UCLV
Universidad Central
"Marta Abreu" de Las Villas

FIE
Facultad de
Ingeniería Eléctrica

Departamento de Telecomunicaciones y Electrónica

TRABAJO DE DIPLOMA

Título: Plataforma para el procesamiento de bioseñales de forma remota

Autor: Adalberto Ortega Eguino

Tutor: Dr.C. Miguel A. Mendoza Reyes, P.T.

Santa Clara

Hago constar que el presente trabajo de diploma fue realizado en la Universidad Central "Marta Abreu" de Las Villas como parte de la culminación de estudios de la especialidad de Ingeniería en Telecomunicaciones y Electrónica, autorizando a que el mismo sea utilizado por la Institución, para los fines que estime conveniente, tanto de forma parcial como total y que además no podrá ser presentado en eventos, ni publicados sin autorización de la Universidad.

_____

Firma del Autor

Los abajo firmantes certificamos que el presente trabajo ha sido realizado según acuerdo de la dirección de nuestro centro y el mismo cumple con los requisitos que debe tener un trabajo de esta envergadura referido a la temática señalada.

_____ _____

Firma del Tutor

Firma del Jefe de Departamento donde se defiende el trabajo

_____

Firma del Responsable de Información Científico-Técnica

PENSAMIENTO

"Lo único que se interpone entre ti y tu sueño, es la voluntad de intentarlo y la creencia de que en realidad es posible"

Marco Aurelio

DEDICATORIA

A mis padres.

AGRADECIMIENTOS

A mis padres, por apoyarme siempre en todo lo que he necesitado.

A mi tutor, por haberme guiado hasta el final y ser un ejemplo a seguir en todo momento.

A todos mis profesores, por haber forjado incansablemente la persona que soy hoy.

A mis amigos, por haber compartido juntos muchos momentos de vida.

TAREA TÉCNICA

1. Realizar una búsqueda bibliográfica de los principales algoritmos de procesamiento de bioseñales.
2. Analizar los parámetros clínicos de interés de las bioseñales seleccionadas a ser determinados por los algoritmos de procesamiento.
3. Revisar los estándares para las redes inalámbricas de sensores.
4. Revisar la disponibilidad de dispositivos de hardware para la implementación de una plataforma para el procesamiento remoto de bioseñales.
5. Seleccionar el dispositivo de hardware para el nodo coordinador de la red de la plataforma para el procesamiento remoto de bioseñales.
6. Seleccionar y asimilar las herramientas que sirven como entorno de desarrollo para la programación del nodo coordinador de la red de la plataforma para el procesamiento remoto de bioseñales.
7. Configurar y/o programar el nodo coordinador de la red de la plataforma para el procesamiento remoto de bioseñales.
8. Implementar los algoritmos de procesamiento de las bioseñales seleccionadas.
9. Evaluar el desempeño del prototipo de plataforma para el procesamiento remoto de bioseñales a partir del uso de señales sintéticas.
10. Escribir el informe del trabajo.

Firma del Autor

Firma del Tutor

RESUMEN

En este Trabajo de Diploma se presenta el diseño y la puesta a punto de un sistema para el procesamiento remoto de bioseñales. Se describen los principales algoritmos para el procesamiento de bioseñales, las principales tecnologías empleadas en redes inalámbricas de sensores en aplicaciones biomédicas y los componentes de hardware necesarios para su despliegue. A partir del análisis de los principales requerimientos de este tipo de redes se eligen las herramientas y procedimientos adecuados para satisfacerlos mediante la propuesta de una plataforma de procesamiento remoto. En el diseño presentado se toma como referencia el estándar WiFi, el protocolo de comunicación *Message Queue Telemetry Transport* (MQTT), componentes de hardware modulares de bajo costo y alta disponibilidad y herramientas de software libre. La metodología seguida para el desarrollo de esta investigación puede servir como guía para el despliegue de sistemas similares en entornos hospitalarios o en el hogar.

Palabras claves: Procesamiento de bioseñales, WiFi, MQTT.

TABLA DE CONTENIDOS

INTRODUCCIÓN

En las últimas décadas ha existido un explosivo crecimiento de las redes de computadores y, en concreto, de las comunicaciones inalámbricas, propiciado por los continuos avances tecnológicos. Existe una constante evolución hacia dispositivos embebidos, cada vez más pequeños y de menor consumo, dotados con mayor potencia de proceso, capacidad de almacenamiento y facilidad para las comunicaciones. En este contexto surgen las redes inalámbricas de sensores (WSN, *Wireless Sensor Network*), las cuales evolucionan rápidamente debido al gran interés suscitado en busca del "entorno inteligente" [1].

La atención médica ubicua es un paradigma que surge para aumentar la eficiencia, precisión y disponibilidad de los tratamientos médicos. La implementación práctica de este modelo es posible gracias a los recientes avances en las comunicaciones inalámbricas y en la electrónica, que ofrecen sensores pequeños e inteligentes que pueden utilizarse en el cuerpo humano y dispositivos cada vez más potentes y multifuncionales para el procesamiento de los datos, como son las computadoras compactas (SBC, *Single Board Computer*) [2].

De esta forma, las WSN evolucionan hacia redes de menor tamaño diseñadas para aplicaciones biomédicas o servicios sanitarios [3]. Actualmente se están empleando muchos esfuerzos que se focalizan en el desarrollo fiable, flexible y barato de dichas redes que puedan utilizarse en aplicaciones médicas.

La ventaja de las WSN en el entorno biomédico es evidente para la comodidad de los pacientes, especialmente para la monitorización de signos vitales a largo plazo [4]. No obstante, satisfacer el potencial de estas redes requiere abordar un número de desafíos técnicos, los cuales se extienden más allá de las limitaciones de recursos a las que se enfrentan todas las WSN en términos de capacidad de red, limitaciones de procesamiento y memoria así como las escasas reservas de energía [5].

En las aplicaciones médicas, las redes inalámbricas de sensores comparten los mismos retos y agregan algunos otros (por ejemplo, la heterogeneidad de los datos, la movilidad de los pacientes, los problemas de limitaciones de energía en los nodos implantables) dependiendo de sus propósitos y aplicación. Además, debido a la naturaleza intrínseca de la los datos que transmiten, deben garantizarse altos niveles de calidad de servicio (QoS, *Quality of Service*) [3]. Específicamente, a diferencia de las aplicaciones en otros ámbitos, las aplicaciones médicas imponen requisitos estrictos en cuanto a fiabilidad, calidad, privacidad y seguridad del sistema [6].

En el presente trabajo se aborda el diseño de una plataforma para el procesamiento de bioseñales de forma remota haciendo uso de tecnologías de bajo costo.

Situación del problema

Existen diversas variantes para realizar una plataforma para el procesamiento remoto de señales biomédicas en dependencia del tipo de tecnología inalámbrica a utilizar, protocolos, aplicación, algoritmos a implementar, etc. La elección de las características que debe tener la plataforma obedece no solo a los requisitos establecidos para la misma sino a la disponibilidad de los recursos para su implementación y su utilización eficiente.

A pesar de haberse desarrollado varios trabajos relacionados con el uso de computadoras de placa única (SBC, *Single Board Computer*) para el procesamiento remoto de bioseñales, no se cuenta con un sistema flexible que permita modificar parámetros de comunicación y procesamiento en función de la aplicación, tales como la cantidad de nodos remotos, el protocolo de comunicación, los algoritmos para el procesamiento de las señales biomédicas, etc. Debido a lo anterior, esta investigación está orientada a dar respuesta a la interrogante siguiente:

¿Cuáles son las opciones viables en nuestro entorno para desarrollar un sistema de procesamiento remoto de señales biomédicas que utilice la información captada por sensores inalámbricos?

Antecedentes

Los sistemas electrónicos de monitorización remota de señales biomédicas han evolucionado durante los últimos años incrementando su autonomía, portabilidad y funcionalidad. Con la progresión de las generaciones tecnológicas y la aplicación de estas a los trabajos de investigación asociados a redes de monitorización, los servicios de telemedicina han experimentado un importante avance en sus prestaciones. Los trabajos abordados en este campo han tenido como objeto de investigación cuestiones como la ubicación de los elementos de red, componentes de hardware y software asociados al diseño, tipo de tecnología inalámbrica a utilizar, entre otros [7]–[12].

Durante los últimos años se han desarrollado diversos estudios e implementado procesos para la adquisición de señales biomédicas con diversos enfoques, apoyados en herramientas digitales con el fin de mejorar la calidad de vida de los pacientes, teniendo en cuenta factores como la comodidad, la adquisición en tiempo real, la economía, portabilidad, capacidad de almacenamiento y precisión al momento de interactuar con las señales de interés [13]–[17].

Existen estudios comparativos de los estándares establecidos para las WSN en la medicina. Aspectos como el consumo energético en los nodos [18], [19], y la latencia en la transmisión de datos también han estado en el centro de atención de las investigaciones en este campo [2].

La seguridad en este tipo de redes es otro de los temas abordados debido a que los datos deben ser accesibles por cualquier persona autorizada desde lugares remotos. La coordinación entre los componentes de hardware y el programa de software es fundamental para proporcionar una comunicación segura y confiable [20].

En las propuestas consultadas de diseño de redes para la monitorización de signos vitales se implementa el procesamiento de los datos en distintas secciones del sistema, tanto en el lugar donde se realiza la adquisición de los bioseñales como de forma remota en un dispositivo con mayores recursos de cómputo.

En Cuba se han realizados varios estudios relacionados con herramientas tecnológicas vinculadas al monitoreo de variables fisiológicas. En estos se han tratado diversos aspectos como el empleo de medios inalámbricos para transmitir los datos captados por los sensores,

el procesamiento y la presentación de los resultados en entornos virtuales y el monitoreo de pacientes en movimiento [21], [22].

Objetivos

Tomando en consideración los antecedentes expuestos, este trabajo se propone cumplir con los objetivos siguientes:

Objetivo general

Proponer una plataforma viable para el procesamiento remoto de señales biomédicas en un entorno de comunicación inalámbrico.

Objetivos específicos

- Definir los requerimientos de las redes inalámbricas de sensores utilizadas en aplicaciones biomédicas.
- Elegir las alternativas tecnológicas para la implementación de la plataforma para procesamiento remoto de señales biomédicas.
- Evaluar la viabilidad de la propuesta mediante el uso de señales sintéticas.

Interrogantes científicas

- ¿Cuáles son los principales algoritmos para el procesamiento de bioseñales que permiten la determinación de parámetros de interés clínico?
- ¿Cuáles son los protocolos de red existentes para la implementación de las redes inalámbricas de sensores en aplicaciones biomédicas?
- ¿Qué dispositivos de hardware pueden cumplir la función de nodo coordinador de una red inalámbrica de sensores?
- ¿Cuáles son las herramientas y recursos necesarios para procesar los datos recibidos en el nodo coordinador de la red?
- ¿Cómo evaluar la viabilidad de la propuesta de plataforma para el procesamiento de bioseñales?

Tareas de investigación

- Realizar una búsqueda bibliográfica de los principales algoritmos de procesamiento de bioseñales.

- Analizar los parámetros clínicos de interés de las bioseñales seleccionadas a ser determinados por los algoritmos de procesamiento.
- Revisar los estándares para las redes inalámbricas de sensores.
- Revisar la disponibilidad de dispositivos de hardware para la implementación de una plataforma para el procesamiento remoto de bioseñales.
- Seleccionar el dispositivo de hardware para el nodo coordinador de la red de la plataforma para el procesamiento remoto de bioseñales.
- Seleccionar y asimilar las herramientas que sirven como entorno de desarrollo para la programación del nodo coordinador de la red de la plataforma para el procesamiento remoto de bioseñales.
- Configurar y/o programar el nodo coordinador de la red de la plataforma para el procesamiento remoto de bioseñales.
- Implementar los algoritmos de procesamiento de las bioseñales seleccionadas.
- Evaluar el desempeño del prototipo de plataforma para el procesamiento remoto de bioseñales a partir del uso de señales sintéticas.
- Escribir el informe del trabajo.

Organización del informe

El informe de la investigación realizada está estructurado en tres capítulos. En el primer capítulo, a partir del análisis de la bibliografía consultada, se muestra una breve caracterización de las bioseñales de interés para el desarrollo del proyecto y se analizan los principales algoritmos para el procesamiento de bioseñales. Se exponen además las principales características y requisitos de las redes inalámbricas de sensores así como los estándares para su implementación. En el segundo capítulo se describe y fundamenta la elección de los componentes para el desarrollo de la propuesta de plataforma para el procesamiento de bioseñales de forma remota. Se muestran los detalles de la aplicación de los métodos y procedimientos utilizados para la evaluación de la propuesta. En el último capítulo se presentan y analizan los resultados de los experimentos para la validación del diseño a partir del uso de señales sintéticas. Finalmente, en las conclusiones se exponen los principales resultados de la investigación y en las recomendaciones se proponen posibles líneas para el trabajo futuro.

1. EL PROCESAMIENTO DE BIOSEÑALES DE FORMA REMOTA

En este capítulo se analizan las tendencias predominantes en la actualidad sobre la implementación del procesamiento de bioseñales en redes inalámbricas de sensores para el monitoreo fisiológico. Se describen las bioseñales de interés para el desarrollo del proyecto, así como los algoritmos comunes para el procesamiento de las mismas. Se caracterizan las redes inalámbricas de sensores y su impacto en el entorno biomédico, identificando las principales tecnologías utilizadas en su despliegue.

1.1 Señales biomédicas

Las señales biomédicas son registros espaciales, temporales o espacio-temporales de eventos tales como el latido del corazón o la contracción de un músculo. La actividad eléctrica, química o mecánica que ocurre durante estos eventos biológicos frecuentemente produce señales que pueden ser medidas y analizadas. En consecuencia las señales biomédicas o bioseñales contienen información que puede ser utilizada para explicar los mecanismos fisiológicos subyacentes en un evento o un sistema biológico específico [23].

Las bioseñales pueden clasificarse según la fuente o naturaleza física que las origina. Esta clasificación resulta útil cuando las características físicas básicas de los procesos subyacentes son de interés, como ocurre cuando se desea establecer un modelo para la señal. De acuerdo a su fuente las bioseñales se clasifican en bioeléctricas, de bioimpedancias, biomagnéticas, bioquímicas, biomecánicas, bioacústicas o bioópticas.

La clasificación también puede ser concebida de acuerdo con el campo de aplicación, que puede ser de interés cuando el propósito es, por ejemplo, el estudio del sistema fisiológico [24].

Cuando el propósito principal es el procesamiento, no es relevante cuál es la fuente de la señal o a qué sistema biomédico esta pertenece; lo que es primordial son las características de la señal [24].

Desde el punto de vista del diagnóstico es esencial el seguimiento de las señales biomédicas. Para esto es necesario realizar correctamente la adquisición, el procesamiento y la presentación de las señales, de manera que pueda obtenerse información sobre la salud del paciente y el personal clínico pueda actuar en beneficio de la misma.

Entre las principales señales biomédicas se encuentran la electrocardiográfica (ECG), electroencefalográfica (EEG), electromiográfica (EMG), la presión arterial y la respuesta galvánica de la piel. Las principales características de las mismas se describen a continuación.

1.1.1 La señal ECG

La señal electrocardiográfica es la señal eléctrica generada por el corazón y adquirida sobre la superficie del cuerpo mediante electrodos localizados siguiendo el criterio de alguno de los sistemas de derivaciones. Esta señal proporciona, de forma no invasiva, información de la actividad eléctrica del corazón y su estudio permite el diagnóstico de las principales enfermedades cardíacas.

Cada ciclo cardíaco se manifiesta como una secuencia ordenada de ondas de características bien definidas, que por convención se designan por letras del alfabeto (P, Q, R, S y T). Las diferentes ondas que conforman la señal ECG están relacionadas con las diferentes etapas de polarización/despolarización del corazón, y a raíz de las cuales se pueden detectar diferentes patologías [25].

La amplitud (voltajes) de las ondas depende de la manera de aplicar los electrodos a la superficie del cuerpo y de la proximidad de los mismos al corazón. Por lo general la amplitud de la señal ECG oscila entre 0.5 mV y 4 mV [26], [27].

El ancho de banda útil de la señal depende del estudio a realizar y de la región de interés. El ancho de banda clínico en el registro del ECG estándar de doce derivaciones para la detección de las ondas P y T**,** y el complejo QRS**,** es de 0.05 Hz a 100 Hz [28]–[30]. Para aplicaciones de monitorización, en pacientes que requieren cuidados intensivos y pacientes ambulatorios, el ancho de banda útil es restringido desde 0.5 Hz hasta 50 Hz [29], [30]. En otras

aplicaciones (ECG de alta resolución) se extiende el ancho de banda hasta los 300 Hz, para permitir la detección de potenciales tardíos presentes después del complejo QRS, que corresponden a señales de baja amplitud y alta frecuencia [31], [32].

Frecuencia cardíaca y variabilidad del ritmo cardiaco

Un paso indispensable en los sistemas para el procesamiento de la señal ECG es la detección de la posición de los latidos. A partir de la determinación de la onda R operan los procedimientos para la determinación de la frecuencia cardíaca y el estudio de su comportamiento, o los dedicados a la detección de latidos anormales o morfológicamente diferentes que presuponen la existencia de condiciones anómalas en el funcionamiento del sistema cardiovascular.

La frecuencia cardíaca es calculada como el número de latidos en un minuto y se expresa en latidos por minutos (lpm, en español o bpm, en inglés). Esta frecuencia se puede determinar instantáneamente como el inverso del intervalo en segundos existente entre dos latidos consecutivos (intervalo RR en el ECG) multiplicado por 60 para llevarla a un minuto, es decir

$$HR_i = \frac{60}{RR_n} \qquad (1)$$

donde RR_n es el período correspondiente al n-ésimo latido [26], [33].

Otra de las características que se pueden extraer de la señal ECG es la variabilidad del ritmo cardíaco (HRV, *Heart Rate Variability*). El estudio de la misma consiste en analizar la variación del período de tiempo entre latidos consecutivos de forma que se puedan inferir propiedades de interés clínico del sistema cardiovascular [34], [35]. Dicho estudio no sólo aporta información acerca del sistema cardiovascular, sino que además permite evaluar el estado de diferentes sistemas fisiológicos cuya medición de forma directa resulta forzosamente invasiva [34].

Parámetros para digitalizar la señal ECG

Basándose en el teorema de muestreo de Nyquist-Shannon, las recomendaciones de la AHA (sigla en inglés de *American Heart Association*) muestran que 500 Hz es la mínima frecuencia de muestreo para digitalizar señales electrocardiográficas de un adulto con un error menor que el 1%. Para la adquisición de ECG en pacientes pediátricos el filtro paso bajo debe tener una frecuencia de corte como mínimo de 250 Hz para mantener los errores

de amplitud en menos de 25 mV en más del 95% de los casos, con lo que la tasa de muestreo deberá ser un mínimo de 750 Hz [36].

En [37] se concluye que para una detección adecuada de la onda P y el análisis de la variabilidad del ritmo cardíaco se requiere una frecuencia de muestreo mínima de 1 kHz y para un análisis de potenciales tardíos se necesita un mínimo de 4 kHz.

La cantidad de bits necesaria para digitalizar una señal electrocardiográfica puede variar en dependencia al error máximo permisible para la medición en cuestión en función del tipo de paciente. En [38], Bearson compara señales de 8 bits y 10 bits de resolución con respecto a una de 12 bits de resolución en 115 pacientes adultos. Basado en los estudios teóricos de Neubert en [39], es necesario digitalizar a una frecuencia entre 650 Hz y 700 Hz con 8 bits de resolución para mantener un error RMS (sigla en inglés de *Root Mean Square*) de 10 µV o menos. Generalmente las señales electrocardiográficas se muestrean con 12 o 16 bits para obtener una buena resolución.

1.1.2 La señal EEG

La señal electroencefalográfica es una medida de las corrientes que fluyen durante las excitaciones sinápticas de las dendritas de muchas neuronas piramidales en la corteza cerebral [40]. Se mide por medio de electrodos de superficies repartidos por el cuero cabelludo y su interpretación permite la detección de ciertas patologías asociadas al funcionamiento del sistema nervioso así como de características psicofisiológicas del individuo [41]–[43].

