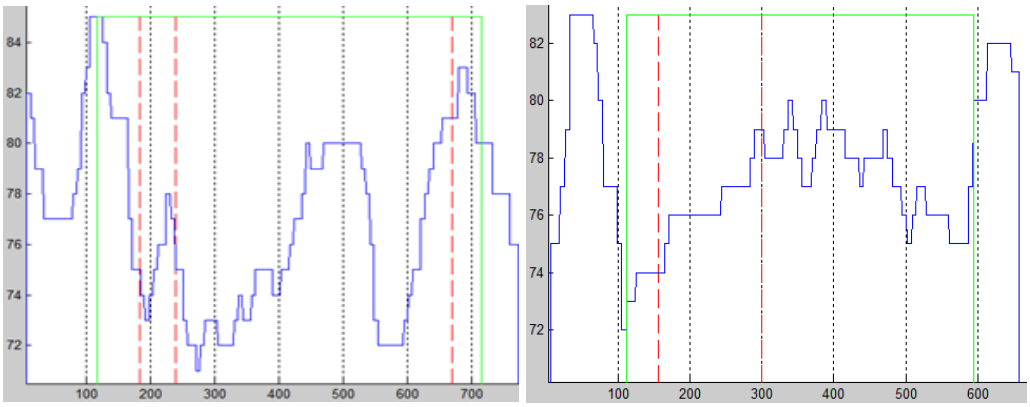


Figura 5. 17 Paciente 1: izquierda pulso miedo, derecha pulso asco

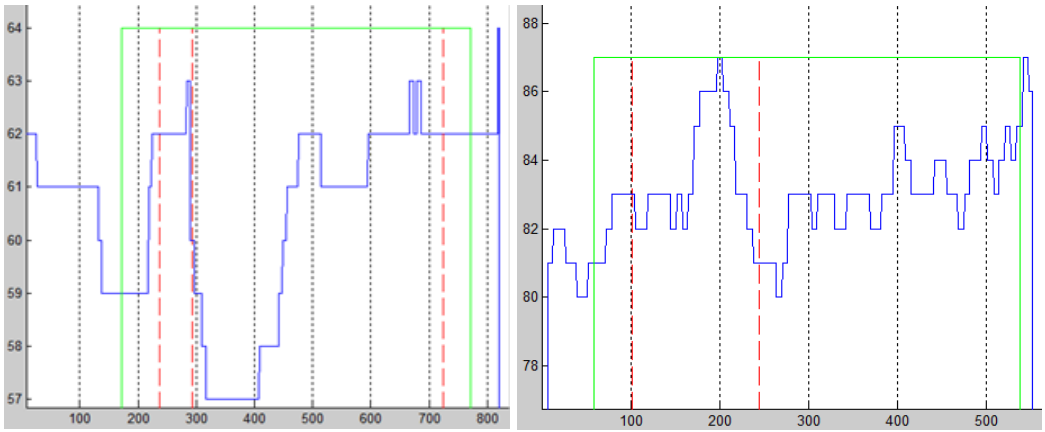


Figura 5. 18 Paciente 2: izquierda pulso miedo, derecha pulso asco

5.2.3 Oxígeno

Aunque nuevamente parece no existir relación entre vídeos, se vuelve apreciar en el vídeo tipo miedo, que el comportamiento una vez comienza el vídeo, cambia. Aspecto interesante, a la hora de discernir entre zona de no estímulo y de cuándo comienza éste.

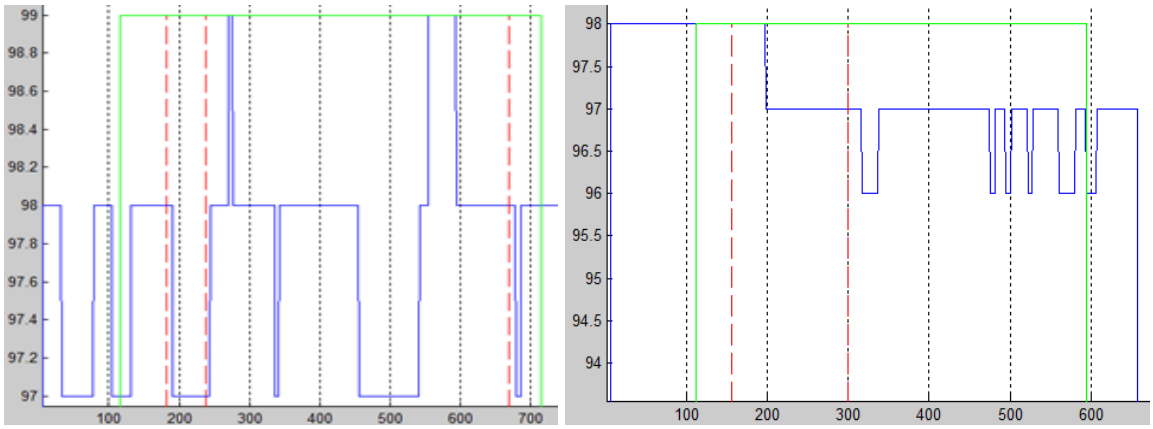


Figura 5. 19 Paciente 1: izquierda oxígeno miedo, derecha oxígeno asco

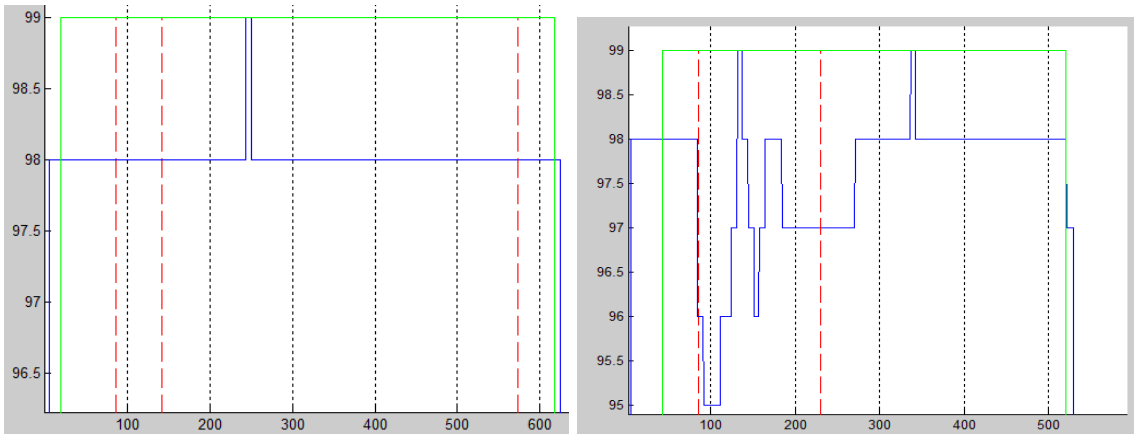


Figura 5. 20 Paciente 2: izquierda oxígeno miedo, derecha oxígeno asco

5.2.4 Flujo de aire

Si se hace notar con el flujo de aire que hay una respuesta similar entre vídeos. Aun así, siguen siendo muy diferentes entre pacientes. Y no parecen, por lo tanto, aptos para satisfacer los objetivos que se buscan en este TFG.

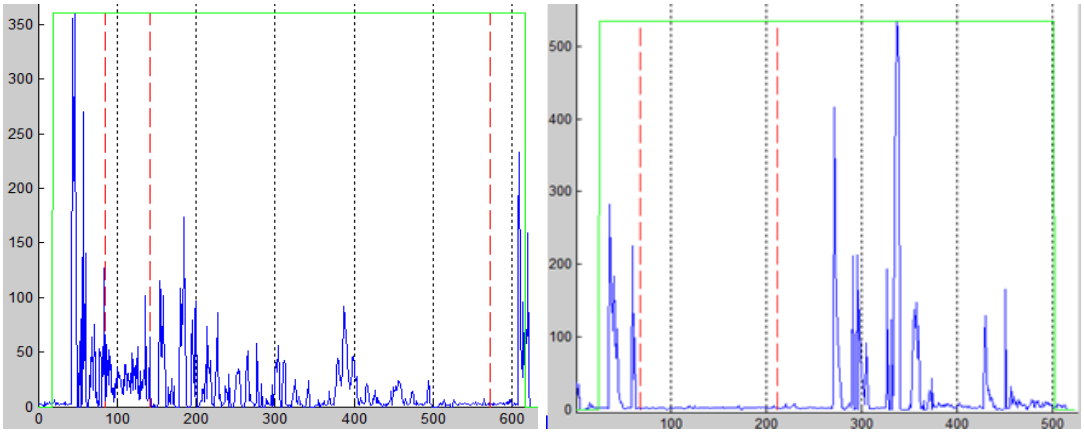


Figura 5. 21 Paciente 1: izquierda flujo de aire miedo, derecha flujo de aire asco

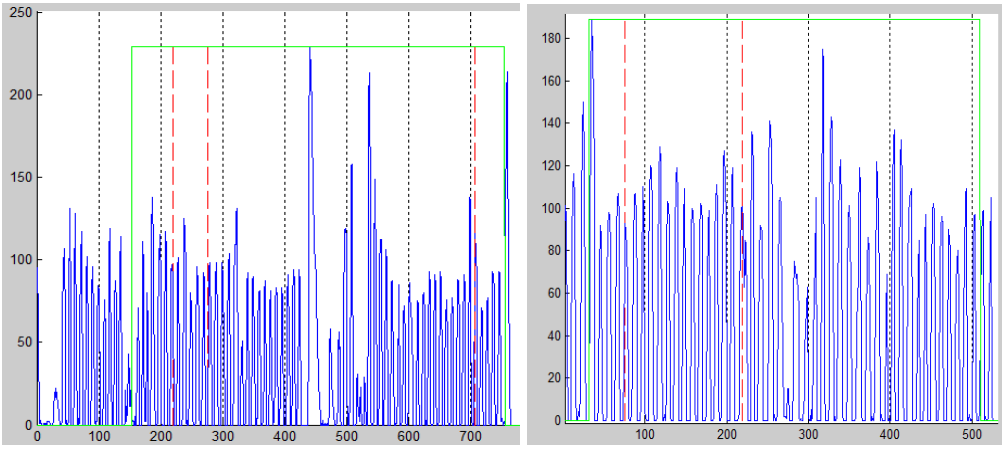


Figura 5. 22 Paciente 2: izquierda flujo de aire miedo, derecha flujo de aire asco

5.2.5 Conductancia

Como se puede observar, aunque en valores absolutos si se ve que hay relación entre vídeos, no así en comportamientos.

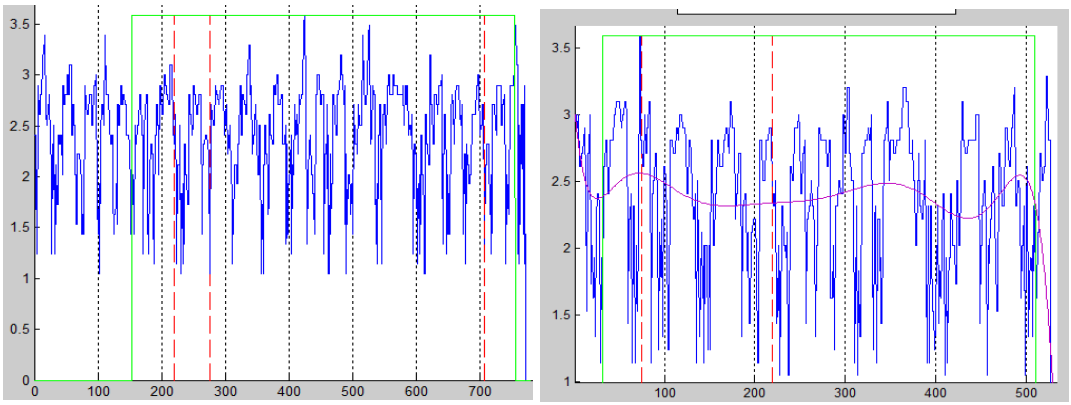


Figura 5. 23 Paciente 1: izquierda conductancia miedo, derecha conductancia asco

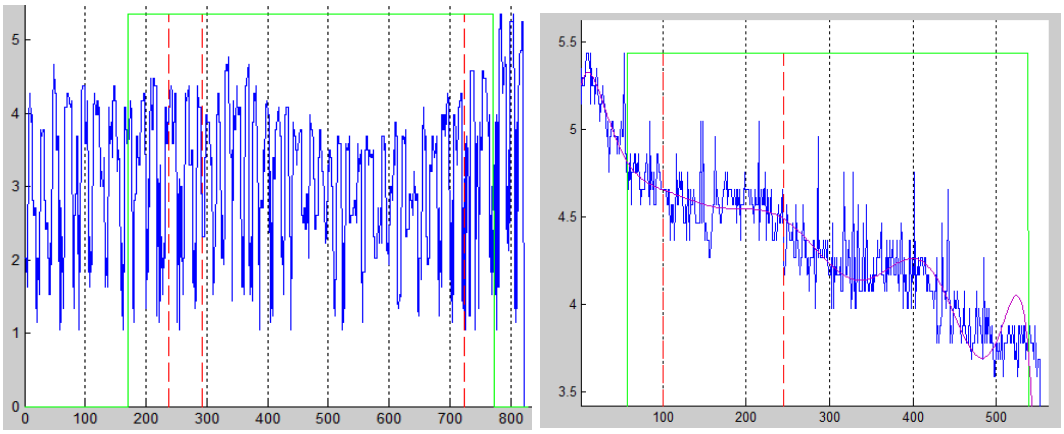


Figura 5. 24 Paciente 2: izquierda conductancia miedo, derecha conductancia asco

A modo resumen, se pueden sacar ciertas conclusiones de éste tipo de análisis. Por un lado, existen ciertas variables que parecen prestarse mejor a la obtención de los objetivos propuestos, mientras que, en cuanto a reacción tras sucederse los críticos (momentos intensos) es complicado encontrar un patrón común a los pacientes. Por otro lado, queda visualmente patente, cuan diferentes pueden ser los resultados, no sólo entre pacientes, sino también entre tipos de vídeos.

Capítulo 6. Segmentación

Con vistas a hacer un futuro análisis estadístico, y por lo tanto más riguroso. Se procede ahora a separar las señales en estados, para así, poder diferenciarlas conceptualmente en tramos. Surgen con esta intención hasta 5 diferentes estados:

1. Antes del vídeo (*estado relax*), desde que se conecta el dispositivo hasta que comienza el vídeo.
2. Inicio del vídeo (estado neutro), desde comienzo vídeo hasta el primer susto/asco.
3. Momento crítico (*emoción*), comienzo susto/asco hasta fin de su reacción (se asignan 3 segundos de reacción).
4. Después del momento crítico (*acomodación de emoción a neutro*), desde último susto/asco hasta fin de vídeo.
5. Después del vídeo (*vuelta al relax*), desde fin de vídeo hasta desconexión.

También se tuvo en cuenta en esta separación de estados, que los puntos que delimitan el estado 2, del 3 y del 4 son los momentos críticos. La colocación de éstos, como se comentó anteriormente, es de una cierta naturaleza subjetiva, ya que a un persona le puede parecer un momento lo suficientemente intenso como para considerarlo crítico, y tal vez a otra persona no. Para paliar de alguna manera esta indecisión de en qué puntos han de separarse éstos estados, se llevó a cabo un solapamiento entre estados, en la que el inicio de uno acoge parte del final del anterior y viceversa.

Por otro lado, la conformación del estado 3 y 4 también admite varios puntos de vista. Por un lado, el que se denominó caso 1, asume que el estado 3 son todas aquellas reacciones de 3 segundos después de cada crítico, y el estado 4 es el que desde la reacción del último crítico hasta que termina el vídeo. En cambio, el caso 2, mantiene el estado 3 igual que en el caso 1, pero el estado 4 se conforma de todas aquellas tramas que se encuentran entre reacción de un crítico y el siguiente. Y por último, el caso 3, mantiene el estado 4 como el del estado 1, y el estado 3 abarca toda la zona que va desde el primer crítico hasta el último incluida su reacción. Todas estas posibilidades y sus solapes se resumen en la siguiente figura.

