ugr | Universidad de Granada

TRABAJO FIN DE GRADO

INGENIERÍA EN TECNOLOGÍAS DE LA TELECOMUNICACION

# Utilización de señales biomédicas para la generación de claves privadas de encriptación de seguridad.

---


**Autor**

Guillermo Pérez Alba

**Directores**

Juan Francisco Valenzuela Valdés

Pablo Padilla de la Torre





Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación


—

Granada, junio de 2017


# Utilización de señales biomédicas para la generación de claves privadas de encriptación de seguridad.

---

**Autor**
Guillermo Pérez Alba

**Directores**
Juan Francisco Valenzuela Valdés
Pablo Padilla de la Torre

# Utilización de señales biomédicas para la generación de claves privadas de encriptación de seguridad.

Guillermo Pérez Alba

**Palabras clave**: IoT, RNG, BAN.

## Resumen

La llegada de las tecnologías IoT (Internet of Things) están haciendo que en los últimos años se haya incrementado considerablemente el número medio de dispositivos interconectados por persona. Un buen ejemplo son los dispositivos "wearable" y dispositivos necesarios para las redes BAN (Body Area Networks). Gran parte del aumento se debe a este tipo de dispositivos, ya que la mayoría de sus tareas las realizan de manera autónoma. Al no necesitar atención humana para estar en uso, los utilizamos sin ser conscientes, lo que provoca este aumento.

La principal ventaja de este tipo de dispositivos y el mayor atractivo que la industria ve en ellos es esta actividad en "segundo plano". Permite la automatización de tareas y la monitorización de señales biométricas, tanto para uso médico como para uso personal.

Estos dispositivos presentan los problemas propios de cualquier dispositivo inalámbrico. Por un lado, problemas relacionados con la seguridad del dispositivo y de la información que transmite. Se trata de información sensible que no debe caer en manos de un tercero, y también por el hecho de que un uso malicioso o indeseado de este tipo de dispositivos puede suponer graves problemas al usuario. Es por esto que se trata de problemas intolerables e inadmisibles. Por otro lado, pero no menos importantes, problemas relacionados con el coste de fabricación y con el hecho de que un dispositivo inalámbrico tiene un uso que viene limitado por su batería. Por lo tanto son también temas a tener en cuenta el bajo coste de fabricación y el coste computacional de los dispositivos.

En este trabajo se propone una solución que intenta solucionar los problemas mencionados anteriormente. Se basa en el uso de las señales biométricas, que estos dispositivos suelen recoger, como RNGs (Random Number Generators). Estas secuencias de números aleatorios generadas se utilizarán en la seguridad de las comunicaciones de estos dispositivos ya que servirán como claves para el cifrado de los mensajes. El bajo coste de los sensores que recogen este tipo de señales biométricas, junto con un procesado de coste computacional muy bajo, hace que los problemas mencionados queden en mayor o menor medida solucionados.

# Utilization of biomedical signals to generate private keys for security encryption.

Guillermo Pérez Alba

**Keywords**: IoT, RNG, BAN.

## Abstract

The appearance of IoT (Internet of Things) technologies is increasing considerably the number of interconnected devices per person in the last years. The best example of IoT technology are wearable devices and devices related to BAN (Body Area Network) technology. Those devices have an autonomous activity. The fact that people can use them without being conscious is one of the main reasons of this increase.

One of the main advantages and the appeal to industry of those devices is the fact that people can programme a lot of activities so that the device does it by itself. Another of the main advantages is the monitoring of medical signals. The industry focus on personal and medical use.

Wireless devices have some problems in common. On the one hand, problems related to the security of the device and of the information it sends. Sometimes it sends very sensitive information so it is inadmissible that a third person could have access to this information or use it in a malicious way. On the other hand, problems related to the manufacturing and to the battery life of the devices. Those devices aren't vital for human life so people are going to buy them only if they have a reasonable price, so the manufacturing must be as cheaper as possible. The battery life must be as long as possible so it is needed a very low computational cost inside the devices.

This research work proposes a possible solution for the above mentioned problems. The proposed solution is based on the use of biometrical signals as RNGs (Random Number Generators). A RNG generates random sequences of bits that can be used in the security and ciphering of the communications of the devices. The mentioned sequences are going to be the coding keys of the communication. Sensors for those kind of biometrical signals are very cheap and a lot of those devices already have those sensors installed for another purpose. Cheap sensors combined with a simple algorithm to process the signal so it can be a random sequence, is the solution proposed in this research work that tries to solve the mentioned problems.

Yo, **Guillermo Pérez Alba**, alumno de la titulación GRADO EN INGENIERIA DE TECNOLOGIAS DE LA TELECOMUNICACION de la **Escuela Técnica Superior de Ingenierías Informáti- ca y de Telecomunicación de la Universidad de Granada**, con DNI 26050711Z, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Grado en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Fdo: Guillermo Pérez Alba

Granada a 18 de junio de 2017 .

D. **Juan Francisco Valenzuela Valdés**, Profesor del área de Ingeniería Telemática del Departamento de Teoría de la Señal, Telemática y Comunicaciones de la Universidad de Granada.

D. **Pablo Padilla de la Torre**, Profesor del área de Teoría de la Señal y Comunicaciones del Departamento de Teoría de la Señal de la Universidad de Granada.

## Informan:

Que el presente trabajo, titulado ***Utilización de señales biomédicas para la generación de claves privadas de encriptación de seguridad*** ha sido realizado bajo su supervisión por **Guillermo Pérez Alba**, y autorizamos la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden y firman el presente informe en Granada a 18 de junio de 2017 .

## Los directores:

**Juan Francisco Valenzuela Valdés** **Pablo Padilla de la Torre**

# Agradecimientos

Me gustaría agradecer principalmente a mi tutor, Juan Francisco, su ayuda y dedicación en cada una de las tutorías además de su disponibilidad en cualquier momento para ayudarme.

También agradecer a mis compañeros Álvaro y Manuel su ayuda en momentos puntuales no sólo durante el proyecto, sino durante toda la etapa universitaria.

Sobre todo agradecer a mis padres Manuel y Encarnación por haberme dado la oportunidad de estudiar.

Por último agradecer a Pablo Padilla, Sandra Sendra y Jesús Minguillón su colaboración prestada en momentos del proyecto.

# Índice general

# Índice de figuras

Capítulo 1

# Introducción y objetivos

Las primeras líneas de esta memoria sirven de toma de contacto con el tema. De la misma manera el contexto hace notar la importancia de las investigaciones centradas en la generación de secuencias aleatorias. Es necesario ver en qué punto se sitúan estas investigaciones y qué líneas de estudio han seguido.

Una vez se tenga claro el contexto, se expondrá el objetivo real del trabajo además de las motivaciones que han llevado a realizar este trabajo de investigación.

## 1.1 Contexto

Realmente no se es consciente de la importancia que en el día a día tienen los RNG (*Random Number Generator).* Realizan su trabajo en segundo plano y no se tiene una interacción directa con ellos. Principalmente desempeñan su papel en tareas relacionadas con la ciberseguridad(generación de claves para encriptación). Sin embargo esta no es la única aplicación que tienen. También son participes de la aleatoriedad en la simulación de entornos reales en estudios e investigaciones, o en el intento de proporcionar la mayor realidad posible en videojuegos y juegos de azar online.

La población actual está siendo espectadora de la evolución y el avance del IoT (*Internet of Things)*. Cada vez son más los dispositivos interconectados a través de internet. En la figura 1.1 se puede observar su avance en tan sólo unos cuantos años y cómo se prevé que estará en el 2020.

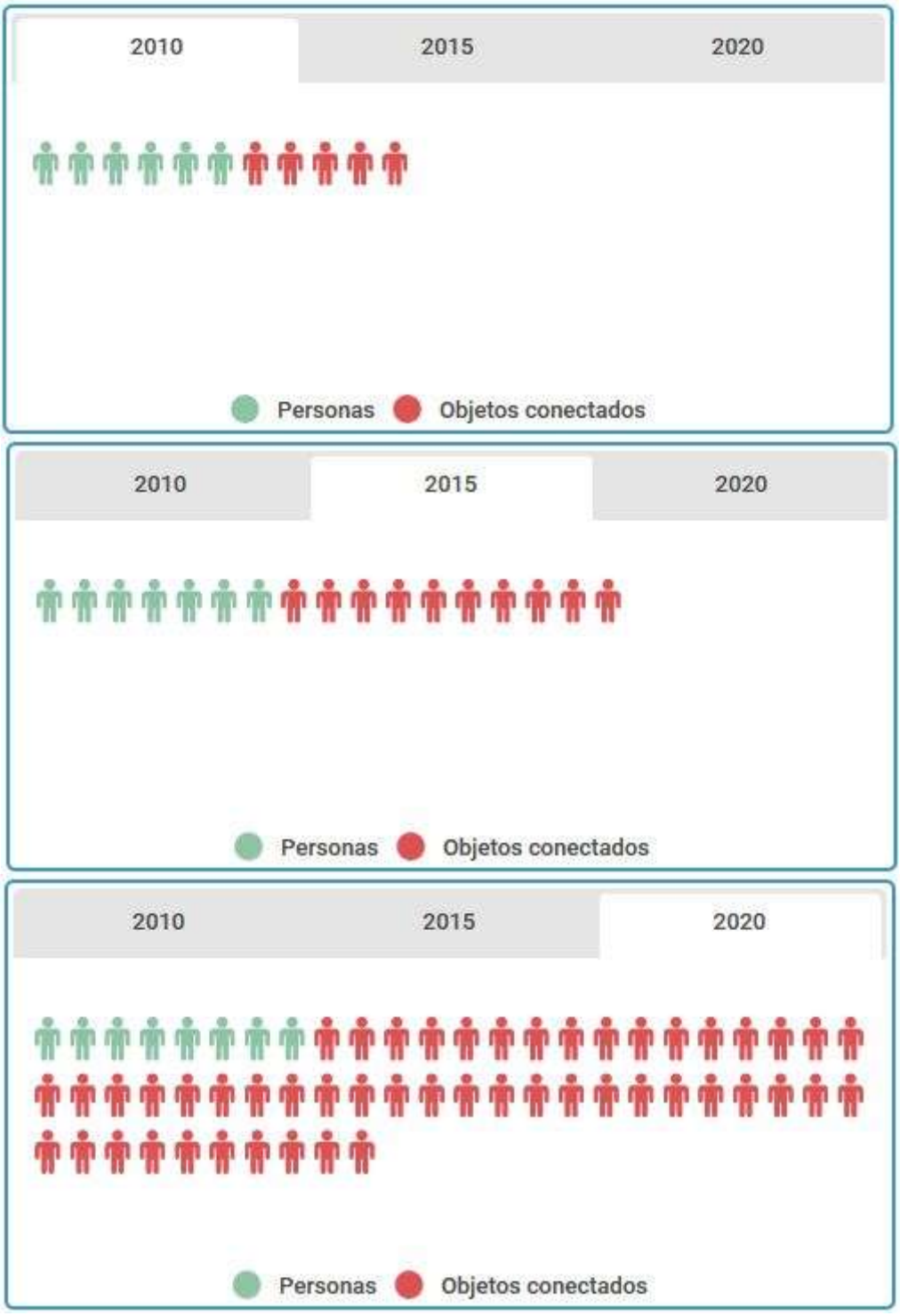


*Figura 1.1: Avance del Internet of Things. Datos obtenidos de [1].*

Todos los focos están puestos en la importancia de su crecimiento y de lo que esto involucra. Cada vez se ceden más tareas a estos dispositivos, pero no se es consciente de lo sensibles que pueden llegar a ser esas tareas. Cuanto más importantes sean las tareas y más sensible sea la información con la que trabajan estos dispositivos, mayor debe ser la seguridad de las comunicaciones a través de las cuáles se le da órdenes.

Sin embargo, aunque la necesidad de estos RNG es bastante evidente, aún no se dispone de RNGs "reales" o perfectos que integrar en estos dispositivos. En cambio se dispone de lo que se conoce como PRNG (*Pseudo-random Number Generator)*. Básicamente es un algoritmo que a través de una "semilla" que se le proporciona, genera una secuencia pseudo-aleatoria. El problema de estos PRNGs es que si un tercero tiene acceso a esta semilla, puede obtener la secuencia completa generada por el PRNG. Más que una solución, estos PRNGs siguen siendo una posible brecha de seguridad. Para paliar esta vulnerabilidad se ha intentado utilizar como semilla señales provenientes de sensores instalados en estos dispositivos. Sensores relacionados con el comportamiento humano como los acelerómetros. El comportamiento humano es más predecible de lo que parece, por lo que hay que ser muy cuidadoso en la elección de estos sensores. [2]

## 1.2 Motivación y objetivos

Si se tiene en cuenta el contexto anterior, es evidente que la motivación para este trabajo no es solamente encontrar una fuente de semillas que puedan funcionar de manera fiable para un PRNG. La motivación real es encontrar distintas señales de las que directamente se obtenga una secuencia realmente aleatoria. Es decir, en lugar de un PRNG con todas las vulnerabilidades citadas antes, se obtendrían directamente señales que funcionan como RNG.

Visto cuál va a ser el objetivo de este trabajo, se va a pasar a ver qué tipo de señales se han elegido para trabajar con ellas. Sobre todo, aclarar la motivación que lleva a elegir este tipo de señales y no otras.

La tendencia de dar conectividad a todos los objetos provocada por el IoT, ha hecho que tanto la ropa como los accesorios (relojes, gafas, colgantes, etc.) sean parte de estos avances. Es entonces cuando nacen los *wearables.* Prendas de ropa o accesorios *smart* a los que se les ha incorporado algún tipo de dispositivo electrónico y que gracias a su interacción con el usuario va a tener alguna función específica. Son dispositivos que se llevan equipados el día entero normalmente. El intercambio de mensajes con estos tiene que tener alguna encriptación que lo haga seguro, ya que es evidente el riesgo que se corre si el control de un *wearable* cae en manos de un tercero. Para aprovechar que se lleven equipados siempre y cumplir con la seguridad, la mejor elección es una señal biométrica. Muchos de estos *wearables* tienen como función obtener este tipo de señales para seguimiento médico, rendimiento físico, etc. Por lo tanto la posibilidad de que estas mismas señales puedan ser utilizadas para encriptar los mensajes que estos transmiten y reciben sería una solución muy óptima.

En el caso de que algún tipo de *wearable* no tenga ningún sensor para las señales que se van a utilizar, incluir alguno de esos sensores no supone un coste desorbitado en comparación con el beneficio que proporciona.

No solamente serían útiles en este tipo de dispositivos, por ejemplo hay una gran cantidad de dispositivos electrónicos que se pueden encontrar en la habitación de un

hospital. Teniendo en cuenta esto, y la necesidad en muchos casos de tener gran cantidad de pacientes en constante observación y monitorización, se vuelve a necesitar un intercambio de mensajes y datos entre varios dispositivos que tienen que tener algún tipo de encriptación y seguridad. En caso contrario se estaría poniendo en peligro la salud del paciente. En este caso estas señales biométricas vuelven a ser una de las mejores opciones para la generación de secuencias aleatorias que actúen como claves.

# Capítulo 2

# Estado del arte

Es un hecho que todos los intercambios de información que se realizan a través de un medio compartido son relativamente seguros, o al menos se pretende que así sean.

La figura 2.1 recoge datos del último informe proporcionado por el Centro Criptográfico Nacional (CCN), organismo estatal que depende del Centro Nacional de Inteligencia (CNI). Se aprecia que a la par que avanza la tecnología avanzan todo tipo de ataques y amenazas, lo que hace que la ciberseguridad sea tema de vital importancia. Se observa en la figura 2.1, que según datos del CCN este tipo de incidencias han aumentado en los últimos años, incrementándose en un 150% en el periodo de 2012 a 2013 y en un 78% de 2013 a 2014 [1].


*Figura 2.1: Incremento de ciberataques en España. [1]*

La figura 2.2 muestra como en efecto el incremento se produce tanto en ataques y amenazas como en dispositivos interconectados.

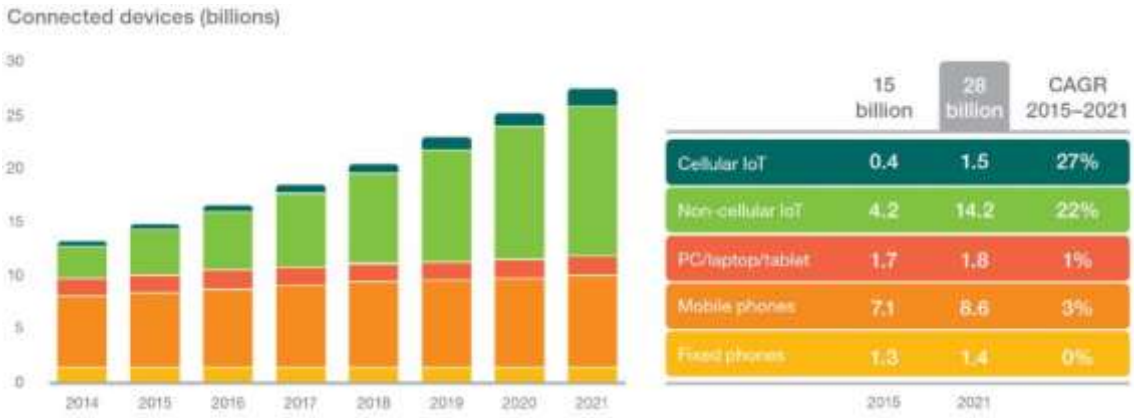


*Figura 2.2. Incremento objetos conectados [3].*

Una actitud muy común ante los riesgos de ciberseguridad existentes es pensar que: "eso no me va a pasar a mi…". Se suele pensar que los únicos objetivos de estos ataques son grandes empresas u organismos, pero lo cierto es que afecta a todos y cada uno de los usuarios de internet.