El análisis de la actividad de las ondas EEG se centra básicamente en el estudio de diferentes bandas de frecuencias, asumiéndose que a mayor actividad cerebral se tiene una mayor frecuencia y una menor amplitud (existe una relación inversa entre ambas características). En término general, las componentes del EEG se dividen en cinco bandas de frecuencias denominadas ondas *delta*, *theta*, *alfa*, *beta* y *gamma* [25], [44], [45]. Cada una de estas bandas está situada en diferentes rangos de frecuencia, siendo la *gamma* la mayor, llegando hasta los 50 Hz. La existencia de cada una de ellas está asociada al tipo de actividad neuronal frente a estímulos y realización de tareas, así como requerimientos cognitivos y procesamiento de información.

Además de los ritmos anteriormente apuntados, que están relacionados con la actividad espontánea del cerebro, existen otros asociados a acontecimientos discretos que se caracterizan por cambios rápidos (respuestas) y de corta duración que se estudian en el dominio temporal. Estos cambios suelen denominarse potenciales evocados relacionados a eventos (ERP, *Event-Related Potential*) y quedan enmascarados por la actividad espontánea del cerebro ya que la amplitud de estos es muy inferior (por lo general menor a 1 μV) [46].

Parámetros para digitalizar la señal EEG

La digitalización de la señal EEG se realiza mediante el empleo de convertidores análogo/digital multicanales. Afortunadamente, el ancho de banda efectivo está limitado a aproximadamente 100 Hz. Para muchas aplicaciones el ancho de banda puede ser considerado incluso la mitad de este valor. Por lo tanto, una frecuencia mínima de 200 Hz (para satisfacer el criterio de Nyquist-Shannon) es a menudo suficiente para muestrear las señales del EEG. En algunas aplicaciones donde se requiere una resolución más alta para la representación de actividades cerebrales en el dominio de la frecuencia, se pueden utilizar frecuencias de muestreo de hasta 2 kHz.

La representación de cada muestra de señal con hasta 16 bits es muy popular para los sistemas de grabación EEG, aunque algunos estudios emplean hasta 8 bits obteniéndose resultados acertados sin mucho margen de error [47].

1.1.3 La señal EMG

La señal electromiográfica es el registro de la actividad eléctrica generada por el músculo estriado y registrada mediante electrodos superficiales o intramusculares. Esta se compone principalmente de los potenciales de acción de las unidades motoras superpuestas. Cada potencial de unidad motora (PAUM) está formado por la suma de los potenciales de todas las fibras que componen una unidad motora (UM) estando cada una compuesta por una célula nerviosa (motoneurona) y por todas las fibras musculares que están inervadas por ella. Estos potenciales son generados durante la contracción muscular [48].

La información que contiene refleja las fuerzas que son generadas por los músculos y la temporización de los comandos motores. Además, puede usarse en el diagnóstico de patologías que afectan al Sistema Nervioso Periférico, las alteraciones funcionales de las

raíces nerviosas, de los plexos y los troncos nerviosos periféricos, así como de patologías del músculo y de la unión neuromuscular [49].

La amplitud de la señal EMG tiene una variedad de aplicaciones, tales como en el control para prótesis mioeléctricas, estimaciones ergonómicas, sistemas de realimentación (*Biofeedback*), y también ha sido usada para estimar el par asociado a una articulación [49].

Parámetros para digitalizar la señal EMG

Las componentes de principal interés de las señales EMG tienen una frecuencia que oscila entre 50 Hz y 150 Hz, de aquí se deduce que la frecuencia de muestreo adecuada no debe ser menor de 300 Hz. Sin embargo, la señal posee componentes armónicos hasta los 500 Hz, pero cabe resaltar la existencia de algunas señales mioeléctricas que están fuera de este rango alcanzando frecuencias superiores a los 10 kHz. En resumen la frecuencia de muestreo debe ser ajustada en función de las características de la señal a analizar [50].

Los equipos modernos trabajan con frecuencias de muestreo superiores a 20 kHz (intervalos de muestreo inferiores a 50 μs). La mayoría de estos cuentan con conversores de 16 bits, con lo cual se pueden obtener medidas del voltaje del orden de los nanovoltios. Sin embargo, con el empleo de 12 bits no se consigue incrementar el margen de error.

1.1.4 La señal de presión arterial

La señal de presión sanguínea arterial (BP, *Blood Pressure*) se define como el registro de la fuerza ejercida por la sangre contra la pared arterial y/o vascular como resultado del flujo sanguíneo desde el corazón. Esta puede medirse utilizando métodos invasivos o no invasivos, siendo estos últimos los más usados en la práctica médica.

La presión arterial varía de forma semejante a una función sinusoidal representando las distintas etapas del ciclo cardíaco lo cual permite distinguir la presión sistólica que es definida como el máximo de la curva de presión en las arterias durante la sístole y la presión arterial diastólica que es el valor mínimo de la curva de presión durante este período. Las componentes fundamentales de frecuencia de esta señal varían desde el nivel de corriente directa (DC, *Direct Current*) hasta 50 Hz.

Valores tales como la presión arterial sistólica, la presión arterial diastólica, la presión arterial media, la ocurrencia temporal de la incisura dicrota y la presión arterial pulsátil proporcionan

información relevante vinculada a la forma de onda y sufren alteraciones con el envejecimiento y el advenimiento de patologías vasculares [51].

El valor de la presión sanguínea arterial es un indicador esencial de las condiciones fisiológicas de un paciente, siendo una de las variables que más comúnmente se mide en estudios clínicos [52], [53]. A partir de la medición de la presión sanguínea arterial puede determinarse, en primera instancia, si el paciente padece de hipertensión arterial (HTA), siendo esta una de las enfermedades que más frecuentemente afecta la salud de los individuos [54].

1.1.5 Respuesta galvánica de la piel

La respuesta galvánica de la piel (GSR, *Galvanic Skin Response*), también denominada actividad electrodérmica (EDA, *Electrodermic Activity*) y conductancia de la piel (SC, *Skin Conductance*), es la medida de las continuas variaciones en las características eléctricas de la piel, por ejemplo la conductancia, causada por la variación de la sudoración del cuerpo humano [55], [56]. Esta respuesta, se manifiesta con un retraso aproximado de un segundo, aumentando rápidamente hasta un punto y bajando lentamente hasta quedar en el estado de inicio, su forma de onda suele ser bifásica o trifásica y dura varios segundos [57], [58], sus componentes de frecuencia se encuentran entre 0,01 Hz y 1 Hz.

La señal EDA está vinculada al procesamiento emocional y cognitivo autónomo, por ello se utiliza como índice sensible al procesamiento emocional y la actividad del sistema nervioso simpático.

La actividad electrodérmica facilita la obtención de varias características cuantificables que permiten conocer la respuesta de la persona hacia los estímulos. Por un lado, el nivel de conductividad de la piel (SCL, *Level Skin Conductivity*), el cual está asociado al nivel tónico de la piel y, por otro lado, las respuestas de conductividad de la piel (SCRs, *Skin Conductivity Response*), asociadas al sistema nervioso simpático en sus cambios más rápidos, lo que produce eventos cuantificables en la señal. El objetivo en el procesamiento de la señal de conductividad de la piel es extraer eventos, frecuencia, amplitud y nivel de estas características.

1.2 Algoritmos de procesamiento de señales biomédicas

La Figura 1.1 muestra un diagrama genérico de las distintas etapas de aplicación del procesamiento digital de señales biomédicas.

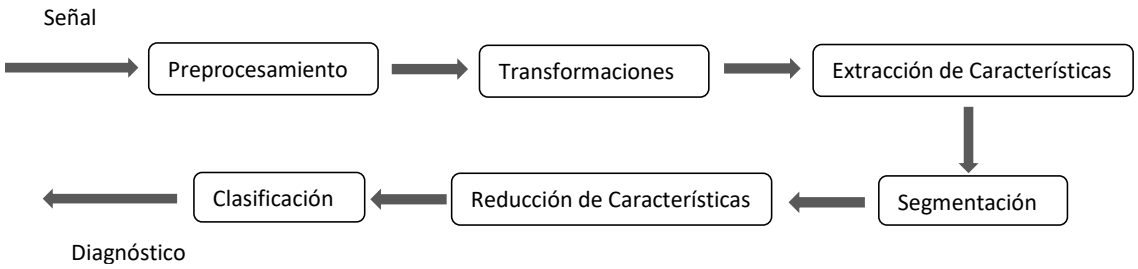

Figura 1.1. Etapas de aplicación del procesamiento digital de bioseñales.

Dependiendo de la aplicación son necesarios sólo determinados bloques. A continuación, se describen brevemente.

1.2.1 Preprocesamiento

En esta etapa se intenta destacar la información deseada del resto de la señal, que usualmente tiene ruido aditivo o multiplicativo asociado, y de la cual puede interesar tan sólo una parte (eliminación del EMG superpuesto al ECG, por ejemplo). Se aplican técnicas de atenuación y cancelación de ruido para lo cual se necesita un conocimiento previo de las características de la señal y del ruido.

Como manifestaciones de ruido se pueden citar la interferencia de la red eléctrica y las variaciones de la línea base. Esta última se produce por cambios en la impedancia de los electrodos, debido a movimientos del paciente durante la adquisición de la bioseñal en cuestión, o a la respiración. Los principales métodos de reducción de ruido se describen a continuación.

Métodos basados en técnicas clásicas de filtrado

Consisten en el empleo de filtros paso bajo, paso alto o su combinación para reducir el ruido presente en la señal, atenuando apreciablemente las componentes de frecuencias no deseadas. Estos filtros se diseñan utilizando las técnicas de respuesta finita al impulso (FIR, *Finite Impulse Respons*,) y respuesta infinita al impulso (IIR, *Infinite Impulse Response*). La ventaja de su uso reside en que las características del filtro pueden cambiarse fácilmente, y su

funcionamiento es independiente del entorno (variaciones de temperatura, tensión de alimentación, tolerancias de los componentes, entre otros).

Métodos basados en filtros de media móvil

Se trata de un método de filtrado muy simple aunque con baja selectividad, además de enmascarar detalles en la alta frecuencia de la señal [59].

Métodos basados en la aproximación mediante funciones

Son técnicas basadas en el reconocimiento de formas, entre estas encontramos la del ajuste de una curva mediante segmentos [60] y el ajuste de curvas mediante funciones ortogonales y polinomios [61].

Métodos basados en la Transformada *Wavelet*

La Transformada *Wavelet* es una herramienta relativamente común en múltiples aplicaciones de procesamiento de señales, entre las que se encuentra la reducción del ruido. Para llevar a cabo esta reducción, prácticamente todos los métodos se basan en los pasos propuestos en [62].

Métodos basados en redes neuronales artificiales

Un método empleado para la eliminación del ruido muscular en señales biomédicas con una estructura multicapa consiste en el empleo de redes neuronales artificiales (RNA). El sistema propuesto en [63] se basa en una red neuronal artificial creciente, que optimiza el número de neuronas en la capa oculta.

Filtros adaptativos

En diferentes investigaciones, como en [64], se describen métodos para la reducción de la variación de la línea base utilizando como técnica el filtrado adaptativo.

Filtros variantes en el tiempo

En los filtros variantes en el tiempo la frecuencia de corte varía en función de algunos parámetros de la señal. Por ejemplo, aplicado a señales electrocardiográficas, en estos filtros la frecuencia de corte va variando según el análisis de las componentes de frecuencias de cada latido, respecto a un promedio de los mismos.

1.2.2 Transformaciones

Antes de emplear algunas técnicas para determinar valores correspondientes a la duración de intervalos, segmentos y puntos característicos de una señal biomédica es necesario hacer alguna transformación del espacio tiempo al de la frecuencia debido a que generalmente el contenido en frecuencias de la forma de onda proporciona más información que en el dominio temporal, incluyendo características útiles para el diagnóstico.

Se pueden citar diferentes métodos de transformaciones aplicadas a las señales biomédicas. Las más usuales son la Transformada Discreta de Fourier (DFT, *Discrete Fourier Transform*), la Transformada de Fourier de Tiempo Corto (STFT, *Short-time Fourier Transform*), o la Transformada *Wavelet* [65]–[73].

La DFT es la herramienta primaria, básica, fundamental del procesamiento digital de señales. Requiere que la función de entrada sea una secuencia discreta y de duración finita. Utilizar la DFT implica asumir que el segmento que se analiza es un único período de una señal periódica que se extiende de forma infinita; si esto no se cumple, se debe utilizar una ventana para reducir los espurios del espectro.

Esta transformación puede ser calculada de forma eficiente en la práctica utilizando el algoritmo de la Transformada Rápida de Fourier (FFT, *Fast Fourier Transform*). El algoritmo impone algunas limitaciones en la señal y en el espectro resultante ya que la señal muestreada y que se va a transformar debe estar compuesta por un número de muestras igual a una potencia de dos para que el procesamiento sea más eficiente. Aunque existen algoritmos rápidos para calcular digitalmente la transformada, los datos del segmento a analizar deben estar almacenados en memoria antes de calcularla.

En el caso de la STFT, la señal se divide en pequeños segmentos, y se calcula la Transformada de Fourier de cada segmento, esta segmentación se realiza mediante la definición de una función de ventana. Este análisis es especialmente importante en aplicaciones en que se requiere estudiar el comportamiento dinámico de su espectro. Uno de los problemas del STFT es que tiene una resolución fija.

Con la Transformada *Wavelet* se observa el comportamiento de distintas partes de la señal, al expandirla en funciones básicas. Provee la representación tiempo-frecuencia de la señal, al brindar información sobre el tiempo y la frecuencia simultáneamente. A diferencia de la

STFT, la Transformada *Wavelet* analiza la señal con distintas resoluciones para las diferentes frecuencias. Su aplicación resulta particularmente útil para el análisis de señales con componentes armónicas de alta frecuencia durante períodos muy cortos de tiempo, y armónicas de baja frecuencia durante largos períodos.

1.2.3 Extracción de características

La extracción de las características de las bioseñales supone una forma de comprimir la señal, eliminando la información redundante y manteniendo todas aquellas propiedades relevantes sin perder información crítica. Con este propósito se utilizan métodos que aproximan la señal tratada, simplificándola en mayor o menor grado.

Existen un gran número de algoritmos de aproximación de señales biomédicas, por ejemplo el empleo de polinomios de grado n, ($n > 1$) para el reconocimiento de las ondas en el pulso carotideo y su compresión [74], la aproximación de la señal electroencefalográfica utilizando funciones de autocorrelación, el empleo de exponentes de Lyapunov [75] o la aproximación de la señal ECG mediante polinomios interpoladores por partes [76].

1.2.4 Segmentación

Frecuentemente, la información relevante de la señal posee una onda característica que se conoce de forma general. Es necesario, en algunos casos, detectar estas ondas características, generalmente como paso previo a algún procesamiento posterior.

A modo de ejemplo, en el análisis de la señal electrocardiográfica es muy importante la detección de ciertos elementos para obtener la duración y amplitud de las ondas con el propósito de aislar los latidos para su posterior clasificación y diagnóstico sobre los resultados obtenidos. En esta etapa son necesarios algoritmos para la detección del complejo QRS [77]–[79], de las ondas P y T. En el procesamiento de señales EEG se emplean métodos de segmentación como son la medición de error espectral (SEM, *Spectral Error Measurement*), el *Generalized Likelihood Ratio (GEM)* o el *Non-linear Energy Operator (NLEO)* [80]. También para la detección de la forma de onda en las señales mioeléctricas se han reportado investigaciones que utilizan métodos estadísticos [81]–[83] y ventanas deslizantes [84].

1.2.5 Reducción de características

La etapa de reducción o selección de las características de una señal resulta ser el paso más importante dentro de un proceso global de agrupamiento. En esta etapa se pretenden extraer de los datos el menor número de características que proporcione la mayor cantidad de información y que permita discriminarlos correctamente durante la posterior fase de clasificación. En la literatura relativa a este tema aparecen una gran cantidad de transformaciones a aplicar sobre los datos [72], [85], [86]. A saber, la Transformada de *Karhunen-Loève* (KLT), la Transformada *Wavelet*, la Transformada de Fourier, el análisis estadístico multivalente: análisis de componentes principales (PCA, *Principal Component Analysis*), análisis multivalente de la varianza, etc. Otro método normalmente empleado para la caracterización de datos son los modelos ocultos de Markov (HMM, *Hidden Markov Models*).

1.2.6 Clasificación

La clasificación permite, a partir de una serie de patrones previamente definidos que establecen las características propias de cada patología, y de las características extraídas de la señal de entrada, determinar a qué clase corresponde la señal. Existen diversos algoritmos de clasificación [40], [73], [86]–[89], a citar ejemplos, *Linear Discriminant Analisis* (LDA), árbol de decisión, el clasificador Bayesiano lineal, el empleo del método *Branch and Bound* y otros más complejos como la utilización de redes neuronales, máquinas de soporte vectorial (SVMs, Support *Vector Machines*), etc. La elección del clasificador depende de la complejidad del problema.

1.3 Redes inalámbricas de sensores

Una red inalámbrica de sensores (WSN, *Wireless Sensor Network*) es una infraestructura constituida por elementos de medición, cómputo y comunicación que permiten instrumentar, observar y responder a eventos y fenómenos en un entorno específico [90].

Las redes inalámbricas de sensores permiten la obtención de la información en el entorno en que se despliegan. En los elementos que la componen la información puede ser procesada localmente y transmitida a través de enlaces inalámbricos hasta un nodo central de

coordinación mediante comunicaciones de un solo salto, *single-hop*, o múltiples saltos, *multi-hop*.

Entre las principales características que definen a una red inalámbrica de sensores se encuentran: la escalabilidad y topología dinámicas, empleo de comunicaciones multisalto, bajo consumo energético del procesador del nodo y de los transceptores de radio, limitaciones de hardware y bajo costo económico.

Estas redes tienen muchas aplicaciones de gran interés para diversos campos. El monitoreo medioambiental, educación infantil, vigilancia, agricultura, aplicaciones comerciales, militares y biomédicas son solo algunos ejemplos.

1.3.1 Componentes de una red inalámbrica de sensores

Existen cuatro componentes básicos en una red de sensores: (1) sensores y actuadores distribuidos o confinados, (2) red de interconexión, no necesariamente totalmente inalámbrica, (3) punto central de agrupamiento de la información o nodo coordinador y (4) conjunto de recursos de cómputo en el punto central (o en otro lugar) para el procesamiento de datos [90].

1.3.1.1 Nodo sensor

Los nodos sensores constituyen una parte principal de la WSN. Su construcción está basada en microcontroladores que son capaces de soportar tanto sensores análogos o digitales, dependiendo cuál sea su requerimiento.

El nodo sensor se encarga de tomar los datos y transmitirlos de manera inalámbrica hacia una estación de mucho mayor poder computacional. También gestionan el acceso al canal inalámbrico y se encargan de enfrentar inconvenientes como interferencia, pérdida de paquetes, etc.

1.3.3.2 Nodo central o coordinador

Generalmente en una red de sensores las funciones de centralización del agrupamiento de la información y el control de la red se realizan en un solo dispositivo con recursos de cómputo apropiados. Una solución cada vez más utilizada es el empleo de computadoras compactas o *single-board computer* (SBC).

Una SBC es una placa de circuitos que contiene lo indispensable para funcionar como una computadora. Esta referencia abarca también las placas de desarrollo que pueden ser programadas y se pueden conectar a varios circuitos [90]. Es un sistema embebido multiuso que corre bajo sistemas operativos como Android, Linux Debian, Linux Ubuntu, Windows 10, entre otros. Posee diversos periféricos tales como USB 2.0, USB 3.0, puertos GPIO, Ethernet, HDMI, Bluetooh, IR, micrófono, entre otros. Estas plataformas se basan en un *System On Chip* (SoC), generalmente de arquitectura ARM, y se encuentran actualmente en formatos de 1 a 8 núcleos.

A diferencia de las computadoras personales de escritorio, las SBC consumen menor cantidad de energía, tienen tamaño reducido, son de bajo costo y tienen un buen desempeño en aplicaciones específicas [91].

Existe una gran comunidad alrededor del uso de las computadoras compactas, formada por entusiastas y profesionales de todo el mundo que comparten proyectos y se ayudan mutuamente. La amplitud de esa comunidad es un factor muy importante desde el punto de vista educativo. Muy a menudo los programas fuentes son publicados como código libre, lo que hace más fácil el desarrollo de proyectos por los interesados [92].

En la Tabla 1.1 se muestran a modo de comparación algunas características de las SBC más utilizadas en los últimos años para el procesamiento de datos.

1.3.2 Estándares inalámbricos

Debido al rápido desarrollo en el campo de las comunicaciones, cada vez más se están desarrollando y adoptando nuevas tecnologías, sobre todo la inalámbricas, que proporcionan una manera más conveniente de transmisión de información que la cableada. A continuación, se caracterizan brevemente los principales estándares inalámbricos usados en las redes de sensores.