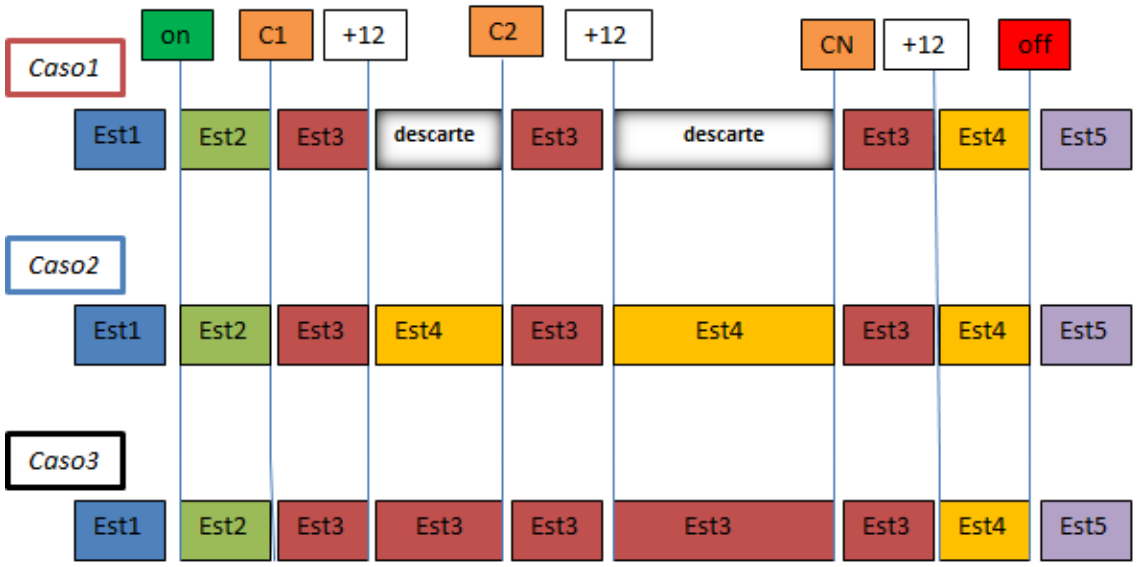


Figura 6. 1 Separación de estados según caso

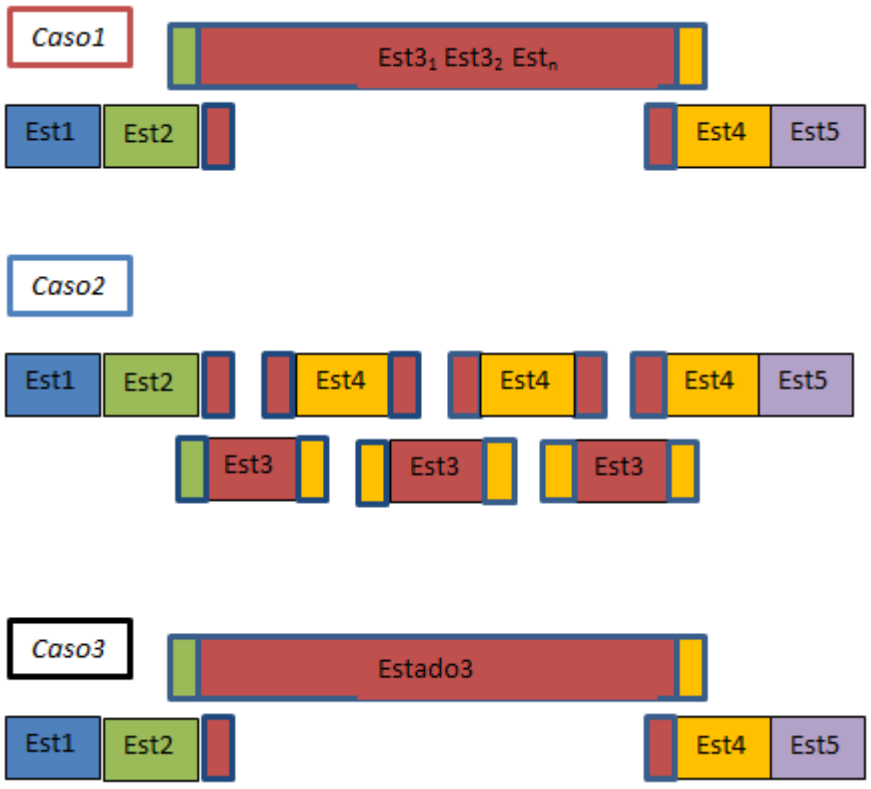


Figura 6. 2 Separación de estados según caso con solape

Todo el procedimiento que se ha explicado, se encuentra en la función *txt_a_parametros*, se llamó así porque también incluye la parametrización de variables

que se explica en el siguiente apartado. Inicialmente, como en *txt_a_graficas*, se separan las variables fisiológicas en variables mediante la función *import*.

Una vez separadas, como el estado1, 2 y 5 son iguales en todos los casos, se procede a la formación de estos estados inicialmente.

```
estado1=datos(1:video_on,1:5);   %el estado1 va desde el inicio de la grabación hasta que comienza

                %*****************Estado2*******************

estado2=datos(video_on+1:video_on +interfaz.criticos_videos(1)-1,1:5);
estado2_solapado=datos(video_on+1:video_on +interfaz.criticos_videos(1)-1+numMuestras_solape,1:5);


                %*****************Estado5*******************

estado5=datos(video_off+1:length(datos),1:5);   %el estado5 va desde el final del vídeo a la desco
```

Código 6. 1 Estados 1,2 y 5

Para el estado 3, se juega con la posición de los críticos.

```
for i=1:ultimo_critico
    critico=datos(video_on+interfaz.criticos_videos(i):video_on+interfaz.criticos_videos(i)+tiempo_reaccion,1:5);
    estado3_caso1=vertcat(estado3_caso1,critico); %la función vertcat me concatena verticalmente los trozos corre
end
```

Código 6. 2 Estado 3

El estado 4, como se dijo, va desde el último crítico hasta el final. Una vez, conformados todos los estados, se hace lo propio con los solapes.

```
estado4_caso1=datos(video_on+interfaz.criticos_videos(ultimo_critico)+(tiempo_reaccion+1):video_off,1:5);
fin_estado2=length(estado2_solapado)-numMuestras_solape;
fin_estado3=length(estado3_caso1);
fin_estado4=length(estado4_caso1);
s3_2=estado2(fin_estado2-numMuestras_solape+1:fin_estado2-1,1:5);  %s3_2 es el solape que se añade al est

if fin_estado4>numMuestras_solape  %Como se puede dar el caso que el estado4 se menor que numMuestras_sol
    s3_4=estado4_caso1(1:numMuestras_solape,1:5);%s3_4 es el solape que se añade al estado3 y que provien
else
    s3_4=estado4_caso1(1:fin_estado4,1:5);
end

s4_3=estado3_caso1(fin_estado3-numMuestras_solape-1:fin_estado3,1:5);%s4_3 es el solape que se añade al e
estado3_caso1_solapado=vertcat(s3_2,estado3_caso1,s3_4);
estado4_caso1_solapado=vertcat(s4_3,estado4_caso1);
```

Código 6. 3 Estado 4 y solapes

El código de los casos 2 y 3, son muy similares, introduciendo los cambios que se explicaron en el comienzo de esta sección.

Capítulo 7. Parametrización

Con el fin de extraer las propiedades estadísticas de las variables fisiológicas, se llevan a cabo una serie de parametrizaciones. Para ello se usarán estadísticos de tipo centralización, dispersión, asimetría y de forma.

7.1 Media

Con este estadístico de centralización se calcula el promedio de cada una de las variables fisiológicas en cuestión, y por lo tanto, representar las propiedades estadísticas de la amplitud aleatoria de la señal. Se puede estimar como el centro de gravedad de cada vector de datos.

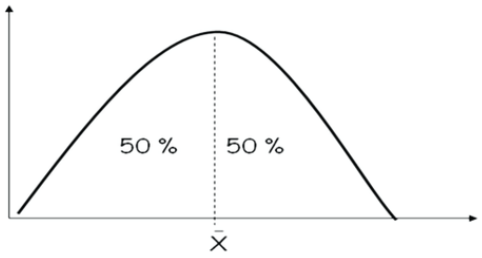


Figura 7. 1 Media

La función Matlab para la media es la función *mean()*.

```
%CASO1
media_estado1_caso1=mean(estado1);
media_estado2_caso1=mean(estado2_solapado);
media_estado3_caso1=mean(estado3_caso1_solapado);
media_estado4_caso1=mean(estado4_caso1_solapado);
media_estado5_caso1=mean(estado5);
```

Código 7. 1 Media de los estados

Para el resto de la casos es igual pero con sus correspondientes estados y solapes. Los estados se agruparán en una matriz por caso de la siguiente manera.

```
MEDIA_CASO1= [media_estado1_caso1 media_estado2_caso1 media_estado3_caso1 media_estado4_caso1 media_estado5_caso1];
MEDIA_CASO2= [media_estado1_caso2 media_estado2_caso2 media_estado3_caso2 media_estado4_caso2 media_estado5_caso2];
MEDIA_CASO3= [media_estado1_caso3 media_estado2_caso3 media_estado3_caso3 media_estado4_caso3 media_estado5_caso3];
```

Código 7. 2 Matrices MEDIA por casos

Para el cálculo del resto de parámetros que se citan a continuación, se lleva a cabo el mismo procedimiento que el de la media pero con la función Matlab correspondiente.

7.2 Varianza

Es una medida de dispersión, por lo que, se logra calcular el promedio de las desviaciones al cuadrado con respecto a la media. Se utilizará la función *var().*

```
%CASO1
varianza_estado1_caso1=var(estado1);
varianza_estado2_caso1=var(estado2_solapado);
varianza_estado3_caso1=var(estado3_caso1_solapado);
varianza_estado4_caso1=var(estado4_caso1_solapado);
varianza_estado5_caso1=var(estado5);
```

Código 7. 3 Varianza de los estados

Con los demás estados se realiza el mismo procedimiento para agruparlos todos finalmente en una matriz varianza para cada uno de los casos.

```
VARIANZA_CASO1=[varianza_estado1_caso1 varianza_estado2_caso1 varianza_estado3_caso1 varianza_estado4_caso1 varianza_estado5_caso1];
VARIANZA_CASO2=[varianza_estado1_caso2 varianza_estado2_caso2 varianza_estado3_caso2 varianza_estado4_caso2 varianza_estado5_caso2];
VARIANZA_CASO3=[varianza_estado1_caso3 varianza_estado2_caso3 varianza_estado3_caso3 varianza_estado4_caso3 varianza_estado5_caso3];
```

Código 7. 4 Matrices VARIANZA por caso

7.3 Curtosis

Con la curtosis se mide cuánto se aplasta la distribución con respecto a la gaussiana o normal, es decir, se mide la mayor o menor concentración de datos alrededor de la media. Por lo tanto, si es nula, la distribución coincide con la de la distribución normal, si es positivo, es más puntiaguda, y es negativo, más achatado [30].

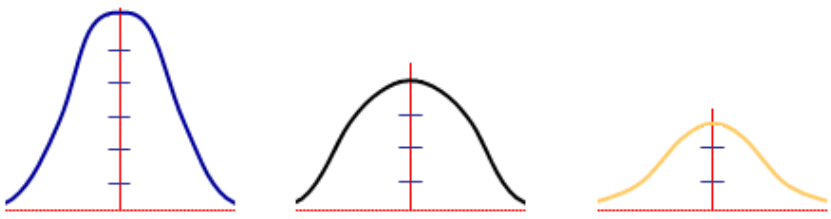


Figura 7. 2 Posibles distribuciones *curtosis*

La función Matlab correspondiente es la de *kurtosis().*

```
%CASO1
kurtosis_estado1_caso1=kurtosis(estado1);
kurtosis_estado2_caso1=kurtosis(estado2_solapado);
kurtosis_estado3_caso1=kurtosis(estado3_caso1_solapado);
kurtosis_estado4_caso1=kurtosis(estado4_caso1_solapado);
kurtosis_estado5_caso1=kurtosis(estado5);
```

Código 7. 5 Curtosis de los estados

Al igual que en los casos anteriores, se lleva a cabo lo mismo para todos los casos y se agrupa en una matriz KURTOSIS por caso.

```
KURTOSIS_CASO1=[kurtosis_estado1_caso1 kurtosis_estado2_caso1 kurtosis_estado3_caso1 kurtosis_estado4_caso1 kurtosis_estado5_caso1];
KURTOSIS_CASO2=[kurtosis_estado1_caso2 kurtosis_estado2_caso2 kurtosis_estado3_caso2 kurtosis_estado4_caso2 kurtosis_estado5_caso2];
KURTOSIS_CASO3=[kurtosis_estado1_caso3 kurtosis_estado2_caso3 kurtosis_estado3_caso3 kurtosis_estado4_caso3 kurtosis_estado5_caso3];
```

Código 7. 6 Matrices KURTOSIS por caso

7.4 Entropía

El concepto de entropía en teoría de la información tiene que ver con la cantidad de desorden que contiene la señal aleatoria. De esta forma, se puede relacionar de la cantidad de información que lleva una señal. Por lo tanto, un pequeño cambio en una de las probabilidades de aparición de algún elemento de uno de nuestros vectores de datos debe cambiar poco la entropía [31]. Por lo tanto, si todos los datos son equiprobables a la hora de aparecer, la entropía tomará su valor máximo. Aparte de la entropía (entropy) se calculará también la entropía local (*entropyfilt*) y la entropía tipo Shannon (*wentropy*). La entropía local en Matlab calcula la entropía dentro de una vecindad de datos de 9 filas por 9 columnas, mientras que la de Shannon determina una información media del conjunto de valores que puede adoptar. Debido a que estas funciones devuelven un solo valor, es decir, cuando se le pasa cada uno de los estados, calculan la entropía tanto del pulso, como del oxígeno, como del resto de las variables, es importante calcular la entropía para cada una de ellas.

```
entropiaPulso_estado1_caso1=entropy(estado1(1:end,1));
entropiaPulso_estado2_caso1=entropy(estado2_solapado(1:end,1));
entropiaPulso_estado3_caso1=entropy(estado3_caso1_solapado(1:end,1));
entropiaPulso_estado4_caso1=entropy(estado4_caso1_solapado(1:end,1));
entropiaPulso_estado5_caso1=entropy(estado5(1:end,1));

entropiaOxigeno_estado1_caso1=entropy(estado1(1:end,2));
entropiaOxigeno_estado2_caso1=entropy(estado2_solapado(1:end,2));
entropiaOxigeno_estado3_caso1=entropy(estado3_caso1_solapado(1:end,2));
entropiaOxigeno_estado4_caso1=entropy(estado4_caso1_solapado(1:end,2));
entropiaOxigeno_estado5_caso1=entropy(estado5(1:end,2));
```

Código 7. 7 Entropía por estados y variables

Creándose la matriz ENTROPIA de los diferentes casos de la siguiente manera.

```
ENTROPIA_CASO1=[entropia_estado1_caso1 entropia_estado2_caso1 entropia_estado3_caso1 entropia_estado4_caso1 entropia_estado5_caso1];
```

Código 7. 8 Matrices ENTROPIA pos casos

Los otros tipos de entropía se llevan a cabo de la misma manera pero usando las funciones que se mencionaron al comienzo de este apartado.

7.5 Asimetría

Una distribución es simétrica si las mitades de su distribución se ven iguales. Si la distribución no es simétrica existirá lo que se llama “cola” de la distribución, que es aquel conjunto de datos que producen la asimetría. Si la asimetría es positiva o negativa dependiendo en qué lado se encuentra la “cola” de la distribución [32].

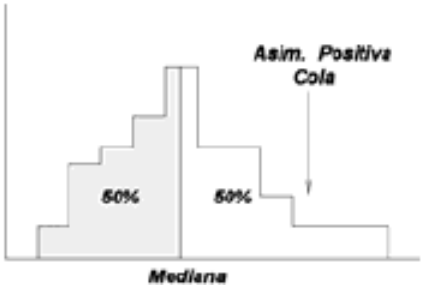


Figura 7. 3Asimetría positiva

La función Matlab a usar es la función *skewness().*

```
oblicuidad_estado1_caso1=skewness(estado1);
oblicuidad_estado2_caso1=skewness(estado2_solapado);
oblicuidad_estado3_caso1=skewness(estado3_caso1_solapado);
oblicuidad_estado4_caso1=skewness(estado4_caso1_solapado);
oblicuidad_estado5_caso1=skewness(estado5);
```

Código 7. 9 Entropía por estados

Las matrices OBLICUIDAD para cada uno de los casos, se realiza nuevamente de la siguiente forma.

```
OBLICUIDAD_CASO1=[oblicuidad_estado1_caso1 oblicuidad_estado2_caso1 oblicuidad_estado3_caso1 oblicuidad_estado4_caso1 oblicuidad_estado5_caso1];
OBLICUIDAD_CASO2=[oblicuidad_estado1_caso2 oblicuidad_estado2_caso2 oblicuidad_estado3_caso2 oblicuidad_estado4_caso2 oblicuidad_estado5_caso2];
OBLICUIDAD_CASO3=[oblicuidad_estado1_caso3 oblicuidad_estado2_caso3 oblicuidad_estado3_caso3 oblicuidad_estado4_caso3 oblicuidad_estado5_caso3];
```

Código 7. 10 Matrices OBLICUIDAD por casos

Por último para obtener una matriz global que reúna todos los parámetros de todas las variables fisiológicas de un cierto paciente, se hará un último paso en la función *txt_a_parámetros*.

```
PARAMETROS_CASO1=[MEDIA_CASO1_VERTICAL VARIANZA_CASO1_VERTICAL KURTOSIS_CASO1_VERTICAL
    ENTROPIA_CASO1_VERTICAL J_ENTROPIA_CASO1_VERTICAL W_ENTROPIA_CASO1_VERTICAL OBLICUIDAD_CASO1_VERTICAL];
```

Código 7. 11 PARAMETROS caso 1

Así, lógicamente para los tres casos. Por lo que al salvarse, se guardarán por usuario tres parámetros, uno por caso. Con la función *uigetdir(),* se le facilitará al usuario de forma interactiva, el salvado de los parámetros en la carpeta que desee.

```
nombre=strcat('parametros_',paciente.nombre);
carp=uigetdir( 'C:\Users\acaymo\Desktop\TFG\Matlab copia 6\VIDEO ESTÍMULOS\','¿en qué
nombre=[carp,'\',nombre];
save(nombre,'parametros_caso1','parametros_caso2','parametros_caso3');
```

Código 7. 12 Función *uigetdir*

Capítulo 8. Clasificación

Para que los parámetros puedan ser definidos por una serie de características que puedan poseer, se usa un clasificador. Las redes neuronales se han usado ampliamente en multitud de aplicaciones, siendo una la capacidad de realizar una clasificación. Por lo tanto, mediante una serie de pasos, una red neuronal, nos puede llegar a decir, cómo es de probable, que nuestros parámetros generados, cumplan una serie de propiedades, que nos sirvan para un determinado experimento [33]. La base del funcionamiento de la red neuronal es su unidad fundamental, el perceptrón. Es un elemento que tiene varias entradas con un cierto peso cada uno, si la suma de todos los pesos es mayor a un determinado número, la salida es uno. Si es menor, es cero. Para encontrar el peso adecuado de cada entrada es preciso entrenar a la red. El entrenamiento consiste en ir ajustando los pesos por retroalimentación hasta que la decisión de la neurona sea coherente [34].

La red neuronal que se usará en este TFG es la herramienta para este fin que proporciona el entorno Matlab. Se usará esta herramienta en una serie de funciones creadas, que son los experimentos que consideramos oportunos para sacar conclusiones que satisfagan los objetivos propuestos.

8.1 Experimentos vídeo tipo asco

Inicialmente se llevarán a cabo una serie de experimentos en los cuales se busque llegar a una serie de conclusiones respecto a una estimulación tipo asco.