Se observa en la figura 2.3 una clasificación de los ataques gestionados en España durante 2014:

- Código dañino: troyanos, spyware, etc.
- Intrusiones: ataques dirigidos a explotar vulnerabilidades e introducirse.
- Recogida de información: primeros pasos para una campaña mayor (vulnerabilidades, ingeniería social).
- Seguridad de la información: violaciones de políticas de seguridad.
- Contenido abusivo: contra la imagen.
- Disponibilidad: daños de imagen y productividad (rendimiento).
- Fraude: propiedad intelectual, protección de datos o suplantación de identidad (phising).

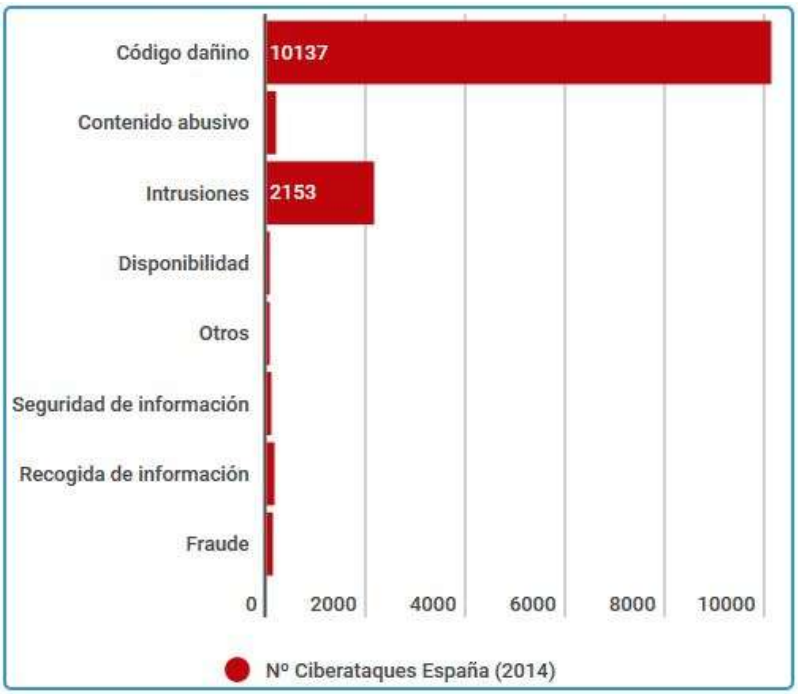


*Figura 2.3: Clasificación tipología ciberataques España (2014). [1]*

El CCN organiza todos los años las Jornadas STIC CCN-CERT. Tienen como objetivo tanto reunir a expertos de la materia como la concienciación social. La sociedad española es cada vez más consciente de la importancia de la ciberseguridad, y esto se ve reflejado en la asistencia recogida en estas jornadas del CCN. Se ve en la figura 2.4 que se han incrementado considerablemente de 2013 a 2014.

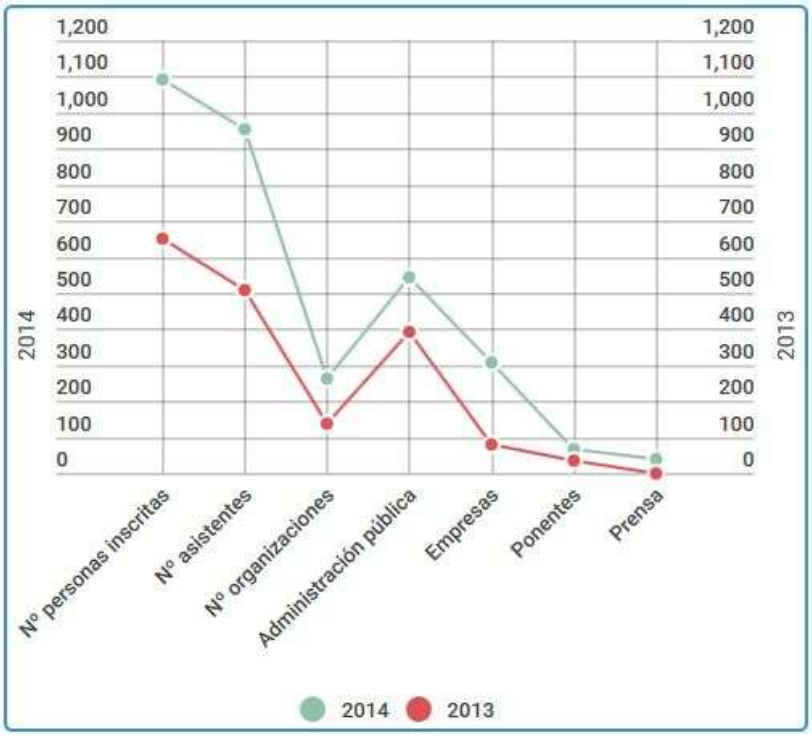


*Figura 2.4: Asistencia Jornadas STIC CNN-CERT. [1]*

A través de la encriptación de los mensajes que se intercambian en las comunicaciones, se pretende que la información sensible que se intercambia no pueda ser obtenida por un tercero que no sea el receptor de la misma. Para ello se hace uso de herramientas criptográficas que combinan criptosistemas de clave pública (con los que se establece una clave de sesión), y criptosistemas de clave privada (con los que se transmiten los mensajes de forma privada).

Se necesita que estas claves públicas y privadas sean seguras para que de ninguna manera un tercero pueda conocerlas. En caso de conocerlas, no sólo tendría acceso a las claves, sino también a toda la información encriptada por estas. Esto haría todo el proceso inútil. Para evitar que un tercero obtenga las claves se debe aumentar el espacio de búsqueda al máximo posible, haciendo igual de probables todos los valores y reduciendo la predictibilidad. Esto se asemeja bastante a la definición de aleatoriedad. Se ve como todo el proceso de seguridad de comunicaciones nace con las secuencias de números aleatorios, muy importantes para cifrado, firma digital, generación de claves para cifrado, etc.

Los generadores de estas secuencias de números aleatorios pueden ser de dos tipos. Según sea su origen se distinguen los deterministas y los no deterministas. Los deterministas, llamados PRNG, se basan en un algoritmo que hay que alimentar con una *semilla* para que generen esa secuencia de números pseudoaleatorios. Los no deterministas se conocen como RNG y toman como fuente de aleatoriedad eventos o comportamientos de la naturaleza o el cuerpo humano que son una fuente de azar. [4]

Hasta el momento en la categoría de PRNGs se ha investigado bastante y entre los algoritmos más conocidos se encuentran algunos como:

*Blum Blum Shub (BBS)*, algoritmo que genera bits con la fórmula:

$$x_i = x_{i-1}^2 mod(N)$$

Su seguridad se basa en el coste computacional de factorizar N al ser muy grande ya que se viene de multiplicar dos primos grandes P y Q. [5]

Otros como son *XOR-shift* o *LFSR* se basan en corrimientos de bits, funciones XOR y retroalimentación lineal. La principal característica de estos es su facilidad de implementación hardware. [4]

Se abre un abanico muy amplio de posibilidades cuando se pasa a hablar de RNGs. El motivo es que estos utilizan como fuente de aleatoriedad cualquier evento de la naturaleza, comportamiento humano, elementos software, señales biométricas, electromagnéticas o cualquier otra que tenga un comportamiento azaroso.

Desintegración radioactiva, ruido térmico, turbulencia del aire en un sistema físico, temperatura, presión… son algunos de los orígenes de señales provenientes de la naturaleza. [4]

De elementos software se obtienen algunas como movimiento del ratón o la señal

proporcionada por el acelerómetro en algún dispositivo.

Este trabajo se centra principalmente en las señales biométricas, señales obtenidas por sensores corporales. Una de las más estudiadas es la señal del *electrocardiograma (ECG)*. Señal que se estudia en este trabajo y proporciona una solución para ser usada como RNG. Otra que se ha usado en numerosas investigaciones es la señal del *electroencefalograma (EEG)*. Señal que también se estudia en este trabajo, pero desde un punto de vista algo distinto al de las investigaciones que hay hasta el momento. Los estudios más repetidos miden las EEGs en un paradigma concreto y en este trabajo el RNG funciona con EEGs medidas en cualquier paradigma.

## 2.1 Criptografía

Tras ver cómo se han obtenido hasta ahora esas secuencias de números aleatorios, sólo queda ver de qué forma se utilizan en la actualidad esas claves, la manera de intercambiarlas, etc.

Se entiende como *Criptografía* al estudio de métodos para el envío de mensajes a los que se les ha hecho algún tipo de transformación de modo que el único que puede leerlo correctamente será su verdadero receptor.

Se denomina *texto plano* al mensaje original que el emisor quiere enviar. *Texto cifrado* al texto plano que ya ha sufrido alguna transformación de modo que no puede ser leído sin eliminar esa transformación. *Cifrado* será el nombre que reciba esa citada transformación que sufre el texto plano y lo convierte en texto cifrado. El *descifrado* será en este caso el proceso que se le realiza al texto cifrado para volverlo a transformar en texto plano y que así sea legible por el receptor. Tanto el cifrado como el descifrado se harán con una *función de cifrado*, y cuyo resultado va a depender de las claves *k* y *k*. [6].

Estos son los elementos que intervienen en un *esquema de cifrado*:

- Al mensaje o texto plano que se quiere enviar se le llama *M*.
- A las funciones de cifrado se las $f$ y $f^{-1}$.
- Una vez se ha cifrado M, se obtiene el texto cifrado, *C*.
- Se denominan *k* y *k'* al par de claves que usan las funciones de cifrado.

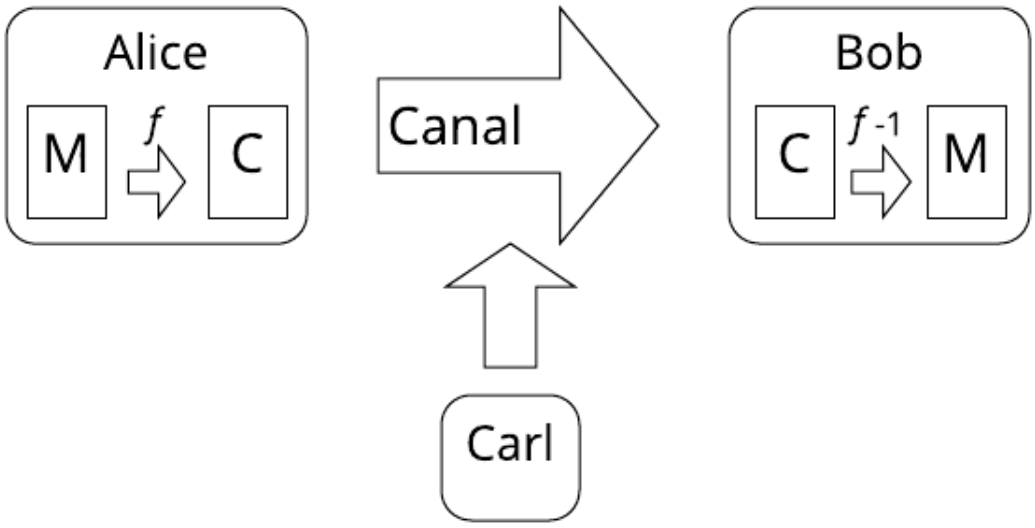


*Figura 2.5: Diagrama cifrado, descifrado y transmisión de mensaje (M).*

Durante todo el proceso de explicación de las distintas técnicas de criptografía aparecen tres individuos llamados Alice, Bob y Carl. Alice quiere enviarle un mensaje a Bob, pero no quiere que Carl, que pretende interceptar la información, vea ese mensaje. Por lo tanto, cada vez que se nombre a Alice, se entenderá que se trata del emisor del mensaje. Bob será el receptor del mensaje enviado por el emisor, y Carl será un tercero que intentará capturar la información dirigida a Bob [6].

### 2.1.1 Cifrado simétrico

Las funciones de cifrado transforman el texto plano en un texto cifrado de acuerdo con una clave *k*. El objetivo es que durante la transmisión del mismo este sea ilegible, y que sólo pueda ser leído por alguien que conozca la clave *k'* y le realice la función de descifrado.

En el cifrado simétrico ambas claves son exactamente iguales (*k=k'*), por lo tanto antes de comenzar a intercambiarse mensajes cifrados, ambos tienen que haberse puesto de acuerdo para utilizar la misma clave. Este proceso por el que se ponen de acuerdo debe realizarse con especial precaución, ya que si Carl consigue de alguna manera la clave *k*, podría descifrar todos los mensajes que Alice envíe a Bob.

### 2.1.2 Cifrado asimétrico

En el cifrado asimétrico, tanto Alice como Bob van a tener dos claves cada uno. Una clave privada ($k_{privada}$) y una clave pública ($k_{pub}$). Un mensaje cifrado con la clave pública de un usuario, únicamente va a poder ser descifrado correctamente con la clave privada del usuario destinatario. Tanto Alice como Bob van a tener una clave pública que va a ser conocida por todos (incluso por Carl) para que todo el que quiera pueda

mandarle un mensaje cifrado y asegurarse de que el único que va a poder leer el mensaje va a ser el verdadero destinatario. La clave privada de cada uno va a ser conocida sólo por ellos mismos y será la que pueda descifrar los mensajes cifrados con su clave pública.

Una transmisión con cifrado asimétrico seguiría los siguientes pasos:

- Alice cifra el mensaje (*M*) con la clave pública de Bob ($k_{privB}$) y obtiene *C*.
- Se transmite $C= k_{privB}(M)$.
- Bob recibe *C* y lo descifra con su clave privada, $k_{pubB}(k_{privB}(M))=M$.

Con este cifrado se garantiza que el receptor del mensaje sea el deseado, pero hay otros muchos problemas como son la autenticación del emisor, el no repudio y similares que son solucionados incluyendo la firma digital en estos cifrados. Sin embargo no se va a profundizar más en esto. Lo que se quiere hacer ver con la explicación del cifrado de mensajes es el hecho de que cada dispositivo que quiera formar parte de una comunicación segura va a tener que generar claves, que en realidad son secuencias de números aleatorios generados por PRNGs o RNGs. La seguridad del intercambio de mensajes va a depender directamente de la "calidad" de su PRNG o RNG.

## 2.2 Números aleatorios.

### 2.2.1 Introducción

Desde la antigüedad el ser humano ha convivido con fenómenos naturales a los que intentaban dar respuesta o comprender. Por ello el azar y la predicción de estos acontecimientos han sido siempre buscados. Herramienta fundamental para estas y otras necesidades del ser humano son los números aleatorios, generados en un primer momento con métodos muy rudimentarios y que, a la vez que el conocimiento humano, han ido avanzando y mejorando.

### 2.2.2 Historia

Fueron muchos los matemáticos que interesados en cuantificar la naturaleza. Se preguntaban cuáles eran las probabilidades de que ciertos fenómenos ocurriesen. Compartían sus ideas y teorías hasta que en el siglo XIX, Pierre Simon formulase la primera teoría general de la probabilidad. Buenos resultados fueron obtenidos de la aplicación de la teoría de la probabilidad en juegos de azar y estudios. Una vez se sabía medir esta aleatoriedad, se sabía cómo generarla [7].

### 2.2.3 ¿Qué son los números aleatorios?

Se considera *número aleatorio* a todo aquel escogido al azar dentro de un conjunto de números posibles. Todos los elementos del conjunto tienen la misma probabilidad de ser escogidos y su elección no ha sido influenciada por ninguna elección anterior ni influenciará a ninguna elección futura.

Los requisitos que debe cumplir una secuencia de números para ser aleatoria se recogen en las siguientes características [7]:

- Secuencias no correlacionadas: Dada una secuencia aleatoria, no se deben encontrar partes o subsecuencias de esta que estén relacionadas entre sí.

- Independencia estadística y equiprobabilidad: Todos los elementos del conjunto deben tener la misma probabilidad de aparecer en la secuencia. Esta aparición no debe excluir la futura aparición de cualquier otro elemento.

- Período máximo: Los generadores de números aleatorios suelen ser cíclicos, por lo que la situación idílica es que cada elemento del conjunto de elementos posibles aparezca exactamente una vez antes de que se vuelva a repetir el ciclo. Nada debe depender del valor inicial o *semilla*.

- Uniformidad: La función densidad de probabilidad de la sucesión de números aleatorios debe "adaptarse" a la función densidad de probabilidad de la variable aleatoria uniforme, $U(r, a=0, b=1)$.

- Eficiencia: El gasto computacional debe ser tal que no influya en los resultados de modo que el generador debe producir los mismos resultados sea cual sea el computador donde se ejecute.

### 2.2.4 Números aleatorios en ordenadores.

El principal atractivo de la generación de números aleatorios es llevar estos generadores a los ordenadores. Se abren un sinfín de posibilidades, ya sea en estudios, videojuegos, simulación, criptografía, etc. Para ello se utilizan algoritmos que dada una *semilla,* que funciona como valor inicial para el algoritmo, generan secuencias de números. Se llaman a estos generadores *pseudoaleatorios.* [8]

Se tiene por lo tanto la necesidad de generar números aleatorios, ya que los usos de estos números aleatorios se extienden a la criptografía y generación de claves.