1.3.2.1 Estándar IEEE 802.11

Las redes inalámbricas de área local (WLAN, *Wireless Local Area Network*) se basan en el estándar IEEE 802.11, en el que se establecen las especificaciones para el nivel físico y para el control de acceso al medio [93].

Tabla 1.1. Características de algunas computadoras de placa única.

SBC	Procesador	GPU	Memoria
Raspberry Pi 3 Modelo B	Broadcom BCM2837 (4x 64-bit ARM Cortex-53 cores @ 1.2 GHz)	VideoCore IV @ 400 MHz	RAM 1 GB SDRAM
Rock64 Media board	Rockchip RK3328 (4x Cortex-A53 cores @ 1.5 GHz)	Mali-450 MP2 GPU	1 GB, 2 GB, or 4 GB LPDDR3-1600 RAM
Pocket Beagle	Octavo Systems OSD335x SiP con TI Sitara AM3358 (1x Cortex-A8 @ 1 GHz)	PowerVR SGX530	512MB RAM
Banana Pi M64	Allwinner A64 (4x Cortex-A53 @ 1.2 GHz)	Mali-400 MP2	2 GB DDR3 RAM; 8 GB a 64 GB eMMC
Odroid-XU4	Samsung Exynos5422 (4x Cortex-A15 @ 2.0 GHz y 4x Cortex-A7 @ 1.4 GHz)	Mali-T628 MP6	2 GB LPDDR3 RAM; opcional hasta 64 GB eMMC
Asus Tinker Board S	Rockchip RK3288 (4x Cortex-A17 @ 1.8 GHz)	Mali-T760	2 GB LPDDR3 RAM

Las redes WLAN se han difundido ampliamente para dar cobertura a hogares, edificios de oficinas, campus docentes, instalaciones hospitalarias, aeropuertos, ferrocarriles, parques, etc., posibilitando hoy en día una forma fácil de acceso a Internet.

Al primer estándar IEEE 802.11 surgido le siguió el IEEE 802.11b, que fue el primero aceptado ampliamente, posteriormente se desarrollaron las versiones mejoradas IEEE 802.11a, IEEE 802.11g, IEEE 802.11n y 802.11ac. Estas versiones difieren en aspectos como las bandas de frecuencias utilizadas, tipo de modulación, velocidad de transmisión del enlace, etc.

La Comisión Federal de Comunicaciones (FCC) ha creado dos bandas de uso libre, conocidas como: bandas Industrial, Científico y Médico (ISM) y las bandas de Infraestructura de Información Nacional sin Licencia (U-NII). Actualmente existen 12 bandas ISM, pero solo la que empieza en 2.4 GHz es utilizada por los estándares IEEE 802.11, IEEE 802.11b, IEEE 802.11g y IEEE 802.11n, mientras que las cuatro bandas U-NII son utilizadas por IEEE 802.11a y IEEE 802.11n.

El rango de frecuencias se encuentra dividido en canales de un ancho definido, los cuales han sido establecidos por los organismos reguladores de las telecomunicaciones de cada país.

La selección de los canales en el diseño de las redes IEEE 802.11 es indispensable. Para evitar interferencias creadas por los mismos elementos de una red, es necesario seleccionar canales cuyas frecuencias no se solapen.

1.3.2.2 Bluetooth

Bluetooth (IEEE 802.15.1) es una especificación para conectividad basada en radiofrecuencias para dispositivos portátiles personales [90]. Pertenece a la especificación industrial para redes inalámbricas de área personal (WPAN, *Wireless Personal Area Network*) y posibilita la transmisión de voz y datos entre diferentes dispositivos mediante un enlace por radiofrecuencia en la banda ISM de los 2,4 GHz. Su uso está enfocado a dispositivos que requieren corto alcance de emisión y basados en transceptores de bajo costo. La especificación Bluetooth define una tecnología de bajos consumo y potencia con el objetivo de facilitar la comunicación entre dispositivos móviles al eliminar los cables y conectores entre estos y facilitar la sincronización de datos entre equipos personales.

La tecnología permite el soporte de conexiones punto-a-punto y punto-a-multipunto, además tiene la posibilidad de transmitir en *full duplex* con un máximo de 1600 saltos/s. A diferencia de otros estándares inalámbricos, Bluetooth incluye las definiciones de la capa de enlace y la capa de aplicación para los desarrolladores de productos.

Esta tecnología tiene limitadas características de rendimiento debido a su diseño; por lo tanto, su aplicabilidad para las redes de sensores está bastante limitada en la mayoría de los casos.

1.3.2.3 ZigBee

ZigBee es el nombre de la especificación de un conjunto de protocolos de comunicación inalámbrica de alto nivel, para su utilización en aplicaciones de radiodifusión digital de bajo consumo, con base en el estándar IEEE 802.15.4 de redes inalámbricas de área personal (WPAN) [94]. ZigBee añade a este estándar cuatro componentes principales: los niveles de red y aplicación, los objetos de dispositivo ZDO (*ZigBee Device Objects*) y objetos de aplicación definidos por el fabricante, que permiten la personalización y adaptación, y

favorecen la integración total. Se pretende su uso en aplicaciones de propósito general con características autoorganizativas y bajo costo (redes en malla, en concreto).

Entre sus principales características están el soporte de velocidades comprendidas entre 20 kb/s y 250 kb/s, rangos de alcance de 10 m a 75 m, larga duración de la batería, alto número de nodos por red y empleo de las bandas libres ISM de 2,4 GHz (Mundial), 868 MHz (Europa) y 915 MHz (EEUU).

1.3.2.4 Estándar IEEE 802.15.6

IEEE 802.15.6 [58] es el primer estándar de redes de área corporal (WBAN, *Wireless Body Area Network*) que sirve para varias aplicaciones médicas y no médicas y admite comunicaciones dentro y alrededor del cuerpo humano. Este estándar es un paso adelante en las redes de sensores inalámbricos portátiles, ya que está diseñado específicamente para su uso con un amplio rango de velocidades de datos, menor consumo de energía, amplio número de nodos (256) por red de área del cuerpo y diferentes prioridades de nodos de acuerdo con los requisitos de aplicación. El acceso al canal se maneja utilizando CSMA/CA o el procedimiento de acceso ALOHA. Proporciona flexibilidad en las características de seguridad [95]. El estándar IEEE 802.15.6 puede alcanzar velocidades de datos de hasta 10 Mbps.

1.3.2.5 Otras tecnologías de radio

En los últimos años otras tecnologías han cobrado auge dentro de las aplicaciones biomédicas. En este contexto se encuentra la tecnología *Ultra Wideband* (UWB) que se utiliza para sistemas de comunicaciones de corto alcance y proporciona un gran ancho de banda. Debido a su complejidad, no es adecuado para aplicaciones portátiles [96].

ANT es otro estándar emergente para aplicaciones de monitoreo de bienestar y salud. Es un protocolo de baja velocidad y baja potencia soportado por varios fabricantes de sensores [97].

La tecnología Zarlink es de potencia ultra baja, lo que la hace adecuada para aplicaciones de implantes médicos que requieren bajas frecuencia y tasas de datos [98], [99].

El protocolo inalámbrico activo Rubee utiliza señales magnéticas de onda larga para enviar y recibir paquetes de datos cortos (128 bytes) en una red local. Rubee no requiere comunicación de línea de vista para su funcionamiento. Además, tiene las ventajas de una

distancia de transmisión eficiente, un alto nivel de seguridad, un consumo de energía ultra bajo, una autonomía de funcionamiento estable y una larga vida útil de la batería, lo que lo hace conveniente para muchas aplicaciones como el monitoreo de pacientes y la atención médica móvil [98].

1.3.3 Protocolos de comunicación

Las redes inalámbricas de sensores requieren protocolos de aplicación que sean eficientes en cuanto a ancho de banda, energía y recursos disponibles. Dos de los protocolos más usados en las redes de sensores por ser muy ligeros son *Message Queue Telemetry Transport* (MQTT) y *Constrained Application Protocol* (CoAP). Estos protocolos junto a HTTP son discutidos a continuación [100].

1.3.3.1 *Hypertext Transfer Protocol* (HTTP)

HTTP es un protocolo de nivel de aplicación que permite la transferencia de información en la World Wide Web (WWW) [101]. HTTP se basa en sencillas operaciones de solicitud/respuesta. Un cliente establece una conexión con un servidor y envía un mensaje con los datos de la solicitud, el servidor responde con un mensaje similar, que contiene el estado de la operación y su posible resultado. Es un protocolo sin estado, es decir, no guarda ninguna información sobre conexiones anteriores. A pesar de ser robusto y escalable, HTTP necesita más ancho de banda y energía, lo que representa un gran problema para la comunicación entre los dispositivos de la red.

1.3.3.2 *Constrained Application Protocol* (CoAP)

CoAP es un protocolo de nivel de aplicación pensado para ser usado en dispositivos electrónicos simples. Su objetivo es transferir datos desde un gran número de nodos con pocos recursos y redes con poco ancho de banda. Este protocolo implementa el modelo REST de HTTP, el cual es un estilo de arquitectura software para sistemas hipermedia. Los servidores hacen los recursos accesibles bajo un URL (sigla en inglés de *uniform resource locator*), y los clientes acceden a esos recursos utilizando los métodos GET, PUT, POST y DELETE. Por otro lado, CoAP utiliza cabeceras reducidas de 4 bytes optimizando la fragmentación en la capa de enlace y limita el intercambio de mensajes, añadiendo soporte UDP. Además, define cuatro tipos de mensajes: *Confirmable*, *Non-Confirmable*, *Acknowledgment* (ACK) y *Reset*. El mensaje *Confirmable* asegura la fiabilidad por medio de

un simple mecanismo de retransmisión usando un *backoff* exponencial cuando no recibe ningún reconocimiento positivo (ACK) de parte del receptor con el mismo identificador de mensaje. CoAP aborda completamente las necesidades de un protocolo extremadamente ligero y con la naturaleza de una conexión permanente, puede ser apropiado para dispositivos que operan con batería o mediante extracción de energía [102].

1.3.3.3 *Message Queue Telemetry Transport* (MQTT)

MQTT es también un protocolo de nivel de aplicación enfocado al envío de datos en aplicaciones donde se requiere muy poco ancho de banda. Además, sus características le permiten presumir de tener un consumo realmente bajo, así como precisar de muy pocos recursos para su funcionamiento. Sigue una topología en estrella, donde existe un nodo central o bróker con capacidad para trabajar con un gran número de clientes.

MQTT es un protocolo orientado a la conexión en el sentido de que requiere que el cliente establezca una conexión con el bróker antes que pueda intercambiar publicaciones y suscripciones con él. Utiliza la arquitectura publicación/suscripción basada en *topics* que el cliente publica cuando crea un mensaje y los nodos que deseen recibirlo deben subscribirse a él. La comunicación puede ser de uno a uno, o de uno a muchos. Los mensajes de MQTT dependen de la pila TCP/IP y son transmitidos con encabezados fijos de 2 bytes.

Una característica de MQTT es que soporta el concepto de "*last will*". En el momento de la conexión, un cliente podría pedirle al bróker que guarde un mensaje "*last will*" junto a un tópico "*testament*". El bróker enviará este mensaje "*last will*" hacia los suscriptores del tópico "*testament*" cuando pierda la conexión con el cliente anormalmente. Las aplicaciones pueden utilizar esta característica para detectar fallas en los dispositivos y enlaces.

En la versión 3.1 de MQTT se añade el uso de usuario y contraseña para que el bróker proporcione servicios de autenticación. Estos se especificarán en el establecimiento de la conexión. Para asegurar privacidad, la conexión TCP puede ser cifrada usando SSL/TLS. El bróker supervisa la vitalidad de la conexión del cliente mediante un temporizador "*keep-alive*", que define el máximo espacio de tiempo que puede transcurrir entre dos mensajes recibidos por ese cliente.

MQTT especifica tres niveles de calidad de servicio (QoS) que determinarán el procedimiento de intercambio de datos entre los clientes y el bróker. El nivel 0 significa que

el mensaje es enviado una sola vez y ningún ACK es requerido. El nivel 1 asegura que el mensaje se reciba por lo menos una vez, aunque pueden ocurrir duplicados. En el nivel 2 se utiliza un mecanismo de reconocimiento de 4 vías (*four-way handshake*) para entregar el mensaje exactamente una vez [103].

En [100] se muestra una comparación preliminar del rendimiento entre MQTT y CoAP. Los resultados muestran que MQTT exhibe menor retraso comparado con CoAP para pérdidas de paquetes pequeñas. Por otra parte, CoAP consume menos ancho de banda que MQTT para transmitir la misma carga útil. La Tabla 1.2 resume la comparación entre los protocolos HTTP, CoAP y MQTT.

Tabla 1.2. Comparación entre HTTP, CoAP y MQTT [104].

Parámetro	**HTTP**	**CoAP**	**MQTT**
Patrón	Solicitud/Respuesta	Solicitud/Respuesta	Publicación/Suscripción
Capa de transporte	TCP	UDP	TCP
Calidad de Servicio (QOS) (confirmación de entrega)	No	Confirmable No-confirmable	Una vez cuando más Al menos una vez Exactamente una vez
Distribución de los datos	Uno-a-Uno	Uno-a-Uno Uno-a-Muchos	Uno-a-Uno Uno-a-Muchos Muchos-a-Muchos

1.4 Las redes inalámbricas de sensores en la medicina

Debido al aumento de la investigación en el área de redes de sensores inalámbricos (WSN), estas han brindado nuevas oportunidades a los dispositivos médicos. Los desarrollos tecnológicos recientes en circuitos integrados de baja potencia, comunicaciones inalámbricas y sensores fisiológicos promueven el desarrollo de dispositivos de monitoreo diminutos, livianos y de potencia ultra baja. En este contexto surgen las redes inalámbricas de área corporal (WBAN, *Wireless Body Area Network*), ofreciendo un nuevo paradigma para la tecnología WSN en cuanto al monitoreo fisiológico, la capa MAC, la capa de red, estrategias de enrutamiento, etc. Consisten en un conjunto de dispositivos de cómputo, tanto sensores

como unidades de procesamiento, que son integrables al cuerpo humano y se comunican de forma inalámbrica [36].

Los requisitos específicos de las WBAN son principalmente confiabilidad, seguridad, bajo consumo de energía y latencia. Algunas aplicaciones médicas que manejan datos de emergencia no pueden tolerar un tiempo de respuesta prolongado, por lo tanto, se requiere transmisión en tiempo real con garantía de rendimiento.

Aunque la tecnología ha encontrado aplicaciones en varios campos, el dominio médico tiene requisitos muy estrictos de calidad y garantía, lo que causa muchos desafíos que se enfrentan al implementar y operar estos sistemas. Entre los principales retos de este tipo de redes se encuentran los varios tipos de infraestructura de comunicación de red, la tolerancia a fallos, la integridad de datos, el bajo consumo de energía, el retardo de transmisión y las posibles fallas de nodo [57].

Una variedad de sensores fisiológicos que monitorean signos vitales, sensores ambientales y un sensor de ubicación pueden integrarse en una WWBAN (sigla en inglés de *Wearable Wireless Body Area Network*). Este es otro concepto que surge como parte del desarrollo tecnológico en el área biomédica. Las WWBAN están constituidas por sensores económicos, livianos y en miniatura, y pueden permitir un monitoreo de salud discreto y ambulatorio a largo plazo con retroalimentación instantánea al usuario sobre el estado actual de salud en tiempo real y sus registros médicos. Cuando se integra en un sistema de telemedicina más amplio el uso de los registros médicos del paciente, las WWBAN muestran una revolución en la investigación médica al enviar todos los datos recopilados. La gran cantidad de datos fisiológicos recopilados permitirá el análisis cuantitativo de varias condiciones y patrones [105].

Las redes de sensores inalámbricas en la medicina se han convertido en un campo prometedor de estudio, en especial porque se puede monitorear y controlar enfermedades previamente detectadas y al garantizar un control proactivo se garantiza que el paciente cuente con una mejor calidad de vida, además se pueden utilizar para reducir los errores médicos, aumentar la calidad de la atención médica, mejorar la eficiencia de los cuidadores y la comodidad de los pacientes.

1.4.1 Aplicaciones reportadas

Diferentes investigaciones se han enfocado en que los datos captados por los sensores sean transmitidos solamente con previa digitalización a una computadora remota ubicada en el centro de salud, donde se realiza el procesamiento de los datos para la correcta interpretación de los parámetros clínicos de la señal [7], [8], [17]. En [9] se presenta un sistema de monitorización remota capaz de captar la señal ECG y de temperatura de un paciente para ser transmitida vía Bluetooth a un módulo de visualización; posteriormente, la información capturada es enviada a través de una red WiFi a una base de datos implementada en un servidor para el almacenamiento y de ser necesario posterior consulta y visualización, una aplicación web permite el acceso a los datos desde cualquier dispositivo conectado a Internet.

Otros enfoques han optado por hacer algún tipo de preprocesamiento de la señal en el nodo sensor, posteriormente los datos son enviados a un dispositivo remoto para su correcta interpretación. Dentro de esta categoría se encuentra la investigación presentada en [106], donde se sensan las señales ECG, fotopletismográfica, y de temperatura para ser multiplexadas y procesadas en un microcontrolador ATMEGA16 y se envían mediante un módulo WiFi a una computadora ubicada en el centro de salud. En [11] se miden bioseñales de forma no invasiva para obtener parámetros de las mismas a través de su análisis en el dominio del tiempo y la frecuencia en un microcontrolador, los datos resultantes son transmitidos mediante el protocolo TCP/IP a una base de datos para realizar un algoritmo de clasificación sobre los mismos.

En [12] los autores presentan el diseño de un sistema que realiza el procesamiento en el nodo sensor, el cual es llamado unidad de cuidado móvil. Este está compuesto por una plataforma *e-health* como sistema de acondicionamiento de señales y un microcontrolador donde se digitalizan y procesan las mismas, los resultados son enviados vía Internet o mediante la red GPRS (*General Packet Radio Service*) a un servidor central para ser almacenados en una base de datos y posteriormente ser mostrados en los dispositivos de visualización para el diagnóstico.

También en [19] se diseña un sistema que obtiene las señales ECG y de temperatura, y mediante circuitos de acondicionamiento son transmitidas a un dispositivo SoC FPGA (*Field Programmable Gate Array*) donde se realiza la conversión análogo/digital y el

procesamiento de las mismas, los parámetros determinados son enviadas mediante Bluetooth a un teléfono inteligente, *smartphone*, donde se programa una interfaz de usuario para su visualización.

1.5 Conclusiones del capítulo

Distintos algoritmos pueden ser implementados para el procesamiento de bioseñales en función de los parámetros clínicos que se necesiten para determinar el estado de salud del paciente, este procedimiento es necesario para construir un diagnóstico en función de las patologías detectadas. La complejidad de los algoritmos seleccionados debe estar en función de la aplicación deseada y de las características de los dispositivos en los cuales se vayan a implementar. Las redes inalámbricas de sensores en el entorno biomédico deben cumplir con requisitos específicos los cuales son objeto del diseño. Entre estos requisitos, uno de los más importantes es la latencia en la transmisión de los datos, aspecto crítico cuando se necesita monitorear las variables fisiológicas en tiempo real. Para el cumplimiento de estos requerimientos deben ser elegidos apropiadamente los recursos a utilizar en el diseño de una plataforma para el procesamiento de bioseñales de forma remota, teniendo en cuenta las tecnologías existentes y las posibilidades reales de su acceso. Entre las tareas que debe garantizar la plataforma se encuentran: (1) recibir y almacenar temporalmente las señales biomédicas; (2) obtener parámetros clínicos a partir del procesamiento de las señales biomédicas; (3) permitir la visualización de los parámetros clínicos obtenidos y (4) controlar la comunicación entre los componentes de red.

2. DISEÑO E IMPLEMENTACIÓN DE LA PLATAFORMA PARA EL PROCESAMIENTO DE BIOSEÑALES DE FORMA REMOTA

En el presente capítulo se presenta la propuesta de diseño de la plataforma para el procesamiento de bioseñales de forma remota. Se seleccionan los componentes de hardware y herramientas de software que permiten conformar la estructura de la red y la ejecución de las funciones deseadas. Además, se describen los detalles del despliegue del diseño propuesto que incluyen la configuración de los componentes de red y puesta a punta del sistema para el procesamiento de bioseñales.