8.1.1 Experimento por caso

En este experimento lo que se busca es qué caso funciona mejor para todas las variables fisiológicas monitorizadas y para todos los parámetros estadísticos realizados a éstas. Cada uno de los experimentos realizados, comienzan siempre con la invocación de dos funciones. La función *parametrosNN*, y la función *crear_Test_Training*. La primera de ellas se encarga de modificar la matriz global PARAMETROS, que es la matriz donde se encuentran los parámetros de todos los usuarios de todos los vídeos, de tal manera que concuerde en forma con la matriz etiqueta. La matriz etiqueta es la matriz con la que se le dice a la red, qué valores deben de tener mis parámetros idealmente. En nuestro caso, la matriz etiqueta es de la siguiente forma.

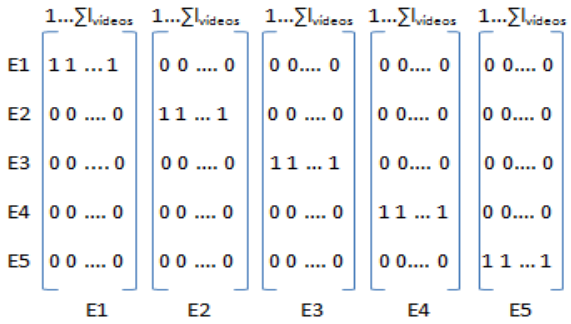


Figura 8. 1 Matriz etiqueta

Como se observa, cuando las columnas correspondientes al estado 1 coincide con la fila del estado 1, sus valores son 1. El resto, de columnas para esta fila son ceros, porque no corresponde a ese estado. Con esta matriz, le estamos diciendo a la red neuronal, que queremos que nos distinga los estados. Por lo tanto, la función *parametrosNN,* se encargará de darle la forma que tiene la matriz etiqueta a nuestra matriz global PARAMETROS. Resultando de la siguiente manera.

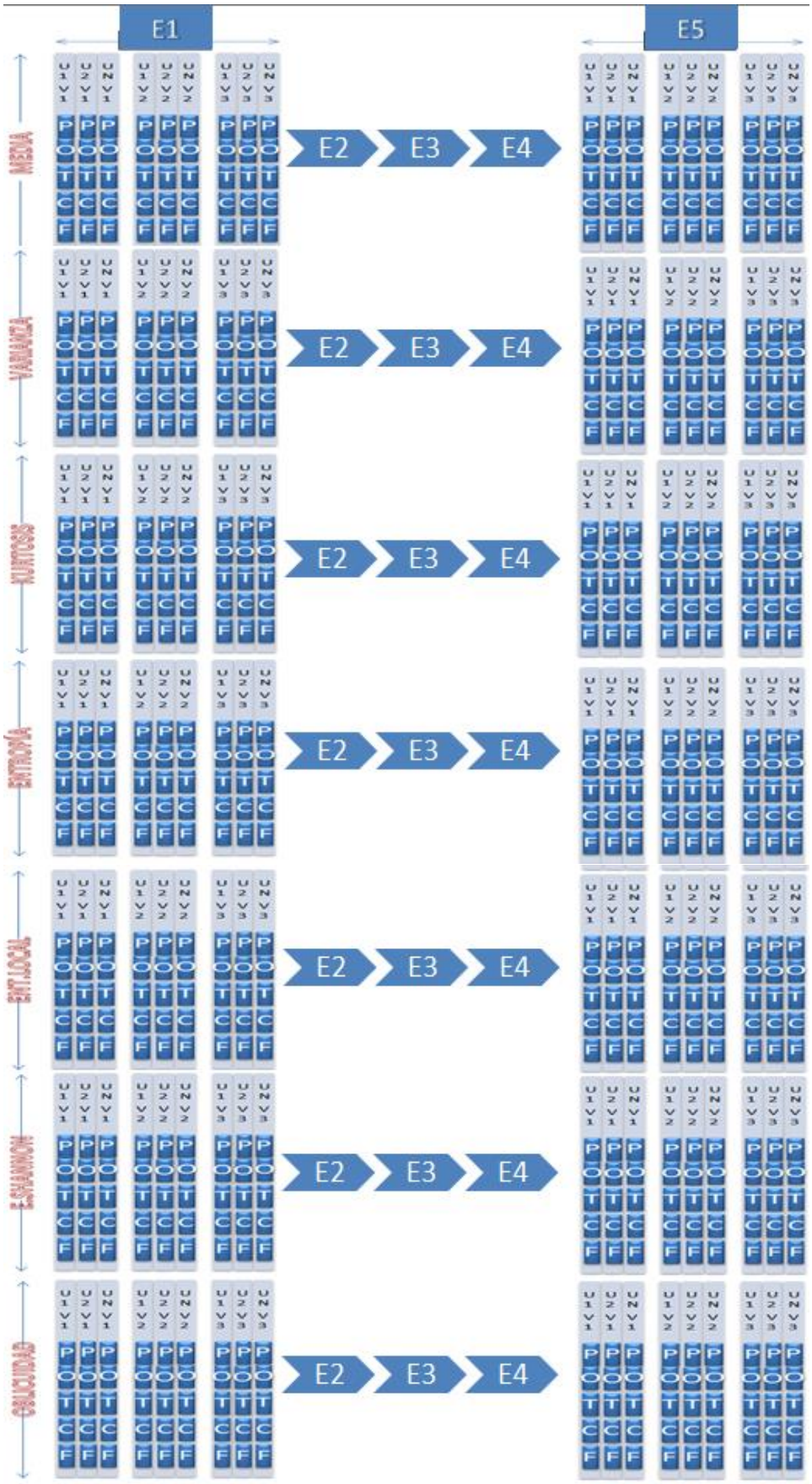


Figura 8. 2 Aspecto de PARAMETROS

Si a su vez, a esta matriz, la modificamos a nuestra semejanza, obtendremos cada uno de los experimentos que se quieren realizar. Es decir, por ejemplo, en este experimento, se quiere ver qué caso funciona mejor. Pues se le introducirá por parámetro a la función *parametrosNN* los diferentes casos. Se crearán las correspondientes matrices a través de *crear_Test_Training* (se explica a continuación) para cada caso, y la red neuronal nos dirá en forma de porcentaje de acierto, qué caso funciona mejor.

```
function experimentoCaso( PARAMETROS,video1,video2, video3,elcaso)
    PARAMETROS_NN = parametrosNN( PARAMETROS,elcaso );
```

Código 8. 1 Llamada a función *parametrosNN*

Como decíamos, la segunda función que se ejecuta en cada inicio de cada experimento es la de *crear_Test_Training.* Esta función se encarga de dividir los parámetros de los tres vídeos que entran por experimento, en dos matrices de parámetros. Una es la matriz de entrenamiento, compuesta por los dos vídeos con menos usuarios, y otra es la matriz de validación, formada por los parámetros del vídeo con más usuarios. A su vez, a cada una de estas matrices hay que generarles sus correspondientes matrices etiqueta, que tendrán la forma que se explicó anteriormente, cada una con su tamaño correspondiente, es decir, una matriz etiqueta tendrá el mismo tamaño que la matriz entrenamiento y otra matriz etiqueta tendrá la de la matriz validación. Todo este proceso es lo que Matlab llama una "función de división de datos". De tal manera, que la red se entrenará usando sólo el subconjunto de la matriz de entrenamiento. Este entrenamiento se dará por finalizado en caso de cumplirse con las condiciones de parada, relacionadas con el subconjunto de la matriz de validación. En el momento que la red se dé por entrenada, se prueba con la matriz etiqueta de manera independiente [35].

Otro aspecto, que mejora el resultado de las redes neuronales, es el de simular los datos de entrenamiento y de validación de forma iterativa, de tal manera, que la salida de esta, sea un resultado promedio mejorado. Se busca un compromiso de número de iteraciones y número de redes para encontrar una salida con la mayor probabilidad de éxito posible. En la siguiente figura se ilustra este hecho.

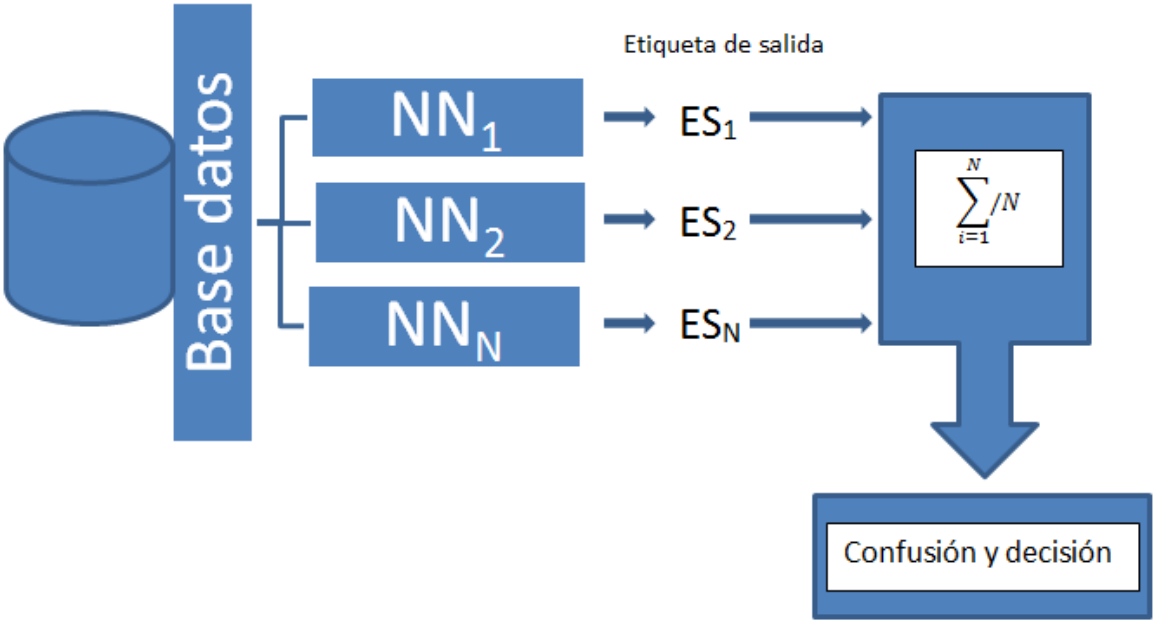


Figura 8. 3 Iteración para una salida mejorada

El código correspondiente a la red neuronal es el siguiente.

```
for i=1:iteraciones
    %Datos entrada de entrenamiento y validacion
    ent = matriz_Training;
    entTest=matrizTest;

    %Se ponen las etiquetas
    etiquetaTrain=matriz_TargetTrain;
    etiquetaTest=matriz_Target_Test;

    %Se normalizan los datos, valores entre [-1,1].
    [pn,ps] = mapminmax(ent);
    [tn,ts] = mapminmax('apply',entTest,ps);

    %RED DETECCION POR ESTADOS
    %Se crea la red (newff) y se entrena
     net = newff(pn,etiquetaTrain,10);
     net = train(net,pn,etiquetaTrain);

    %Se simula la red para obtener salida
    etiquetasalida=sim(net,tn);
    etiquetaSalidaSum=etiquetaSalidaSum+etiquetasalida;
end
    etiquetaSalidaIntegrada=etiquetaSalidaSum/iteraciones;
    figure;plotconfusion(etiquetaTest,etiquetaSalidaIntegrada);
```

Código 8. 2 Iteración en red neuronal

Dónde una vez normalizadas las matrices, se crea la red con la función *newff()*, con un cierto número de neuronas, se entrena con la función *train(),* y se simula con *sim()*, el resultado de esta simulación se suma al de la iteración anterior, para finalmente ser dividida por el número de iteraciones, de tal manera, que el resultado se ha promediado. Este promedio se representa con un *plot* especial, el *plotconfusion*. El resultado se muestra a continuación.

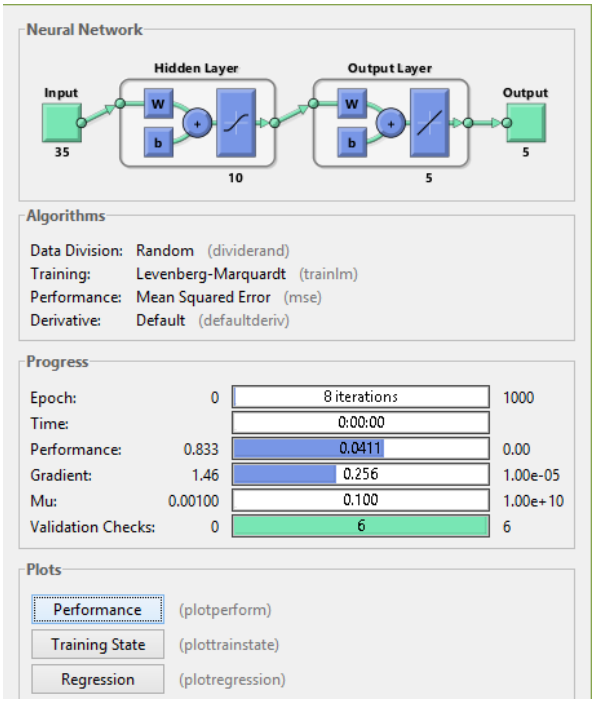


Figura 8. 4Ejemplo de simulación

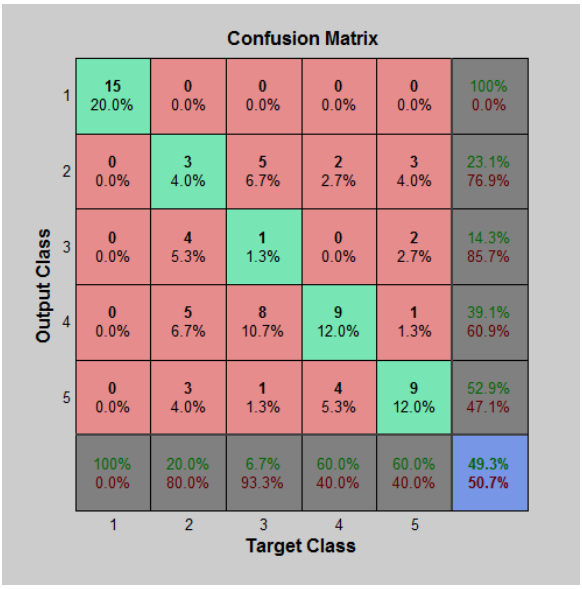


Figura 8. 5 Resultado *plotconfusion*

Como se observa en la figura 8.4, nuestra entrada es de 35 (7 parámetros de 5 variables fisiológicas), acto seguido se encuentra la red de 10 neuronas, el siguiente diagrama se corresponde al aproximador de funciones (compara con la matriz test de 5 estados), por lo que, la salida será de tamaño 5. La figura 8.5 es el resultado del *plotconfusion().* Idealmente, la diagonal de la matriz (verde), debe contener los valores más altos, eso significa, que la red a reconocido cada uno de los estados. El cuadrado azul indica el porcentaje de acierto (número verde) y de confusión (número rojo).

Los resultados para este experimento, tras simular con diferente número de neuronas es el siguiente.

Nºneuronas	%éxito	%éxito	%éxito	%éxito	%éxito	Media
10	28.6	34.3	35.7	30	21.4	30
15	37.1	25.7	34.3	37.1	24.3	31.7
20	31.4	37.1	37.1	30	37.1	34.5
25	24.3	24.3	24.3	25.7	35.7	26.9
30	34.3	30	27.1	34.3	27.1	30.6
35	38.6	24.3	24.3	34.3	34.3	31.1

Tabla 8. 1 Resultados vídeo asco caso1

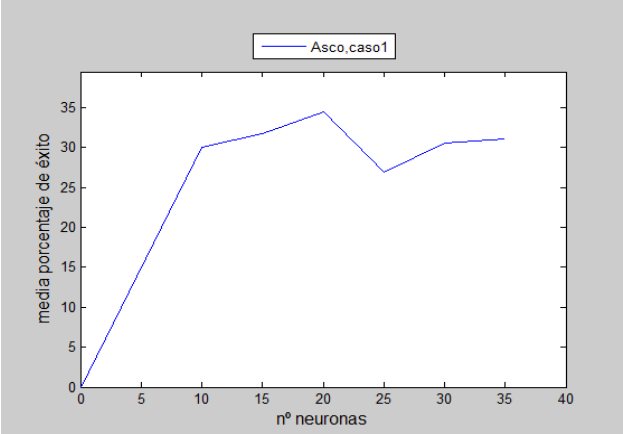


Figura 8. 6 Resultados vídeo asco caso1

Nºneuronas	%éxito	%éxito	%éxito	%éxito	%éxito	Media
10	38.6	42.9	25.7	44.3	47.1	39.7
15	38.6	40	37.1	35.7	47.1	39.7
20	25.7	47.1	47.1	30	41.4	38.2
25	41.4	35.7	51.4	31.4	32.9	38.6
30	48.6	45.7	40	30	28.6	38.6
35	42.9	38.6	35.7	37.1	28.6	36.6

Tabla 8. 2 Resultados vídeo asco caso 2

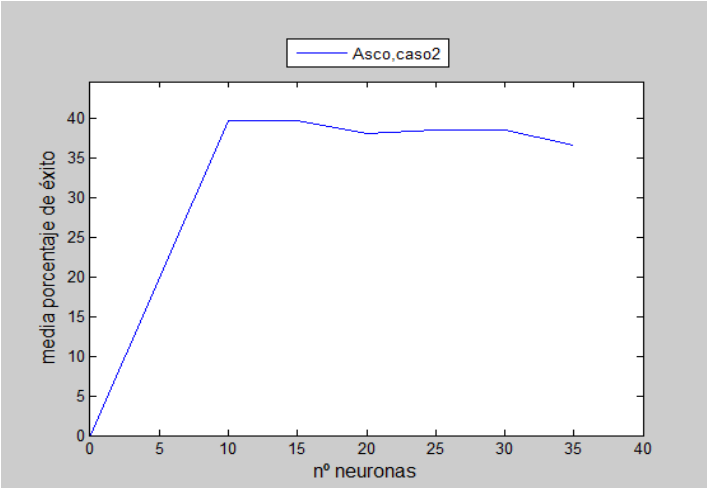


Figura 8. 7 Resultados vídeo asco caso 2

Nºneuronas	%éxito	%éxito	%éxito	%éxito	%éxito	Media
10	26.6	32.9	24.3	24.3	28.6	27.7
15	25.7	27.1	34.3	31.4	35.7	30.8
20	31.4	27.1	34.3	31.4	35.7	30.8
25	17.1	30	30	31.4	44.3	30.6
30	22.9	35.7	38.6	28.6	25.7	30.3
35	32.9	34.3	31.4	34.3	32.9	33.1

Tabla 8. 3 Resultados vídeo asco caso 3

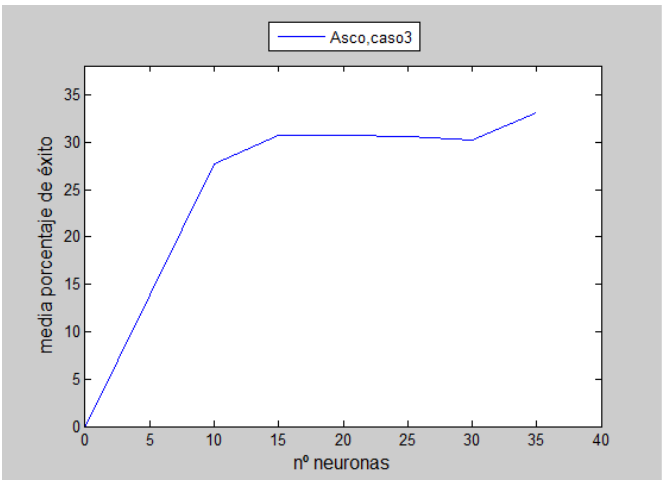


Figura 8. 8 Resultados vídeo asco caso 3

Como se ve, obtenemos un 10% más de acierto si usamos el caso 2 en el caso de introducir vídeos tipo asco.