# Capítulo 3

# Señales biométricas

## 3.1 Introducción

La aplicación de la tecnología a la medicina está sufriendo un desarrollo enorme. Tanto universidades y grupos de investigación como las fuentes de financiación de estos proyectos son conscientes de la importancia que tienen.

Se habla de la salud y esperanza de vida de la población, de detección temprana de enfermedades, de comportamientos de individuos o de medidas tomadas a estos que indican la existencia de una patología o la manera de solucionarla… Con el estudio de señales biométricas se ha visto que existen indicadores o medidas capaces de detectar enfermedades o de indicar cuantitativamente el estado de salud del individuo. Esto supone un gran avance en medicina.

Se necesita que las señales utilizadas cumplan una serie de requisitos. Hay temas como la estabilidad de la señal frente a cambios de salud y la robustez con el paso del tiempo que debemos tener en cuenta antes de elegir qué tipo de señales se van a utilizar [9].

Tradicionalmente se ha investigado de manera separada en cada uno de los campos, aunque en realidad tienen interacción entre ellos. Lo ideal sería abordar el estudio de la biometría de manera más global. Si se desarrolla un dispositivo que monitorice señales biométricas de algún paciente con el objetivo de alertar si algo no va bien y aparece algún tipo de riesgo, sería muy interesante que de la seguridad de las comunicaciones de este dispositivo se encargasen estas mismas señales. Si tenemos en cuenta temas relacionados con salud a la vez que los relacionados con la seguridad, los avances en una de las dos ramas pueden repercutir de manera positiva en la otra.

Sin más, se pasa a ver qué tipo de señales biométricas se han elegido para este trabajo y el porqué de la elección.

## 3.2 Electrocardiograma (ECG)

El electrocardiograma registra las diferencias en el voltaje que ocurren durante la despolarización y repolarización en las células. Esta actividad eléctrica generada por el corazón se detecta con electrodos colocados en la superficie corporal. Este impulso produce la contracción rítmica del corazón, cuya actividad se produce periódicamente latido tras latido [10] [11].

Esta actividad eléctrica es del orden de milivoltios, y produce una forma de onda periódica como la que se observa en la figura 3.1:

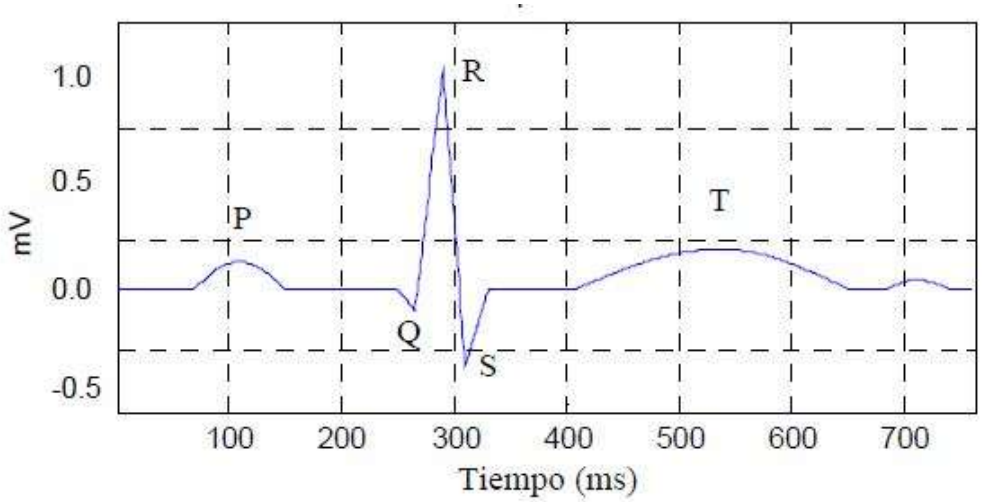


*Figura 3.1: Forma de onda ECG tomada de [12].*

La señal ECG es periódica y cada pulso tiene tanto tramos que varían rápidamente como lentamente. La parte que cambia con más velocidad se corresponde con los puntos QRS de la señal, y los puntos P y T son los que se observa cambian con menos velocidad.

Se divide la señal ECG en los siguientes intervalos [13]:

- Onda P. Representa la despolarización de las aurículas y tiene una duración media de 0,1 s. Un aumento de la amplitud o duración de la onda P indica que existe algún problema auricular.

- Complejo QRS. Representa la despolarización de los ventrículos y tiene una duración de unos 120 ms aproximadamente.

- Onda T. Representa la repolarización de los ventrículos. Cualquier anomalía marca un desequilibrio hidroelectrolítico.

El significado de cada uno de los intervalos de la onda ECG hace que se pueda localizar el lugar dónde se está produciendo alguna anomalía y que se pueda tener automatizada su detección en un dispositivo destinado a ello. Todo esto hace que los estudios en este campo sean mucho más objetivos y a la vez eficientes. Por lo tanto el significado de estos puntos PQRST es muy importante para estudios en salud. Sin embargo este trabajo se centra en la seguridad. Si la intención es buscar una fuente de aleatoriedad en esta señal para proporcionar seguridad, las características de estos puntos (amplitud y tiempo) son una buena opción. Esta va a ser la fuente de aleatoriedad que se utilizará para uno de los procesados hecho para estas señales.

Otra de las posibles fuentes de aleatoriedad que se observan y en la cual se centra el segundo procesado es el ruido. Sea cual sea el dispositivo con el que se ha medido, el electrocardiograma va a presentar ruido. Son varios los factores que lo generan: ruido por el contacto electrodo-piel, ruido térmico generado por los dispositivos utilizados para las medidas, ruido provocado por las señales de electromiografía generadas por los músculos o incluso interferencias generadas por la frecuencia de la red de electricidad [11].

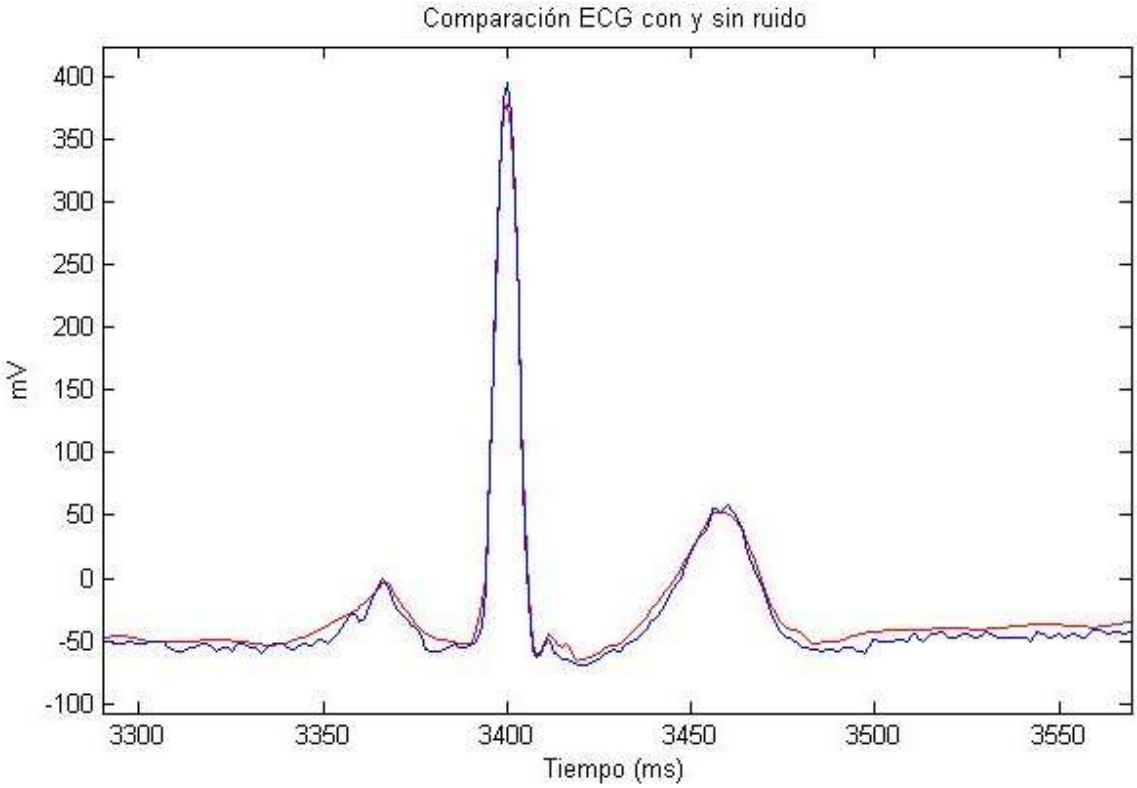


*Figura 3.2: En azul la señal ECG con ruido y en rojo el ECG sin ruido.*

## 3.3 Electromiografía (EMG)

El corazón es un músculo, por lo que, si mediante sensores se ha podido captar la actividad eléctrica que este tiene en nuestro cuerpo, también se va a poder captar la que tengan otros músculos. Aunque esta actividad no va a ser tan "fuerte", se va a poder captar a través de sensores ya que se trata de músculos mucho más superficiales.

Se define la electromiografía (EMG) como la captura, registro y posterior análisis de la actividad eléctrica generada por la activación y contracción de las fibras musculares activas. De esta activación se encarga el sistema nervioso central y periférico, que actúa como un generador de impulsos que activan unidades motoras. Las unidades motoras transforman estos impulsos en potenciales de acción de unidad motora. Estos potenciales son los que llegan al músculo y lo hacen contraerse. Lo que se mide al colocar electrodos en un músculo, es la suma de todos los potenciales que llegan a ese músculo. Los músculos generan tensiones cercanas a 100 mV al contraerse que, aunque son muy atenuadas por el tejido y la piel, son suficientes para poder ser capturadas a nivel superficial de la piel. Por ejemplo una señal EMG de un músculo como el bíceps gira en torno a 2.1 mV [14] [15].

Estas señales no están exentas de ruido e interferencias. Ruido generado por factores similares a los citados en el electrocardiograma. Hay que añadirle interferencias provocadas por la generación de actividad en músculos cercanos al que se está estudiando. Las fuentes de potenciales no tienen límite de zonas por las que transmitir. Cuando se toman medidas en un músculo en concreto, los electrodos colocados en este van a recibir los potenciales de dicho músculo y potenciales que en realidad están destinados a músculos cercanos.

En este caso, para los procesados se va a volver a utilizar como primera fuente de aleatoriedad el ruido generado a la hora de las medidas. En el segundo procesado se usarán directamente las amplitudes de las muestras tomadas.

## 3.4 Electroencefalograma (EEG)

De nuevo una señal biométrica que registra la actividad eléctrica del cuerpo, en este caso concreto registra la actividad eléctrica cerebral. Cualquier actividad cerebral se manifiesta como un estímulo eléctrico que se transmite entre las neuronas. Este estímulo se corresponde con un campo eléctrico perpendicular a la superficie del cráneo, que desde que se genera hasta que llega a la superficie craneal sufre atenuación. Es en la superficie del cráneo donde se colocan una serie de electrodos encargados de captar esa señal eléctrica [16].

Tenemos millones de neuronas. Todas ellas, sea cual sea la actividad que se está realizando, generan actividad eléctrica constantemente, por lo que captar esta señal eléctrica en un único punto no tiene sentido. El estándar "Sistema internacional 10/20" aprobado por la Federación Internacional de Sociedades de Encefalografía y Neurofisiología Clínica establece las divisiones del cráneo en las que se deben colocar los electrodos hasta un máximo de 64 electrodos.

***Sistema Internacional 10/20***

Recibe este nombre debido a la separación que deben tener los electrodos. La superficie craneal está dividida en 6 zonas: frontal (F), parietal (P), frontopolar (Fp), central (C), occipital (O) y temporal (T). La línea que une el nasión (entrecejo) y el inión (nuca), separa el cráneo en dos secciones llamadas frontal y parietal, y la longitud de esta línea marca la distancia de referencia en base a la cual se calculan los porcentajes (10% y 20%) para la separación de los electrodos.

Los electrodos se colocarán como se observa en la figura 3.3 y de acuerdo a la siguiente nomenclatura [16]:

- Letra que marca el lóbulo cerebral.
    - F: lóbulo frontal.
    - O: lóbulo occipital.
    - P: lóbulo parietal.
    - T: lóbulo temporal.
    - C: zona central.
    - Z: línea central.
- Número que marca el hemisferio cerebral.
    - Números pares: hemisferio derecho.
    - Números impares: hemisferio izquierdo.

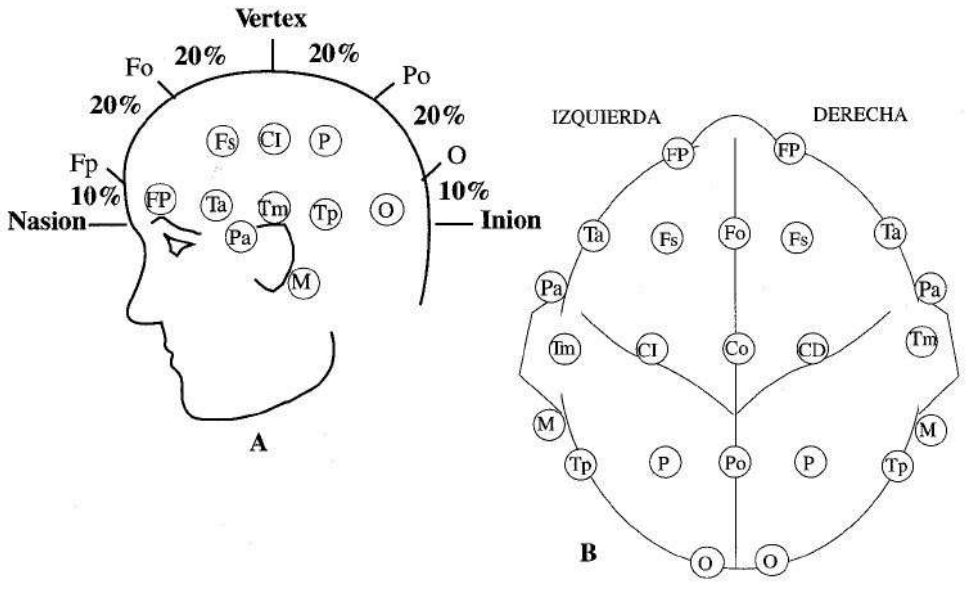


*Figura 3.3: Sistema colocación electrodos. A. Vista perfil. B. Vista superior. Imagen tomada de [17].*

La señal eléctrica que cada electrodo recoge no se corresponde únicamente con la de las neuronas sobre las que está situado, sino que va a ser la suma de todas las señales eléctricas producidas por cada neurona. La amplitud que va a tener esta señal no va a estar ligada expresamente al nivel o complejidad de la actividad cerebral desarrollada, sino a la sincronización con que se produzcan estas señales. Puede darse el caso de que se produzcan interferencias constructivas o destructivas. La amplitud se va a mover dentro del rango de 5uV hasta 100uV [18].

Está relacionado con la actividad realizada la frecuencia de la señal recogida, siendo frecuencias altas propias de actividades relativamente complejas que requieran una mayor actividad neuronal y frecuencias bajas propias de actividades tales como dormir.

Muchos de los estudios que se han llevado a cabo se basan en la reacción que tienen las neuronas ante algún estímulo (visual, auditivo,…). Este estímulo se va a ver reflejado en la señal de nuestro EEG. Otros estudios se basan en la reacción que se manifiesta en el EEG cuando se piensa en una contraseña para iniciar sesión en alguna cuenta. Obviamente queda aún mucho que estudiar y que mejorar hasta que realmente se puedan llevar a cabo estas técnicas, aunque sin duda, son más que factibles. En el estudio del uso del EEG como RNG se continúa con la misma línea de trabajo y de nuevo usan como fuente de aleatoriedad tanto la propia señal medida como el ruido que se ha generado durante la captura de la misma.

# Capítulo 4

# Herramientas.

## 4.1 Introducción.

La principal herramienta utilizada para el trabajo es Nist Test Suite. Un software desarrollado para evaluar la aleatoriedad de secuencias de bits. Dispone de una serie de test que evalúan distintos aspectos de las secuencias en cuanto a aleatoriedad.

Cualquier trabajo de investigación necesita una buena base de datos, lo suficientemente heterogénea dentro de su ámbito de estudio. Base de datos para realizar numerosas pruebas con el objetivo de tener cubierta la diversidad de datos diferentes que se pueden encontrar y dar validez estadística. Una buena base de datos se puede conseguir de alguna fuente de datos online que sea fiable y que cumpla con el estándar y protocolo para realizar cada una de las medidas. También si se dispone directamente de los medios y dispositivos necesarios para realizar medidas experimentales.

Para este estudio se han combinado ambas fuentes de datos. Se disponía de los dispositivos necesarios para realizar medidas durante un tiempo limitado, por lo que también se ha usado una base de datos online. Para hacer numerosas pruebas y dar validez estadística y se han usado los datos de la fuente online. Las medidas reales tomadas experimentalmente sirven como forma de corroborar los resultados.

## 4.2 Nist Test Suite

Hasta ahora, se ha mencionado constantemente el hecho de que una secuencia sea aleatoria o no. Se han visto las propiedades de las secuencias aleatorias y una idea aproximada de los tipos de generadores de números aleatorios que existen hasta ahora.

En ningún momento se ha hablado de cómo saber si una secuencia es aleatoria o no. La herramienta Nist Test Suite fue desarrollada con el objetivo de evaluar la aleatoriedad de secuencias generadas por RNGs o PRNGs.