2.1 Componentes del diseño

La plataforma para el procesamiento de bioseñales de forma remota se diseña para su despliegue en la forma mostrada en la Figura 2.1. Está constituida por el nodo coordinador de la red donde se ejecutan las funciones de recepción, almacenamiento y procesamiento de las bioseñales. Estas deben ser adquiridas por sensores específicos colocados en el cuerpo del paciente.

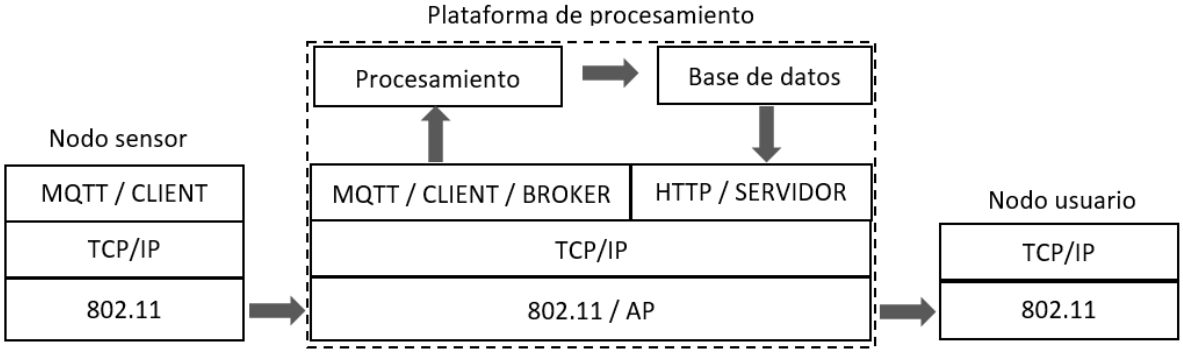

Figura 2.1. Entorno de despliegue de la plataforma para el procesamiento de bioseñales.

En los nodos sensores se acondicionan y digitalizan las variables captadas y son transmitidas hacia el nodo coordinador de red de forma inalámbrica. La plataforma para el procesamiento de bioseñales está formada por dos secciones, una que cumple las funciones de red y otra dedicada al procesamiento de las bioseñales.

2.1.1 Estándar inalámbrico

El estándar seleccionado para la implementación de la plataforma es el IEEE 802.11. Este protocolo está implementado prácticamente en todos los dispositivos móviles y en muchos de los equipos electrónicos que se utilizan diariamente: teléfonos, tabletas, laptops; además de que provee un gran ancho de banda y altas velocidades de transmisión en comparación con otros protocolos analizados. Este estándar demanda un alto consumo de energía. A pesar de esta desventaja, el IEEE 802.11 proporciona la confiabilidad y seguridad necesarias para aplicaciones en el área de la telemedicina, tales como el monitoreo remoto de variables fisiológicas, la recuperación en el hogar de pacientes post-operatorios, entre otras. Además, es el estándar que mayor integración permite con las tecnologías desplegadas.

2.1.2 Protocolo de comunicación

El protocolo de nivel de aplicación seleccionado es MQTT. Este está diseñado para sistemas con limitaciones de memoria y de procesamiento de CPU, también para redes con un ancho de banda mínimo y alta latencia. MQTT consta de tres actores diferentes: cliente-publicador que publica mensajes a un tópico específico, servidor-bróker que recibe los mensajes publicados y los reenvía hacia los destinatarios, que son los clientes-suscritores [92]. El nodo coordinador de la red de la plataforma actúa como bróker, configurado sin autenticación, y en este se ejecuta un cliente-suscriptor que recibe los datos transmitidos por los nodos sensores, los cuales funcionan como clientes-publicadores. En correspondencia con los requisitos para el monitoreo de variables fisiológicas, como parámetro de conexión se especifica el nivel 2 de calidad de servicio en el que además de existir garantías de los mensajes distribuidos se asegura a los suscriptores que no reciban mensajes duplicados.

2.1.3 Nodo coordinador

La selección del hardware como nodo coordinador de red depende del protocolo de comunicación a implementar. Debido a las características de MQTT se necesita un procesador capaz de alojar el bróker MQTT, con suficiente capacidad de cómputo para

manejar las conexiones con los clientes, y que necesariamente tendría que estar encendido todo el tiempo debido a que el procesamiento de las señales se realiza en tiempo real.

El dispositivo seleccionado también debe contar con alta capacidad de procesamiento que permita la ejecución de los algoritmos seleccionados. Con este fin, debido a las ventajas que ofrecen las computadoras compactas, ha sido seleccionado el Raspberry Pi.

El Raspberry Pi es una computadora de placa única con procesador ARM y software libre que se puede utilizar para desempeñar muchas de las funciones que una PC convencional puede hacer. Ha sido desarrollado por la Fundación Raspberry Pi del Reino Unido con la intención de estimular la enseñanza de la informática básica en las escuelas.

En este proyecto se dispuso de un Raspberry Pi - 3 modelo B (Figura 2.2) que utiliza la variante Raspbian del sistema operativo Debian GNU/Linux Jessie de 64 bits, disponible en la página oficial de Raspberry Pi [107].

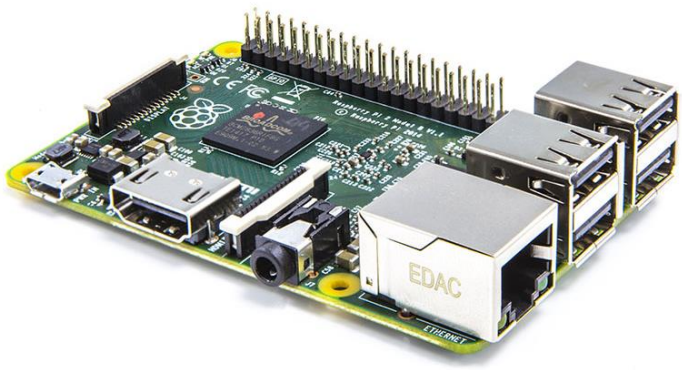

Figura 2.2. Raspberry Pi – 3 modelo B.

2.1.4 Software

Las herramientas de desarrollo y entornos de programación utilizados en el diseño de la plataforma se seleccionaron en base a su disponibilidad, experiencias de usuarios y características. En la Tabla 2.1 se listan las utilizadas en este proyecto y las funciones que cumplen. En las próximas secciones se ofrecen más detalles de los programas seleccionados.

2.1.4.1 Mosquitto

Para la implementación de la comunicación mediante el protocolo MQTT se utiliza el servidor (bróker) Mosquitto. Este es un software de código libre que implementa las

versiones 3.1 y 3.1.1 del protocolo MQTT, es liviano y adecuado para su uso en todos los dispositivos, desde computadoras de placa única de baja potencia hasta servidores completos [108].

Tabla 2.1. Programas utilizados en el desarrollo del proyecto.

Plataforma software	**Objetivo**
Mosquitto	Protocolo de comunicación entre nodos
SQlite	Gestor de base de datos
Nginx	Servidor de HTTP
Python	Recepción, almacenamiento y procesamiento de bioseñales

2.1.4.2 SQlite

SQlite es el gestor de base de datos utilizado en el proyecto para almacenar los parámetros clínicos de las bioseñales obtenidos después del procesamiento. SQlite es un sistema de gestión de bases de datos relacional de código abierto contenida en una relativamente pequeña biblioteca escrita en C. No necesita un proceso separado funcionando como servidor ya que lee y escribe directamente sobre archivos que se encuentran en el disco duro. El formato de la base de datos es multiplataforma e indistintamente se puede utilizar el mismo archivo en sistemas de 32 y 64 bits [109].

2.1.4.3 Nginx

Para la implementación de la página web que permite visualizar los resultados se selecciona el servidor web de código libre Nginx [110]. Nginx soporta múltiples protocolos, entre ellos HTTP(S), WebSocket, IMAP, POP3 y SMTP. Es conocido por su alto rendimiento, estabilidad, configuración simple y bajo consumo de recursos. A diferencia de los servidores tradicionales, no se basa en subprocesos para manejar las solicitudes, en su lugar, utiliza una arquitectura mucho más escalable basada en eventos (asíncrona). Esta arquitectura lo hace ideal para el manejo de múltiples páginas web.

Nginx es uno de los pocos servidores web (junto con Node.js) que es capaz de solucionar el problema de la escalabilidad conocido como C10K (un término acuñado en 1999 por Don

Kegel para 10,000 conexiones concurrentes). Una de sus características importantes es que puede reconfigurarse y actualizarse sin interrumpir la actividad del cliente.

2.1.4.4 Python

Python es un lenguaje de *scripting*, independiente de plataforma y con licencia de código libre, preparado para realizar cualquier tipo de programa, desde aplicaciones *Windows* a servidores de red o incluso páginas web. Es un lenguaje interpretado, lo que significa que no se necesita compilar el código fuente para poder ejecutarlo, que tiene ventajas como la rapidez de desarrollo e inconvenientes como una menor velocidad. Se basa en la programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional [111]. Python ha sido seleccionado como entorno de programación de las funciones de almacenamiento y procesamiento de las señales biomédicas.

2.2 Configuración de las funciones de red

La sección de red está constituida por el nodo coordinador, que actúa como punto de acceso al cual se conectan los nodos sensores y los nodos usuarios. El Raspberry Pi que funciona como coordinador se encargará de manejar las conexiones con los nodos sensores ya que en él reside el bróker MQTT. Los nodos usuario, que pueden ser una computadora portátil o un dispositivo Android, pueden acceder a la red para visualizar el comportamiento de los parámetros clínicos de las bioseñales obtenidos después del procesamiento. Estos parámetros son almacenados en una base de datos para su visualización. A modo de resumen, las funciones realizadas por el Raspberry Pi son:

- Punto de acceso, AP, de la red inalámbrica.
- Bróker del protocolo MQTT.
- Gestor de Base de Datos.
- Servidor de HTTP.

A continuación, se describen los pasos para la configuración del dispositivo que permiten dar cumplimiento a las funciones referidas anteriormente.

2.2.1 Punto de acceso WiFi

La elección del Raspberry Pi para cumplir la función de punto de acceso WiFi de la red inalámbrica de la plataforma está fundamentada por el objetivo de prescindir de un equipo dedicado a ese fin, reduciendo el costo de la red, aunque pueda suponer una disminución de su desempeño. Para configurar el Raspberry Pi como AP se siguieron los pasos propuestos en [104] teniendo en cuenta los parámetros indicados en la Tabla 2.2.

Tabla 2.2. Parámetros principales de configuración del punto de acceso en el Raspberry Pi.

Parámetro	**Valor**
Interfaz	wlan0
SSID	raspi_ap
Contraseña	raspberrypi
Código del país	US
Modo de operación	g
Canal	6
Protocolo de seguridad	WPA2
Mecanismo de clave	WPA-PSK

2.2.2 Instalación de Mosquitto

La instalación del bróker MQTT en el Raspberry Pi permite la comunicación entre dicho dispositivo y los nodos usuario. En el sistema operativo Raspbian la instalación de Mosquitto es sencilla y se realiza mediante la ejecución en el terminal de los comandos:

sudo apt-get update

sudo apt-get install mosquitto mosquitto-clients

La configuración de Mosquitto se puede realizar de dos formas. La primera es mediante el archivo de configuración por defecto ubicado en **/etc/mosquitto/mosquitto.conf**. Es recomendable hacer copia de seguridad de este archivo si se va a modificar. El segundo mecanismo es más seguro y consiste en editar un archivo con extensión **.conf** ubicado en **/etc/mosquitto/conf.d**. Este fichero contiene ejemplos de configuraciones por defecto, que

se pueden editar para uso local. En este caso se ha utilizado la configuración por defecto que se obtiene al instalar el bróker.

2.2.3 Base de datos

Para crear la base de datos se utilizó SQLite3. La base de datos se ha nombrado **PACIENTES.DB**, y dentro de esta residen las tablas correspondientes a los pacientes cuyas variables fisiológicas están siendo monitoreadas. En este caso se creó la tabla **Paciente_1** en la que se almacenan los parámetros clínicos obtenidos como resultado del procesamiento de las señales biomédicas en las columnas: **ECG**, **EEG**, **EMG**, **EDA**, **BP**, y **tiempo**. Los pasos para la creación de la base de datos se indican en la Tabla 2.3.

Tabla 2.3. Pasos para la creación de la base de datos en el Raspberry Pi.

Paso	**Observación**
Instalación de SQLite3	Terminal: **sudo apt-get update** **sudo apt-get install sqlite3 libsqlite3-dev**
Creación de la base de datos	Terminal: **sqlite3 $HOME/PI/PACIENTES.DB "CREATE TABLE Paciente_1 (ECG text, EEG text, EMG text, EDA text, BP text, tiempo text);"**

La actualización de los valores de la base de datos se realiza como parte de la ejecución del programa de la sección de procesamiento el cual se explicará posteriormente.

2.2.4 Servidor HTTP

La principal razón para crear la página web es facilitar la visualización de los parámetros clínicos de las bioseñales almacenados en la base de datos desde cualquier dispositivo que cuente con un navegador web y que esté conectado a la red del Raspberry Pi.

En el diseño de la página se utilizó HTML, CSS, JavaScript y PHP. El servidor web Nginx, de código libre, está disponible en los módulos de instalación del Raspberry Pi. El procedimiento para la instalación y configuración del servidor web en el coordinador se indica en el Anexo I.

La página web puede desplegarse en un dispositivo usuario accediendo mediante un navegador al URL http://xx.xx.xx.xx/index.html, donde "**xx.xx.xx.xx**" es la dirección IP del Raspberry Pi. Los códigos HTML para la creación de la página web y PHP para la conexión con la base de datos implementados en el Raspberry Pi se muestran en el Anexo II.

2.3 Sección de procesamiento

La sección de procesamiento está condicionada por la ejecución de un programa escrito en Python que se ejecuta automáticamente en el Raspberry Pi. El diagrama de flujo del programa es mostrado en la Figura 2.3. Mediante el mismo se configura el cliente MQTT que recibe los datos provenientes de los nodos sensores y son almacenados en un *buffer* cíclico temporalmente para su correspondiente procesamiento. Los resultados del procesamiento son almacenados en una base de datos. En el Anexo III se muestra el *script* implementado que ejecuta dichas funciones. Para la ejecución automática del programa se modifica el archivo **/etc/rc.local** añadiendo el comando **sudo python3.6 /home/pi/cliente_mqtt.py &**.

En los epígrafes siguientes se describen las herramientas usadas en la aplicación.

2.1.5 Cliente MQTT

Para la configuración del cliente MQTT en Python se utiliza la biblioteca *paho* del proyecto Eclipse Paho, que proporciona implementaciones de código libre de los protocolos de mensajería MQTT y MQTT-SN dirigidas a aplicaciones para el Internet de las Cosas (IoT, *Internet of Things*) [112].

Esta biblioteca provee una clase de cliente que permite a las aplicaciones conectarse a un intermediario MQTT (mediante las versiones 3.1 y 3.1.1 del protocolo) para publicar mensajes, suscribirse a temas y recibir mensajes publicados. También proporciona algunas funciones de ayuda para que la publicación de mensajes únicos en un servidor MQTT sea muy sencilla. Es compatible con las versiones de Python a partir de la 2.7.9 y 3.4.

Como característica configurable de MQTT se usó el valor del parámetro *keep-alive* que viene fijado por defecto, el cual es igual a 60 segundos.

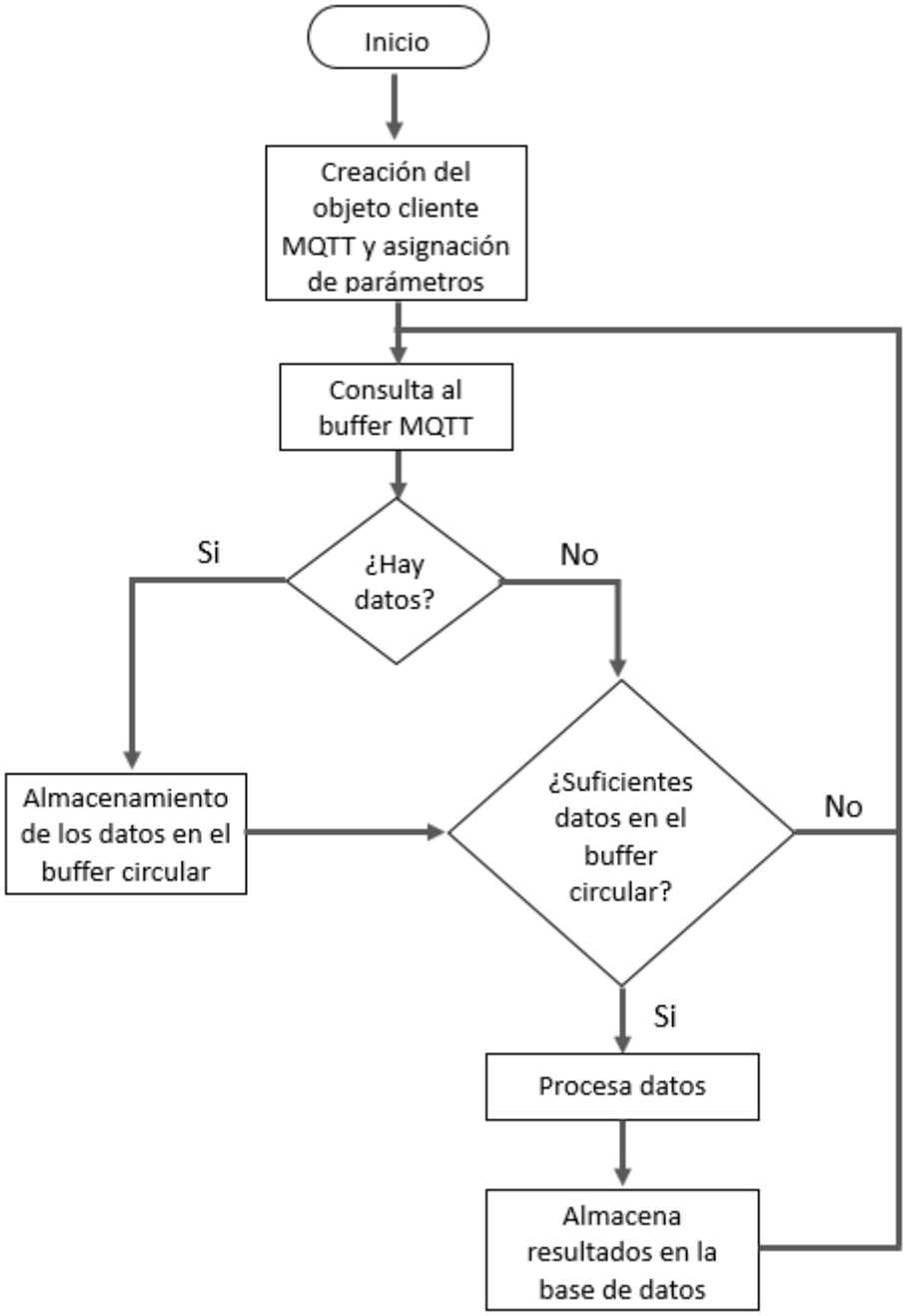

Figura 2.3. Diagrama de flujo de la sección de procesamiento.

2.1.6 Almacenamiento temporal de bioseñales

Una vez recibidos los datos de los nodos sensores correspondientes a las señales biomédicas, es necesario almacenarlos temporalmente en una estructura de memoria que permita el acceso hacia los mismos de forma tal que los procesos de lectura y escritura sean

independientes. Para realizar esta función se usa un *buffer* cíclico que no es más que un arreglo de tamaño fijo el cual adopta su nombre por la forma en la que se insertan o extraen los elementos. El *buffer* diseñado utiliza los punteros *next_index* y *read_index* para acceder a sus elementos, ambos tienen avance incremental y cíclico. Cuando se establecen las condiciones de *buffer* lleno o vacío estos índices siguen ciertas reglas para evitar que se lea información incorrecta del *buffer* o que se escriban datos destruyendo información útil. El *script* implementado corresponde a una clase de Python que es llamada a dicho entorno mediante el comando **import**, debe localizarse en una de las rutas de directorios y archivo del programa. Para conocer dichas rutas se puede usar el módulo *sys* observando la salida del método **sys.path** en el intérprete de Python, dicho módulo proporciona acceso a algunas variables utilizadas o mantenidas por el intérprete. En el Anexo IV se muestra el código del *buffer* cíclico implementado. El tamaño máximo se fija en 20000 muestras, este valor puede ser modificado a conveniencia.