8.1.2 Experimento por variable fisiológica

El siguiente experimento que puede interesar, sería el de ver qué variable fisiológica responde mejor. Para ello, se crea una función que llamamos *experimentoSenyal()*, la cual separa de los parámetros la señal especificada y acto seguido se dan los pasos explicados en el experimento anterior. Los resultados de cada variable por separado, y para vídeos de asco, se muestran a continuación.

Nºneuronas	%éxito	%éxito	%éxito	%éxito	%éxito	Media
10	41.4	41.4	41.4	32.9	45.7	40.6
15	38.6	34.3	37.1	40	47.1	39.4
20	35.7	44.3	48.6	48.6	31.4	41.7
25	38.6	48.6	45.7	40	54.3	45.4
30	41.4	24.3	44.3	31.4	37.1	35.7
35	34.3	37.1	40	35.7	37.1	36.8

Tabla 8. 4 Pulso asco caso 1

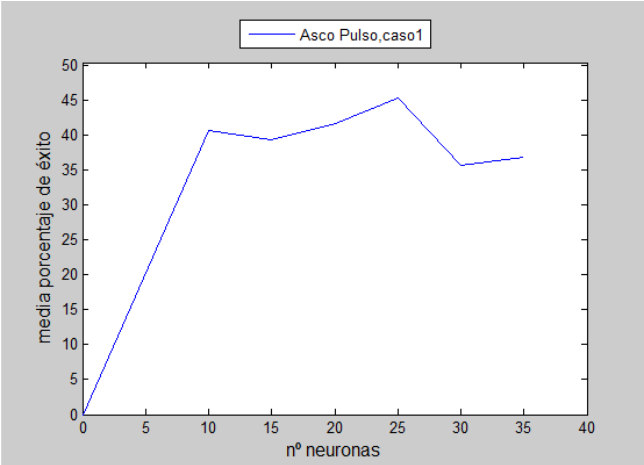


Figura 8. 9 Pulso asco caso 1

Nºneuronas	%éxito	%éxito	%éxito	%éxito	%éxito	Media
10	55.7	48.6	64.3	60	58.6	57.4
15	61.4	60	58.6	52.9	58.6	58.3
20	50	57.1	65.7	55.7	52.9	56.3
25	50	55.7	54.3	55.7	57.1	54.6
30	58.6	52.9	58.6	57.1	57.1	56.8
35	54.3	58.6	51.4	47.1	54.3	53.1

Tabla 8. 5 Pulso asco caso 2

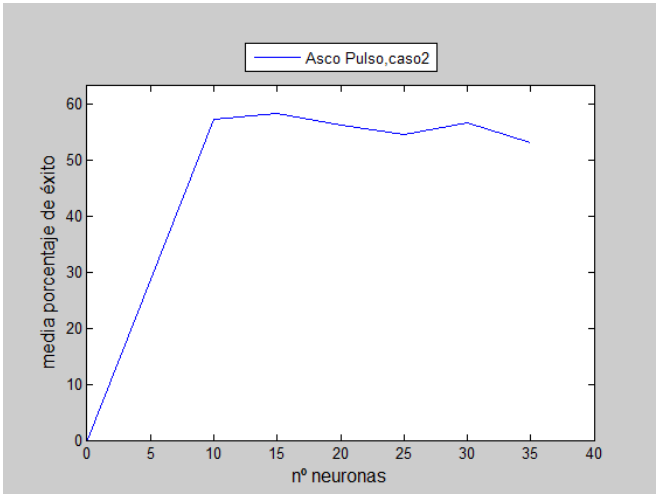


Figura 8. 10Pulso asco caso 2

Nºneuronas	%éxito	%éxito	%éxito	%éxito	%éxito	Media
10	52.9	32.9	32.9	34.3	27.1	36
15	34.3	38.6	28.6	31.4	31.4	32.9
20	34.3	41.4	27.1	32.9	34.3	34
25	35.7	25.7	32.9	34.3	30	31.7
30	27.1	35.7	30	31.4	38.6	32.6
35	50	34.3	31.4	27.1	34.3	35.4

Tabla 8. 6Pulso asco caso 3

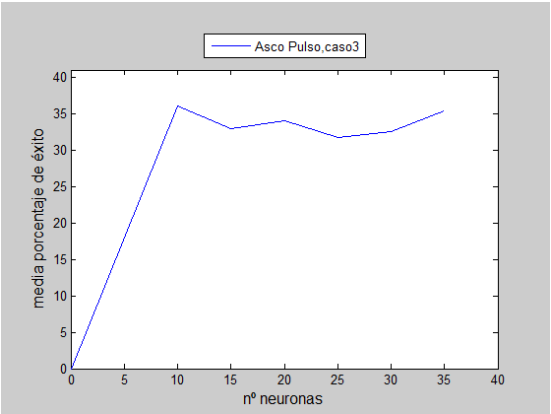


Figura 8. 11 Pulso asco caso 3

Se observa ahora, que separando el pulso para el caso 2, la red acierta más que confundirse con el reconocimiento de estados. Acercándose con 15 neuronas al 60% de acierto. Por lo tanto, como ya se intuía en el análisis gráfico, el pulso es una señal interesante para los propósitos de este TFG. Se observa a continuación el mismo experimento pero seleccionando la variable saturación de oxígeno en sangre.

Nºneuronas	%éxito	%éxito	%éxito	%éxito	%éxito	Media
10	38.6	35.7	44.3	51.4	37.1	41.4
15	40	34.3	44.3	44.3	45.7	41.7
20	32.9	40	47.1	47.1	28.6	45.1
25	34.3	45.7	44.3	42.9	30	39.4
30	58.6	42.9	45.7	38.6	32.9	43.7
35	35.7	41.4	50	45.7	44.3	43.2

Tabla 8. 7 Oxígeno asco caso 1

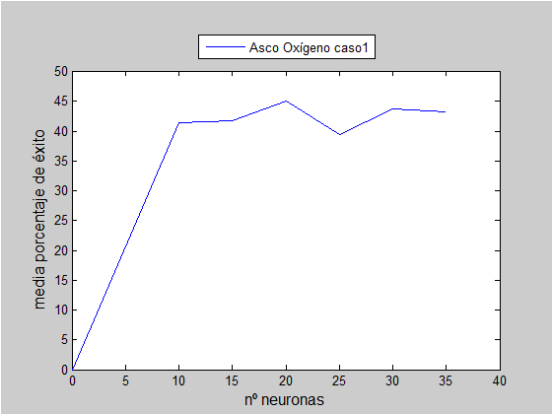


Figura 8. 12 Oxígeno asco caso 1

Nºneuronas	%éxito	%éxito	%éxito	%éxito	%éxito	Media
10	65.2	57.1	51.4	58.6	58.6	58.1
15	54.3	60	57.1	62.9	61.4	59.1
20	50	52.9	64.3	51.4	52.9	54.3
25	51.4	60	55.7	51.4	62.9	56.3
30	57.1	60	54.3	57.1	40	53.7
35	51.4	55.7	50	52.9	50	52

Tabla 8. 8 Oxígeno asco caso 2

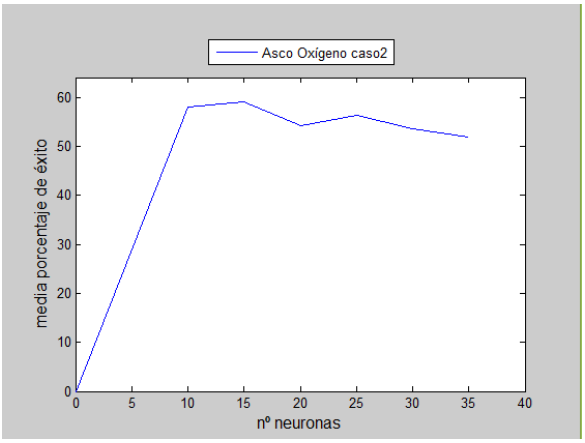


Figura 8. 13 Oxígeno asco caso 2

Nºneuronas	%éxito	%éxito	%éxito	%éxito	%éxito	Media
10	44.3	51.4	47.1	45.7	51.4	48
15	47.1	38.6	52.9	50	50	47.7
20	44.3	41.4	42.9	45.7	37.1	42.3
25	48.6	40	50	42.9	38.6	44
30	44.3	47.1	41.4	44.3	45.7	44.6
35	50	47.1	55.7	45.7	41.4	48

Tabla 8. 9 Oxígeno asco caso 3

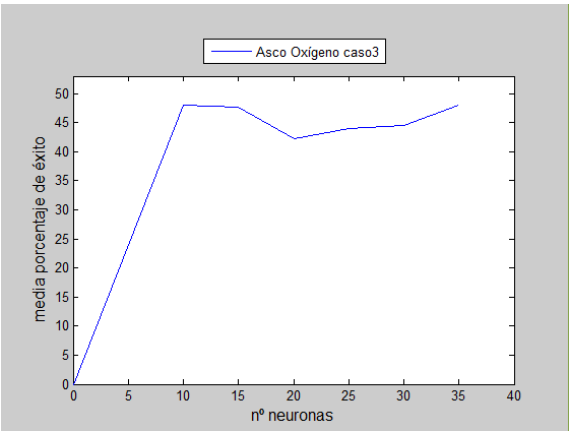


Figura 8. 14 Oxígeno asco caso 3

Nuevamente, volvemos a tener mejores valores que con todas las variables juntas, tan positivos como en el caso de la variable pulso. Vuelve apoyar la red neuronal lo que se observó en las gráficas de oxígeno. Testeemos ahora la temperatura.

Nºneuronas	%éxito	%éxito	%éxito	%éxito	%éxito	Media
10	30	25.7	22.9	20	21.4	24
15	22.9	21.4	22.9	20	24.3	22.3
20	25.7	27.1	25.7	21.4	27.1	25.4
25	17.1	17.1	14.3	15.7	24.3	17.7
30	20	18.6	32.9	21.4	28.6	24.3
35	17.1	22.9	30	11.4	34.3	23.1

Tabla 8. 10 Temperatura asco caso 1

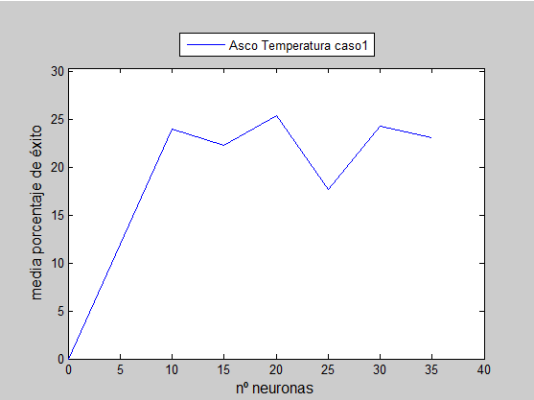


Figura 8. 15 Temperatura asco caso 1

Nºneuronas	%éxito	%éxito	%éxito	%éxito	%éxito	Media
10	35.7	41.4	38.6	32.9	35.7	36.9
15	32.9	25.7	30	37.1	28.6	30.9
20	24.3	22.9	30	37.1	28.6	30.9
25	40	40	31.4	27.1	25.7	34.8
30	34.3	38.6	42.9	35.7	40	38.3
35	32.9	30	27.1	28.6	40	31.7

Tabla 8. 11 Temperatura asco caso 2

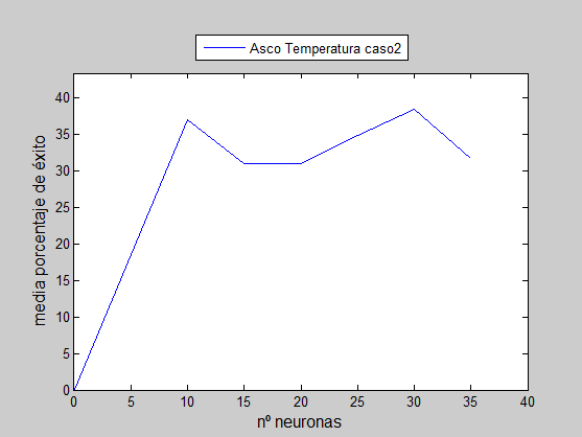


Figura 8. 16 Temperatura asco caso 2

Nºneuronas	%éxito	%éxito	%éxito	%éxito	%éxito	Media
10	30	21.4	28.6	18.6	24.3	24.6
15	21.4	24.3	18.6	20	27.1	22.3
20	27.1	10	27.1	25.7	24.3	22.8
25	24.3	25.7	27.1	25.7	25.7	25.7
30	24.3	229	20	15.7	22.9	21.1
35	22.9	22.9	22.9	20	25.7	22.9

Tabla 8. 12 Temperatura asco caso 3

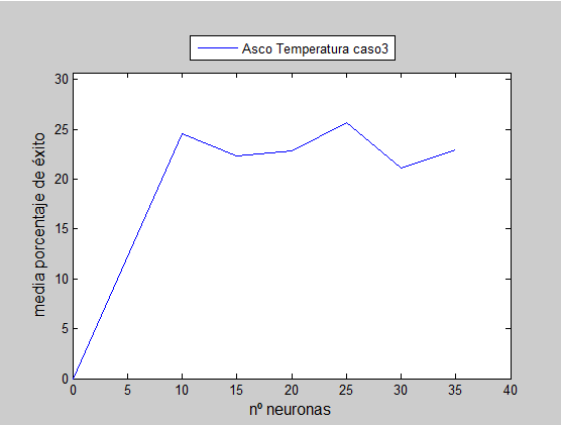


Figura 8. 17 Temperatura asco caso 3

La probabilidad de éxito con la temperatura baja considerablemente. Veamos qué ocurre con la conductancia.

Nºneuronas	%éxito	%éxito	%éxito	%éxito	%éxito	Media
10	21.4	24.3	22.9	24.3	24.3	23.4
15	14.5	20	24.3	22.9	27.1	21.8
20	15.2	20	22.9	11.4	25.7	19
25	17.1	18.6	27.1	21.4	20	20.9
30	21.4	24.3	32.9	15.7	21.4	23.1
35	22.9	14.3	21.4	15.7	20	18.9

Tabla 8. 13 Conductancia asco caso 1

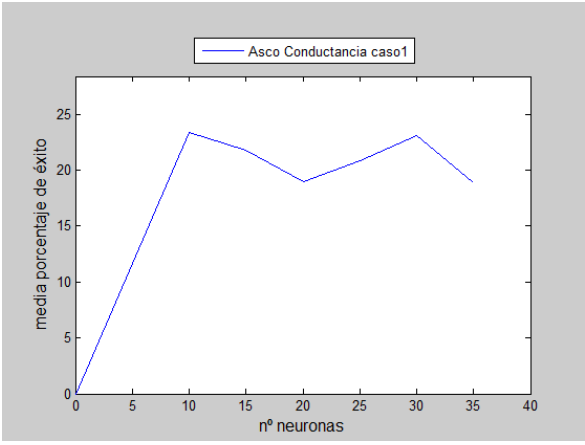


Figura 8. 18 Conductancia asco caso 1

Nºneuronas	%éxito	%éxito	%éxito	%éxito	%éxito	Media
10	35.7	34.3	38.6	34.3	42.9	37.1
15	34.3	37.1	18.6	25.7	32.9	29.7
20	37.1	32.9	25.7	32.9	37.1	33.1
25	35.7	30	32.7	28.6	28.6	31.7
30	30	38.6	32.9	34.3	35.7	32.3
35	31.4	38.6	24.3	35.7	32.9	32.6

Tabla 8. 14 Conductancia asco caso 2

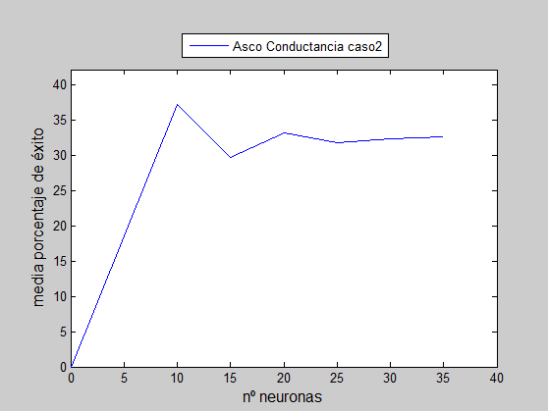


Figura 8. 19 Conductancia asco caso 2

Nºneuronas	%éxito	%éxito	%éxito	%éxito	%éxito	Media
10	24.3	20	17.1	11.4	21.4	18.8
15	20	25.7	18.6	15.7	24.3	20.9
20	20	17.1	21.4	18.6	21.4	19.7
25	15.7	25.2	15.7	24.3	21.4	20.5
30	22.9	18.6	21.4	21.4	15.7	20
35	24.3	24.3	8.6	22.9	17.1	19.9

Tabla 8. 15 Conductancia asco caso 3

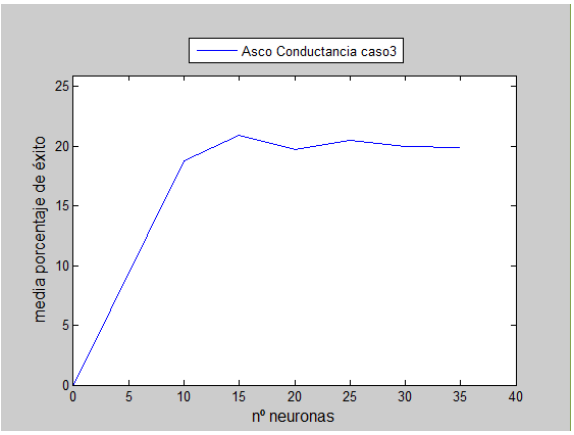


Figura 8. 20 Conductancia asco caso 3

Como en la temperatura, vuelve a bajar el porcentaje de éxito. Queda comprobar el flujo de aire.