### 4.2.1 ¿Cómo funciona?

La aleatoriedad es una propiedad probabilística. Significa que las propiedades de las secuencias aleatorias pueden ser descritas en términos de probabilidad y cuantizadas. Una vez se tienen cuantizadas y se sabe cuáles son las probabilidades de secuencias realmente aleatorias se pueden establecer unos límites a partir de los cuáles se consideran superadas las pruebas realizadas.

### 4.2.2 Ejecución de la herramienta.

Se desarrollaron una serie de test que evalúan el número de veces que se repiten algunos comportamientos que se encuentran en secuencias no aleatorias dentro de una secuencia dada.

Al ejecutar la herramienta se  pide lo siguiente:

En la ruta en la que se haya instalado la herramienta se van a copiar los ficheros que contendrán las secuencias de unos y ceros que vamos a evaluar. Así una vez se lanza la herramienta, lo único que hay que indicar es el nombre del fichero que se quiera evaluar.

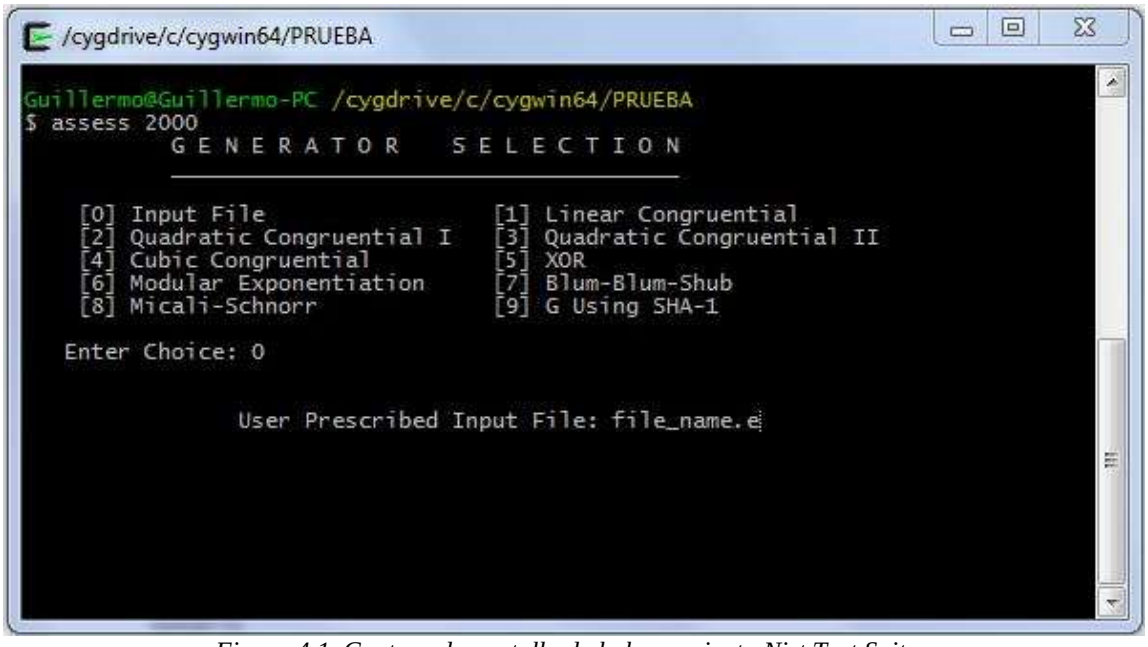


*Figura 4.1. Captura de pantalla de la herramienta Nist Test Suite.*

Lo siguiente que aparece es una lista con todos los test con los que se puede evaluar el fichero. Se debe indicar cuáles de ellos se quieren utilizar en esta ejecución.

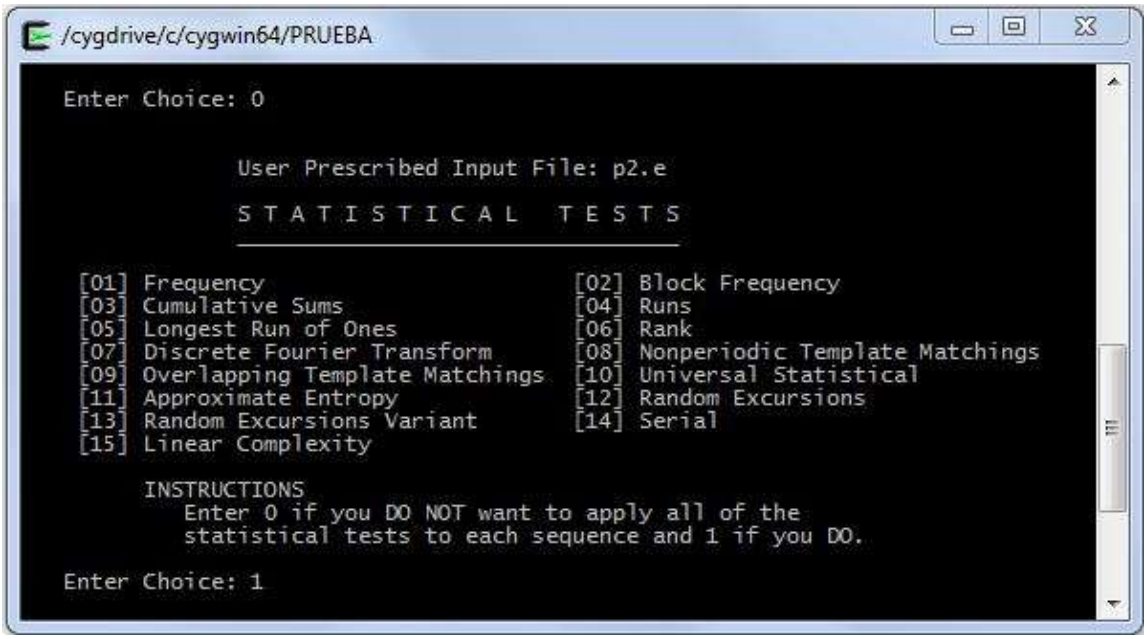


*Figura 4.2. Captura de pantalla de la herramienta Nist Test Suite.*

Una serie de características para la ejecución de los test aparecen en el siguiente cuadro. Distintas longitudes de bloque con los que se van a realizar los test.

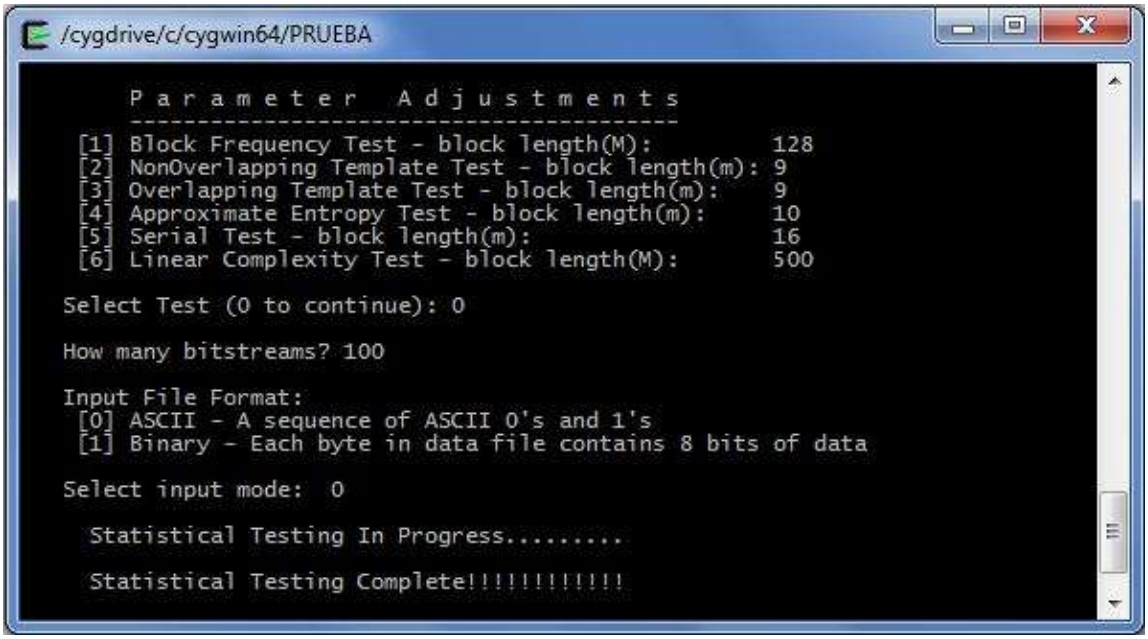


*Figura 4.3. Captura de pantalla de la herramienta Nist Test Suite.*

Al realizar los test no se va a evaluar la secuencia completa de una vez. La secuencia completa se fragmentará en tantas partes cómo se quiera. A cada una de ellas se le realizarán todos los test y de esta forma se puede obtener un porcentaje de los fragmentos que han superado los test de manera exitosa. Si sólo se realizase una prueba a toda la secuencia completa podría existir la posibilidad de que esta vez en concreto superase los test por fortuna [19] [20].

Lo último que le vamos a indicar antes de ejecutar los test es el formato del fichero que contiene la secuencia a evaluar.

### 4.2.3 Descripción de los test.

La herramienta Nist Test Suite cuenta con un total de 15 tests. Este trabajo utiliza los 6 más importantes, los cuales evalúan al completo todas las propiedades propias de la aleatoriedad y en consecuencia los más difíciles de superar para los RNGs [19].

**Frequency (monobit) test**

El primer test mide la proporción de unos y ceros de la secuencia completa. El número de unos y ceros de una secuencia realmente aleatoria esperado es *n/2* (siendo *n* la longitud total de la secuencia). El conjunto de elementos posibles es solamente unos y ceros y la probabilidad de ambos debe ser la misma, o sea $p_0 = p_1 = 0,5$.

Del éxito o fracaso de este test dependen el resto de test, ya que es el más importante y casi un requisito para los demás.

El tamaño de secuencia recomendado para este test es n>100 bits.

**Frequency Test within a Block**

La idea de este test es similar al del test de frecuencia. La única diferencia con el anterior es que en este caso no se evalúa la secuencia al completo, sino que se parte la secuencia en trozos de longitud *M*. Este es el primero de los parámetros que se puede ajustar en la ejecución de la herramienta y por defecto viene *M=128 bits.*

La recomendación para este test, como para el de frecuencia es de una longitud n>100 bits.

**Binary Matrix Rank Test**

El objetivo de este test es evaluar la dependencia linear de la secuencia. Para ello divide la secuencia en distintas subsecuencias o tramas con las que forma matrices. Comprueba si son linealmente dependientes o no con el cálculo del rango.

**Discrete Fourier Transform Test**

Este test se centra en los picos altos que se producen en la Transformada Discreta de Fourier con el objetivo de detectar rangos periódicos. Esto ayuda a saber si una secuencia es aleatoria o no.

En este caso la longitud de la secuencia debe ser n>1000 bits.

**Linear Complexity Test**

Las secuencias aleatorias están caracterizadas por tener LFSRs (*Linear Feedback Shift Register),* por lo que una secuencia con un LFSR muy corto denota no aleatoriedad. Este test se encarga de calcularla y ver de este modo si la secuencia es lo suficientemente compleja como para ser considerada una secuencia aleatoria.

**Approximate Entropy Test**

Este test es el encargado de comparar la frecuencia de bloques superpuestos con dos tamaños consecutivos (*m* y *m+1)* con la frecuencia esperada de una secuencia aleatoria. El valor de m también se puede ajustar como parámetro en la ejecución de la herramienta, aunque hay valores de *m* inaceptables dependiendo de la longitud de la secuencia.

## 4.3 Physionet.org

La base de datos usada en este trabajo ha sido "*Physionet.org*" ya que se adaptaba muy bien a las necesidades. Tanto por los tipos de señales biométricas de que dispone como por el tamaño de esos registros.

Más en concreto se han utilizado registros de su base de datos llamada "*PhysioBank*". Contiene más de 90.000 registros en su base de datos. Algo más de 4 terabytes de señales biométricas organizadas en más de 80 tipos de bases de datos de registros biométricos. Todo esto organizado en lo que llaman "*PhysioBank ATM*". Apartado de su página web donde clasifican por tipos los distintos registros que forman su base de datos. Se puede tanto visualizar las formas de onda de las distintas señales como descargar estos registros en varios formatos (".mat", ".txt",…) [21].

Tanto el *National Institute of General Medical Sciences (NIGMS)* como el *National Institute of Biomedical Imaging and Bioengineering (NIBIB)* respaldan la base de datos de Physionet, por lo que se puede estar seguro que todas las medidas tomadas han seguido todos los protocolos y estándares necesarios [21].

## 4.4 Fotopletismografía con Arduino.

Uno de los métodos usados para la toma de datos experimental consiste en un sensor de fotopletismografía instalado sobre Arduino 1 con el que se han tomado muestras de señales del electrocardiograma.

La fotopletismografía consiste en un diodo emisor de luz infrarroja que será reflejada por el flujo sanguíneo cutáneo y que posteriormente recibe un fotodetector situado junto al diodo emisor. Un aumento de flujo sanguíneo se traduce en un aumento de la luz reflejada, por lo que gracias a esto se puede tener una medida del flujo sanguíneo. Se puede así obtener el electrocardiograma, ya que las contracciones del corazón se ven reflejadas en aumentos de flujo sanguíneo.

Pues bien, una placa Arduino 1 a la que se le ha conectado, como se ve en la figura 4.4, una ranura para una tarjeta microSD. Sobre esta se van a ir almacenando las muestras tomadas por el sensor de fotopletismografía que se le ha conectado también. El sensor consiste de un diodo emisor de luz infrarroja y un fotodetector.

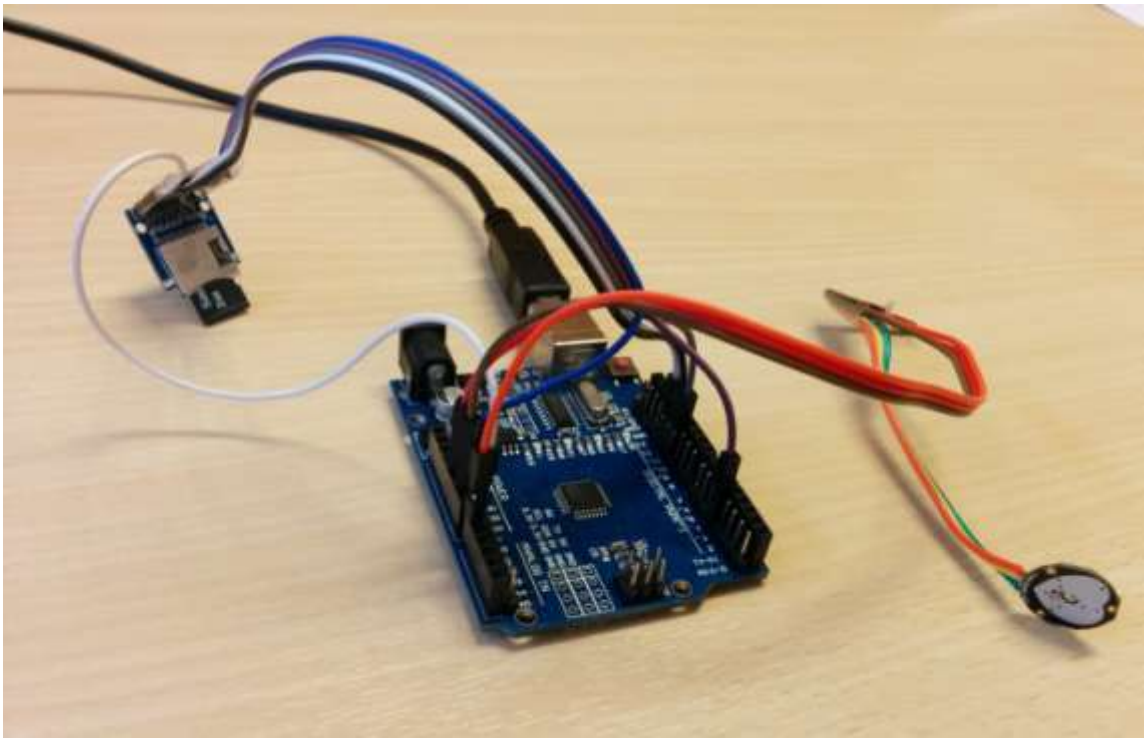

*Figura 4.4. Sensor de pletismografía instalado en placa Arduino 1.*

Esto se conecta por USB a un computador y automáticamente comienza a recoger muestras que se almacenan en la microSD. Se coloca en el sensor la yema del dedo por ser uno de los puntos donde los vasos capilares son más superficiales, por lo que se puede medir el ECG más fácilmente.

En la figura 4.5 se puede ver como la forma de onda de la señal recogida se asemeja bastante a la forma de onda de un ECG. Se pueden observar todos sus puntos característicos.