2.1.7 Procesamiento de bioseñales

Para realizar el procesamiento de las señales biomédicas y determinar los parámetros de interés clínico se utiliza la biblioteca de código libre *biosppy*. La misma es un *toolbox* para el procesamiento de señales biomédicas escrito en Python que reúne numerosos algoritmos de procesamiento de señales y reconocimiento de patrones especializados en el análisis de señales biomédicas [113]. La biblioteca es de código abierto y es accesible tanto para fines académicos como comerciales.

La biblioteca se carga desde el entorno Python usando el comando **import**, y está compuesta por los módulos: *biometrics*, *clustering*, *metrics*, *plotting*, *signals.bvp*, *signals.ecg*, *signals.eda*, *signals.eeg*, *signals.emg*, *signals.resp*, *signals.tools*, *storage* y *utils*. En este trabajo se utilizan los módulos *signals.ecg* para el procesamiento de señales electrocardiográficas y *signals.tools* donde se aplican algunas funciones de filtrado.

El módulo *signals.ecg* incluye funciones para la lectura de señales de electrocardiograma, preprocesamiento de señales (filtrado de los componentes que interfieren con la señal, como la actividad muscular, el ruido de la red eléctrica y la variación en la línea de base) y algoritmos para la detección del complejo QRS (Gamboa, Hamilton, *Slope Sum Function*,

etc.) [79], [114], [115]. La función para la estimación de la frecuencia cardíaca instantánea también está contenida en este módulo.

Instalación de *biosppy*

Uno de los requerimientos de *biosppy* es el uso de la biblioteca de Python *scipy*. Esta se basa en paquetes de C y Fortran que deben compilarse en la misma arquitectura. Por lo general, el comando **pip install** permite obtener paquetes precompilados, pero no son compatibles con la arquitectura ARM de Raspberry Pi. Debido a esta limitación se instaló la versión 3.6 de Python que permite la compilación de *scipy* desde la fuente GitHub. En los Anexos V y VI se muestran los pasos realizados para instalar Python 3.6 y *scipy* respectivamente. Una vez realizado este procedimiento se procede a la instalación de *biosppy* mediante la ejecución en un terminal del comando: **sudo pip3.6 install biosppy**.

Modificaciones realizadas

El módulo *signal.ecg* está optimizado para el procesamiento *offline*, por lo que no se toman en cuenta las muestras comprendidas entre el último y primer pico R de segmentos de análisis consecutivos. Con el objetivo de realizar el procesamiento de la señal electrocardiográfica en tiempo real se modificó el método **ecg** de dicho módulo, ubicado en la carpeta de instalación de la biblioteca *biosppy*, la cual se localiza en **/usr/local/lib/python3.6/site-packages**. Dicho directorio puede ser consultado mediante el uso del método **sys.path** en el intérprete de Python. En el Anexo VII se muestran los cambios realizados.

2.4 Escenario de prueba

Para la validación del presente proyecto se realizó el análisis de la representación de la frecuencia cardíaca instantánea de manera secuencial utilizando el algoritmo Hamilton para la segmentación de la señal, por lo que el resultado almacenado en la base de datos será el valor instantáneo de la frecuencia cardíaca y el valor de tiempo correspondiente. Este se conoce como tacograma y su análisis permite tener una medida de la variabilidad de ritmo cardíaco.

Para suplir la necesidad de realizar pruebas con una bioseñal real se utiliza una señal de prueba que corresponde a un segmento de señal ECG tomada en la derivación I. Aunque presenta buena calidad, está mezclada con ruido de la red eléctrica de 50 Hz y componentes

de corriente directa resultados del dispositivo de adquisición. La señal presenta además la influencia de la respiración que produce variabilidad de las amplitudes de los picos R. En la Tabla 2.4 se muestran sus principales parámetros.

El envío de la información correspondiente a las señales biomédicas de prueba hacia la plataforma para el procesamiento remoto se realiza mediante el módulo inalámbrico ESP-01 el cual es una de las configuraciones más extendidas del ESP8266. En la Figura 2.4 se muestra el encapsulado utilizado y su distribución de pines.

Tabla 2.4. Parámetros de la señal de prueba.

Parámetro	**Valor**
Cantidad de muestras	15000
Duración	15 segundos
Valor máximo	2506
Valor mínimo	1987
Frecuencia de muestreo	1 kHz
Resolución	12 bits

El módulo ESP8266, diseñado por Espressif Systems, es un sistema autónomo para soluciones WiFi de bajo consumo y bajo coste que funciona mediante el protocolo TCP/IP, incluye un microcontrolador Tensilica Xtensa LX106 para manejar dicho protocolo y el software necesario para la conexión inalámbrica. Posee potentes capacidades de procesamiento y almacenamiento que le permiten integrarse con sensores y dispositivos específicos de aplicación a través de sus GPIOs. Además el kit de desarrollo de software o SDK (*Software Development Kit*) es libre, permitiendo un desarrollo más rápido de aplicaciones para su uso [116]. A pesar de que es posible la actualización del *firmware* se utiliza el que viene preinstalado por defecto debido a su compatibilidad con el IDE (*Integrated Development Environment*) de Arduino. Esta herramienta de desarrollo es la utilizada para la programación del nodo sensor [117].

El *sketch* de programación desarrollado se muestra en el Anexo VIII y su instalación en el módulo ESP-01 se realizó siguiendo los pasos propuestos en [104].

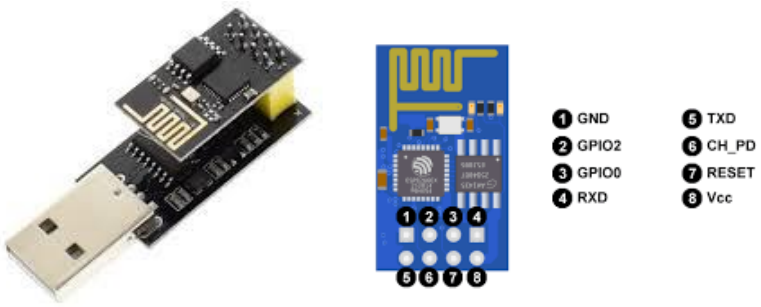

Figura 2.4. Encapsulado y distribución de pines del ESP-01.

Debido a las limitaciones del módulo ESP-01 en cuanto a la cantidad de datos permitidos en el buffer de transmisión, para validar el diseño de la plataforma se envían las señales de prueba mediante un programa en Matlab que se comunica mediante el puerto serie con dicho módulo, este programa corresponde a una interfaz gráfica en la que se muestran los datos transmitidos. El programa desarrollado se muestra en el Anexo IX.

En la Figura 2.5 se indica la distribución de los componentes del escenario de prueba.

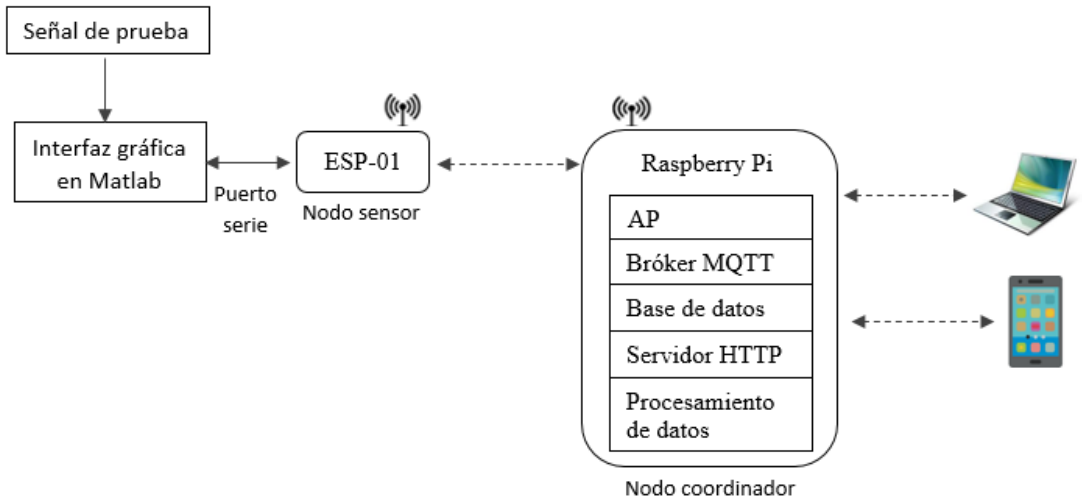

Figura 2.5. Escenario de prueba.

2.5 Conclusiones

La elección de los elementos de hardware y software en la conformación de la plataforma para el procesamiento de señales biomédicas permite asegurar la flexibilidad, fiabilidad y facilidad en el desarrollo de la misma. El uso de componentes de hardware de bajo costo y

herramientas de desarrollo de software libre garantizan la actualización y mantenimiento de las funciones de la plataforma para su uso en aplicaciones biomédicas. La gestión de los parámetros clínicos de interés de las bioseñales mediante una base de datos facilita el acceso a estos valores y su presentación gráfica desde dispositivos diversos.

3. RESULTADOS Y DISCUSIÓN

En este capítulo se exponen los resultados obtenidos en la validación de las distintas secciones que conforman la plataforma para el procesamiento de bioseñales de forma remota. Las pruebas realizadas se enfocan en verificar: (1) la comunicación Matlab/nodo inalámbrico; (2) la comunicación nodo inalámbrico/nodo coordinador; (3) el *buffer* cíclico; (4) el procesamiento de los datos; (5) la operación de la base de datos y (6) la presentación de resultados. También se realiza un análisis de desempeño de la plataforma en un ambiente donde existan múltiples nodos inalámbricos en transmisión y finalmente se presenta un análisis económico de la propuesta.

3.1 Comunicación entre Matlab y el módulo ESP-01

Para comprobar la correcta funcionalidad de la interfaz gráfica desarrollada se conecta el módulo ESP-01 a un puerto USB, utilizando un adaptador USB-serie, en la misma computadora donde se ejecuta el programa en Matlab. La interfaz gráfica diseñada permite visualizar el puerto serie mediante el cual se realiza la comunicación con el módulo ESP-01 así como los datos transmitidos, incluyendo las opciones manuales de comienzo y fin de la transmisión, Figura 3.1. Antes de enviar algún dato es necesario establecer la conexión WiFi entre el módulo ESP-01 y el Raspberry Pi.

Una vez realizada la conexión serie entre Matlab y el módulo inalámbrico se carga la señal de prueba en forma de arreglo con extensión **.mat** hacia el directorio de trabajo de Matlab y se divide en bloques de tamaño fijo de 1000 muestras, los cuales son transmitidos cíclicamente. La cantidad de muestras seleccionadas para formar cada mensaje corresponde a un período del ciclo cardíaco. El módulo ESP-01 envía una confirmación de recepción de

datos la cual es mostrada en la interfaz gráfica, en la Figura 3.1 se muestra el resultado del envío del bloque 2.

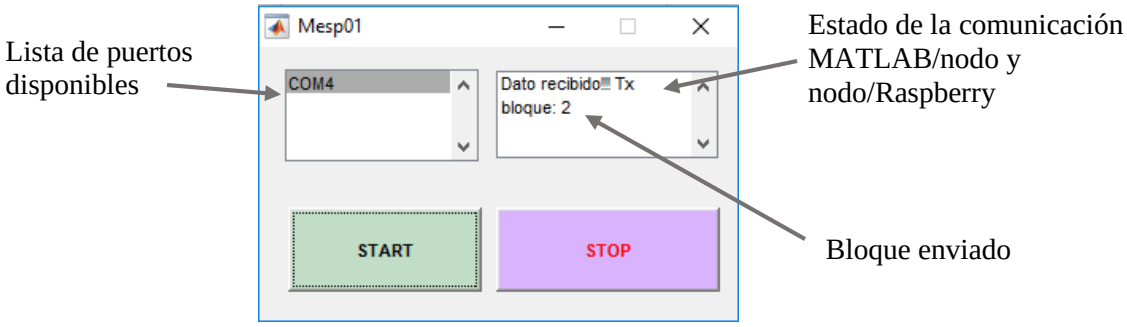

Figura 3.1. Interfaz gráfica en Matlab para el envío de datos mediante el módulo ESP-01.

3.2 Comunicación entre el módulo ESP-01 y el Raspberry Pi

Para la creación del cliente MQTT en el Raspberry Pi se asigna la dirección IP y el puerto del bróker, además se especifican los *topics* del cliente y a los que se subscribe, estos últimos corresponden a los distintos nodos sensores que envían las bioseñales de cada paciente. En la Tabla 3.1 se muestran los parámetros empleados para la configuración de la conexión entre los clientes.

Tabla 3.1. Parámetros de conexión entre los clientes MQTT.

Parámetro	**Valor**
Dirección IP del bróker	10.5.5.1
Puerto del bróker	1883
Topic del cliente/nodo sensor	hospital/sala1/paciente1
ID cliente/nodo sensor	nodo_inalámbrico_1
Topic al que se suscribe el cliente/nodo coordinador	hospital/sala1/#
ID cliente/nodo coordinador	nodo coordinador

En la Figura 3.2 se muestra la ejecución del cliente MQTT en el Raspberry Pi en la que se observa el arribo de mensajes de prueba enviados por un módulo ESP-01. Los mensajes se encuentran identificados por una línea discontinua. Los datos son recibidos en formato

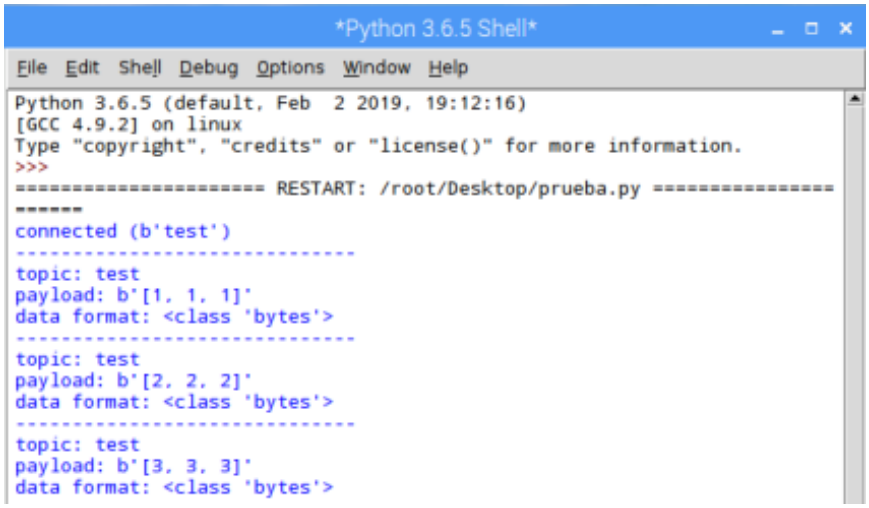

Figura 3.2. Datos recibidos por el cliente MQTT configurado en el Raspberry Pi.

bytes por lo que es necesario decodificarlos antes de realizar cualquier tipo de operación que los involucre.

3.3 Operación del *buffer* cíclico

El *buffer* cíclico posee el método *add_element ()* para almacenar los valores recibidos de los nodos sensores, estos son almacenados uno a la vez y con formato entero. Para extraer los datos se emplea el método *remove_elements ()* pasándole como argumento a la función la cantidad de datos a leer. Para validar el funcionamiento del *buffer* se definió una capacidad de 20 muestras y se realizaron las pruebas indicadas a continuación.

Lectura sobre *buffer* vacío

Antes de almacenar algún dato se intenta extraerlo mostrando que el resultado de lectura sobre el *buffer* vacío produce una salida nula, detectable por el algoritmo de lectura, tal como se indica en el resultado de esta prueba mostrado en la Figura 3.3.

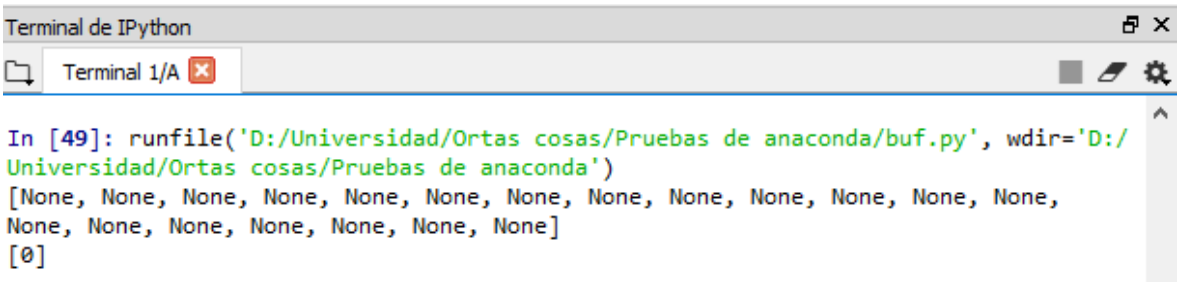

Figura 3.3. Lectura de valores ante la condición de *buffer* vacío.

Lectura secuencial

Se almacenan 20 valores y se extraen en grupos de 6 mostrando que no se leen datos repetidos ni valores incorrectos, la lectura es secuencial. En la Figura 3.4 el primer arreglo mostrado es el contenido del *buffer* y los demás los valores extraídos excepto el último que corresponde a un intento incorrecto de lectura.

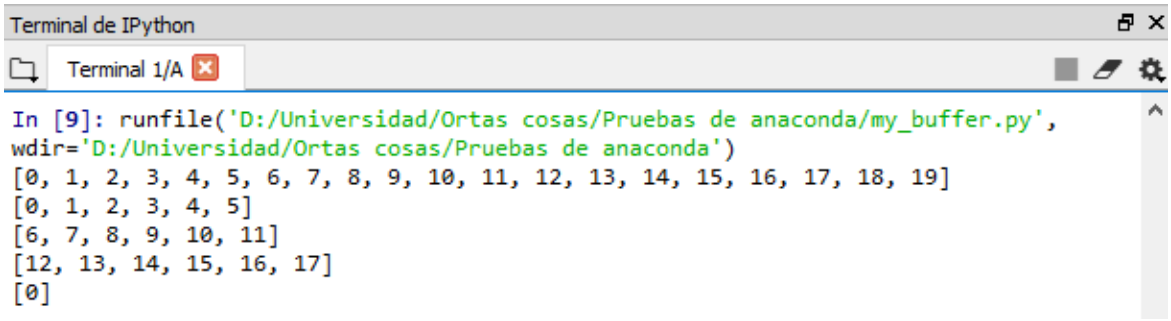

Figura 3.4. Lectura de valores en el *buffer* de forma secuencial.

Relación entre los punteros

En la prueba mostrada (Figura 3.5) se almacenan solo 10 valores en el *buffer*, quedando las 10 capacidades restantes vacías. En una primera operación se extraen 5 valores y posteriormente se intenta extraer 6, el resultado muestra una salida nula debido a que el puntero de lectura no puede sobrepasar al de escritura

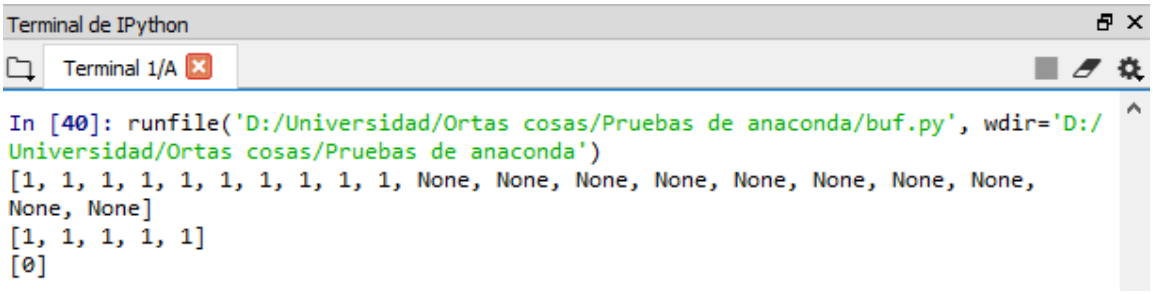

Figura 3.5. Dependencia del puntero de lectura del de escritura en el *buffer* circular.

Lectura cíclica

En esta prueba se realizaron las siguientes operaciones: se guardan 10 valores y se leen 5, posteriormente se almacenan 3 grupos de 5 datos y se leen 15. Esto muestra que la lectura es cíclica y que las operaciones de los punteros de lectura y escritura son independientes siempre

que la posición del primero no sobrepase a la del segundo, ya que de ser así el resultado de la lectura sería una salida nula.