Nºneuronas	%éxito	%éxito	%éxito	%éxito	%éxito	Media
10	21.4	14.3	28.6	25.7	20	22
15	21.4	20	20	18.6	27.1	21.4
20	21.4	28.6	28.6	12.9	21.4	22.6
25	27.1	17.1	22.9	18.6	24.3	22
30	21.4	20	24.3	22.9	27.1	23.1
35	28.6	35.7	27.1	35.7	27.1	30.9

Tabla 8. 16 Flujo de aire asco caso 1

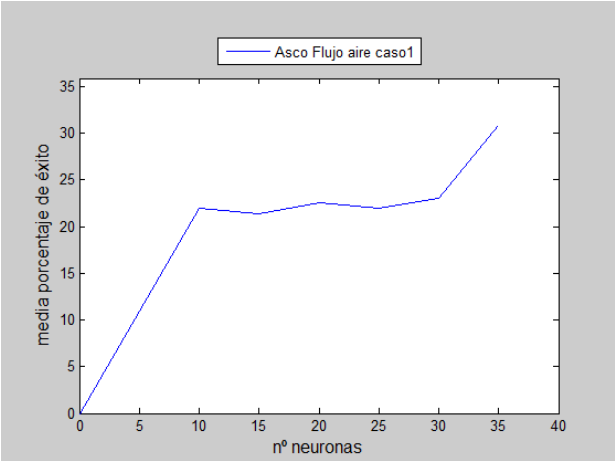


Figura 8. 21 Flujo de aire asco caso 1

Nºneuronas	%éxito	%éxito	%éxito	%éxito	%éxito	Media
10	40	27.1	34.3	35.7	41.4	35.7
15	34.3	25.7	40	28.6	37.1	33.1
20	34.3	24.3	34.3	24.3	31.4	29.7
25	25.7	32.9	22.9	34.3	34.3	30
30	25.7	27.1	35.7	34.3	30	30.6
35	28.6	31.4	34.3	28.6	24.3	29.4

Tabla 8. 17 Flujo de aire asco caso 2

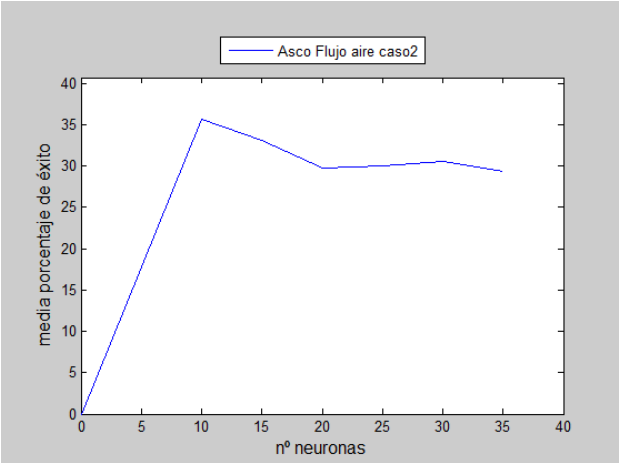


Figura 8. 22 Flujo de aire asco caso 2

Nºneuronas	%éxito	%éxito	%éxito	%éxito	%éxito	Media
10	31.4	24.3	31.4	25.7	28.6	28.8
15	20	20	24.3	21.4	25.7	22.3
20	22.9	24.3	28.6	24.3	27.1	25.4
25	21.4	17.1	14.3	25.7	22.9	20.3
30	10	17.1	20	20	21.4	17.7
35	21.4	20	22.9	15.7	21.4	20.3

Tabla 8. 18 Flujo de aire asco caso 3

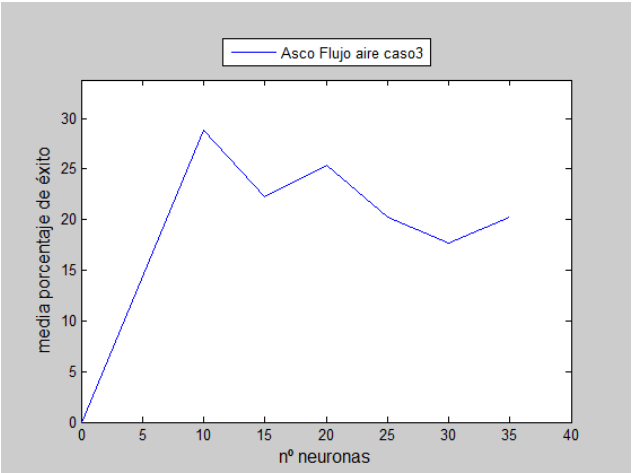


Figura 8. 23 Flujo de aire asco caso 3

Definitivamente, el pulso y el oxígeno, son las variables fisiológicas que más se prestan para este TFG. Esto queda patente en el siguiente resumen.

Mejores señales por medias				
1	2	3	4	5
Pulso	**Oxígeno**	**Temperatura**	Conductancia	Flujo aire
56.08	**55.6**	**33.9**	32.7	31.4
caso2	**caso2**	**caso2**	caso2	caso2

Mejores señales por máximos				
1	2	3	4	5
Oxígeno	**Pulso**	**Temperatura**	Conductancia	Flujo aire
59.1	**58.3**	**38.3**	37.1	33.1
caso2	**caso2**	**caso2**	caso2	caso2

Figura 8. 24 Cuadro resumen experimento por señal

8.1.3 Experimento 2 mejores señales

Las conclusiones del experimento anterior invitan a ver qué resultados arroja la red si le introducimos las variables fisiológicas de pulso y oxígeno fusionados. Se logra programando la función *experimento2Senyal()*, la cual es muy similar a la anterior, pero lógicamente, seleccionando dos variables fisiológicas por parámetro de entrada. Debido a que en los experimentos anteriores, el caso 3 siempre ha sido el más pobre, se descarta para los siguientes experimentos de vídeos tipo asco.

Nºneuronas	%éxito	%éxito	%éxito	%éxito	%éxito	**Media**
10	47.1	47.1	47.1	28.6	44.3	44.8
15	40	32.9	31.4	51.4	31.4	37.4
20	44.3	45.7	50	42.9	31.4	42.9
25	44.3	44.3	40	41.4	45.7	43.1
30	38.6	34.3	44.3	41.4	41.4	40
35	40	41.4	42.9	41.4	34.3	41

Tabla 8. 19 Pulso y oxígeno asco caso 1

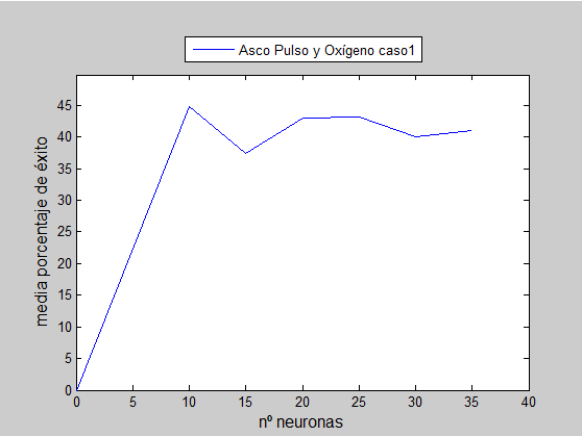


Figura 8. 25 Pulso y oxígeno asco caso 1

Nºneuronas	%éxito	%éxito	%éxito	%éxito	%éxito	Media
10	52.9	52.9	54.3	55.7	55.7	54.3
15	50	54.3	55.7	54.3	57.1	54.3
20	54.3	54.3	50	58.6	54.3	54.3
25	50	45.7	58.6	54.3	61.4	54.5
30	65.7	57.1	55.7	51.4	54.3	56.8
35	54.3	57.1	45.7	55.7	50	52.6

Tabla 8. 20 Pulso y oxígeno asco caso 2

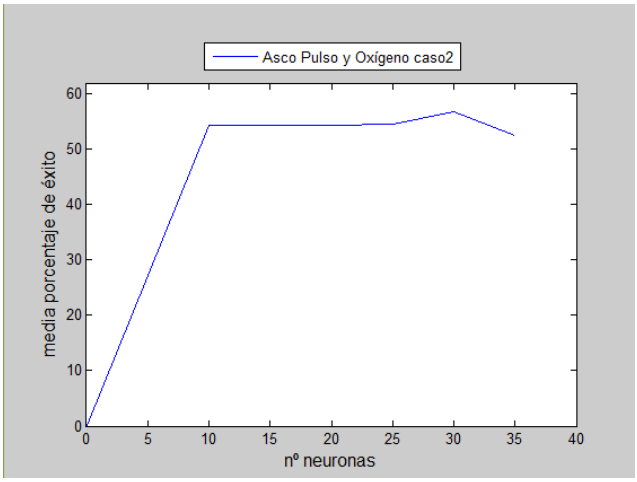


Figura 8. 26 Pulso y oxígeno asco caso 2

Los resultados son parecidos, a cuando se tiene el pulso y el oxígeno por separado.

8.1.4 Experimento 3 mejores señales

Veamos ahora qué ocurre si se le añade también la tercera mejor señal, es decir, la temperatura. Ya sólo con el caso 2.

Nºneuronas	%éxito	%éxito	%éxito	%éxito	%éxito	Media
10	47.1	51.4	50	50	54.3	50.6
15	47.1	54.3	51.4	44.3	54.3	50.3
20	47.1	48.6	42.9	48.6	47.1	46.9
25	42.9	58.6	30	41.4	57.1	45.8
30	44.3	47.1	47.1	55.7	45.7	48
35	37.1	51.4	57.1	37.1	48.6	46.3

Tabla 8. 21 Pulso, oxígeno y temperatura caso 2

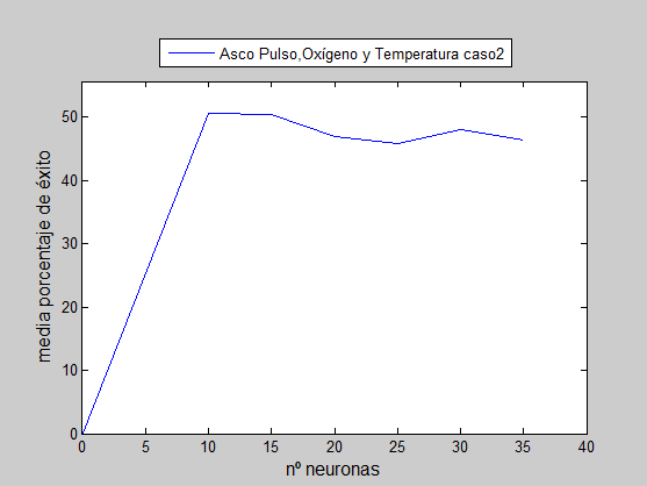


Figura 8. 27 Pulso, oxígeno y temperatura caso 2

Como cabía de esperar, al añadir una señal considerablemente peor que las dos anteriores, el conjunto empeora.

8.1.5 Experimento por parámetro

Una vez se sabe cuáles son las mejores variables fisiológicas, se podría investigar qué parámetros estadísticos funcionan mejor para el reconocimiento por parte de la red de los diferentes estados. Siguiente experimento, *experimentoSolo1Parametro()*, el cuál selecciona de los parámetros las filas correspondientes a las medias realizadas.

Nºneuronas	%éxito	%éxito	%éxito	%éxito	%éxito	Media
10	31.4	34.3	38.6	28.6	28.6	32.3
15	32.9	27.1	31.4	32.9	28.6	32.6
20	38.6	31.4	35.7	28.6	30	32.9
25	34.3	32.9	28.6	30	31.4	31.4
30	30	34.3	35.7	40	30	34
35	28.6	31.4	31.4	32.9	25.7	28.6

Tabla 8. 22 Media caso 1

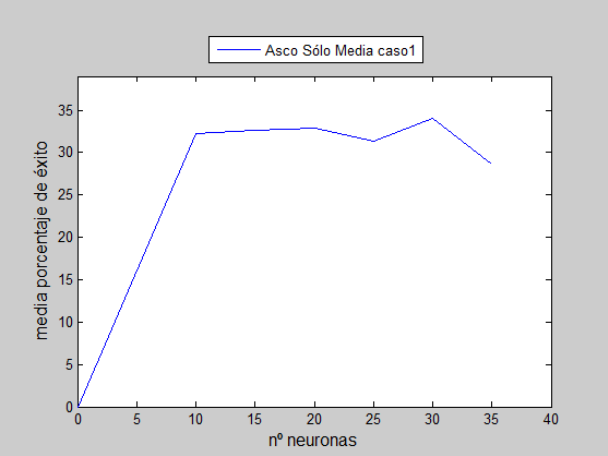


Figura 8. 28 Media caso 1

Nºneuronas	%éxito	%éxito	%éxito	%éxito	%éxito	Media
10	34.3	30	35.7	37.1	31.4	33.7
15	25.7	31.4	34.3	37.1	31.4	32
20	32.9	30	31.4	34.3	31.4	32
25	32.9	28.6	31.4	31.4	31.4	31.1
30	32.9	32.9	34.3	32.9	37.1	34
35	25.7	25.7	27.1	31.4	32.9	26.6

Tabla 8. 23 Media caso 2

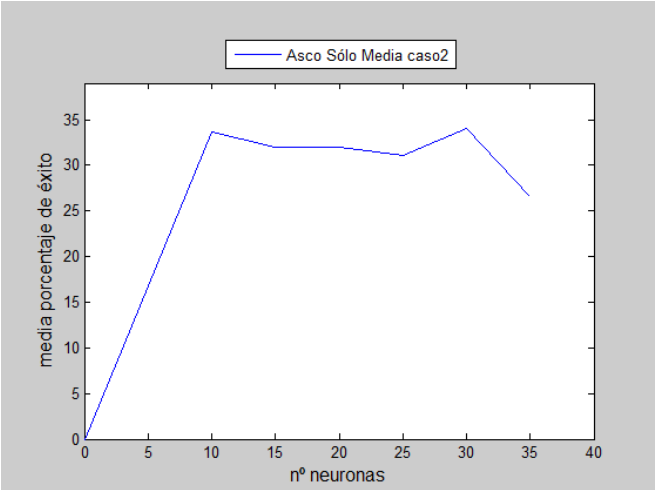


Figura 8. 29 Media caso 2

Resultados considerablemente pobres. Veamos la varianza.

Nºneuronas	%éxito	%éxito	%éxito	%éxito	%éxito	Media
10	42.9	40	44.3	42.9	41.4	42.3
15	28.6	40	50	41.4	42.9	40.6
20	42.9	47.1	35.7	48.6	45.7	44
25	38.6	41.4	45.7	41.4	35.7	40.6
30	37.1	41.4	34.3	42.9	42.9	39.7
35	32.9	35.7	35.7	42.9	35.7	36.6

Tabla 8. 24 Varianza caso 1

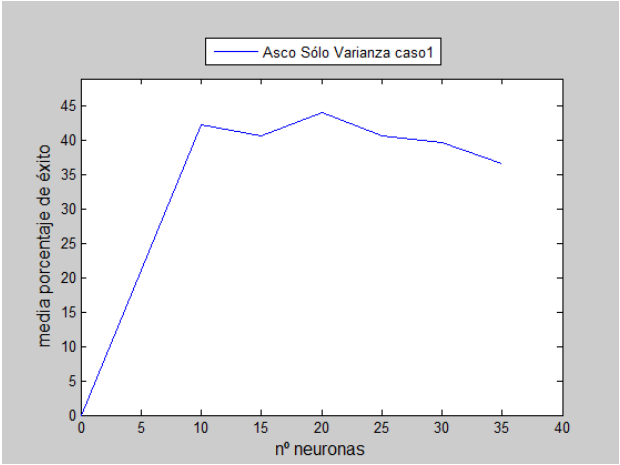


Figura 8. 30 Varianza caso 1

Nºneuronas	%éxito	%éxito	%éxito	%éxito	%éxito	Media
10	41.4	42.9	41.4	32.9	31.4	38
15	35.7	40	41.4	38.6	32.9	37.7
20	40	40	41.4	47.1	34.3	40.6
25	45.7	34.3	37.1	42.9	40	40
30	34.3	32.9	42.9	38.6	41.4	38
35	38.6	34.3	42.9	41.4	34.3	38.3

Tabla 8. 25 Varianza caso 2

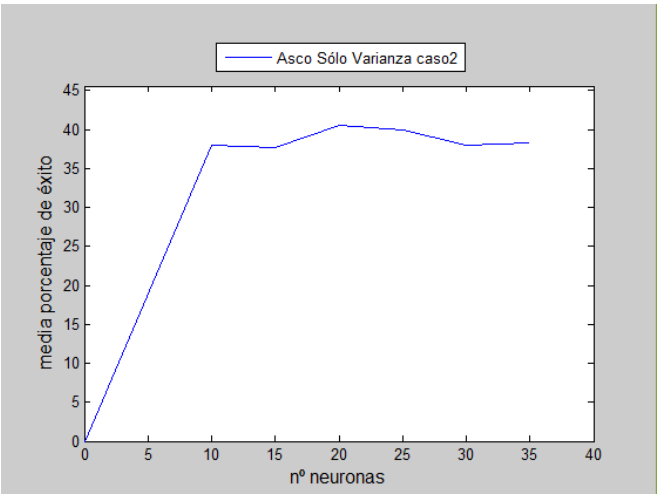


Figura 8. 31 Varianza caso 2

La varianza mejora respecto a la media. Turno de la curtosis.