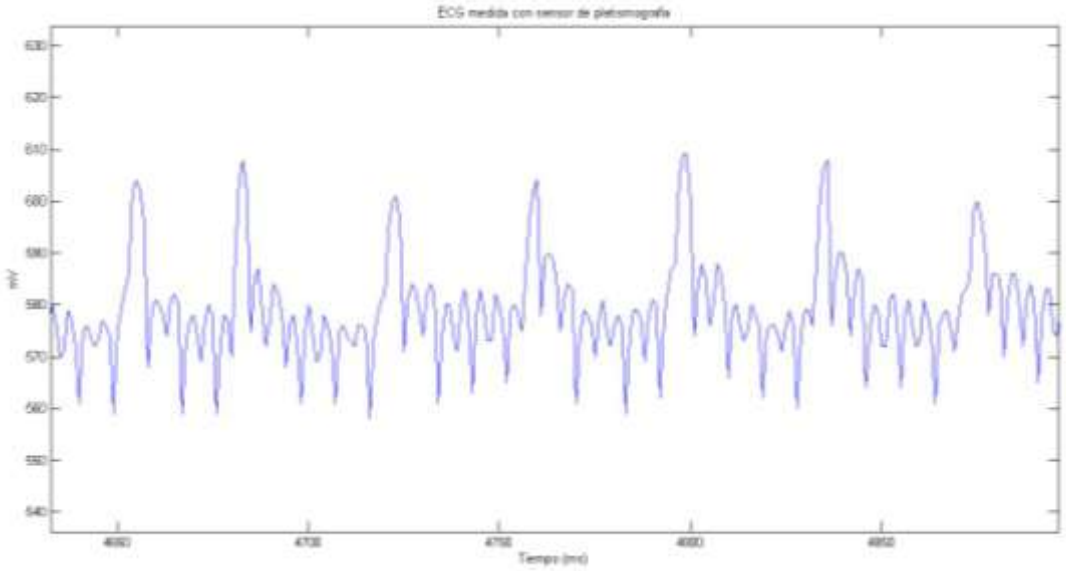


*Figura 4.5. Señal ECG medida con sensor de pletismografía.*

## 4.5 Plat EEG

Un proyecto de investigación de la Universidad de Granada desarrolló la "*Plat EEG*". Proyecto llamado "Plataforma de altas prestaciones para la adquisición, extracción y procesamiento inteligente de señales EEG". Gracias a uno de los integrantes del equipo de investigación, Jesús Minguillón, se pudo acceder puntualmente a este dispositivo, para con su ayuda poder tomar algunas muestras reales.

El *Plat EEG* es un sistema inalámbrico, portátil y de coste reducido que permite la captación de biopotenciales a través de los electrodos. Comunicación por Bluetooth para que mediante el software apropiado el usuario pueda establecer diversos parámetros.

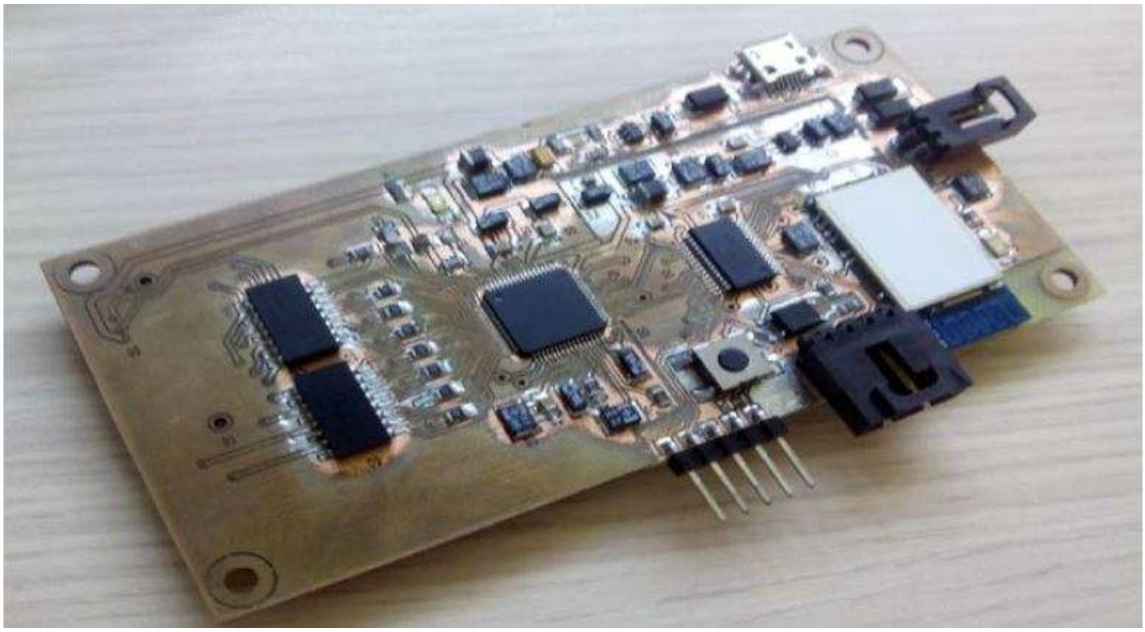

*Figura 4.6.Placa de la Plat EEG. [22].*

Aunque parezca contradictorio, con la Plat EEG no se han capturado señales EEG, sino señales ECG y EMG. Se disponía de un tiempo limitado para recoger muestras y para una señal EEG se utiliza un casco mallado con gran cantidad de electrodos por lo que el tiempo no era suficiente. En cambio para ECGs y EMGs sólo se necesitan dos electrodos adhesivos. Dado que el dispositivo capta cualquier tipo de biopotencial era totalmente factible la toma de muestras.

Los protocolos para la colocación de los electrodos para electromiografía superficial son de dos tipos: longitudinal y transversal con respecto al músculo. Ambas son igual de válidas, así que como se observa en las figuras 4.7, 4.8 y 4.9, se hizo de manera longitudinal colocando el punto de referencia en el lóbulo de la oreja. Se realizan las medidas en diferentes músculos y diferentes sujetos para así dar más validez y poder comparar.

*Figura 4.7.Medidas en músculos extensores del brazo*


*Figura 4.8. Medidas en bíceps.*

Para las medidas del ECG se mantiene el punto de referencia en el lóbulo de la oreja y se coloca un electrodo en algún punto donde se puedan captar las pulsaciones con facilidad. En nuestro caso se coloca, como se ve en la figura 4.9, en la muñeca.

*Figura 4.9. Localización punto de referencia.*

En ambas medidas se han utilizado electrodos desechables de superficie Ag/AgCl, muy utilizados para todas estas experiencias ya que captan la actividad eléctrica con un nivel de ruido adecuado.

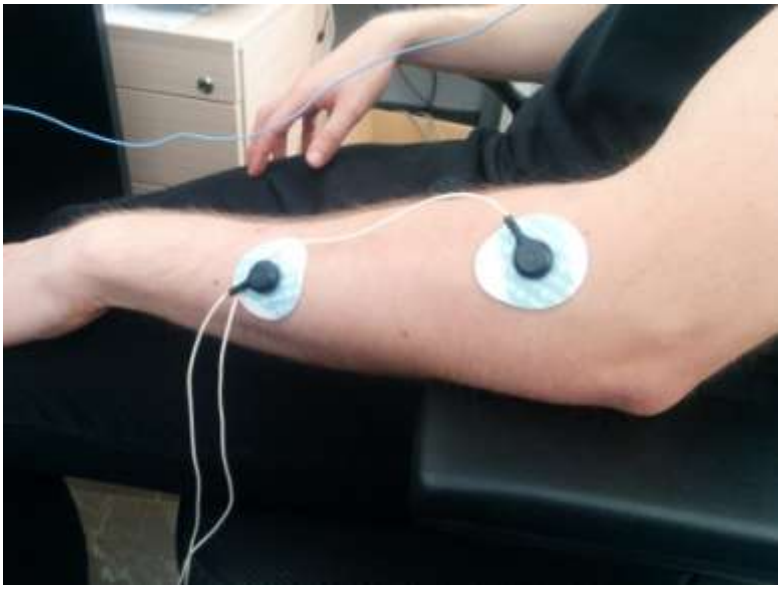

*Figura 4.10. Electrodos de superficie Ag/AgCl.*

En las figuras 4.11 y 4.12 se puede observar la forma de onda de las señales registradas.

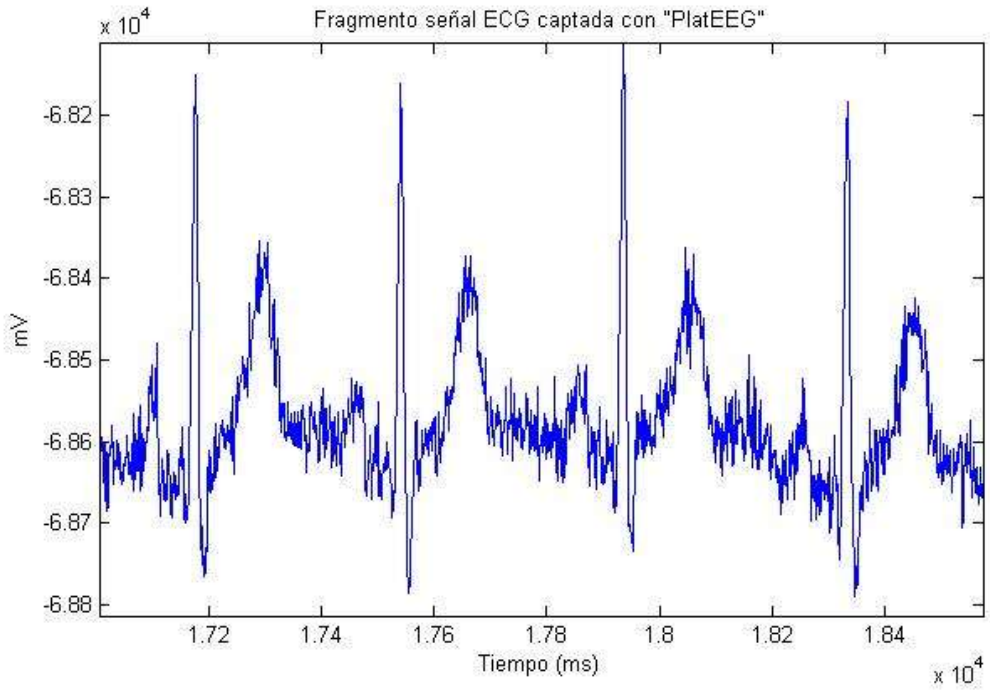


*Figura 4.11. Señal ECG obtenida con PlatEEG.*

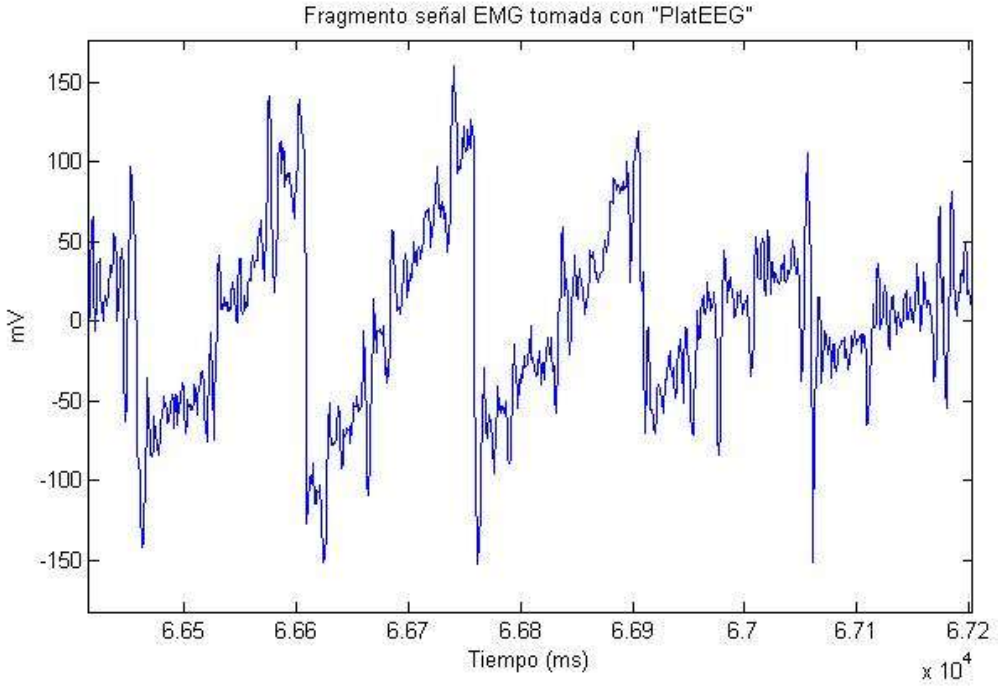


*Figura 4.12.Señal EMG obtenida con PlatEEG.*

# Capítulo 5

# Solución propuesta.

## 5.1 Introducción.

En capítulos previos a este se ha presentado primero la necesidad real de RNGs y PRNGs que se tiene en la actualidad y que se va a incrementar de manera que avanza la industria del IoT. Posteriormente se ha visto la motivación que lleva a elegir un tipo de señales con un grado muy alto de aplicabilidad en determinados sectores. Se han explicado características de cada tipo de señal elegida y se ha visto cómo hemos registrado las señales utilizadas para el estudio o dónde se han conseguido.

Una vez se tenía claro todo esto, solamente faltaba una herramienta con la que evaluar los distintos procesados para las señales que se iban realizando. Se presenta para esto la herramienta Nist Test Suite y se describen los test que se van a tener en cuenta para entender el porqué de algunas de las partes del procesado. A groso modo esto es todo lo que se necesita saber para poder entender las distintas partes de código que se van a explicar y la solución al problema de generación de números aleatorios que se propone.

El objetivo en un primer momento es desarrollar un procesado que común y útil para todos los tipos de señal que van a utilizar. Por lo tanto para el primer procesado que se va a desarrollar se va a intentar no utilizar características propias de un tipo de señal en concreto, ya que no se podría utilizar el mismo procesado para el resto de las señales. Con esto se quiere conseguir que si se utiliza este algoritmo, en algún dispositivo que capture este tipo de señales pueda utilizar tanto una como otra señal. Muy útil en caso de que alguna no se pueda captar en algún momento concreto. Además si se utiliza un único algoritmo, el desarrollo e implantación tanto software como hardware va a ser mucho menos costoso y va a verse una mejora a nivel de costes de producción. Se ha visto en el capítulo que describe las señales biométricas utilizadas, que la característica que tienen en común todas estas señales es que todas tienen ruido e interferencia provocado por diversos factores. En realidad, no preocupa qué produce ese ruido ni porqué, sino que esté siempre presente.

Ligado al uso del ruido como fuente de aleatoriedad hay un posible factor de riesgo sobre el que hay algunos estudios. Si se usa una señal que no sea la original en el

mismo medio, se va a poder obtener ese ruido de la misma manera que si fuese la señal original. Este tipo de ataques es más propio de otras técnicas basadas en el medio inalámbrico, ya que el medio va a ser mucho más accesible que en el caso de señales biométricas. Habría que tener acceso al mismo sensor en un mismo momento en el caso de señales biométricas. También el medio va a ser distinto, ya que uno de los tipos de ruido e interferencia que se inserta es propia del cuerpo humano, huesos, tejidos, etc. Esto hace que se disminuya mucho el riesgo de sufrir uno de estos ataques. Pero para cubrir esta pequeña posibilidad o para otros posibles ataques que se basen en el mismo fundamento se va a estudiar un procesado que en lugar de utilizar el ruido, use la señal en sí, es decir, los valores de las muestras registradas.

## 5.2 Procesado del electrocardiograma

### 5.2.1 Procesado señal

La primera solución con la que se han obtenido unos resultados que se asemejan a los deseados consiste en obtener una secuencia aleatoria basada en los picos periódicos que se repiten en las señales ECG.

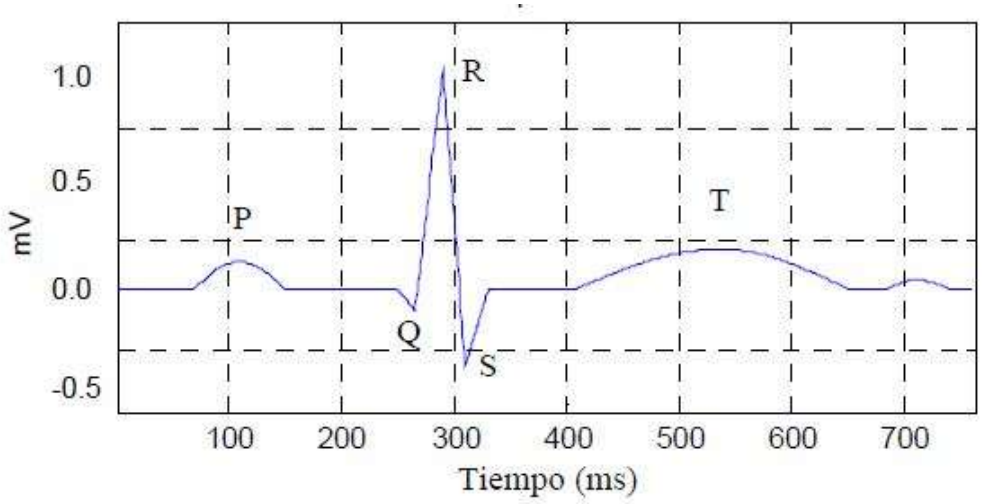


*Figura 5.1. Forma de onda señal ECG [12].*

Como se mencionó en el capítulo 3, una de las características de las señales ECG es la periodicidad. Una serie de "picos" de amplitud se repiten cada cierto tiempo. El periodo de la señal ECG es variable. Esto quiere decir que los "picos" se suceden tras un tiempo que no es fijo, y con una amplitud variable también.

Una característica periódica de la que se pueden obtener dos parámetros con los que generar aleatoriedad. Se observa en la figura 5.2 que ni el valor de amplitud ni de tiempo es un valor fijo.