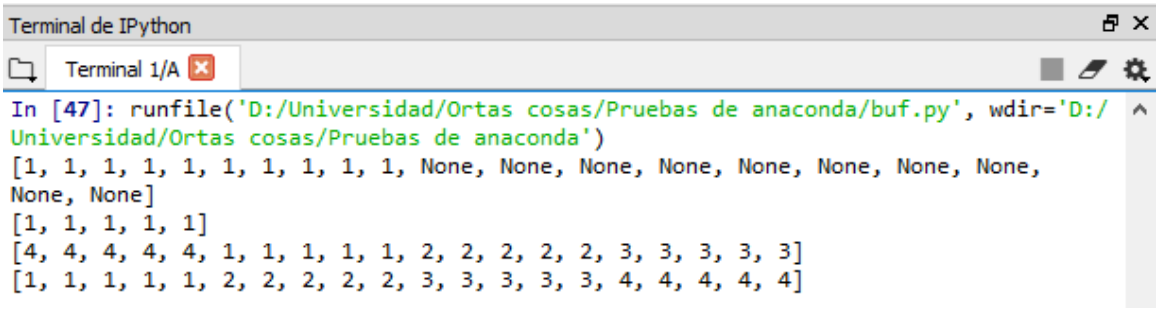

Figura 3.6. Lectura de valores en el *buffer* de forma cíclica.

3.4 Etapa de procesamiento

El procesamiento de los datos de realiza sobre un segmento de señal ECG que contiene 5000 muestras, su longitud es especificada por el usuario en función de la aplicación deseada. La extracción de la frecuencia cardíaca instantánea y su unidad de tiempo correspondiente es resultado de la llamada al método **ecg.ecg** de la biblioteca *biosppy*, pasándole como argumentos el segmento de señal extraído del *buffer* cíclico y la frecuencia de muestreo correspondiente.

La disminución de los niveles de ruido se consigue con un filtrado pasabanda de tipo FIR de orden 300, con frecuencias de corte de 3 y 45 Hz. Posteriormente se determina la posición de los picos R a partir de la implementación del algoritmo Hamilton. Mediante este método se establecen niveles umbrales estimados de picos de ruido y complejo QRS que son utilizados para determinar la posición de los latidos cardíacos y se siguen reglas basadas en duraciones de intervalos para evitar falsas detecciones. Una vez determinada la posición correcta de los picos R se calcula la frecuencia cardíaca instantánea a partir del inverso del intervalo en segundos entre dos latidos consecutivos multiplicado por 60, el resultado es expresado en latidos por minuto. La base de tiempo es calculada a partir de la frecuencia de muestreo y es acumulativa, esto quiere decir que el tiempo de ocurrencia de los latidos cardíacos es contado a partir del latido inicial del primer segmento de señal procesado.

3.5 Base de datos

El estado de la base de datos se actualiza cada vez que se procesan nuevos valores del *buffer* cíclico. Después de realizar la prueba con un módulo transmitiendo durante una hora se almacenaron 4230 registros formados por la frecuencia cardíaca instantánea y su unidad de tiempo correspondiente, resultando en un incremento de 40 kB en el espacio de almacenamiento del Raspberry Pi. Esto muestra que es necesario borrar en algún momento el contenido de las tablas creadas para evitar agotar su capacidad de memoria ya que llegado a este punto el usuario no visualizaría ningún resultado. Las tablas correspondientes a cada paciente se borran con cada reinicio del programa de la sección de procesamiento.

3.6 Presentación de los resultados

La visualización del comportamiento de la frecuencia cardíaca en el tiempo se realiza accediendo mediante el URL http://10.5.5.1/index.html. Durante la configuración del Raspberry Pi como servidor DHCP se configuró esta dirección IPv4 debido a que es de tipo clase A privada, ideal para redes de área local (LAN, *Local Area Network*).

La página web es resultado de la gestión de la base de datos implementada. El comportamiento de la variable mostrada puede ser en tiempo real con cada ciclo de procesamiento en el que existan valores a almacenar en la base de datos, la elección de mostrar los resultados de forma estática o dinámica se realiza mediante la activación de un *push button* en la parte superior derecha del espacio para el gráfico. En una misma ventana se permite la visualización de hasta 30 segundos del registro de la señal electrocardiográfica, este parámetro se puede modificar a conveniencia del usuario. En la Figura 3.7 se muestra el registro obtenido a partir de la señal trasmitida por un módulo. Después de 15 segundos se repite el comportamiento de la variable debido a la transmisión cíclica de la señal de prueba.

3.7 Análisis de desempeño

Para analizar el desempeño del sistema se evaluó el tiempo para el procesamiento de los datos bajo condiciones normales de operación, además se determinó la influencia que tiene en el mismo la cantidad de nodos conectados al coordinador.

Figura 3.7 Registro de la variación de frecuencia cardíaca a partir de la señal de prueba.

En las Figuras 3.8, 3.9 y 3.10 se muestran gráficos de cajas de los valores del tiempo de procesamiento de cada bloque de señal con uno, dos y tres nodos inalámbricos transmitiendo simultáneamente durante una hora. Los rectángulos corresponden al rango de valores típicos de tiempo y los puntos distantes a valores atípicos. Durante la primera prueba el promedio del tiempo de procesamiento es 2,4610 segundos y el rango de los valores típicos prácticamente no varía, acercándose al comportamiento ideal. A medida que aumenta la cantidad de nodos, varía en proporción directa el rango de valores típicos y se presentan valores atípicos de tiempo cada vez más distantes de los primeros, desplazando el valor promedio y aumetando la dispersión. En la Tabla 3.2 se presentan los principales índices estadísticos obtenidos.

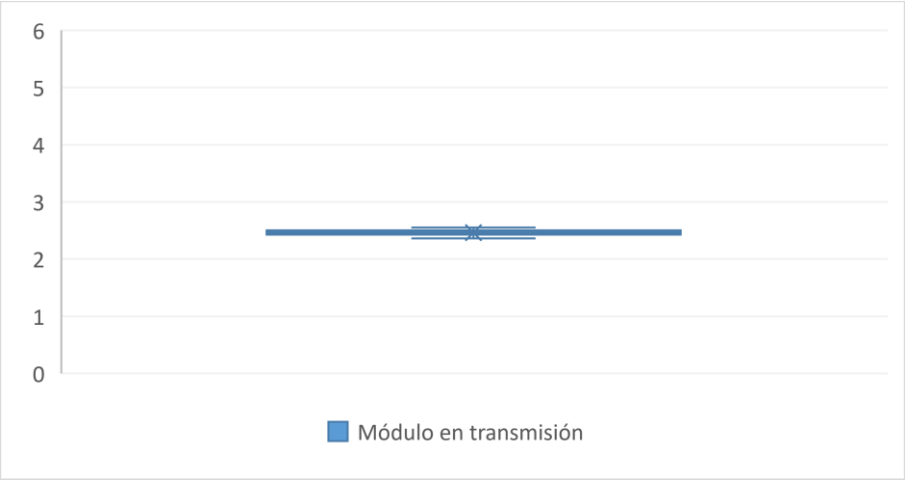

Figura 3.8 Tiempo de procesamiento con un módulo ESP-01.

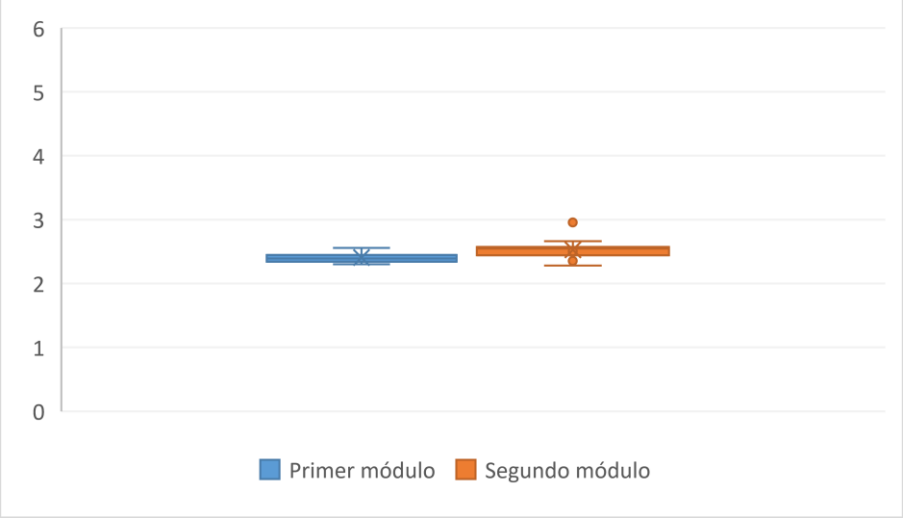

Figura 3.9 Tiempos de procesamiento con dos módulos ESP-01.

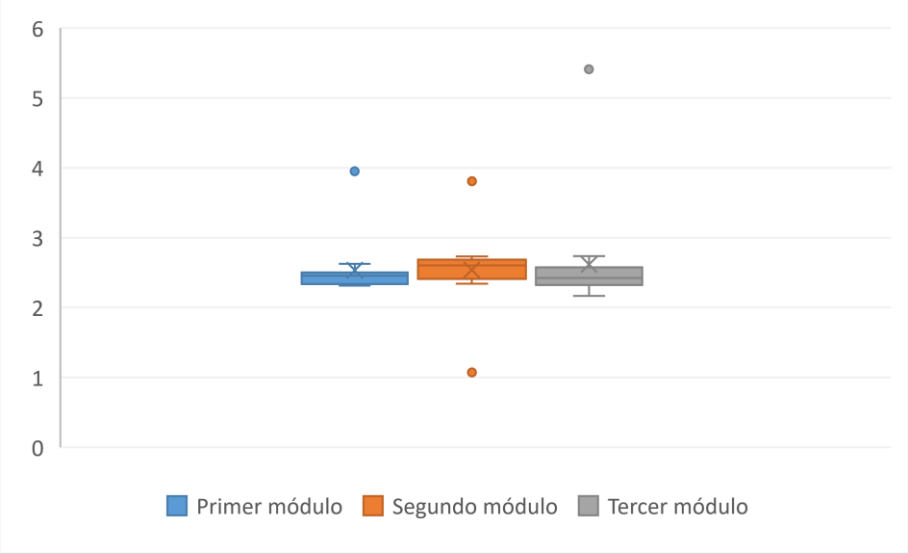

Figura 3.10 Tiempos de procesamiento con tres módulos ESP-01.

Tabla 3.2 Valores estadísticos de tiempos de procesamiento.

Cantidad de nodos	**Valor máximo (segundos)**	**Valor mínimo (segundos)**	**Promedio (segundos)**	**Desviación**
1	2,5507	2,3592	2,4610	0,0528
2	2,9562	2,2808	2,4702	0,1241
3	5,4103	1,0732	2,5648	0,5655

3.8 Análisis económico

El precio de los componentes utilizados en el diseño de la plataforma para el procesamiento de bioseñales de forma remota es el elemento determinante en el costo del sistema propuesto debido a que las herramientas de software que se utilizan son de uso libre. En la Tabla 3.3 se muestran algunos de los precios en USD por unidad de los componentes disopnibles en algunos sitios de internet: Amazon, Deal Extreme y AliExpress

Tabla 3.3 Precios en USD de los componentes utilizados en el diseño.

Componente	**Amazon**	**DealeXtreme**	**AliExpress**
Raspberry Pi modelo B	35,81	49,92	37,69
Adaptador de alimentación	9,99	6,10	4,28
Tarjeta Micro SD 32 GB	13,95	-	-
ESP-01	5,35	3,21	1,48

3.9 Conclusiones

Las pruebas realizadas corroboraron la validez de la propuesta realizada. El *buffer* cíclico permite acomodar las bioseñales recibidas hasta que puedan ser procesadas, logrando independencia entre los eventos de recepción y procesamiento de datos.

El uso de una base de datos para gestionar las variables fisiológicas obtenidas facilita su visualización desde dispositivos usuario diversos, estableciendo las bases para la monitorización de múltiples pacientes. También facilita el estudio de registros históricos.

La opción propuesta es económicamente más atractiva y más viable debido al uso de componentes de bajo costo y alta disponibilidad, además de usar herramientas de software de uso libre.

Debido a que el tiempo para el procesamiento de los datos es menor que el tiempo de arribo del segmento de señal correspondiente para obtener un resultado, es posible bajo estas condiciones la operación de la plataforma en tiempo real.

CONCLUSIONES Y RECOMENDACIONES

La propuesta de plataforma para el procesamiento de bioseñales de forma remota presentada en este trabajo cumple con los requisitos planteados en el proyecto inicial. Las herramientas y métodos usados para el diseño y los resultados obtenidos en las pruebas realizadas permiten plantear las conclusiones siguientes:

1 Se ha diseñado una plataforma para el procesamiento remoto de bioseñales formada por nodos inalámbricos dedicados a la transmisión de bioseñales y un nodo coordinador de la red dedicado al procesamiento de las bioseñales y la presentación y registro de los parámetros resultantes.

2 Las alternativas tecnológicas elegidas para el despliegue de la plataforma permiten a esta cumplir con los requisitos establecidos para este tipo de redes, cumpliendo las funciones de recepción y almacenamiento de las bioseñales captadas, el procesamiento de las mismas y la visualización y el registro de los parámetros obtenidos. El uso de herramientas de software libre y de dispositivos de hardware facilita la asimilación y la ampliación de las potencialidades del diseño desarrollado.

3 Las soluciones de software y hardware planteadas para la realización de la plataforma hacen que la propuesta presentada sea además flexible en cuanto a la implementación de otros algoritmos para el procesamiento de bioseñales. El empleo de la tecnología WiFi conjuntamente con el protocolo MQTT facilita el acceso a la red desde diversos dispositivos de usuario y lo extiende al hacerlo compatible con entornos IoT.

4 El uso del toolbox *biosppy* para el procesamiento de bioseñales confiere a la plataforma herramientas para el análisis de múltiples variables fisiológicas, lo que posibilita contar con un sistema de monitoreo integral e implementable en distintos escenarios.

Recomendaciones

Con el propósito de dar continuidad al presente trabajo, se propone:

1 Utilizar nodos sensores que capten las bioseñales directamente del cuerpo del paciente para analizar el comportamiento del sistema propuesto en un entorno más real.

2 Realizar pruebas donde se empleen distintas frecuencias de muestreo para digitalizar las bioseñales.

REFERENCIAS BIBLIOGRÁFICAS

[1] E. E. Flores Carbajal, «Redes de Sensores Inalámbricas Aplicado a la Medicina», Msc Thesis, Universidad de Cantabria, España, 2012.

[2] R. Negra, I. Jemili, y A. Belghith, «Wireless Body Area Networks: Applications and Technologies», *Procedia Computer Science*, vol. 83, pp. 1274-1281, ene. 2016.

[3] C. Abreu, M. Ricardo, y P. Mendes, «Framework for QoS performance assessment on biomedical wireless sensor networks», presentado en BIODEVICES, 2012.

[4] M. S. Wegmüller, «Intra-body communication for biomedical sensor networks», PhD Thesis, ETH Zurich, 2007.

[5] H. Elayan, R. M. Shubair, y A. Kiourti, «Wireless sensors for medical applications: Current status and future challenges», en *2017 11th European Conference on Antennas and Propagation (EUCAP)*, 2017, pp. 2478-2482.

[6] J. Ko, C. Lu, M. B. Srivastava, J. A. Stankovic, A. Terzis, y M. Welsh, «Wireless Sensor Networks for Healthcare», *Proceedings of the IEEE*, vol. 98, n.º 11, pp. 1947-1960, nov. 2010.

[7] R. S. Dilmaghani, H. Bobarshad, M. Ghavami, S. Choobkar, y C. Wolfe, «Wireless sensor networks for monitoring physiological signals of multiple patients», *IEEE transactions on biomedical circuits and systems*, vol. 5, n.º 4, pp. 347–356, 2011.

[8] N. S. Shivakumar y M. Sasikala, «Design of vital sign monitor based on wireless sensor networks and telemedicine technology», en *2014 International Conference on Green Computing Communication and Electrical Engineering (ICGCCEE)*, 2014, pp. 1–5.

[9] J. P. Tello, O. Manjarres, M. Quijano, A. Blanco, F. Varona, y M. Manrique, «Remote monitoring system of ECG and human body temperature signals», *IEEE Latin America Transactions*, vol. 11, n.º 1, pp. 314–318, 2013.

[10] A. Rizal, V. Suryani, J. Jondri, y S. Hadiyoso, «Development of Wireless Patient's Vital Sign Monitor Using Wireless LAN (IEEE. 802.11. b/g) Protocol», *International Journal of Electrical and Computer Engineering*, vol. 4, n.º 6, p. 893, 2014.

[11] S.-W. Liou, D. Kurniadi, B.-R. Zheng, W.-Q. Xie, C.-J. Tien, y G.-J. Jong, «Classification of biomedical signal on IoT platform using support vector machine», en *2018 IEEE International Conference on Applied System Invention (ICASI)*, 2018, pp. 50–53.

[12] M. Abo-Zahhad, S. M. Ahmed, y O. Elnahas, «A wireless emergency telemedicine system for patients monitoring and diagnosis», *International journal of telemedicine and applications*, vol. 2014, p. 4, 2014.

[13] J. C. Abib y J. C. Anacleto, «Improving Communication in Healthcare: a case study», en *2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2014, pp. 3336-3341.

[14] M. Aminian y H. R. Naji, «A Hospital Healthcare Monitoring System Using Wireless Sensor Networks», *Journal of Health & Medical Informatics*, vol. 4, n.º 2, pp. 1-6, feb. 2013.

[15] E. Freitas y A. C. de Azevedo, «Wireless Biomedical Sensor Networks : The Technology», presentado en Proceedings of the 2nd world congress on electrical engineering and computer systems and science (EECSS'16), Budapest, Hungary, 2016, pp. 1-134.

[16] S. E. C. Bastidas y J. M. L. Peláez, «ESTUDIO DE REDES DE SENSORES Y APLICACIONES ORIENTADAS A LA RECOLECCIÓN Y ANÁLISIS DE SEÑALES BIOMÉDICAS.», *1*, vol. 12, n.º 33, pp. 85-99, oct. 2013.

[17] P. Wang, «The real-time monitoring system for in-patient based on zigbee», en *2008 Second International Symposium on Intelligent Information Technology Application*, 2008, vol. 1, pp. 587–590.

[18] T. Sheltami, A. Mahmoud, y M. H. Abu-Amara, «An ad hoc wireless sensor network for telemedicine applications», *The Arabian Journal for Science and Engineering*, vol. 32, n.º 1, pp. 131-146, abr. 2007.

[19] R. Joaquinito y H. Sarmento, «A wireless biosignal measurement system using a SoC FPGA and Bluetooth Low Energy», en *2016 IEEE 6th International Conference on Consumer Electronics-Berlin (ICCE-Berlin)*, 2016, pp. 36–40.

[20] A. Sudarsono, M. U. H. A. Rasyid, y H. Hermawan, «An implementation of secure wireless sensor network for e-healthcare system», en *2014 International Conference on Computer, Control, Informatics and Its Applications (IC3INA)*, 2014, pp. 69-74.

[21] D. F. D. Díaz, Á. D. B. Varela, A. C. Rodríguez, y Y. O. S. Franco4, «Diseño de un módulo para la adquisición y transmisión inalámbrica de bioseñales.», *Revista Telemática*, vol. 16, n.º 2, pp. 39-53, dic. 2017.

[22] M. J. G. Valdés, «Desarrollo de Monitores de Paciente en Cuba», *Revista Científica de Ingeniería Electrónica, Automática y Comunicaciones ISSN: 1815-5928*, vol. 31, n.º 1, pp. 42-48, 2011.

[23] D. Sánchez Morillo, «Procesado y transmisión de señales biomédicas para el diagnóstico de trastornos y enfermedades del sueño», PhD Thesis, Universidad de Cádiz, España, 2008.

[24] W. S. Aronow, «Cardiovascular manifestations seen in obstructive sleep apnea», *Comprehensive therapy*, vol. 33, n.º 2, p. 82, 2007.

[25] L. Sörnmo y P. Laguna, *Bioelectrical signal processing in cardiac and neurological applications*, vol. 8. Academic Press, 2005.

[26] A. C. Guyton, J. E. Hall, y M. J. Moreno, *Tratado de fisiologia médica*, vol. 9. Interamericana México, 1971.

[27] C. Ramírez y M. Hernández, «Procesamiento en tiempo real de variables Fisiológicas», *Universidad Nacional de Experimental de Táchira, Decanato de Investigación, Grupo de Biomédica*.