Nºneuronas	%éxito	%éxito	%éxito	%éxito	%éxito	Media
10	38.6	38.6	34.3	18.6	37.1	33.4
15	30	32.9	31.4	31.4	40	33.1
20	34.3	24.3	32.9	42.9	40	34.9
25	30	34.3	30	35.7	32.9	32.6
30	32.9	32.9	34.3	25.7	37.1	32.6
35	30	37.1	31.4	35.7	31.4	33.1

Tabla 8. 26 Curtosis caso 1

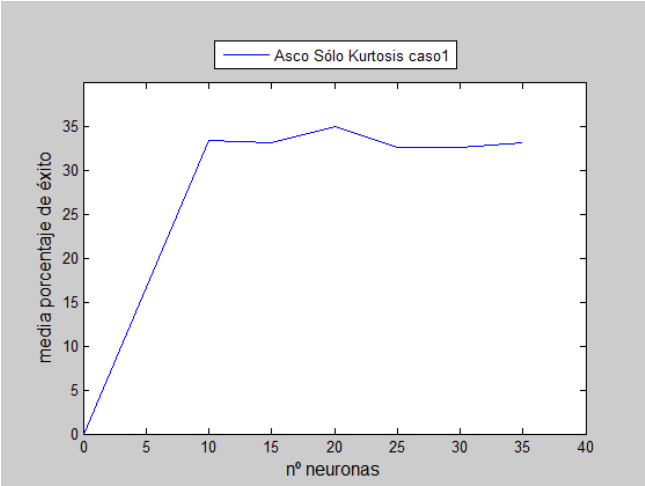


Figura 8. 32 Curtosis caso 1

Nºneuronas	%éxito	%éxito	%éxito	%éxito	%éxito	Media
10	40	37.1	37.1	31.4	27.1	34.5
15	34.3	35.7	38.6	34.3	24.3	33.4
20	24.3	31.4	37.1	32.9	28.6	30.9
25	27.1	32.9	28.6	24.3	34.3	29.4
30	24.3	40	27.1	32.9	41.4	33.1
35	34.3	34.3	34.3	30	34.3	33.4

Tabla 8. 27 Curtosis caso 2

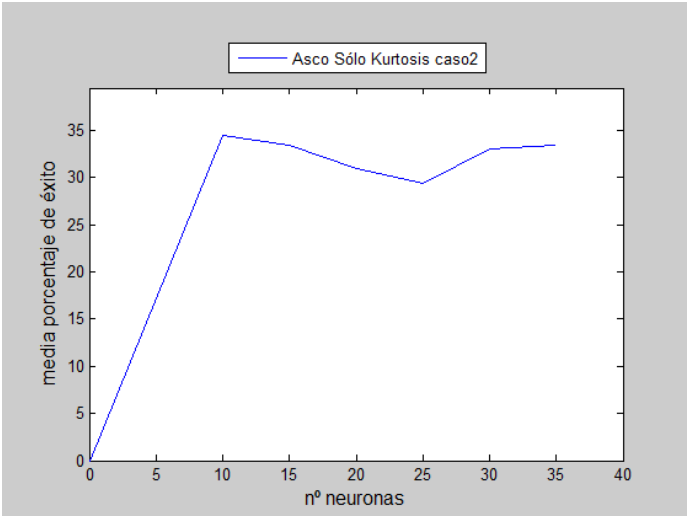


Figura 8. 33 Curtosis caso 2

Se vuelve a tener valores similares a la media. A continuación, la entropía.

Nºneuronas	%éxito	%éxito	%éxito	%éxito	%éxito	Media
10	34.3	38.6	44.3	40	37.1	38.9
15	41.4	44.3	38.6	38.6	44.3	41.4
20	44.3	45.7	40	41.4	40	42.3
25	44.3	40	35.7	41.4	44.3	41.1
30	40	27.1	42.9	32.9	37.1	36
35	45.7	37.1	42.9	40	41.4	41.4

Tabla 8. 28 Entropía asco caso 1

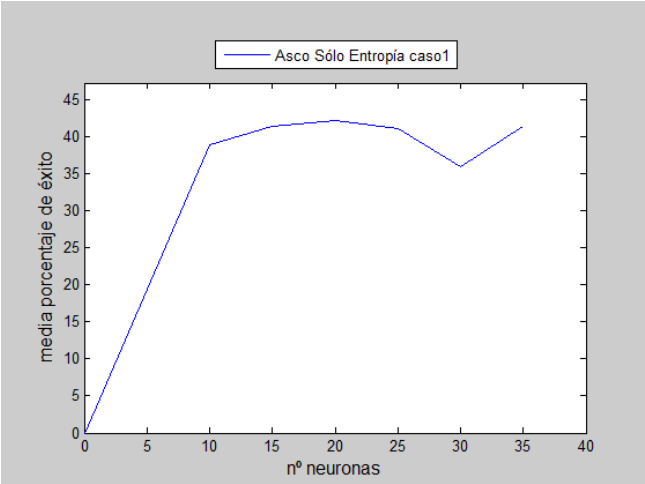


Figura 8. 34 Entropía asco caso 1

Nºneuronas	%éxito	%éxito	%éxito	%éxito	%éxito	Media
10	44.3	45.7	41.4	40	44.3	43.1
15	40	38.6	34.3	30	41.4	36.9
20	40	37.1	34.3	25.7	38.6	35.1
25	37.1	34.3	47.1	37.1	32.9	37.7
30	44.3	40	35.7	37.1	34.3	38.3
35	40	42.9	38.6	35.7	40	39.4

Tabla 8. 29 Entropía asco caso 2

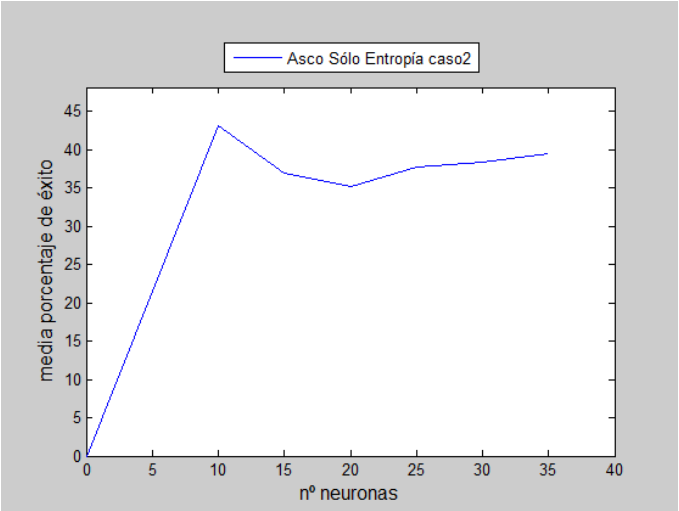


Figura 8. 35 Entropía asco caso 2

Como en la varianza, la entropía vuelve a mejorar los resultados. Como no se ha dado aún que el caso 1 sea mejor que el 2, será este el mostrado. El siguiente es la entropía local.

Nºneuronas	%éxito	%éxito	%éxito	%éxito	%éxito	Media
10	20	20	20	20	20	20
15	20	20	20	20	20	20
20	20	20	20	20	20	20
25	20	20	20	20	20	20
30	20	20	20	20	20	20
35	20	20	20	20	20	20

Tabla 8. 30 Entropía local asco caso 2

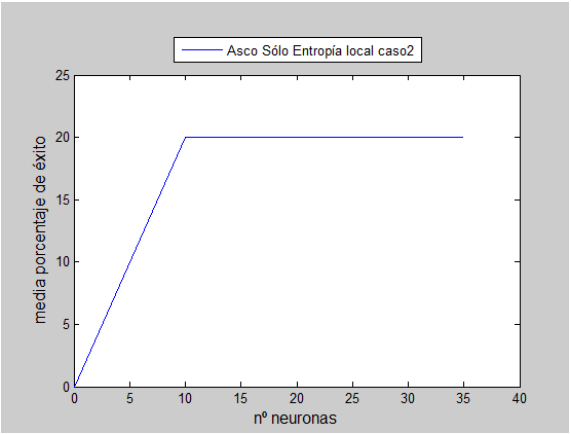


Figura 8. 36 Entropía local asco caso 2

Muy pobre. Veamos a ver qué tal la entropía Shannon.

Nºneuronas	%éxito	%éxito	%éxito	%éxito	%éxito	Media
10	42.9	42.9	31.4	41.4	37.1	39.1
15	37.1	27.1	32.9	45.7	38.6	36.3
20	37.1	41.4	28.6	37.1	34.3	35.7
25	38.6	47.1	38.6	38.6	24.3	37.4
30	42.9	30	40	42.9	37.1	38.6
35	41.4	38.6	50	35.7	38.6	40.9

Tabla 8. 31 Entropía Shannon asco caso 2

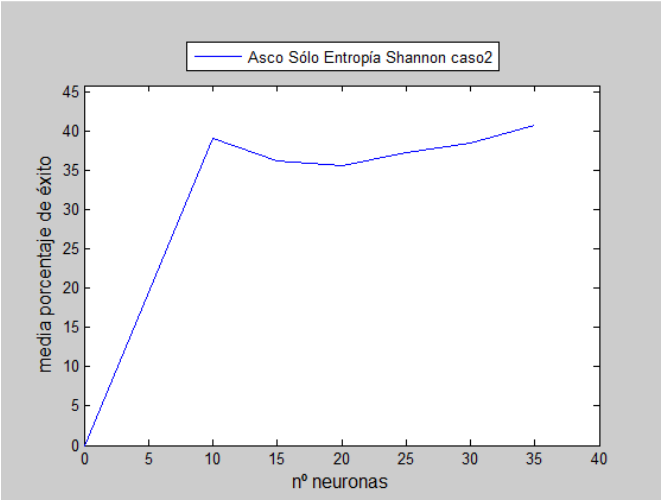


Figura 8. 37 Entropía Shannon asco caso 2

Considerablemente mejor. Por último queda la asimetría o también conocida como sesgo u oblicuidad.

Nºneuronas	%éxito	%éxito	%éxito	%éxito	%éxito	Media
10	35.7	32.9	35.7	28.6	50	36.6
15	28.6	32.9	37.1	30	35.7	32.9
20	35.7	38.6	37.1	40	42.9	38.9
25	37.1	40	32.9	37.1	28.6	35.1
30	35.7	31.4	34.3	32.9	24.3	31.7
35	41.4	32.9	28.6	28.6	31.4	32.6

Tabla 8. 32 Asimetría asco caso 2

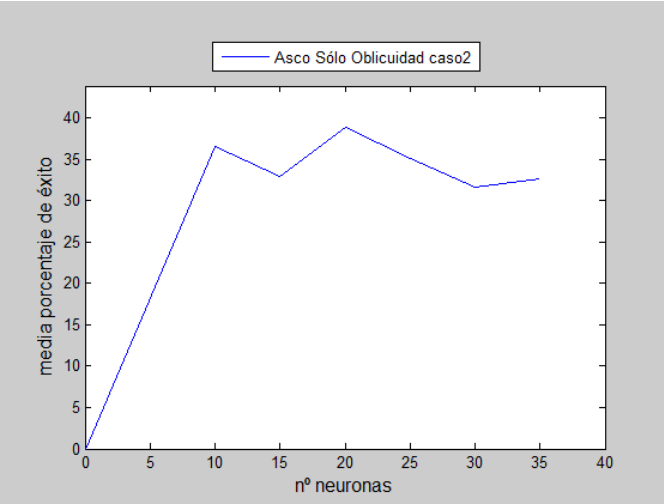


Figura 8. 38 Asimetría asco caso 2

A modo resumen:

Mejores parámetros por medias						
1	2	3	4	5	6	7
Varianza	Entropía	Entropía Shannon	Oblicuidad	Media	Kurtosis	Entropía local
38.8	38.4	38	34.1	31.6	32.5	20
caso2	caso2	caso2	caso2	caso2	caso2	caso2

Mejores parámetros por máximos						
1	2	3	4	5	6	7
Varianza	Entropía	Entropía Shannon	Oblicuidad	Kurtosis	Media	Entropía local
44	43.1	40.9	38.9	34.5	34	20
caso1	caso2	caso1	caso2	caso2	caso2	caso2

Tabla 8. 33 Resumen parámetros asco

8.1.1 Experimento señales-parámetros

Una vez en este punto, dónde se conocen las mejores opciones en cuanto a señal y parámetros, se realizan una serie de combinaciones con ellos, para ver cuál es la mejor posibilidad. Para ellos se usan las funciones *experimento2Senyal2Parametro()* y *experimento2Senyal3Parametro().*

Nºneuronas	%éxito	%éxito	%éxito	%éxito	%éxito	Media
10	52.9	60	62.9	52.9	60	57.7
15	62.9	60	60	51.4	57.1	58.2
20	61.4	64.3	57.1	57.1	55.7	59.1
25	58.6	57.1	52.9	64.3	62.9	59.1
30	55.7	60	64.3	60	61.4	60.3
35	61.4	57.1	60	57.1	55.7	58.2

Tabla 8. 34 Pulso y oxígeno; varianza, entropía y entropía Shannon caso 2

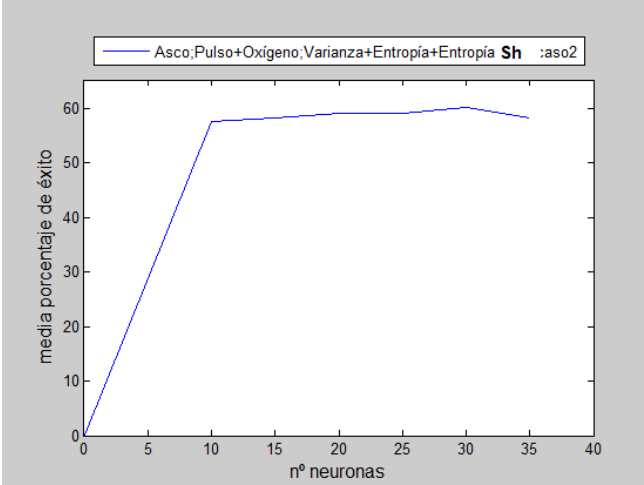


Figura 8. 39 Pulso y oxígeno; varianza, entropía y entropía Shannon caso 2

Nºneuronas	%éxito	%éxito	%éxito	%éxito	%éxito	Media
10	60	61.4	55.7	61.4	58.6	59.4
15	61.4	61.4	55.7	64.3	57.1	60
20	55.7	50	60	62.9	51.4	56
25	45.7	51.4	55.7	55.7	54.3	52.5
30	60	61.4	64.3	57.1	55.7	59.7
35	61.4	54.3	58.6	52.9	58.6	57.2

Tabla 8. 35 Pulso y oxígeno; entropía y entropía Shannon caso 2

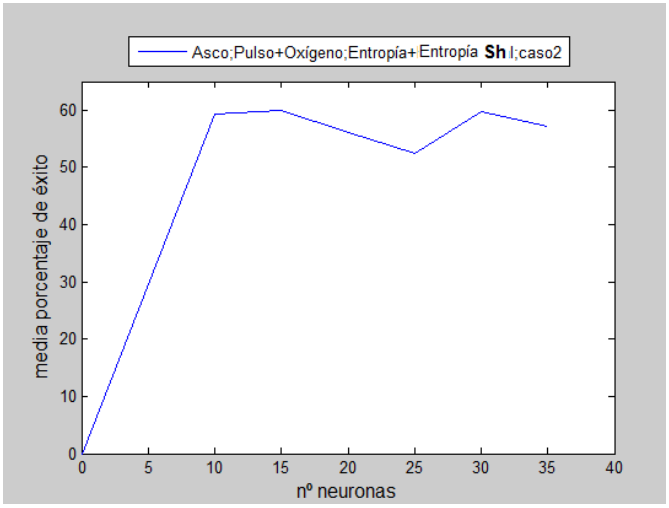


Figura 8. 40 Pulso y oxígeno; entropía y entropía Shannon caso 2

Nºneuronas	%éxito	%éxito	%éxito	%éxito	%éxito	Media
10	67.1	61.4	60	61.4	58.6	61.7
15	60	52.9	58.6	57.1	51.4	56
20	58.6	58.6	52.9	60	52.9	56.6
25	55.7	58.6	61.4	57.1	58.6	58.3
30	55.7	58.6	54.3	47.1	64.3	56
35	57.1	58.6	50	52.9	62.9	56.3

Tabla 8. 36 Pulso y oxígeno; varianza y entropía caso 2

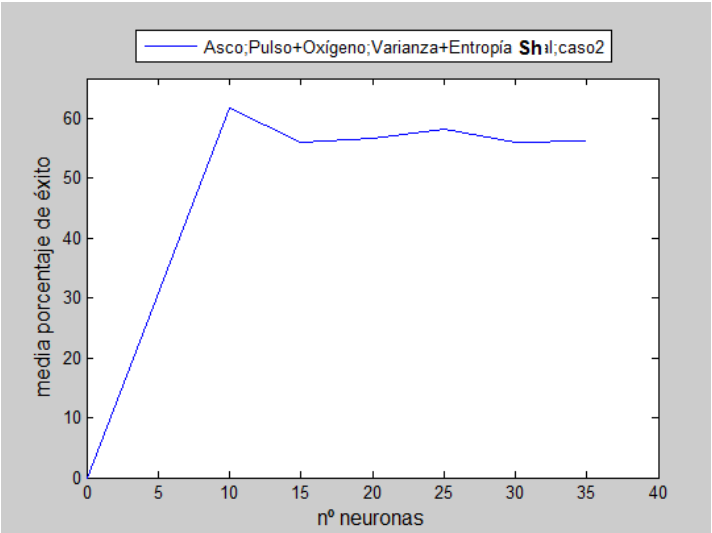


Figura 8. 41 Pulso y oxígeno; varianza y entropía caso 2

Como se ve, agrupando las mejores opciones el resultado mejora bastante.