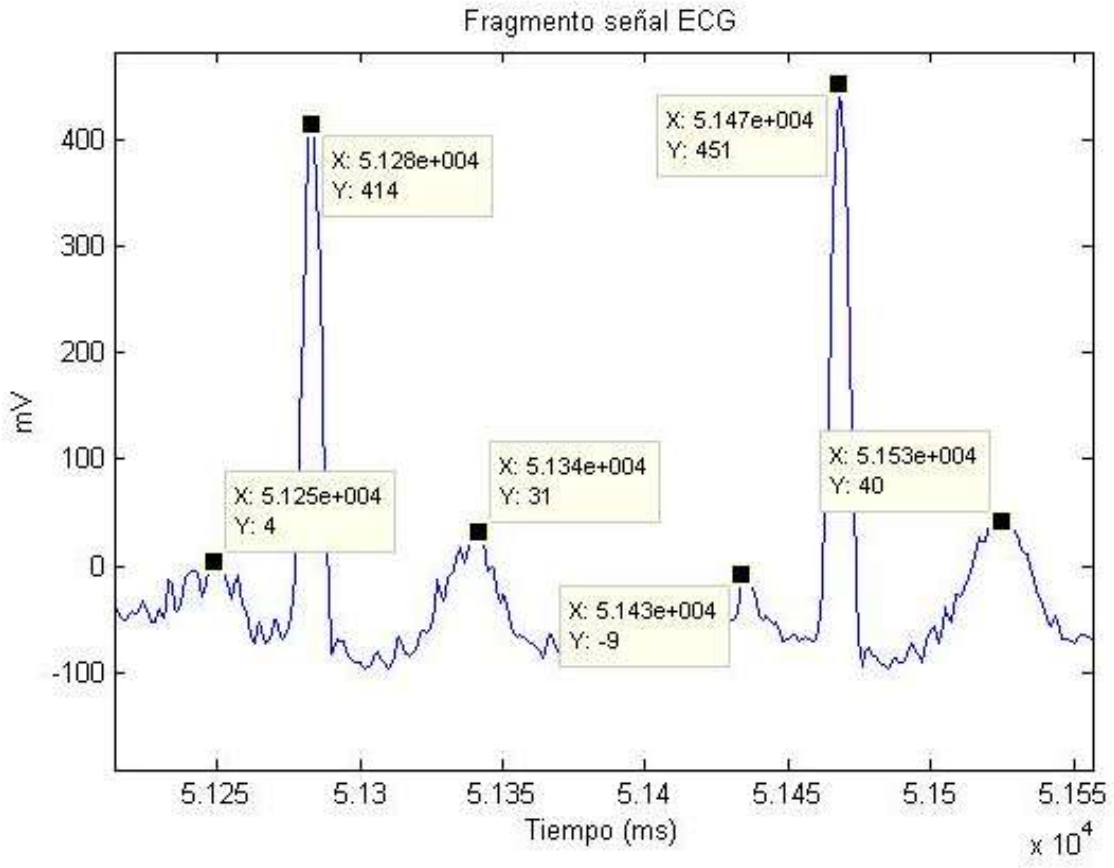


*Figura 5.2. Fragmento ECG que muestra valores en tiempo y amplitud.*

**Detección de picos PQRST**

La primera parte del procesado va a consistir en la detección de todos los picos que se produzcan en la señal. Para esto se utiliza la función *findpeaks* de Matlab:

$$[Amplitud, Tiempo] = findpeaks\ (señal, 'MinpeakDistance', d)$$

Esta función detecta todos los picos que tenga el vector *señal*, con una diferencia de tiempo mínima de valor *d*, y devuelve dos vectores, uno con los valores en amplitud y otro en tiempo.

Se observa en la figura 5.1 que los picos con mayor amplitud, y por lo tanto los que la función *findpeaks* va a detectar, son los picos R. Una vez se tiene un vector con todos los tiempos en los que se tiene un pico R lo siguiente es calcular dónde se producen el resto de picos. Para ello se crea un bucle que va tomando fragmentos de la señal tanto antes como después de que se produzca el pico R. Con esto lo que se consigue es un fragmento de la señal en el que no aparece el trozo de onda R y que así la función *findpeaks* sea capaz de detectar el resto de picos de amplitud menor a la de R. Para la detección de los picos tanto Q como S es necesario pasarle a *findpeaks* el fragmento de la señal invertida en amplitud, ya que en este caso se trata de mínimos de la señal en lugar de máximos como para el resto de picos.

Una vez se tienen todos los picos detectados finaliza la parte del procesado de detección de picos PQRST en el que se tiene como resultado 5 vectores con los valores de tiempo donde se producen los picos.

**Generación secuencia intervalos**

El siguiente paso consiste en generar 5 vectores (RP, RQ, RR, RS, RT) que van a contener las diferencias de tiempos de los picos PQST con respecto a R y de este pico R con respecto al anterior pico R+1. Se observa en la figura 5.3 con más claridad.

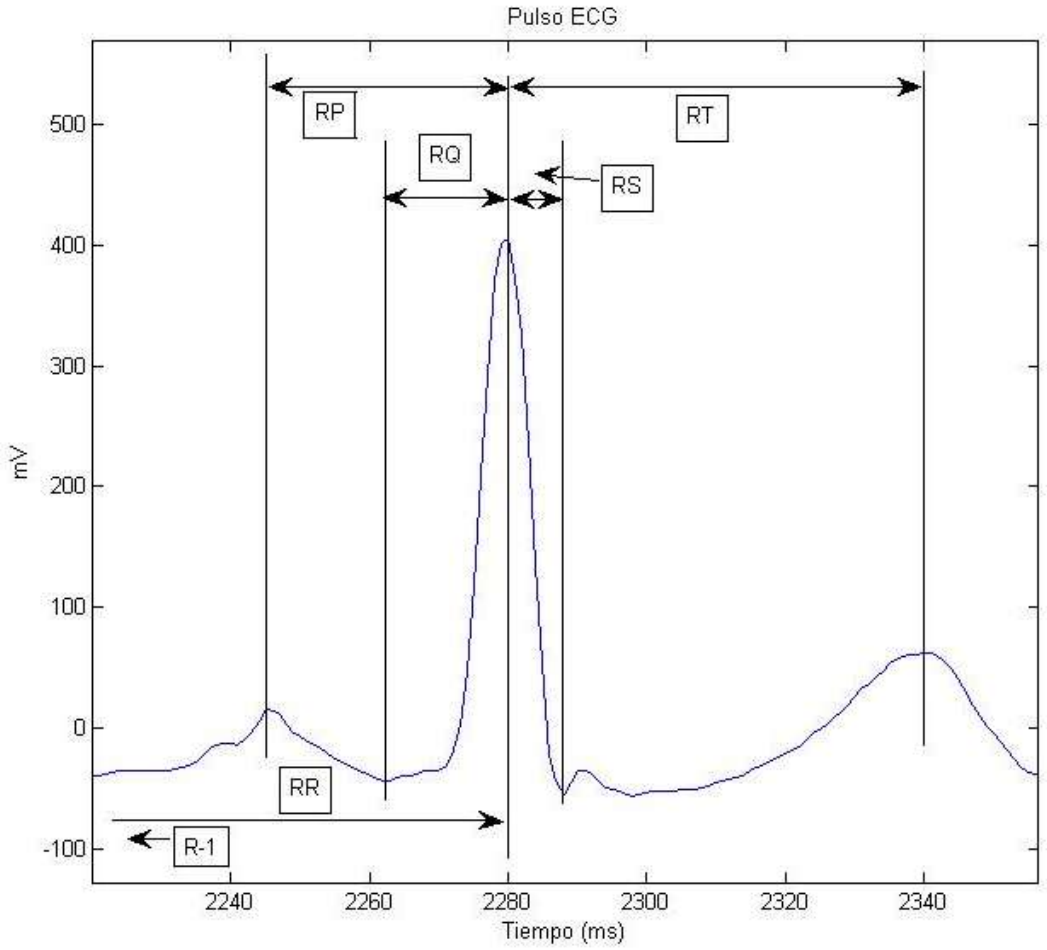


*Figura 5.3. Intervalos RP, RQ, RR, RS y RT.*

Se afirmó al principio de la explicación y se observó en la figura 5.2 que los picos no se producían ni con un valor de intervalo fijo ni tampoco en amplitud. Si bien es cierto que no es un valor fijo, también es cierto como se observa en la figura 5.4, que los cambios son mínimos.

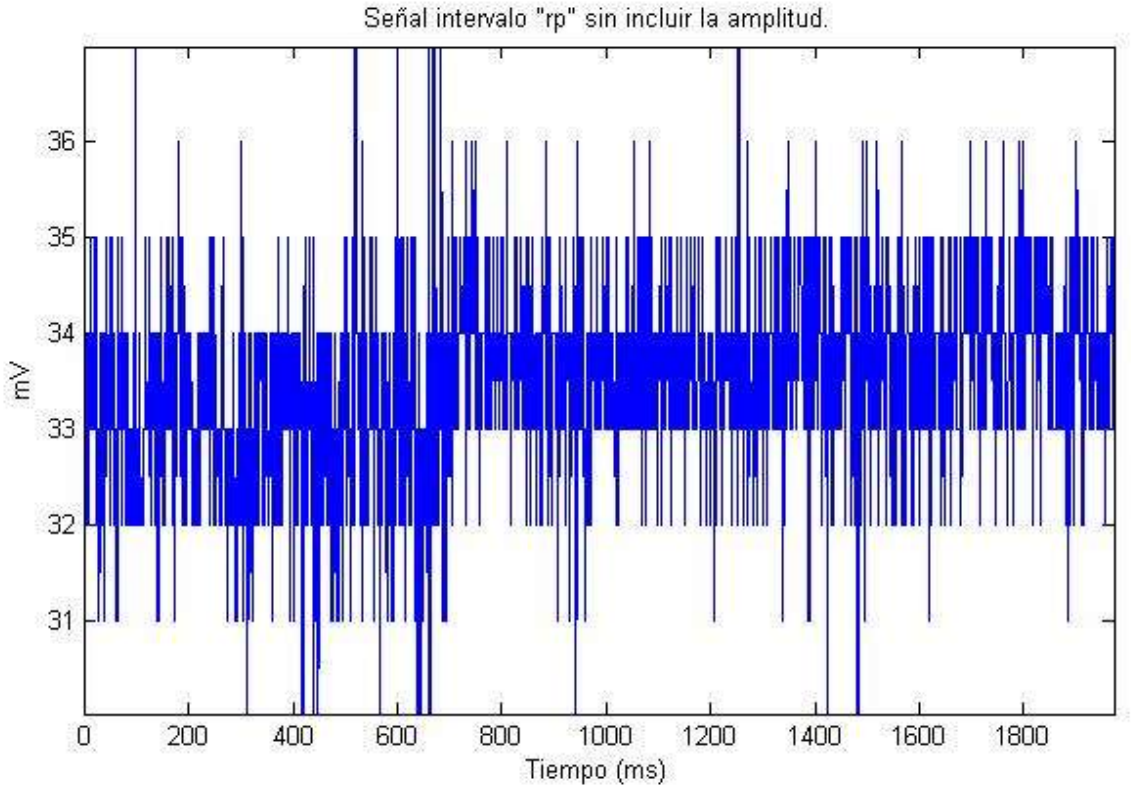


*Figura 5.4. Señal intervalo RP.*

Se pretende utilizar estos 5 vectores para generar secuencias aleatorias, por lo que un vector con valores que oscilan entre 5 valores distintos no va a ser válido. Esto se debe a que la precisión de las muestras de la señal es limitada, por lo tanto las variaciones detectadas son mínimas.

Se utiliza para dar más aleatoriedad a esta señal la otra característica que se mencionó anteriormente, la amplitud. Para ello al calcular cada valor a insertar en el vector se procede de la siguiente manera. Un valor del vector RP se calcula como:

$$RP_i = \left|t_{R_i} - t_{P_i}\right| + ecg(t_{P_i})$$

Como se observa en la figura 5.5, la forma de onda para la señal RP contiene valores que oscilan en un rango mayor, por lo que mejorará la aleatoriedad.

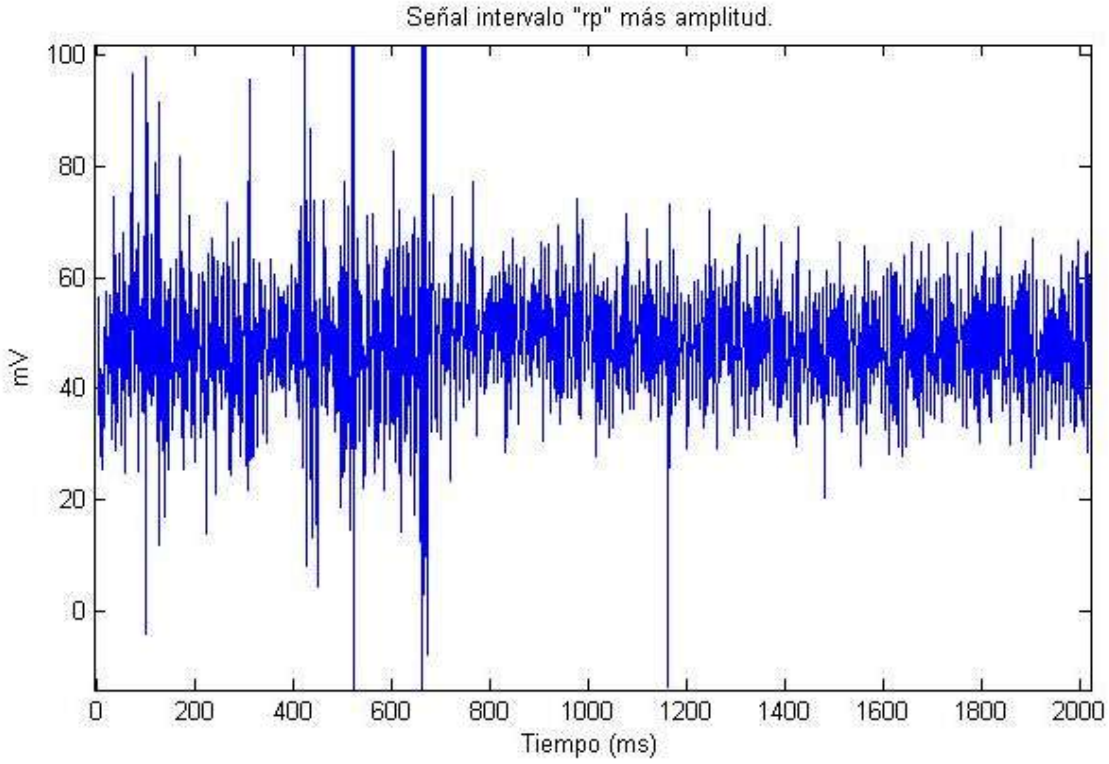


*Figura 5.5. Señal RP más amplitud.*

Mejora notablemente el resultado de la señal RP al incluir valores de amplitud ya que se incrementa bastante el rango de valores entre los que oscila la señal.

El resultado de esta parte del procesado son 5 vectores que forman señales con apariencia aleatoria que se van a combinar en la siguiente parte del procesado para formar una secuencia aleatoria.

**Generación señal salida**

El objetivo de esta parte del procesado es encontrar una forma no periódica de combinar las 5 señales antes obtenidas. Hay que hacer especial énfasis en la no periodicidad de la combinación ya que esta haría inútil el algoritmo.

Se crea un bucle en el que cada iteración va ir añadiendo un fragmento de longitud variable de una de las 5 señales de las que se dispone. Para que ni la longitud del fragmento ni la señal de la que toma el fragmento sean fijas se hace uso del vector *ruido.*

En el apartado 5.2.2 se explica con todo lujo de detalles cómo se obtiene el vector *ruido*, por lo que se explica ahora únicamente cómo se utiliza.

En primer lugar hay que elegir de cuál de las 5 señales vamos a tomar el fragmento. Se toma el vector *ruido* como suficientemente aleatorio para realizar esta función. Lo más práctico para elegir aleatoriamente entre 5 posibles opciones es llevar la señal *ruido* a valores entre 1-5 y redondear los resultados con la función *fix* de Matlab:

$$R_{-1,1} = \frac{R}{\max |R|}$$

$$R_{1-5} = 2'5 * R_{-1,1} + 3'5$$

La función *fix*($R_{1-5}$) redondea los valores a la baja, lo que explica los valores de la última ecuación. Se tiene un vector que oscilará entre los valores 1, 2, 3, 4 y 5, y cuyas componentes se usarán para elegir en cada iteración qué señal elegir de las 5 disponibles.

Una vez se elige qué señal se va a utilizar en una determinada iteración, lo siguiente es elegir de qué longitud se va a extraer el fragmento. Para ello se sigue la misma estrategia que para elegir señal, con la única diferencia que en este caso los valores de ruido se llevan al rango 1-128. Así se añaden a la señal generada fragmentos de longitud variable entre 1 y 128 muestras.

El resultado obtenido de esta parte del procesado es un vector con muestras de las 5 señales que se tenían anteriormente, combinadas. Este vector será utilizado en los siguientes pasos para generar una secuencia de bits.

**Eliminar media por tramos**

Antes de pasar a generar la secuencia de bits unos y ceros se observa en la señal que hay tramos en los que se producen acumulaciones en amplitud. Se observa también en algunos tramos de la señal como aparecen tendencias alcistas o bajistas. Dentro de una tendencia alcista es fácil predecir cuál va a ser el valor aproximado de la siguiente muestra.