[28] J. D. Bronzino y Chemical Rubber Company, Eds., *The biomedical engineering handbook. Vol. 1:* ..., 2. ed. Boca Raton, Fla.: CRC Press [u.a.], 2000.

[29] W. J. Tompkins, Ed., *Biomedical Digital Signal Processing: C Language Examples and Laboratory Experiments for the IBM PC*, Har/Dskt edition. Englewood Cliffs, N.J: Prentice Hall, 1993.

[30] D. Prutchi y M. Norris, *Design and development of medical electronic instrumentation: a practical perspective of the design, construction, and test of medical devices*. John Wiley & Sons, 2005.

[31] G. Breithardt *et al.*, «Standards for analysis of ventricular late potentials using high-resolution or signal-averaged electrocardiography: A statement by a task force committee of the European Society of Cardiology, the American Heart Association, and the American College of Cardiology», *Journal of the American College of Cardiology*, vol. 17, n.º 5, pp. 999-1006, abr. 1991.

[32] A. T. Crispi, «Improving ventricular late potentials detection effectiveness», PhD Thesis, Ph. D. Thesis, The University of New Brunswick, Canada, 2002.

[33] G. D. Clifford, F. Azuaje, y P. Mcsharry, «ECG statistics, noise, artifacts, and missing data», *Advanced methods and tools for ECG data analysis*, vol. 6, p. 18, 2006.

[34] J. M. Gascón, «Análisis de la variabilidad del ritmo cardiaco: Representación temporal e índices clínicos», PhD Thesis, Universidad de Zaragoza, Departamento de Ingeniería Electrónica y Comunicaciones, España, 1999.

[35] M. Malik *et al.*, «Heart rate variabilityStandards of measurement, physiological interpretation, and clinical use», *Eur Heart J*, vol. 17, n.º 3, pp. 354-381, mar. 1996.

[36] P. R. Rijnbeek, J. A. Kors, y M. Witsenburg, «Minimum bandwidth requirements for recording of pediatric electrocardiograms», *Circulation*, vol. 104, n.º 25, pp. 3087–3090, 2001.

[37] P. Kligfield *et al.*, «Recommendations for the standardization and interpretation of the electrocardiogram: part I: the electrocardiogram and its technology a scientific statement from the American Heart Association Electrocardiography and Arrhythmias Committee, Council on Clinical Cardiology; the American College of Cardiology Foundation; and the Heart Rhythm Society endorsed by the International Society for Computerized Electrocardiology», *Journal of the American College of Cardiology*, vol. 49, n.º 10, pp. 1109–1127, 2007.

[38] A. S. Berson, J. M. Wojick, y H. V. Pipberger, «Precision requirements for electrocardiographic measurements computed automatically», *IEEE Transactions on Biomedical Engineering*, n.º 4, pp. 382–385, 1977.

[39] J. Rawles, «Computer ECG analysis: Towards standardization: edited by JL Willems, JH van Bemmel and C. Zywietz, Elsevier Science Publishers, Amsterdam, 1986. 400 pp. US $57.00. ISBN 0-444-87866-1», *International Journal of Bio-Medical Computing*, vol. 20, n.º 1, p. 153, 1987.

[40] S. Sanei, «Fundamentals of EEG Signal Processing», en *Adaptive Processing of Brain Signals*, John Wiley & Sons, Ltd, 2013, pp. 37-44.

[41] A. Gevins, M. E. Smith, L. McEvoy, y D. Yu, «High-resolution EEG mapping of cortical activation related to working memory: effects of task difficulty, type of processing, and practice», *Cereb. Cortex*, vol. 7, n.º 4, pp. 374-385, jun. 1997.

[42] T. Fernández *et al.*, «EEG activation patterns during the performance of tasks involving different components of mental calculation», *Electroencephalography and Clinical Neurophysiology*, vol. 94, n.º 3, pp. 175-182, mar. 1995.

[43] T. W. Picton, Ed., «Endogenous event-related potentials», en *Handbook of Electroencephalography and Clinical Neurophysiology*, 1 edition., vol. 3, Amsterdam ; New York: Elsevier Science Ltd, 1988, pp. 361-426.

[44] L. C. Arangüena y J. I. Dorado, *Psicofisiología: fundamentos metodológicos*. Pirámide, 1997.

[45] P. L. Nunez y R. Srinivasan, *Electric Fields of the Brain: The Neurophysics of EEG*, 2 edition. Oxford ; New York: Oxford University Press, 2005.

[46] M. M. Monge y A. J. M. Cantero, «Procesamiento y caracterización de bioseñales para su uso en interfaces de control y afectividad», PhD Thesis, Universidad de Sevilla, España, 2015.

[47] G. R. Bermúdez, P. J. G. Laencina, D. Brizion, y J. R. Dorda, «Adquisición, procesamiento y clasificación de señales EEG para el diseño de sistemas BCI basados en imaginación de movimiento», *Jornadas de introducción a la investigación de la UPCT*, n.º 6, pp. 10–12, 2013.

[48] J. E. Moore y G. Zouridakis, Eds., *Biomedical technology and devices handbook*. Boca Raton: CRC Press, 2004.

[49] R. Merletti, P. A. Parker, y P. J. Parker, *Electromyography: physiology, engineering, and non-invasive applications*, vol. 11. John Wiley & Sons, 2004.

[50] R. B. Northrop y N. R. B, «Measurement of Electrical Potentials and Magnetic Fields from the Body Surface», en *Noninvasive Instrumentation and Measurement in Medical Diagnosis*, Edición: 1., Boca Raton: CRC Press, 2001, pp. 58-62, 107-114.

[51] L. J. Cymberknop, W. Legnani, F. M. Pessana, y R. L. Armentano, «Procesamiento no lineal de señales cardiovasculares: marco conceptual para la detección de patologías1», PhD Thesis, UNIVERSIDAD TECNOLÓGICA NACIONAL BUENOS AIRES, Argentina, 2013.

[52] G. Beevers, G. Y. Lip, y E. O'Brien, «Blood pressure measurement. Sphygmomanometry: factors common to all techniques», *BMJ*, vol. 322, pp. 981-985, 2001.

[53] R. J. Marchiando y M. P. Elston, «Automated Ambulatory Blood Pressure Monitoring: Clinical Utility in the Family Practice Setting», *AFP*, vol. 67, n.º 11, pp. 2343-2350, jun. 2003.

[54] M. D. Pérez Caballero, A. Dueñas Herrera, J. Alfonso Guerra, A. Vázquez Vigoa, D. Navarro Despaigne, y M. Hernández Cueto, «Guía cubana para la prevención, diagnóstico y tratamiento de la hipertensión arterial», *La Habana: ECIMED*, 2008.

[55] B. T. Shahani, J. J. Halperin, P. Boulu, y J. Cohen, «Sympathetic skin response--a method of assessing unmyelinated axon dysfunction in peripheral neuropathies.», *Journal of Neurology, Neurosurgery & Psychiatry*, vol. 47, n.º 5, pp. 536-542, may 1984.

[56] M. Baba, Y. Watahiki, M. Matsunaga, y K. Takebe, «Sympathetic skin response in healthy man», *Electromyogr Clin Neurophysiol*, vol. 28, n.º 5, pp. 277-283, jul. 1988.

[57] A. Natarajan, M. Motani, B. de Silva, K.-K. Yap, y K. C. Chua, «Investigating Network Architectures for Body Sensor Networks», en *Proceedings of the 1st ACM SIGMOBILE International Workshop on Systems and Networking Support for Healthcare and Assisted Living Environments*, New York, NY, USA, 2007, pp. 19–24.

[58] I. S. Association, «IEEE Standard for Local and Metropolitan Area Networks—Part 15.6: Wireless Body Area Networks», *IEEE Standard for Information Technology, IEEE*, vol. 802, n.º 6, pp. 1-271, 2012.

[59] A. V. Oppenheim y R. W. Schafer, *Discrete-Time Signal Processing*, 3 edition. Upper Saddle River: Pearson, 2009.

[60] A. Koski y M. Juhola, «Segmentation of digital signals based on estimated compression ratio», *IEEE transactions on Biomedical Engineering*, vol. 43, n.º 9, pp. 928–938, 1996.

[61] S. Olmos, J. García, R. Jané, y P. Laguna, «Anáalisis de la señnal ECG con expansiones ortogonales de reducido núumero de coefcientes», en *XV Congreso de la Sociedad Española de Ingeniería Biomédica*, Valencia, 1997, pp. 89-92.

[62] D. L. Donoho, «De-noising by soft-thresholding», *IEEE transactions on information theory*, vol. 41, n.º 3, pp. 613–627, 1995.

[63] J. Mateo, J. Rieta, A. Torres, y C. Sánchez, «Sistema basado en perturbación simultánea para la disminución de ruido muscular presente en el electrocardiograma», *XXIII Simpósium Nacional de la Unión Científica Internacional de Radio URSI*, 2008.

[64] R. Jané, P. Laguna, N. V. Thakor, y P. Caminal, «Adaptive baseline wander removal in the ECG: Comparative analysis with cubic spline technique», en *Proceedings Computers in Cardiology*, 1992, pp. 143–146.

[65] A. F. Quiceno Manrique, «Análisis tiempo-frecuencia por métodos no paramétricos orientado a la detección de patologías en bioseñales.», PhD Thesis, Universidad Nacional de Colombia-Sede Manizales, Colombia, 2009.

[66] F. Tedin y J. Fraire, «Procesamiento de bioseñales en tiempo real en universos interactivos», Instituto Tecnológico de Buenos Aires, Argentina, 2017.

[67] M. Akay, Ed., *Time Frequency and Wavelets in Biomedical Signal Processing*, 1 edition. Piscataway, NJ: Wiley-IEEE Press, 1997.

[68] M. Akay, *Nonlinear Biomedical Signal Processing Vol. II: Dynamic Analysis and Modeling*, 1st ed. Wiley-IEEE Press, 2000.

[69] P. S. Hamilton, «A comparison of adaptive and nonadaptive filters for reduction of power line interference in the ECG», *IEEE Transactions on Biomedical Engineering*, vol. 43, n.º 1, pp. 105–109, 1996.

[70] G. Tognola, F. Grandori, y P. Ravazzani, «Wavelet analysis of click-evoked otoacoustic emissions», *IEEE Transactions on Biomedical Engineering*, vol. 45, n.º 6, pp. 686–697, 1998.

[71] B. Cicchino y A. Noelia, «Técnicas de procesamiento de EEG para detección de eventos», Msc Thesis, Universidad Nacional de La Plata, Facultad de Ingeniería, Argentina, 2013.

[72] L. G. Doroshenkov, V. A. Konyshev, y S. V. Selishchev, «Classification of human sleep stages based on EEG processing using hidden Markov models», *Biomedical Engineering*, vol. 41, n.º 1, pp. 25–28, 2007.

[73] R. D. P. Morales, D. Á. Morales, y V. H. Grisales, «Caracterización de señales electromiográficas para la discriminación de seis movimientos de la mano», *Scientia et technica*, vol. 15, n.º 42, pp. 278–283, 2009.

[74] W. Philips y G. De Jonghe, «Data compression of ECG's by high-degree polynomial approximation», *IEEE Transactions on Biomedical Engineering*, vol. 39, n.º 4, pp. 330–337, 1992.

[75] J. Kang, C. H. Lee, y S. Kim, «EEG feature selection and the use of Lyapunov exponents for EEG-based biometrics», en *2016 IEEE-EMBS International Conference on Biomedical and Health Informatics (BHI)*, 2016, pp. 228-231.

[76] R. Nygaard y D. Haugland, «Compressing ECG signals by piecewise polynomial approximation», en *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP'98 (Cat. No. 98CH36181)*, 1998, vol. 3, pp. 1809–1812.

[77] G. M. Friesen, T. C. Jannett, M. A. Jadallah, S. L. Yates, S. R. Quint, y H. T. Nagle, «A comparison of the noise sensitivity of nine QRS detection algorithms», *IEEE Transactions on biomedical engineering*, vol. 37, n.º 1, pp. 85–98, 1990.

[78] J. Pan y W. J. Tompkins, «A real-time QRS detection algorithm», *IEEE Trans. Biomed. Eng*, vol. 32, n.º 3, pp. 230–236, 1985.

[79] P. S. Hamilton, «Open source ECG analysis software documentation», *Computers in cardiology*, vol. 2002, pp. 101–104, 2002.

[80] L. Wong y W. Abdulla, «Time-frequency evaluation of segmentation methods for neonatal EEG signals», en *2006 International Conference of the IEEE Engineering in Medicine and Biology Society*, 2006, pp. 1303-1306.

[81] S. D. Nandedkar y D. B. Sanders, «Median averaging of electromyographic motor unit action potentials: comparison with other techniques», *Med. Biol. Eng. Comput.*, vol. 27, n.º 6, pp. 566-571, nov. 1989.

[82] J. Navallas *et al.*, «An algorithm for optimal discharge selection for MUAP waveform extraction», en *XVI Congress of the International Society of Electrophysiology and Kinesiology*, 2006, pp. 12-13.

[83] J. Navallas, A. Malanda, L. Gila, J. Rodríguez, y I. Rodríguez, «Evaluación del algoritmo de promediado ordenado de descargas para la extracción de la forma de onda del PAUM», en *XXIV Congreso Anual de la Sociedad Española de Ingeniería Biomédica*, 2006, pp. 181-184.

[84] A. Malanda, J. Navallas, L. Gila, J. Rodríguez, y I. Rodríguez, «Extraction of representative potentials in MUAP sets», en *XVIIth Congress of the international society of electrophysiology and kinesiology*, 2008, vol. 54.

[85] D. Cuesta-Frau, J. C. Pérez-Cortes, G. Andreu-García, y D. Novák, «Feature extraction methods applied to the clustering of electrocardiographic signals. a comparative study», en *Object recognition supported by user interaction for service robots*, 2002, vol. 3, pp. 961–964.

[86] H. Kurniawan, A. V. Maslov, y M. Pechenizkiy, «Stress detection from speech and Galvanic Skin Response signals», en *Proceedings of the 26th IEEE International Symposium on Computer-Based Medical Systems*, 2013, pp. 209-214.

[87] V. N. Vapnik, *Statistical Learning Theory*, 1 edition. New York: Wiley-Interscience, 1998.

[88] P. Micó, «Nuevos desarrollos y aplicaciones basados en métodos estocásticos para el agrupamiento no supervisado de latidos en señales electrocardiográficas», PhD Thesis, Universidad Politécnica de Valencia, España, 2008.

[89] S. Alzate Marín, «Análisis de presión y rigidez arterial de forma no invasiva», PhD Thesis, Universidad Nacional de Colombia-Sede Manizales, Colombia.

[90] K. Sohraby, D. Minoli, y T. Znati, *Wireless Sensor Networks: Technology, Protocols, and Applications*, 1 edition. Hoboken, N.J: Wiley-Interscience, 2007.

[91] A. C. Gonzalez, «Procesamiento embebido de señales cerebrales relacionadas con la imaginación de movimientos para aplicaciones de BCI», PhD Thesis, Universidad Javeriana, Colombia, 2013.

[92] L. Hobring y P. Söderberg, «A sensor network for the Internet of Things Integrated with a bidirectional backend», Msc Thesis, Blekinge Institute of Technology, Sweden, 2014.

[93] «IEEE 802.11, The Working Group Setting the Standards for Wireless LANs». [En línea]. Disponible en: http://www.ieee802.org/11/. [Accedido: 14-mar-2019].

[94] C. LINO RAMIREZ, «Diseño de una arquitectura para redes de sensores con soporte para aplicaciones de detección de eventos», PhD Thesis, Universidad Politécnica de Valencia, España, 2012.

[95] A. Boulemtafes y N. Badache, «Design of Wearable Health Monitoring Systems: An Overview of Techniques and Technologies», en *mHealth Ecosystems and Social Networks in Healthcare*, A. A. Lazakidou, S. Zimeras, D. Iliopoulou, y D.-D. Koutsouris, Eds. Cham: Springer International Publishing, 2016, pp. 79-94.

[96] R. Chávez-Santiago, A. Khaleghi, I. Balasingham, y T. A. Ramstad, «Architecture of an ultra wideband wireless body area network for medical applications», en *2009 2nd International Symposium on Applied Sciences in Biomedical and Communication Technologies*, 2009, pp. 1-6.

[97] E. Jovanov y A. Milenkovic, «Body Area Networks for Ubiquitous Healthcare Applications: Opportunities and Challenges», *J Med Syst*, vol. 35, n.º 5, pp. 1245-1254, oct. 2011.

[98] S. Adibi, Ed., *Mobile Health: A Technology Road Map*, 2015 edition., vol. 5. New York: Springer, 2015.

[99] S. M. R. Al Masud, «Study and analysis of scientific scopes, issues and challenges towards developing a righteous wireless body area network», *International Journal Of Soft Computing And Engineering (IJSCE)*, vol. 3, n.º 2, 2013.

[100] S. M. Bari y M. M. Ahmed, «Application Protocols of Wireless Sensor Networks (WSNs) for Smart Grid Applications», *International Journal of Scientific Engineering and Technology*, vol. 5, n.º 8, pp. 421–424, 2016.

[101] I. Fielding *et al.*, «RFC 2616: Hypertext Transfer Protocol--HTTP/1.1», *Internet Engineering Task Force*, 1999.

[102] Z. Shelby, K. Hartke, C. Bormann, y B. Frank, «RFC 7252: The constrained application protocol (CoAP)», *Internet Engineering Task Force*, 2014.

[103] O. Standard, «MQTT version 3.1. 1», *URL http://docs. oasis-open. org/mqtt/mqtt/v3*, vol. 1, 2014.

[104] M. Mendoza Miranda, «Propuesta de red inalámbrica de sensores y actuadores para el hogar», Thesis, Universidad Central «Marta Abreu» de Las Villas, Facultad de Ingeniería Eléctrica, Departamento de Electrónica y Telecomunicaciones, 2017.

[105] A. Milenković, C. Otto, y E. Jovanov, «Wireless sensor networks for personal health monitoring: Issues and an implementation», *Computer Communications*, vol. 29, n.º 13, pp. 2521-2533, ago. 2006.

[106] A. Rizal, V. Suryani, J. Jondri, y S. Hadiyoso, «Development of Wireless Patient's Vital Sign Monitor Using Wireless LAN (IEEE. 802.11. b/g) Protocol», *International Journal of Electrical and Computer Engineering*, vol. 4, n.º 6, p. 893, 2014.

[107] R. P. Foundation, «Raspberry Pi — Teach, Learn, and Make with Raspberry Pi», *Raspberry Pi*. [En línea]. Disponible en: https://www.raspberrypi.org. [Accedido: 29-mar-2019].

[108] R. Light, «Mosquitto-an open source mqtt v3. 1 broker», *URL: http://mosquitto. org*, 2013.

[109] Y. Chen, W. Shen, H. Huo, y Y. Xu, «A Smart Gateway for Health Care System Using Wireless Sensor Network», en *2010 Fourth International Conference on Sensor Technologies and Applications*, 2010, pp. 545-550.

[110] R. Soni, *Nginx: from beginner to Pro*. Berkeley, CA: Apress, 2016.

[111] G. van Rossum y F. L. Drake, *The Python Language Reference Manual*. Network Theory Ltd., 2011.

[112] «Eclipse Paho - MQTT and MQTT-SN software». [En línea]. Disponible en: https://www.eclipse.org/paho/. [Accedido: 29-mar-2019].

[113] «Welcome to BioSPPy — BioSPPy 0.6.1 documentation». [En línea]. Disponible en: https://biosppy.readthedocs.io/en/v0.6.1/. [Accedido: 29-mar-2019].

[114] A. Lourenço, H. Silva, P. Leite, R. Lourenço, y A. L. Fred, «Real Time Electrocardiogram Segmentation for Finger based ECG Biometrics.», en *Biosignals*, 2012, pp. 49–54.

[115] I. I. Christov, «Real time electrocardiogram QRS detection using combined adaptive threshold», *Biomedical engineering online*, vol. 3, n.º 1, p. 28, 2004.

[116] R. K. Kodali y S. Soratkal, «MQTT based home automation system using ESP8266», en *2016 IEEE Region 10 Humanitarian Technology Conference (R10-HTC)*, 2016, pp. 1-5.

[117] J. Purdum, *Beginning C for Arduino: Learn C Programming for the Arduino*, 1st ed. edition. Berkeley, CA : New York: Apress, 2012.