8.2 Experimentos vídeo tipo miedo

A continuación se procede a la misma secuencia de experimentos pero para los vídeos tipo miedo. Aunque en su momento se hicieron todas las simulaciones, lo que se expone aquí, son los mejores resultados.

Mejores señales por medias				
1	2	3	4	5
Pulso	**Oxígeno**	Temperatura	Conductancia	Flujo aire
51.3	**51**	33.7	30	29.2
caso1	**caso1**	caso1	caso1	caso1

Mejores parámetros por medias						
1	2	3	4	5	6	7
Oblicuidad	**Varianza**	**Entropía**	Kurtosis	Media	Entropía Shannon	Entropía local
35.3	**34.3**	**33.7**	33.1	29	28.8	20
caso1	**caso1**	**caso1**	caso1	caso1	caso1	caso1

Tabla 8. 37 Resumen parámetros y señales miedo

Resultando las 2 mejores combinaciones, las de pulso y oxígeno en el caso 1, y pulso más oxígeno sin la entropía local.

Nºneuronas	%éxito	%éxito	%éxito	%éxito	%éxito	Media
10	58.5	56.9	52.3	53.8	53.8	55
15	53.8	53.8	53.8	53.8	58.5	54.7
20	56.9	50.8	61.5	49.2	50.8	53.9
25	52.3	50.8	53.8	50.8	50.8	51.7
30	50.8	53.8	50.8	53.8	49.2	51.7
35	61.5	55.4	55.4	53.8	55.4	56.3

Tabla 8. 38 Pulso y oxígeno, miedo caso 1

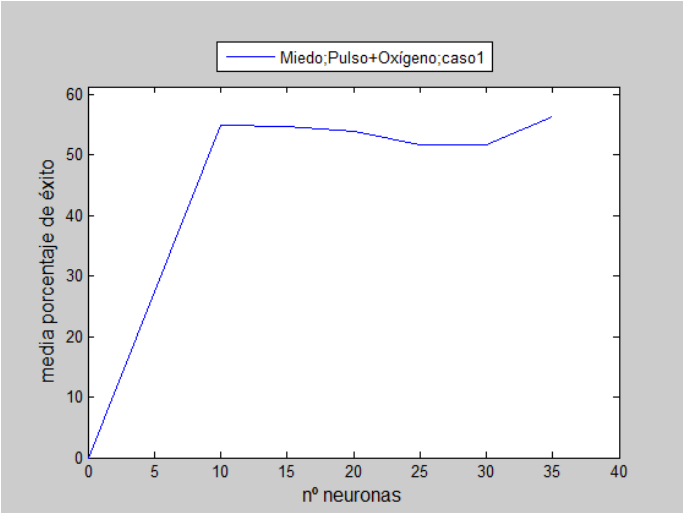


Figura 8. 42 Pulso y oxígeno, miedo caso 1

Nºneuronas	%éxito	%éxito	%éxito	%éxito	%éxito	Media
10	47.7	53.8	49.2	50.8	53.8	51
15	55.4	50.8	55.4	55.4	55.4	54.5
20	52.3	49.2	52.3	53.8	52.3	52
25	60	53.8	56.9	47.7	52.3	54.1
30	53.8	53.4	55.4	44.6	50.8	51.6

Tabla 8. 39 Pulso y oxígeno, sin entropía local, miedo caso 1

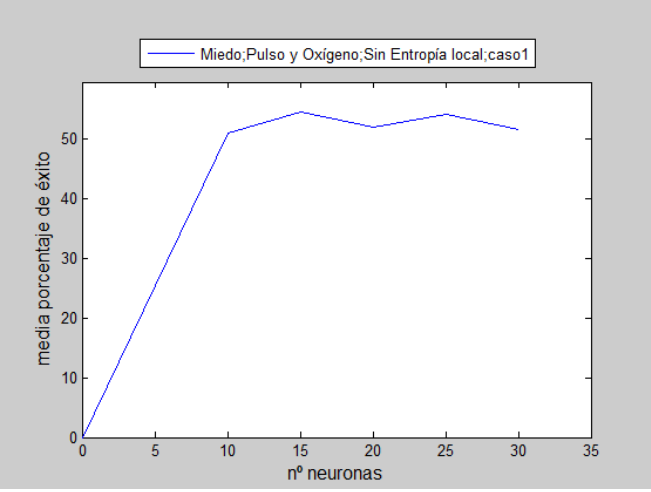


Figura 8. 43 Pulso y oxígeno, sin entropía local, miedo caso 1

En líneas generales, los vídeos tipo asco y miedo tiene porcentajes de éxito similares. Aunque con estas simulaciones se obtienen conclusiones interesantes a nivel de parámetros y señales idóneas, el porcentaje de éxito no es suficiente para decir con certeza que la red detecte claramente cada uno de los estados, con lo que implica cada estado a nivel conceptual.

8.3 Experimentos 2 estados

A pesar de obtener conclusiones bastante interesantes en los experimentos anteriores, se busca aún alguna opción que busque soluciones más rotundas en concepto de cumplir uno de los objetivos planteados inicialmente, que no es otro, que el de detectar una reacción frente a un estímulo. Por ejemplo, se puede dividir ahora las matrices de entrenamiento, a dos estados, uno que recoge el concepto de no emoción (el vídeo no está en reproducción) y el otro el de emoción (reproducción del vídeo). A modo resumen:

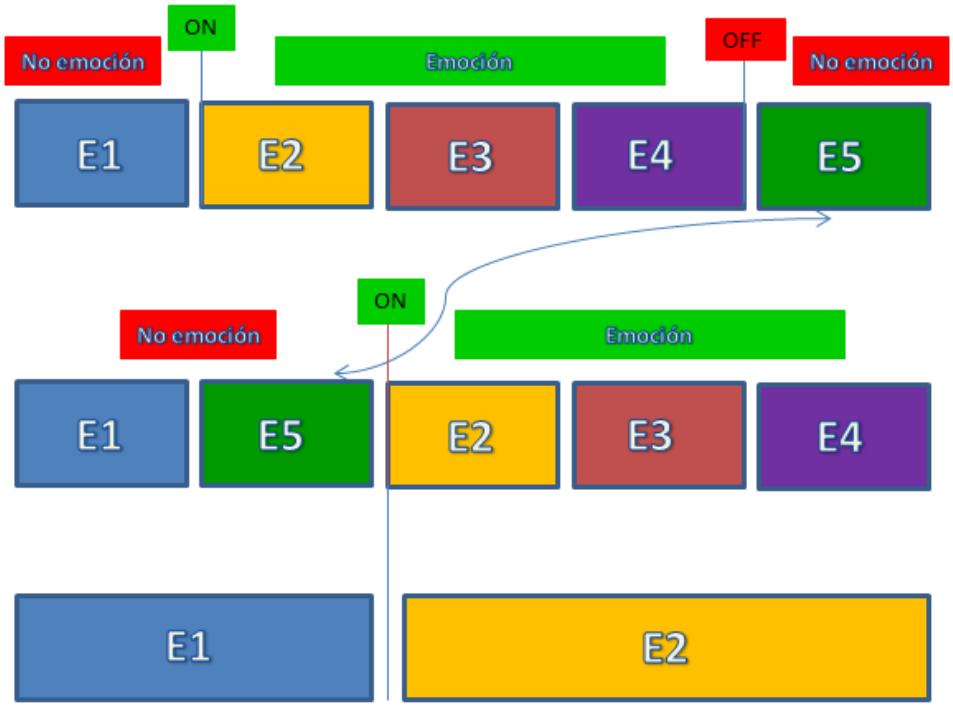


Figura 8. 44 Transformación de estados

Es decir, ahora el estado 5 se coloca acto seguido del 1, y conforman un estado de no emoción. El estado 2 se desplaza hacia delante y se une al 3 y al 4 para formar el estado de emoción. A nivel software, esto se conseguirá en una función nueva llamada *crearTest_Training2estados()*, en la que se hace este desplace de estados de la siguiente manera.

```
for i=1:5
    for j=posicionVideo:posicionVideo+longVideo-1
    parametrosNN_video(:,contador)=PARAMETROS_NN(:,j);
    contador=contador+1;
    end
    if (i==1)
        posicionVideo=posicionVideo+(4*67);

    elseif (i==2)
        posicionVideo=68;
    else
         posicionVideo=posicionVideo+67;
    end
end
```

Código 8. 3 Desplazamiento estado 5

Los experimentos realizados con mejores resultados son los siguientes.

Nºneuronas	10it.	20it.	30it.
10	88.6	90	90
15	88.6	88.6	88.6
20	90	90	90
25	90	90	90
30	90	90	88.6

Tabla 8. 40 Pulso y oxígeno; entropía, entropía. Shannon y varianza; asco caso 3

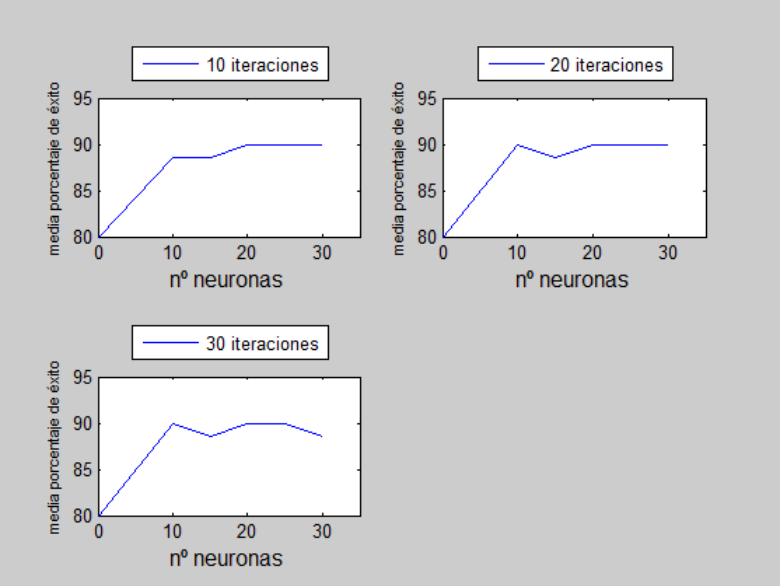


Figura 8. 45 Pulso y oxígeno; entropía, entropía Shannon y varianza; asco caso 3

Nºneuronas	10it.	20it.	30it.
10	93.8	95.4	90.8
15	90.8	92.3	93.8
20	93.8	90.8	90.8
25	92.3	92.3	95.4
30	90.8	92.3	93.8

Tabla 8. 41 Pulso y oxígeno caso 2

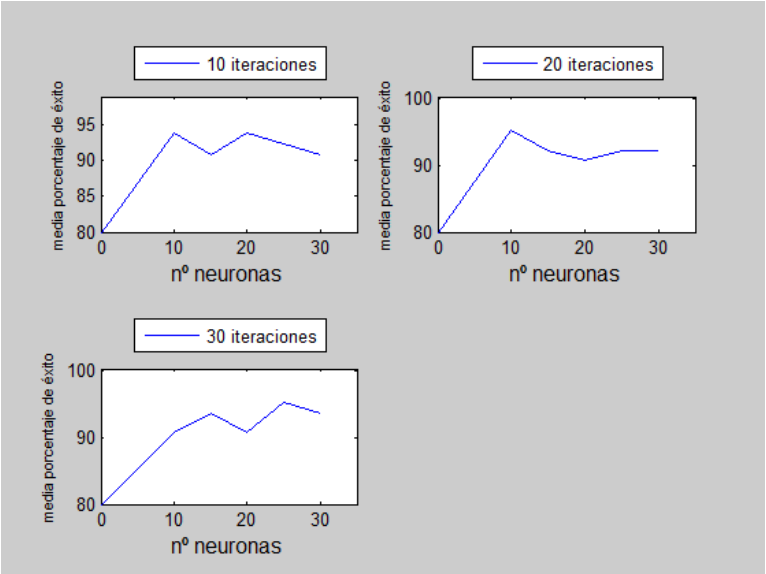


Figura 8. 46 Pulso y oxígeno caso 2

Como vemos, la diferencia emoción-no emoción la detecta muy bien la red. Obteniéndose porcentajes de acierto mucho más elevados.

8.3.1 Experimento mezcla asco-miedo

Podría ser también de interés mezclar las dos emociones de asco y miedo.

Nºneuronas	10it.	20it.
10	90.8	92.3
15	93.8	92.3
20	89.2	90.8
25	89.2	93.8
30	89.2	90.8

Tabla 8. 42 Vídeo carretera y exorcismo con vídeo moco; pulso y oxígeno caso 1

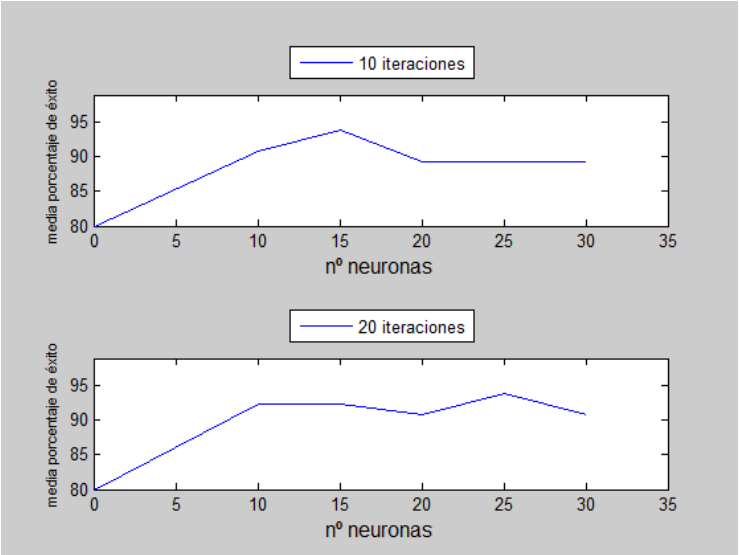


Figura 8. 47 Vídeo carretera y exorcismo con vídeo moco; pulso y oxígeno caso 1

8.3.2 Experimento por número de críticos

En los experimentos anteriores siempre nos hemos centrado en el tipo de estimulación, parámetro utilizado o señal fisiológica capturada, pero también se puede considerar de qué manera es el vídeo de intenso. Es decir, si arroja mejores resultados usar un vídeo con pocos pero intensos momentos intensos de susto/asco o en cambio que este sentimiento sea más continuado en el tiempo.

Nºneuronas	10it.	20it.
10	92.9	91.4
15	90	91.4
20	90	92.9
25	90	90
30	88.6	91.4

Tabla 8. 43 Pocos críticos (moco, carretera y espinilla); pulso y oxígeno caso 1

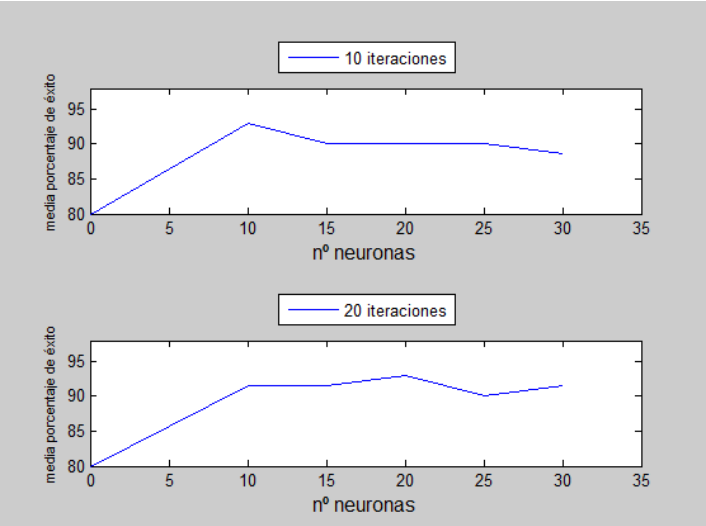


Figura 8. 48 Pocos críticos (moco, carretera y espinilla); pulso y oxígeno caso 1

Nºneuronas	10it.	20it.
10	87.3	85.5
15	85.5	85.5
20	89.1	85.5
25	89.1	87.3
30	85.5	89.1

Tabla 8. 44 Muchos críticos (*jackass*, luces fuera y exorcismo); pulso y oxígeno caso 1

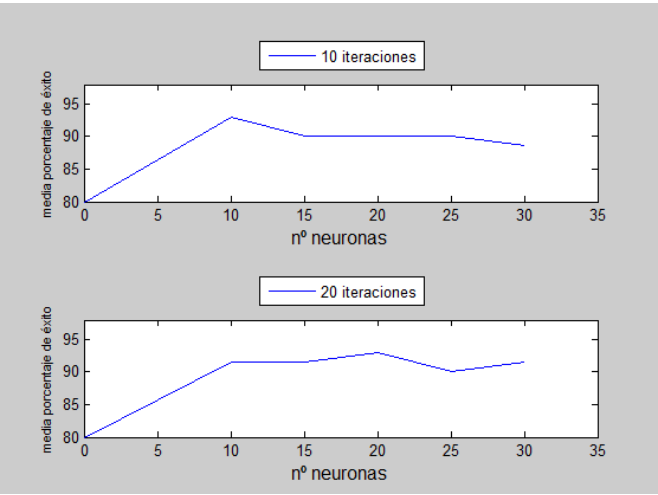


Figura 8. 49 Muchos críticos (*jackass*, luces fuera y exorcismo); pulso y oxígeno caso 1

Parece dar mejores resultados, un vídeo que tenga menos críticos que aquel en el que la sensación de asco/miedo sea más continuada. En el siguiente capítulo se incidirá con más detalle en las conclusiones aquí mencionadas.

Capítulo 9 Conclusiones y líneas futuras

9.1 Conclusiones

En el sentido de la realización de los objetivos propuestos, las conclusiones son bastante positivas. La transferencia de datos desde los sensores mediante Arduino, se realiza de una manera perfectamente sincronizada con la recogida de las tramas en Matlab. A nivel de interfaz de usuario, el resultado también es bastante satisfactorio, ya que, por un lado, la representación dinámica de las variables fisiológicas se lleva acabo de una manera fiable, mientras que por el otro, la ejecución de cada uno de los botones presentes da pie a una monitorización cómoda y automatizada. Sin embargo, como no podía ser de otra manera, la sección de análisis fue la que proporciona, sin lugar a dudas, mayor número de conclusiones. De los experimentos realizados con la red neuronal podemos concluir lo siguiente.