El bucle que elimina la media por tramos consiste en un bucle en el que por cada iteración le restamos la media a un tramo de la señal de una longitud que comienza con la longitud total de la señal y que se va haciendo cada vez más pequeño en cada iteración. Con esta longitud variable se consigue eliminar tanto tendencias muy largas como las de longitud pequeña. El bucle finaliza cuando la longitud de tramo llega a un mínimo fijado en 100 ms, ya que longitudes más pequeñas no tenían sentido.

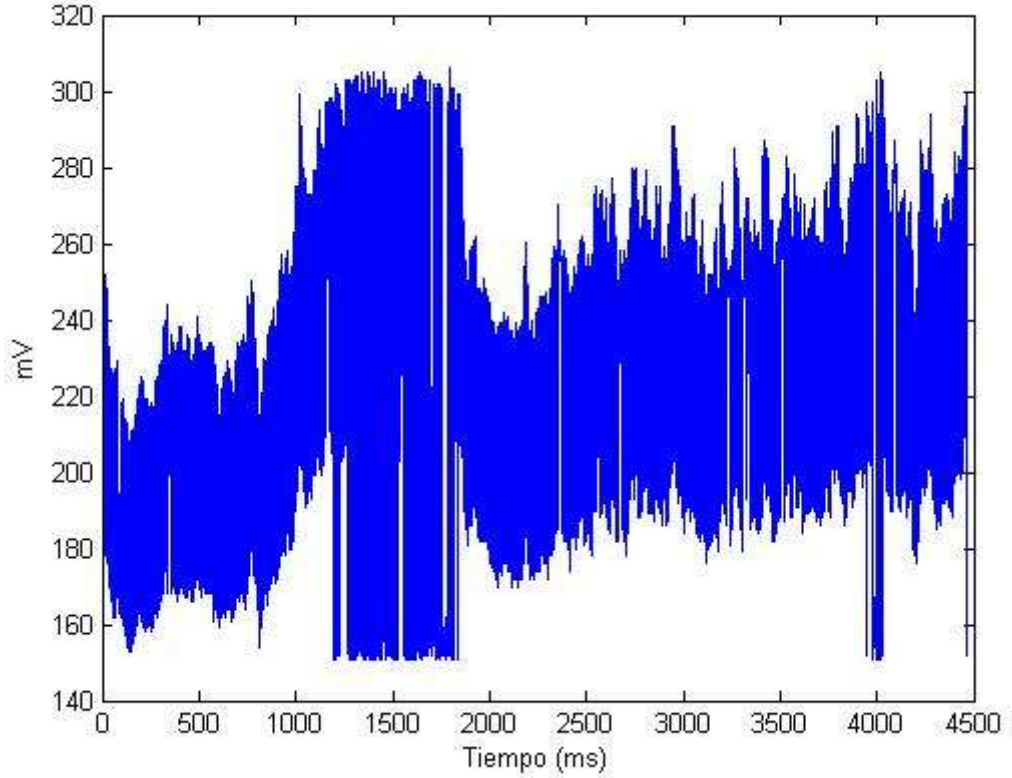


*Figura 5.6. Señal salida con acumulaciones.*

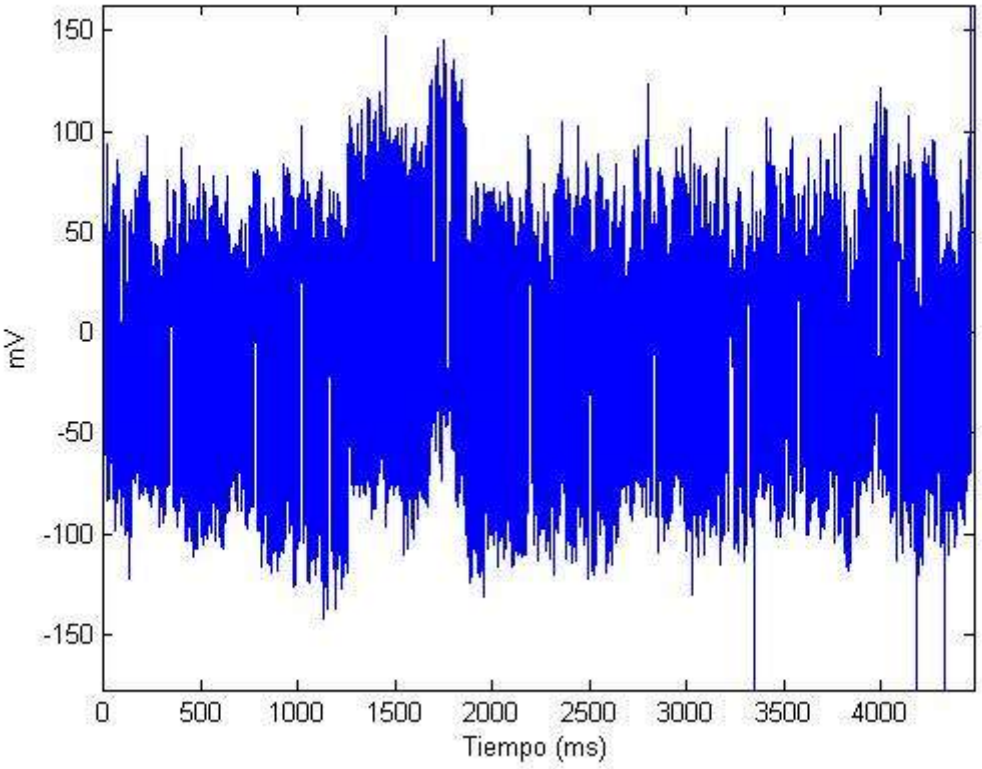


*Figura 5.7. Señal salida sin acumulaciones.*

**Recortar señal**

Una vez se ha eliminado la media por tramos se vuelve a observar la señal que obtenemos para intentar encontrar mejoras que hacerle a la señal para buscar aleatoriedad.

En la figura 5.8 vemos como la mayoría de los valores se acumulan en torno a cero en amplitud y la intensidad del color azul nos ayuda a hacernos una idea de valores a partir de los cuales la probabilidad de que aparezca un valor es mucho menor.

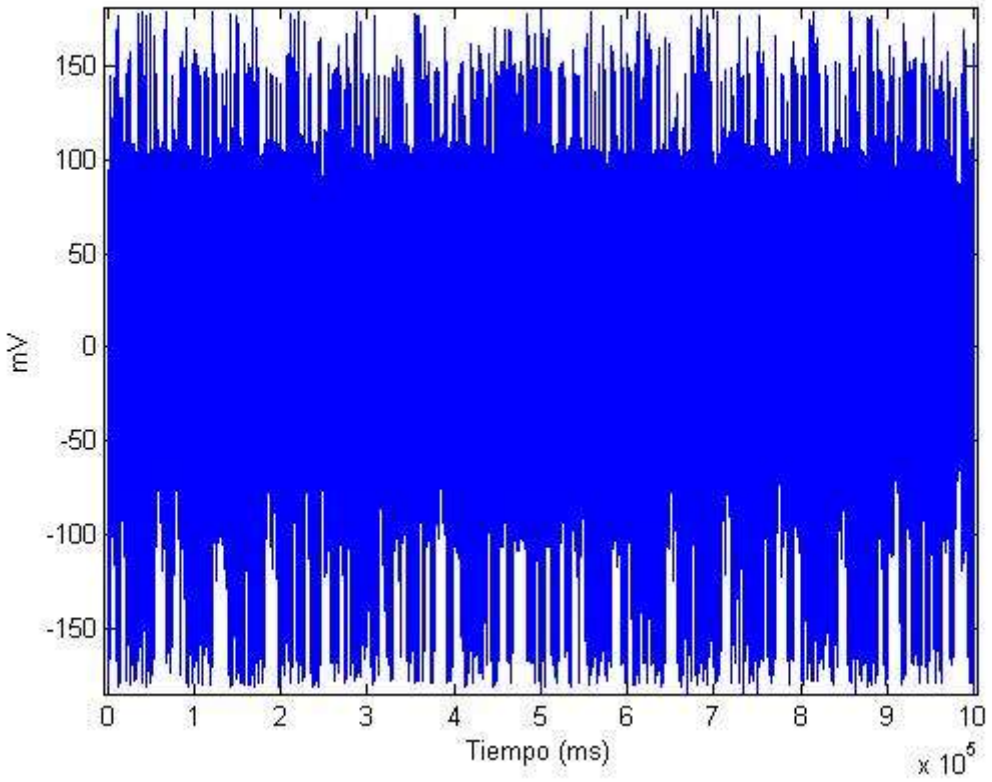


*Figura 5.8. Señal salida sin acotar en amplitud.*

Que los valores no estén distribuidos uniformemente en cuanto a amplitud es un factor a tener en cuenta, ya que todos los valores entre el máximo y mínimo de la señal deben tener la misma probabilidad. En caso contrario, cuando se normalice la señal en uno de los últimos pasos del procesado, para transformar los valores en bits unos y ceros, nos aparecerían muchos más unos que ceros, ya que la probabilidad de valores de baja amplitud sería mucho mayor que la de valores de amplitud alta.

Por lo tanto el objetivo va a ser intentar obtener ese valor anteriormente citado, a partir del cual aparecen muchas menos muestras. Para ello, se realizan dos bucles. El primer bucle crea dos vectores, uno con todos los valores positivos y otro con todos los valores negativos, y obtiene el valor de corte como:

$$Corte = \frac{1}{2}\left(mean(X^{+}) + mean(X^{-})\right)$$

El segundo bucle simplemente va descartando cualquier muestra que en valor absoluto supere este valor corte anteriormente calculado. Se observa en la figura 5.9 como ahora, a simple vista, todos los valores aparecen con la misma probabilidad.

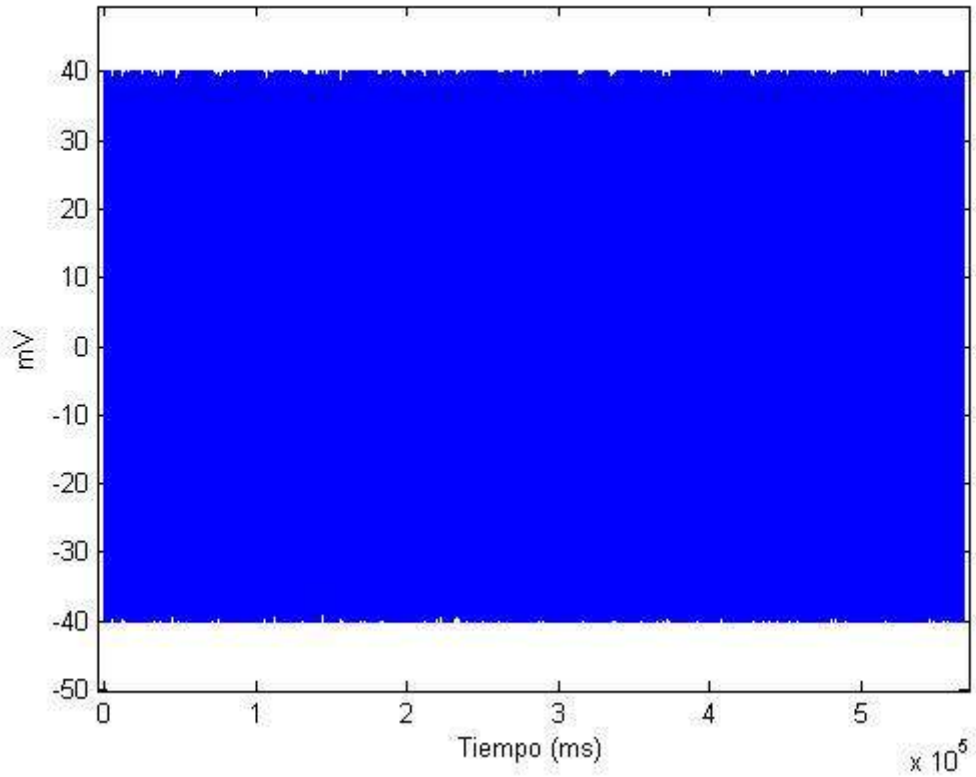


*Figura 5.9. Señal salida acotada en amplitud.*

Para ver esta mejora de manera más clara y representativa, vamos a usar la función *sort* de Matlab, que nos va a ordenar de menor a mayor las muestras de la señal ruido antes y después de recortarla. Efectivamente en las figuras 5.10 y 5.11 vemos como antes de recortar la señal, los valores se acumulan en torno a cero, mientras que después de recortarla los valores están distribuidos a lo largo de todos los posibles valores con la misma probabilidad.

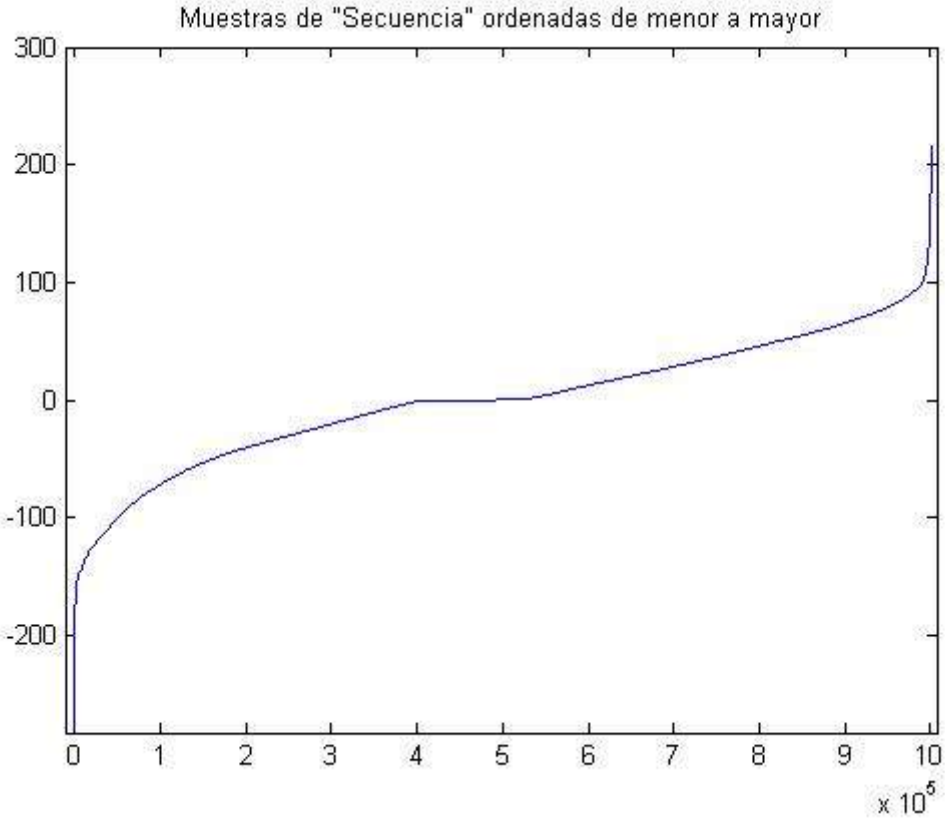


*Figura 5.10. Muestras de la señal salida ordenadas de menor a mayor*

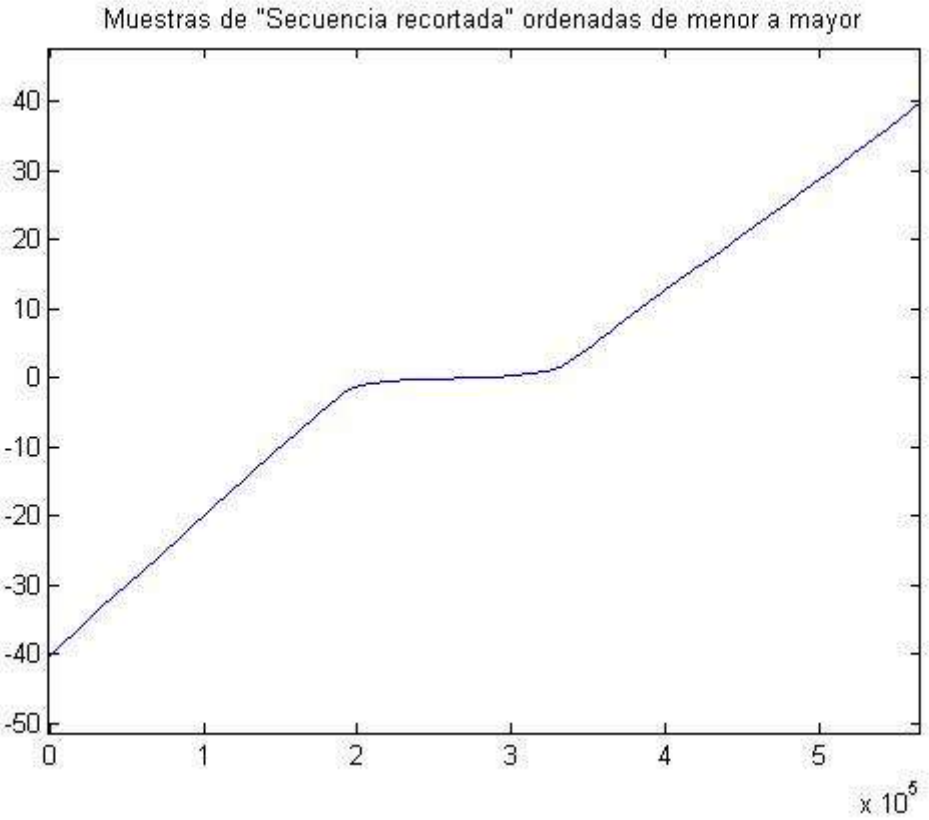


*Figura 5.11. Muestras de la señal salida recortada ordenadas de menor a mayor.*

Esta señal ya está lista para ser convertida a bits unos y ceros, por lo que la última parte del procesado va a consistir en normalizar la señal para que los valores estén entre 0 y 1:

$$Y_{-1,1} = \frac{Y}{\max |Y|}$$

El siguiente paso es llevar todos los valores entre 0 y 255:

$$Y_{0-255} = 127'5 * Y_{-1,1} + 127'5$$

Con esto conseguiremos tener la misma señal pero con sus valores distribuidos en $2^8$=256 diferentes y así con la función *dec2bin* transformar estos valores de decimal a bits unos y ceros.