ANEXOS

Anexo I Procedimiento para la configuración del servidor HTTP.

Paso	**Observación**
Instalación de Nginx	Terminal: **apt-get install nginx**
Instalación de PHP y habilitación de la conexión con SQlite	Terminal: **sudo apt-get install php5-fpm** **sudo apt-get install php5-sqlite**
Configuración de Nginx para dar soporte a los *scripts* PHP	Editar: **/etc/nginx/sites-enabled/default** Bloque *server*: **root /var/www/html;** **index index.php index.html index.htm index.nginx-debian.html;** **location ~ \.php$ {** **include snippets/fastcgi-php.conf;** **fastcgi_pass unix:/run/php/php5.0-fpm.sock;** **}**
Se prueba el archivo de configuración de errores de sintaxis	Terminal: **sudo nginx -t**
Se reinician Nginx y PHP para guardar los cambios realizados	Terminal: **sudo systemctl reload nginx** **sudo service php5-fpm restart**

Anexo II Archivos para la creación de la página web en el Raspberry Pi

var/www/html/index.html

```
<!DOCTYPE html>
<html>
<head>
   <meta charset="utf-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <title>Título</title>
   <meta content="width=device-width, initial-scale=1, maximum-scale=1, user-
scalable=no" name="viewport">
   <link rel="stylesheet" href="css/bootstrap.min.css">
   <link rel="stylesheet" href="css/font-awesome.min.css">
   <link rel="stylesheet" href="css/ionicons.min.css">
   <link rel="stylesheet" href="css/AdminLTE.min.css">
   <link rel="stylesheet" href="css/_all-skins.min.css">
   <link rel="stylesheet"

href="https://fonts.googleapis.com/css?family=Source+Sans+Pro:300,400,600,700,300itali
c,400italic,600italic">
</head>
<body class="hold-transition skin-blue sidebar-mini">
<div class="wrapper">
   <div class="content-wrapper" style="width: 100%; margin-left: 0;">
      <section class="content-header">
         <h1>
            Monitor de signos vitales
         </h1>
      </section>

      <!--este bloque es el seleccionador de opciones de paciente -->
      <section class="col-md-6 ">
         <div class="row">
            <div class="" style="margin-left: 50px; margin-top: 32px; margin-bottom: 50px">
               <label>Selecione el paciente a monitorear:</label>
               <select class="form-control" style="max-width: 200px">
                  <option>Paciente 1</option>
                  <option>Paciente 2</option>
                  <option>Paciente 3</option>
                  <option>Paciente 4</option>
                  <option>Paciente 5</option>
               </select>
            </div>
         </div>
      </section>

      <!--este bloque es el seleccionador de opciones de variables fisiologicas -->
      <section class="col-md-6 ">
         <div class="row">
            <div class="" style="margin-left: 50px; margin-top: 32px; margin-bottom: 50px">
```

```
                <label>Selecione variable fisiológica:</label>
                <select class="form-control" style="max-width: 200px">
                  <option selected>Tacograma</option>
                  <option>Variable 2</option>
                  <option>Variable 3</option>
                  <option>Variable 4</option>
                  <option>Variable 5</option>
                </select>
              </div>
            </div>
          </section>
          <section class="content">
            <div class="row">
              <div class="col-xs-12">
                <!-- bloque de graficar -->
                <div class="box box-primary">
                  <div class="box-header with-border">
                    <i class="fa fa-bar-chart-o"></i>
                    <h3 class="box-title">Tacograma</h3>
                    <div class="box-tools pull-right">
                      Real time
                      <div class="btn-group" id="realtime" data-toggle="btn-toggle">
                        <button type="button" class="btn btn-default btn-xs active" data-
toggle="on">On</button>
                        <button type="button" class="btn btn-default btn-xs" data-
toggle="off">Off</button>
                      </div>
                    </div>
                  </div>
                  <div class="box-body">
                    <div id="interactive" style="height: 300px;"></div>
                  </div>
                </div>
              </div>
            </div>
          </section>
        </div>
      </div>
      <script src="js/jquery.min.js"></script>
      <script src="js/bootstrap.min.js"></script>
      <script src="js/fastclick.js"></script>
      <script src="js/adminlte.min.js"></script>
      <script src="js/demo.js"></script>
      <script src="js/jquery.flot.js"></script>
      <script src="js/jquery.flot.resize.js"></script>
      <script src="js/jquery.flot.pie.js"></script>
      <script src="js/jquery.flot.categories.js"></script>
      <script src="js/jquery.flot.time.js"></script>
      <script>
        var interactive_plot;
        $(function () {
```

```
/*
 * Flot Interactive Chart
 * ----------------------
 */
// Este script obtiene los datos de un archivo externo y los grafica
var data = [], totalPoints = 50;

//Funcion para obtener los datos que se van a graficar
function getData() {
    $.ajax({url: "read.php", dataType: "json", success: function(result){
            $('#interactive').empty();
            interactive_plot = $.plot('#interactive', [result], {
                grid: {
                    borderColor: '#f3f3f3',
                    borderWidth: 1,
                    tickColor: '#f3f3f3'
                },
                series: {
                    shadowSize: 0,
                    color: '#3c8dbc'
                },
                lines: {
                    fill: true,
                    color: '#3c8dbc'
                },
                yaxis: {
                    min: 45,
                    max: 75,
                    show: true
                },
                xaxis: {
                    mode: 'time',
                    timeBase: "milliseconds",
                    show: true
                }
            });
            if (realtime === 'on')
                setTimeout(update, updateInterval);
            console.log(result);
        }, error: function () {
            alert('Error en el pedido a Base de Datos');
        }});
}
interactive_plot = $.plot('#interactive', [], {
    grid: {
        borderColor: '#f3f3f3',
        borderWidth: 1,
        tickColor: '#f3f3f3'
    },
    series: {
        shadowSize: 0,
```

```
            color: '#3c8dbc'
        },
        lines: {
            fill: true,
            color: '#3c8dbc'
        },
        yaxis: {
            min: 50,
            max: 70,
            show: true
        },
        xaxis: {
            mode: 'time',
            timeBase: "milliseconds",
            show: true
        }
    });
    var updateInterval = 1000; //Obtener datos cada x milisegundos
    var realtime = 'on'; //Si realtime está en on, obtener datos cada x milisegundos
    function update() {
        getData();
    }

    //Obtencion de los datos a gradicar
    if (realtime === 'on') {
        update()
    }

    //Condicion de tiempo real
    $('#realtime .btn').click(function () {
        if ($(this).data('toggle') === 'on') {
            realtime = 'on'
        }
        else {
            realtime = 'off'
        }
        update()
    });
});
</script>
</body>
</html>
```

var/www/html/read.php

```
<?php
$dir = 'sqlite:PACIENTES.DB';
$dbh  = new PDO($dir) or die("No se pudo abrir la Base de Datos");
```

```
// seleccionar registros de la base de datos
$query =  "SELECT `ECG`, `tiempo` FROM `Paciente1` WHERE id>((SELECT MAX(id) FROM `Paciente1`)-30) ORDER BY `id` ASC";
$resp = $dbh->query($query);
$jquery = array();

// acomodar los datos en forma de arreglos
foreach ($resp as $key => $row)
{
   $jquery[$key] = array(((int)$row[1])*1000, (int)$row[0]);
}
$dbh = null; //This is how you close a PDO connection
echo json_encode($jquery);
```

Anexo III *Script* de la sección de procesamiento.

home/pi/cliente_mqtt.py

```
#!/usr/bin/env python
import sys
import time
import sqlite3
from my_buffer import *
import paho.mqtt.client
from biosppy.signals import ecg
from biosppy import storage

# conexión con el objeto que representa la base de datos
temp = sqlite3.connect("/home/pi/PACIENTES.DB")

# objeto buffer cíclico y definición de variables
buf=CircularBuffer(20000)
last_time_0=0
last_time_1=0

# función para el procesamiento de los datos
def procesar():
        muestra = []
        muestra = buf.remove_elements(5000)
        if  muestra == [0]:
                return
        else:
                hr, ts_hr = ecg.ecg(signal=muestra, sampling_rate=1000., show=False)
                ts_hr=ts_hr+last_time_0
                last_time_1=ts_hr[-1]
                last_time_0=last_time_1
                # almacenamiento del resultado en la base de datos
                cursor = temp.cursor()
                for i in hr:
                        i=str(i)
                        cursor.execute("INSERT INTO Paciente1 (ECG) VALUES ("+i+")")
```

```
            for i in ts_hr:
                i=str(i)
                cursor.execute("INSERT INTO Paciente1 (tiempo) VALUES ("+i+")")
            temp.commit()

# respuesta a la solicitud de conexión del bróker
def on_connect(client, userdata, flags, rc):
    print ('connected (%s)' % client._client_id)
    client.subscribe(topic='casa/sensores/cocina/testpulse', qos=2)

# se genera cuando se recibe un mensaje del bróker
# se almacenan los datos en el buffer cíclico
def on_message(client, userdata, message):
    datos=message.payload
    datos=datos.decode("utf-8")
    sep=";"
    for i in (datos.split(sep)):
        buf.add_element(int(i))

# función principal del programa
def main():

    # parámetros de la conexión
    client = paho.mqtt.client.Client(client_id='nodo coordinador', clean_session=False)
    client.on_connect = on_connect
    client.connect(host='127.0.0.1', port=1883)
    client.loop_start()

    # consulta al buffer de recepción MQTT
    while True:
        client.on_message = on_message
        procesar()
if __name__ == '__main
```

Anexo IV *Script* para la creación del buffer cíclico.

```
class CircularBuffer:
    #variables
    def __init__(self, size):
        self.size = size
        self.buffer = [None] * size
        self.next_index = 0
        self.last_index = self.size - 1
        self.read_index =0
        self.paso = False

    # añadir datos al buffer
    def add_element(self, el):
        self.buffer[self.next_index] = el
        self.next_index += 1
```

```
        self.paso = False
        if self.next_index == self.last_index + 1:
                self.next_index = 0

# extraer datos del buffer
def remove_elements(self, quantity):
        a=[]
        b=[]
        r=[0]
        start_index = self.read_index
        tail = self.size - start_index
        remainder = quantity - tail;
        for el in range(start_index, self.size):
                a.append(self.buffer[el])
        if self.paso is True:
                return r
        elif (start_index < self.next_index and start_index + quantity > self.next_index):
                return r

        elif (remainder > 0 and (self.next_index < remainder)):
                return r
        elif remainder > 0:
                for el in range(0, remainder):
                        a.append(self.buffer[el])
                self.read_index=remainder
                if self.read_index==self.next_index:
                        self.paso = True
                return a
        else:
                for el in range(start_index, start_index + quantity):
                        if self.buffer[el] == None:
                                return r
                        else:
                                b.append(self.buffer[el])
                self.read_index=start_index + quantity
                if self.read_index==self.next_index:
                        self.paso = True
                return b
```

Anexo V Procedimiento para la instalación de Python 3.6 en el Raspberry Pi.

Paso	Observación
Instalar las herramientas de construcción requeridas	**sudo apt-get update** **sudo apt-get install build-essential tk-dev libncurses5-dev libncursesw5-dev libreadline6-dev libdb5.3-dev libgdbm-dev libsqlite3-dev libssl-dev libbz2-dev libexpat1-dev liblzma-dev zlib1g-dev**
Descargar la versión más reciente de Python 3.6, disponible en el sitio oficial	Terminal: **wget https://www.python.org/ftp/python/3.6.5/Python-3.6.5.tar.xz**

Anexo VI Procedimiento para la instalación de la librería *scipy*.

Paso	Observación
Instalación de los requisitos de paquetes Python	Terminal: **sudo pip3.6 install numpy** **sudo pip3.6 install cython** **sudo pip3.6 install setuptools**
Instalación de los requisitos del sistema	Terminal: **sudo apt-get install libblas-dev liblapack-dev** **sudo apt-get install python3libatlas-base-dev gcc gfortran g++**
Aumentar la memoria de intercambio. El *script* de compilación ocupa la totalidad de la memoria RAM y de la memoria de intercambio al generar múltiples procesos	Terminal: **sudo /bin/dd if=/dev/zero of=/var/swap.1 bs=1M count=1024** **sudo /sbin/mkswap /var/swap.1** **sudo chmod 600 /var/swap.1** **sudo /sbin/swapon /var/swap.1**
Instalación desde GitHub	Terminal: **git clone https://github.com/scipy/scipy.git** **cd scipy** **git clean -xdf** **sudo python3.6 setup.py install**
Se elimina el espacio de intercambio agregado y se restaura el predeterminado	Terminal: **sudo swapoff /var/swap.1** **sudo rm /var/swap.1**

Anexo VII Modificaciones realizadas al módulo *signal.ecg*.

Modificaciones	Observación
global resto_0 **global resto_1**	Se crean variables temporales para almacenar las muestras no analizadas.
resto_0 = len (signal) – rpeaks [-1]	Se almacenan las muestras posteriores al último pico R detectado.
for i in range (len (rpeaks)): ** rpeaks [i] = rpeaks [i] + resto_1** **if resto_1 !=0:** ** rpeaks = np.insert (rpeaks, 0, 1)** **else:** ** pass** **resto_1 = resto_0**	Se desplaza la posición de los picos R en el segmento analizado y se inserta el latido correspondiente.
length = len (signal) + resto_1	Se modifica el vector de tiempo según el latido añadido.

Anexo VIII Sketch para la programación del módulo ESP-01 del nodo sensor

```
// transmite wifi segmentos de datos recibidos vía serial desde una app (Matlab)
// SECCIÓN DE CONFIGURACIÓN
// Definicion de Bibliotecas
#include <ESP8266WiFi.h>    // componentes de manejo de la WiFi
#include <PubSubClient.h>   // cliente MQTT

// Definición de constantes
// constantes MQTT
const char* client_id = "nodo_inalámbrico_1";           // ID del cliente MQTT (debe ser unico en la red)
const char* pub_testpulse = "hospital/sala1/paciente1";
const long interval = 1000;                             // intervalo de tiempo entre lecturas en milisegundos (un segundo en este caso)

// constantes WiFi
const char* ssid = "raspi_ap";                  // SSID de la red
const char* password = "raspberrypi";           // contraseña de la red
const char* mqtt_server = "10.5.5.1";       // direccion IP o nombre del servidor MQTT

// Definición de variables
unsigned long testpulse = 0;
bool firstTime = true;                  // indica si es la primera vez que se ejecuta la funcion loop()
unsigned long previousMillis = 0;       // guarda el tiempo de la ultima lectura

// global variables
byte i = 0;
```

```
WiFiClient espClient;
PubSubClient client(espClient);
String cadena ="";

// función de configuración
void setup() {
  Serial.begin(115200);                    // establecer la velocidad de la conexion Serial
  setup_wifi();                            // configuracion de la WiFi
  client.setServer(mqtt_server, 1883);  // definición del broker MQTT
  client.setCallback(callback);           // definición de la funcion para manejar los mensajes
recibidos
}

/////// SECCIÓN DE EJECUCIÓN CONTINUA
void loop() {
  // si no hay conexión con el broker se llama a la funcion reconnect()
  if (!client.connected()) {
    reconnect();
  }
  client.loop();
  if (Serial.available()) {
    i++;
    // lectura de segmento a transmitir
    cadena = Serial.readString();
    delay(1);
    Serial.print("Dato recibido!!! ");
    // publicacion de datos por MQTT
    Serial.print("Tx bloque: ");
    Serial.println(i);
    //Serial.println(cadena);
    client.publish(pub_testpulse,cadena.c_str());
  }

/// fin de sección de ejecución continua

/////       SECCIÓN DE FUNCIONES LOCALES  /////
// función de configuración de la WiFi
void setup_wifi() {
  delay(10);

  // definición del modo STA (estación) solamente
  WiFi.softAPdisconnect();
  WiFi.disconnect();
  WiFi.mode(WIFI_STA);
  delay(100);

  Serial.println();
  Serial.print("Conectando a: ");
  Serial.println(ssid);

  // establecimiento de la conexion
```

```
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("");
  Serial.println("Conectado a la WiFi");
  Serial.print("dirección IP: ");
  Serial.println(WiFi.localIP());
}

// función de conexión con el broker MQTT
void reconnect() {
  // si no hay conexión se intenta conectarse
  while (!client.connected()) {
    Serial.print("Intentando conectar con MQTT...");
    // intento de conexión
    if (client.connect(client_id)) {
      Serial.println("conectado!!!");
    } else {
      Serial.print("failed, rc= ");
      Serial.print(client.state());
      Serial.println(" intente de nuevo en 5 segundos");

      // esperar 5 segundos antes de reintentar
      delay(5000);
    }
  }
}

// función que maneja los mensajes recibidos
void callback(char* topic, byte* payload, unsigned int length) {
  Serial.print("Message arrived [");
  Serial.print(topic);
  Serial.println("] ");
}
```

Anexo IX Programa en Matlab para transmisión de datos hacia la plataforma mediante el módulo ESP-01.

```
function varargout = Mesp01(varargin)

% etapa de inicialización
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Mesp01_OpeningFcn, ...
                   'gui_OutputFcn',  @Mesp01_OutputFcn, ...
```

```
                'gui_LayoutFcn',  [] , ...
                'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end

function Mesp01_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = hObject;

hwINFO = instrhwinfo('serial');
handles.portlist.String = hwINFO.SerialPorts;
handles.start.Enable = 'off';
guidata(hObject, handles);

function varargout = Mesp01_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;

function start_Callback(hObject, eventdata, handles)
load ecg   % datos en la variable ecg_array de 15000 muestras
handles.log.String = 'ECG data read';
handles.log.String = 'Connection started';
ind=0;
try
    % lazo de escritura
    while ~handles.stop.Value ;
        % esperar por disponibilidad del puerto
        while handles.figure1.UserData.status ~= 'open'; end
        indices = [1:1000]+1000*mod(ind,3); % 3 : cantidad de segmentos a repetir
        cadena = num2str(ecg_array(indices),'%d;');
        cadena = cadena(1:length(cadena)-1);
        % escritura de datos hacia el modulo ESP-01
        fwrite(handles.figure1.UserData,cadena);
        pause(1);
        % lectura de posible mensaje a enviar por el ESP-01
        if handles.figure1.UserData.BytesAvailable > 0
            handles.log.String = ['Bytes Available: '
num2str(handles.figure1.UserData.BytesAvailable)];
            handles.log.String = ['Block: ' num2str(ind) ' Indices: ' num2str(min(indices)) ' to '
num2str(max(indices))];
            %v_read = fscanf(ESPserial,'*',ESPserial.BytesAvailable);
            v_read =
fread(handles.figure1.UserData,handles.figure1.UserData.BytesAvailable);
            handles.log.String = char(v_read)'
        end
        ind=ind+1;
```

```
    end
catch
    handle.log.String = ['Could not sent data to ESP01. Connection failed'];
end

function stop_Callback(hObject, eventdata, handles)
try
    fclose(handles.figure1.UserData);
    delete(handles.figure1.UserData);
    clear handles.figure1.UserData;
    handles.log.String=[handles.portlist.String{handles.portlist.Value} ' port is Closed']
catch
    handles.log.String=[handles.portlist.String{handles.portlist.Value} ' port not available!!!'];
end

function portlist_Callback(hObject, eventdata, handles)
oldSerial = instrfind('Port', handles.portlist.String{handles.portlist.Value})
if (length(oldSerial)>0)
    delete(oldSerial);
    clear oldSerial;
end
try
    handles.figure1.UserData =
serial(handles.portlist.String{handles.portlist.Value},'BaudRate',115200,'DataBits',8);
    set(handles.figure1.UserData,'OutputBufferSize',10000);
    fopen(handles.figure1.UserData);
    pause(1)
    handles.log.String = ['Port ' handles.portlist.String{handles.portlist.Value} ' is ready'];
    handles.start.Enable = 'on';
catch
    handles.log.String = (['Port ' handles.portlist.String{handles.portlist.Value} ' is not
available!!!']);
end

function portlist_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function log_Callback(hObject, eventdata, handles)

function log_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function figure1_CloseRequestFcn(hObject, eventdata, handles)
try
```

```
    fclose(handles.figure1.UserData);
    delete(handles.figure1.UserData);
    clear handles.figure1.UserData;
    handles.log.String=[handles.portlist.String{handles.portlist.Value} ' port is Closed']
catch
    handles.log.String=[handles.portlist.String{handles.portlist.Value} ' port not available!!!'];
    disp(['Could not close port ' handles.portlist.String{handles.portlist.Value}])
end
delete(hObject);
```