- De las señales fisiológicas elegidas inicialmente para la monitorización del paciente, el pulso y el oxígeno son las más propicias para detectar cambios ante estímulos externos, tanto de tipo asco como de tipo miedo.

- Para los estímulos tipo asco, de toda la parametrización que se le hizo a los datos, los parámetros que mayor porcentaje de éxito dieron son, la varianza, la entropía y la entropía Shannon, mientras que para los tipo miedo, son la asimetría, la varianza y la entropía. En el otro extremo, la entropía local proporciona porcentajes muy pobres tanto para el asco como para el miedo.

- Cuando se fusionan los mejores parámetros y señales en un experimento, tanto el miedo como el asco dan porcentajes de éxito similares. De tal manera, que se pueden mezclar vídeos de miedo y asco en una monitorización que los resultados son igualmente válidos que si no estuvieran mezclados.

-

- Independientemente del tipo de estímulo al que pertenezcan, los vídeos con pocos momentos puntuales, pero intensos, arrojan mayor éxito que aquellos que tienen estímulos más continuados en el tiempo.

- Las conclusiones anteriores están basadas en obtener porcentajes de éxito mayores en comparación a otros experimentos. Pero si nos basamos en fiabilidad del éxito, por términos absolutos, el hecho de separar los datos en dos estados, separándolos conceptualmente en estado de emoción y estado de no emoción, fue el que mayor porcentaje de éxito proporcionó con valores cercanos al 90%.

  Este hecho, da a pensar, que realmente el sistema, pueda llegar a ser útil, a la hora de poder diagnosticar, de manera precoz, algún tipo de enfermedad o afección de naturaleza neurológica, que implique una reacción deficiente o atenuada ante un estímulo externo.

9.2 Líneas futuras

Desde el punto de vista de fiabilidad de los datos obtenidos, se pueden plantear un calibrado de los sensores de temperatura y conductancia. Aunque si es verdad, que estas variables no resultaron ser muy interesantes para los objetivos planteados, sería conveniente realizar un calibrado a estos dos sensores. El de temperatura debido a que daba valores demasiados altos y el de conductancia por el hecho de ser también un sensor de tipo ohmímetro. En el capítulo 2 de este TFG y en la web del fabricante se detalla cómo se lleva esta calibración a cabo.

Desde un punto de vista clínico, la base de datos puede aumentar en tamaño significativamente, por lo que, añadir algún sistema que la gestione, para poder almacenar y recoger datos de forma rápida y estructurada, sería seguramente conveniente.

Teniendo en cuenta el carácter *“low-cos”t* del sistema, la monitorización en los domicilios de los pacientes sería perfectamente factible, por lo que, un dispositivo *bluetooth*, para transferir los datos a un sistema móvil tipo Android, etc., o cualquier módulo *wifi* que permita la trasmisión de datos, sería también, bajo mi punto de vista, implicaría dotar al sistema de una funcionalidad bastante mayor.

Bibliografía

[1] «elmundo.es,» [En línea]. : https://www.elmundo.es/salud/2015/09/24/56043966ca47410d398b459c.html.

[2] «Opimec,» [En línea]. : https://www.opimec.org/glosario/chronic-diseases/.

[3] «eldigitalsur,» [En línea]. : https://eldigitalsur.com/canarias/canarias-sistema-telemedicina-controla-100-pacientes-dialisis-hemodialisis/.

[4] «europa press,» [En línea]. : https://www.europapress.es/islas-canarias/noticia-cabildo-invierte-casi-500000-euros-impulsar-telemedicina-descentralizacion-pruebas-aligerar-tf-20180509142227.html.

[5] «lavanguardia,» [En línea]. : https://www.lavanguardia.com/vida/salud/20180919/451886168260/hablar-medico-desde-ordenador-posible-espana-brl.html.

[6] «20minutos,» [En línea]. : https://www.20minutos.es/noticia/3297986/0/sensor-dientes-medir-comida/.

[7] «20 minutos,» [En línea]. : https://www.20minutos.es/noticia/3588544/0/lanzan-plantillas-inteligentes-que-miden-el-comportamiento-del-pie-para-prevenir-lesiones/.

[8] Á. S. S. Mario E. Casado García, «Estado del arte de la telemedicina en España y Europa».

[9] H. Barragán, «¿Qué es Wiring?,» 2011. [En línea]. : https://revistas.uniandes.edu.co/doi/pdf/10.18389/dearq8.2011.16. [Último acceso: 2019].

[10] R. H. Herrador, Universidad de Córdoba, 2009. [En línea]. : http://electroship.com/documentos/Arduino_user_manual_es.pdf. [Último acceso: 2019].

[11] [En línea]. : https://www.mcielectronics.cl/shop/product/microcontolador-atmega328p-wcon-arduino-optiboot-uno-10683.

[12] [En línea]. : http://arduino.cl/arduino-uno/.

[13] [En línea]. : http://www.ehu.eus/biomoleculas/buffers/buffer4.htm.

[14] «Hipoxemia,» [En línea]. : https://www.hipoxemia.net/.

[15] «Ruby Villar-Documet,» [En línea]. : https://www.rvd-psychologue.com/es/biofeedback-ecg-gestion-emociones.html.

[16] A. M. Regueiro, Universidad de Málaga, [En línea].: https://www.uma.es/media/files/tallerestr%C3%A9s.pdf.

[17] D. M. GMbH, «www.draeger.com,» [En línea].: https://www.draeger.com/Library/Content/t-core-bk-9101301-es-1604-1.pdf.

[18] «infobae,» [En línea]. : https://www.infobae.com/2013/12/31/1534178-el-calor-del-cuerpo-segun-nuestras-emociones/.

[19] «SciElo revista cubana de medicina,» [En línea].: http://scielo.sld.cu/scielo.php?script=sci_arttext&pid=S0034-75232004000200007.

[20] S. U. d. Roma, «Brain Signs,» [En línea].: https://www.brainsigns.com/es/science/s2/technologies/gsr.

[21] F. X. M. C. y. M. B. Vázquez, «Variables emocionales implicadas en el control de la diabetes,» Universidad de Murcia, [En línea]. : file:///C:/Users/acaymo/Downloads/29671-Article%20Text-137851-1-10-20080716%20(1).pdf.

[22] N. M. Alemán-Soler, «Biometric Approach Based on Physiological Human Signals,» ULPGC, Noida, Delhi, India, 2016.

[23] Soloelectronicos. [En línea]. : http://soloelectronicos.com/2012/12/15/sensor-de-pulso-y-oxigeno-en-la-sangrespo2-para-arduino/.

[24] «Arduino en español,» [En línea]. : http://manueldelgadocrespo.blogspot.com/p/description-digital-pins-with.html.

[25] «Cooking Hacks Forum,» [En línea]. : https://www.cooking-hacks.com/forum/viewtopic.php?f=43&t=7102.

[26] M. specialities, «datasheet 405-736475 sensortemp.pdf».

[27] e.-H. schematics, «cooking hacks,» [En línea]. : https://www.cooking-hacks.com/documentation/tutorials/ehealth-biometric-sensor-platform-arduino-raspberry-pi-medical.

[28] «Hetpro,» [En línea]. : https://hetpro-store.com/TUTORIALES/puerto-serial/.

[29] U. d. Madrid, «um.es,» [En línea]. : https://www.um.es/geograf/sigmur/sigpdf/temario_9.pdf.

[30] U. d. Madrid, «Tec2,» [En línea]. : https://www.um.es/docencia/pguardio/documentos/Tec2.pdf.

[31] «dsi.uclm.es,» [En línea]. : https://www.dsi.uclm.es/personal/MiguelFGraciani/mikicurri/Docencia/Bioinformatica/web_BIO/Teoria/Teoria%20de%20la%20informacion/Entropia.htm.

[32] «Descartes 2D,» [En línea]. : http://recursostic.educacion.es/descartes/web/materiales_didacticos/unidimensional_lbarrios/asimetria_est.htm.

[33] «Teoría de clasificadores,» [En línea]. : http://bibing.us.es/proyectos/abreproy/70448/fichero/05_Capitulo4.pdf.

[34] L. r. n. q. s. y. p. q. e. volviendo, «Xataca,» [En línea]. : https://www.xataka.com/robotica-e-ia/las-redes-neuronales-que-son-y-por-que-estan-volviendo.

[35] V. G. V. García, «Universidad de Granada,» [En línea]. : http://ceres.ugr.es/~alumnos/esclas/.

Pliego de condiciones

El conjunto de herramientas hardware, software así como firmware usados para el desarrollo de este TFG, son expuestas a continuación.

PC.1 Elementos hardware

En la siguiente tabla se presenta el listado de elementos hardware utilizados.

Equipo	Modelo	Fabricante
Portátil	Aspire E1-571	Acer
Plataforma sensores	*e-Health*	Libelium

Tabla PC. 1 Elementos hardware

PC.2 Elementos software

En la siguiente tabla se presenta el listado de las aplicaciones software utilizadas.

Aplicación	Versión
Windows	8
Matlab	R2012a
Arduino IDE	1.0.5-r2
Microsoft Office	2010
Gimp	2.8

Tabla PC. 2 Aplicaciones utilizadas

Presupuesto

Se presentan los gastos generados en la realización de este TFG. Los gastos son desglosados de la siguiente manera:

- Trabajo tarifado por el tiempo empleado.
- Amortización del inmovilizado material.
- Amortización del material hardware.
- Amortización del material software.
- Redacción del documento
- Derechos de visado del Colegio Oficial de Ingenieros Técnicos de
- Telecomunicación (de aquí en adelante COITT).
- Gastos de tramitación y envío.
- Material fungible.

Analizados todos los puntos anteriormente presentados, se aplicarán los impuestos vigentes y se calculará el coste total del TFG.

P.1 Trabajo tarifado por el tiempo empleado

Se contabilizan los gastos referentes a la mano de obra, atendiendo al salario correspondiente a la hora de trabajo de un Ingeniero Técnico de Telecomunicación. Para el cálculo de los honorarios totales se hace uso de la siguiente fórmula:

$$H = C \cdot (74{,}88 \cdot H_n + 96{,}72 \cdot H_e) \quad \text{(P1)}$$

Donde:

- H_n, número de horas trabajadas dentro de la jornada laboral.
- H_e, número de horas trabajadas fuera de la jornada laboral.
- C, factor de corrección en función de las horas trabajadas.

Al desarrollo de este TFG se ha invertido un total de 300 horas. Todas ellas han sido realizadas durante la jornada laboral. En la siguiente tabla se muestra los factores de corrección, impuestos por el COITT, que se han de aplicar según el número de horas trabajadas.

Horas	Factor de corrección
Hasta 36	1,00
Exceso de 36 a 72	0,90
Exceso de 72 a 108	0,80
Exceso de 108 a 144	0,70
Exceso de 144 a 180	0,65
Exceso de 180 a 360	0,60

Tabla PC. 3 Factor de corrección por horas trabajadas según el COITT

De acuerdo con lo establecido por el COITT, según se indica en la tabla anterior, el factor de corrección C que se debe aplicar es de 0,60. Se sustituyen los datos en la fórmula P1:

H=0,60 · (74,88 · 300+96,72 · 0)=13.478,40 €

El trabajo tarifado por tiempo empleado asciende a la cantidad de *TRECE MIL CUATROCIENTOS SETENTA Y OCHO EUROS CON CUARENTA CÉNTIMOS*.

P.2 Amortización del inmovilizado material

Se tendrán en consideración los recursos, tanto hardware como software, empleados para el desarrollo de este TFG.

Para calcular el coste de amortización en un periodo de 3 años se utiliza un sistema de amortización lineal, en el que se supone que el inmovilizado material se deprecia de forma constante a lo largo de su vida útil. El cálculo de la cuota de amortización anual se obtiene de la siguiente fórmula:

$$CuotaAnulla = \frac{Valordeadquisición - Valorresidual}{Añosdevidaútil} \quad \text{(P2)}$$

El valor residual es un valor teórico el cuál se supone que tendrá el elemento en cuestión después de su vida útil.

P.2.1 Amortización del material hardware

El periodo de desarrollo de este TFG ha sido de 4 meses, periodo muy inferior a los 3 años que se estipulan para el coste de amortización. Por dicha razón se calculan los costes sobre la base de los derivados de los primeros 4 meses.

En la siguiente tabla se listan los elementos hardware amortizables necesarios para el desarrollo de este TFG. En dicha lista se indican el valor de adquisición del elemento así como su amortización para el tiempo de 4 meses indicado.

Elemento	Valor de adquisición (€)	Amortización (€)
Portatil	1500,00	200,00
Plataforma *e-Health*	300,00	55,56
	Total	**255,56**

Tabla PC. 4 Amortización del material hardware

El coste total del material hardware es de *DOSCIENTOS CINCUENTA Y CINCO EUROS CON CINCUENTA Y SEIS CÉNTIMOS.*

P.2.2 Amortización del material software

Tal como se comentó en el apartado 6.3.2.1, sólo se tendrá en cuenta los costes derivados de los 4 primeros meses.

En la siguiente tabla se listan las aplicaciones necesarias para el desarrollo de este TFG. En dicha lista se indican el valor de adquisición así como el valor de amortización.

Aplicación	Valor de adquisición (€)	Amortización (€)
Sistema operativo	0,00	0,00
Microsoft Office	0,00	0,00
Matlab	0,00	0,00
Gimp	0,00	0,00
	Total	**0,00**

Tabla PC. 5 Amortización del material software

El coste total de las aplicaciones necesarias es de CERO EUROS. Esto se debe a que todo el software usado bajo licencia estudiante.

P.3 Redacción del documento

Para determinar el coste de la redacción del documento se usa la siguiente fórmula:

$$R = 0{,}07 \cdot P \cdot C_n \qquad \text{(P3)}$$

Donde:

- P, es el presupuesto.
- C_n, es el coeficiente de ponderación en función del presupuesto.

El valor del presupuesto equivale a la suma de los costes de trabajo tarifados por tiempo con las amortizaciones del inmovilizado material. En la siguiente tabla se muestras el cálculo del presupuesto.

Concepto	Coste (€)
Tarificación por tiempo empleado	13.478,40
Amortización del inmovilizado material	255,56
Total	**13.733,94**

Tabla PC. 6 Presupuesto según la tarificación por tiempo y la amortización de material

Según el COITT, el factor de ponderación, para presupuestos cuyo valor es inferior a 30.050,00€, es de 1,00, por lo que los costes derivados de la redacción de documento se obtienen sustituyendo los valores en la ecuación P3.

$R = 0{,}07 \cdot 13.748{,}64 \cdot 1{,}00 = 962{,}40$ €

El coste de la redacción del documento tiene un coste de *NOVECIENTOS SESENTA Y DOS EUROS CON CUARENTA CÉNTIMOS*.

P.4 Derechos de visado del COITT

Para proyectos de carácter general, los derechos de visado para el año 2018 se calculan de acuerdo a la siguiente fórmula.

$$V = 0{,}006 \cdot P_1 \cdot C_1 + 0{,}003 \cdot P_2 \cdot C_2 \qquad \text{(P4)}$$

Donde:

- P_1, es el valor del presupuesto para este proyecto.
- C_1, es el coeficiente reductor en función del presupuesto.
- P_2, es el presupuesto de ejecución material que corresponde a la obra civil.
- C_2, es el coeficiente reductor en función del presupuesto correspondiente a obra
- civil.

Tal como se comentó en el apartado 6.3.3, el coeficiente C_1 está fijado a 1,00. En el desarrollo de este TFG no se ha requerido obra civil por lo cual el valor de P_2 es 0,00€. En la siguiente tabla se muestra el valor del presupuesto hasta el momento.

Concepto	Coste (€)
Tarificación por tiempo empleado	13.478,40
Amortización del inmovilizado material	255,56
Redacción del documento	962,40
Total	**14.696,36**

Tabla PC. 7 Presupuesto según la tarificación por tiempo, la amortización de material y la redacción del documento.

Se sustituyen los valores en la ecuación P4.

$V = 0{,}006 \cdot 14.696{,}36 \cdot 1{,}00 + 0{,}003 \cdot 0{,}00 \cdot C_2 = 88{,}17\ €$

Los costes por derecho de visado del presente TFG son de *OCHENTA Y OCHO* EUROS CON DIECISIETE CÉNTIMOS.

P.5 Gastos de tramitación y envío

Cada documento visado por vía telemática tiene un coste de 6,00€.

P.6 Material fungible

Durante el desarrollo de este TFG se han usado otros materiales a parte de los recursos hardware y software comentados anteriormente. En la siguiente tabla se listan los costes derivados de estos recursos.

Concepto	Coste (€)
Folios	10,00
Tóner para la impresora	30,00
Encuadernado	5,00
Total	**45,00**

Tabla PC. 8 Material fungible

El coste del material fungible es de *CUARENTA Y CINCO EUROS*.

P.7 Aplicación de impuestos y coste total

El impuesto del presente TFG está regulado por el Impuesto General Indirecto Canario con un valor del 6.5%. Teniendo en cuenta la aplicación de dicho impuesto, en la siguiente tabla se realiza el cálculo total del proyecto.

Concepto	Coste (€)
Tarificación por tiempo empleado	13.478,40
Amortización del inmovilizado material	255,56
Redacción del documento	962,40
Derechos de visado del COITT	88,17
Gastos de tramitación y envío	6,00
Costes de material fungible	45,00
Subtotal	**14.835,53**
I.G.I.C.	**964,31**
Total	**15.799,84**

Tabla PC. 9 Coste total del proyecto

El coste total para el desarrollo del TFG "Diseño de una plataforma web basada en ESP32 para adquisición y procesado de eventos sonoros" tiene una valor de *QUINCE MIL SETECIENTOS NOVENTA Y NUEVE EUROS CON OCHENTA Y CUATRO CÉNTIMOS*.