**Equilibrar cantidad de unos y ceros**

Una vez se transforma el valor decimal a binario, los 4 bits más significativos cambian mucho menos que los 4 bits menos significativos. Se toman por tanto solamente los 4 bits menos significativos para la secuencia, descartando los otros 4. Esta decisión mejorará sobre todo los resultados para el test Entropy, ya que se consigue que no se repitan secuencias de bits muy seguidas. Se explica en un apartado del Capítulo 6 dedicado expresamente a ello.

En este punto los resultados de los test de frecuencia no eran los buscados en algunos casos. Se decide insertar un bucle que tenga como objetivo que la cantidad de unos y ceros sea lo más parecida posible. Se necesita que conforme se van insertando bits en la secuencia se vaya haciendo un recuento de unos y ceros para en un determinado punto equilibrar la balanza. El número de bits que se insertan hasta que se decide insertar unos o ceros (según haya ido el recuento para ese tramo), no puede ser fijo ya que empeorarían los resultados. Por lo tanto se vuelve a utilizar el vector *ruido*. En este caso se normaliza y se llevan las muestras del vector al rango 1-128. En resumidas cuentas, para cada iteración del bucle se va a tomar una componente del vector *ruido128* que va a marcar la longitud del tramo a analizar. Se va a analizar haciendo un recuento de unos y ceros. Al final del tramo analizado se inserta un 1 o un 0 según haya ido el recuento.

Ya con la secuencia de bits, lo último será crear el fichero que posteriormente se analizará con la herramienta Nist Test Suite.

En el Anexo I se encuentra el código Matlab del procesado.

## 5.2.2 Procesado ruido

### Obtención del ruido

Como se ha mencionado varias veces a lo largo de la memoria, una de las fuentes de aleatoriedad que encontramos en todas las señales con las que se va a trabajar es el ruido.

La primera parte de este procesado consiste por lo tanto en obtener la señal limpia y filtrada para así poder restársela a la original y obtener un vector cuyas muestras van a ser los valores de ruido.

Se como primera función para la señal la función *smooth*, que se encarga de "suavizar" la señal con algo parecido a un interpolado de las muestras como se observa en la figura 5.12, en la que se muestra la diferencia de una señal seno a la que se le ha introducido un poco de ruido y esta misma señal tras haberle pasado la función *smooth.*

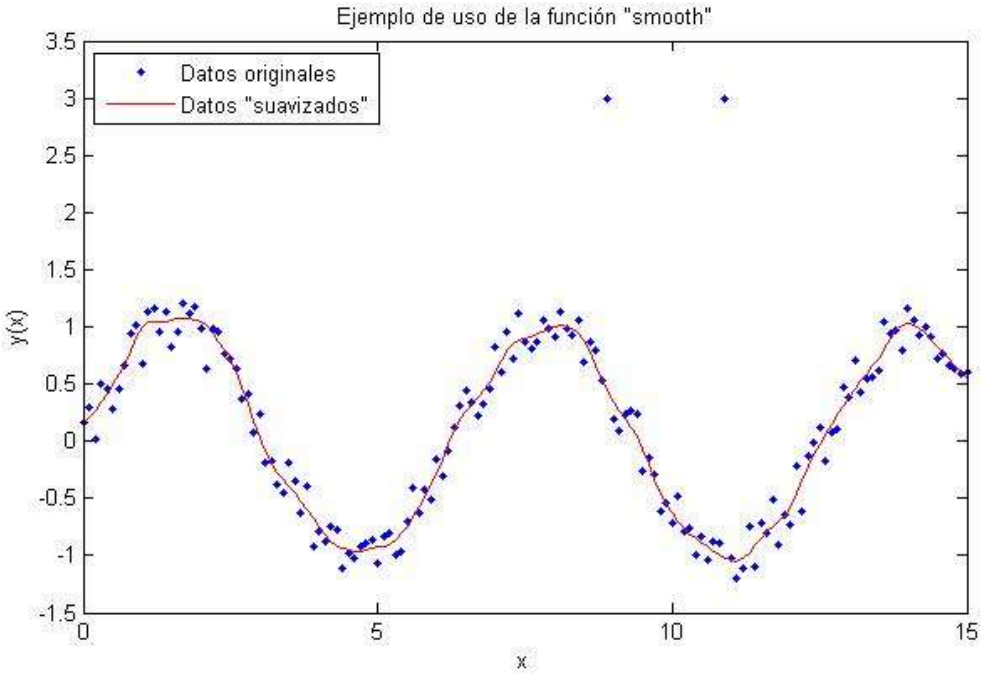


*Figura 5.12.Ejemplo función smooth.*

Matlab incluye otra función que se encarga de realizar un filtrado a la señal para intentar eliminar de esta cualquier resto del ruido que haya podido quedar, esta es la función *wdencmp*. En la figura 5.13 se observa la forma de onda de un pulso ECG antes y después de pasarle la función *wdencmp*. Los cambios son difíciles de observar ya que previamente se le había realizado la función smooth a la señal.

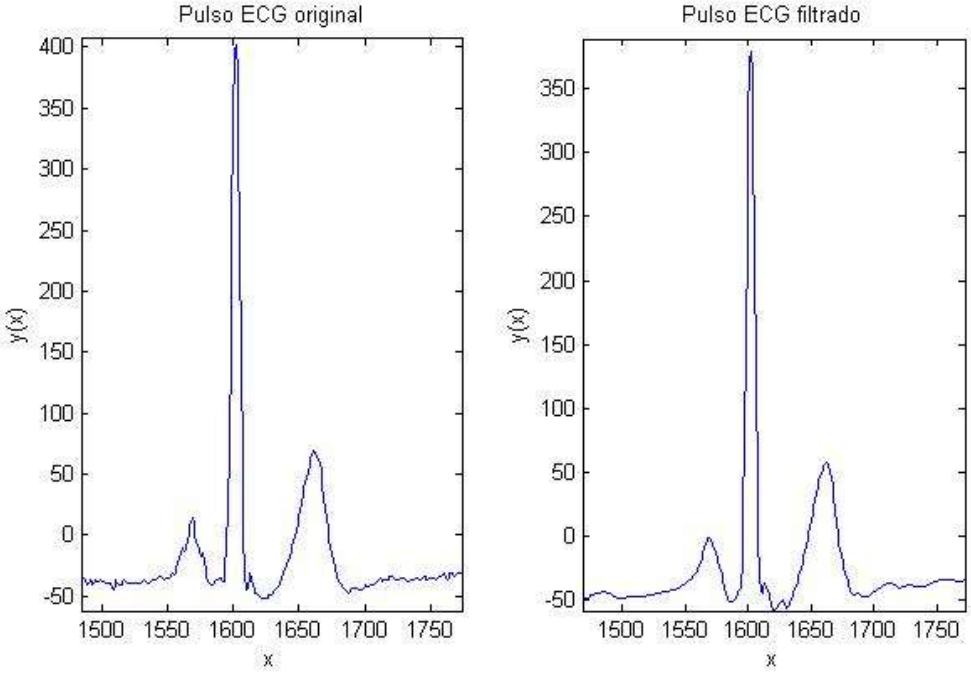


*Figura 5.13. Ejemplo función wdencmp.*

Una vez se le han pasado estas dos funciones, se obtiene la señal limpia denominada *cleanecg* en el código. Por lo tanto si lo que se quiere es obtener la parte del ruido de la señal lo único que hay que hacer es restarle a la señal original esta señal *cleanecg*. En la figura 5.14 la forma de onda de la señal ruido.

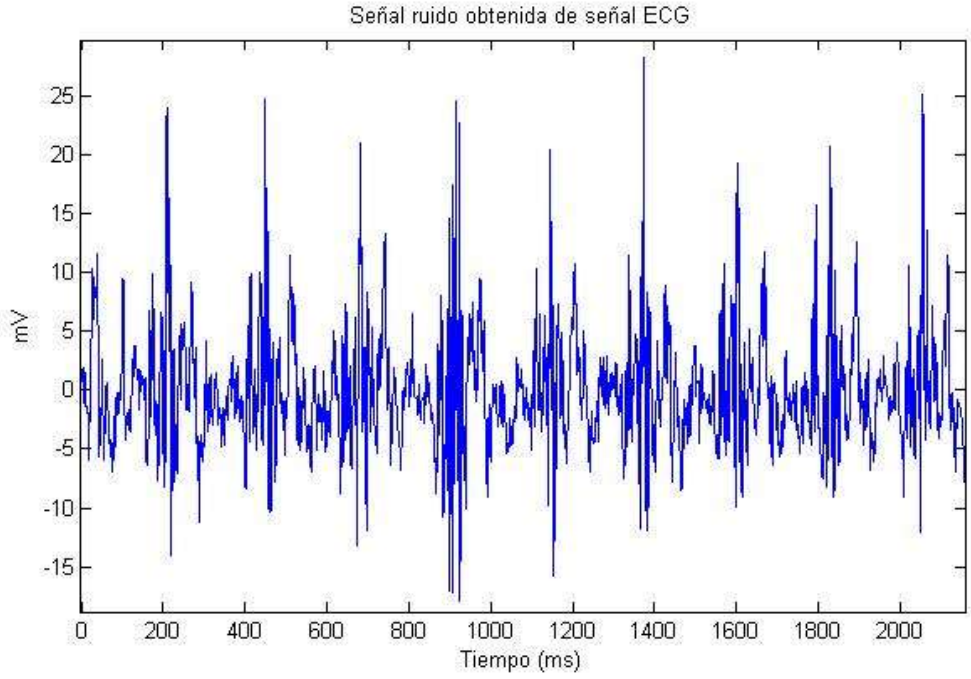


*Figura 5.14. Señal ruido obtenida de señal ECG*

Aún se puede intuir cierta periodicidad en la señal que coincide con los picos R de los pulsos PQRST. Esto hace que en la señal ruido aparezcan ciertas tendencias, tanto con pendiente positiva como negativa, que obviamente van a afectar a la aleatoriedad. Para solucionar esto, realizamos la siguiente parte del procesado.

**Eliminar media por tramos y recortar**

De nuevo se recurre a la parte final del procesado anterior. Aunque en la señal a simple vista no se observan acumulaciones o la necesidad de recortar en amplitud, si se analiza la señal haciendo "zoom" se observan las mismas irregularidades que en el caso anterior. Estos dos bucles se convierten en parte esencial para cualquier algoritmo que pretenda generar aleatoriedad, ya que las posibilidades de que la señal empeore son mínimas.

Se compara a través de las figuras 5.15 y 5.16 el cambio producido en la señal antes y después de realizarle las transformaciones propias de los dos bucles explicados en el apartado "Procesado de la señal".

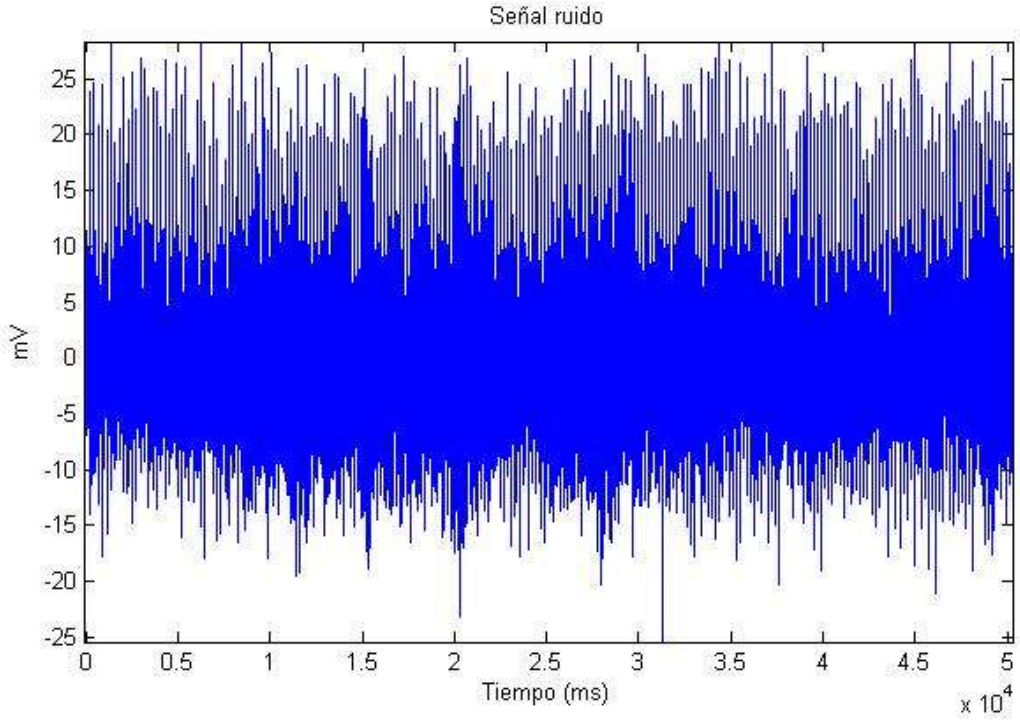


*Figura 5.15. Señal ruido.*

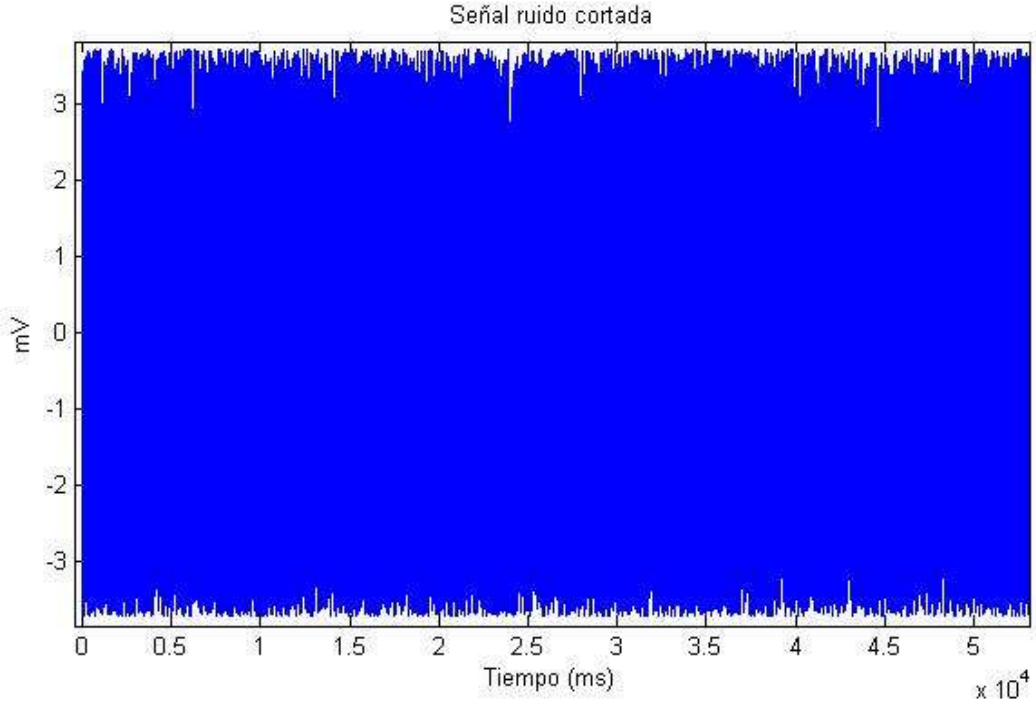


*Figura 5.16. Señal ruido cortada.*

Esta señal ya está lista para ser convertida a bits unos y ceros, por lo que la última parte del procesado va a consistir en normalizar la señal para que los valores estén entre 0 y 1:

$$Y_{-1,1} = \frac{Y}{\max |Y|}$$

El siguiente paso es llevar todos los valores entre 0 y 255:

$$Y_{0-255} = 127'5 * Y_{-1,1} + 127'5$$

Con esto conseguiremos tener la misma señal pero con sus valores distribuidos en $2^8$=256 diferentes y así con la función *dec2bin* transformar estos valores de decimal a bits unos y ceros. De nuevo se toman los 4 bits menos significativos y se descartan los 4 más significativos.

Ya con la secuencia de bits, lo último será crear el fichero que posteriormente analizaremos como dijimos con la herramienta Nist Test Suite.

Una vez redactado puede parecer que las decisiones tomadas en cuanto al procesado son muy evidentes e intuitivas, pero detrás de la solución óptima ha habido decisiones tomadas que también parecían evidentes cuyo resultado no ha sido el esperado. En cada "punto de inflexión" del procesado se han realizado numerosas pruebas y evaluaciones con la herramienta Nist Test Suite para tener una referencia objetiva de si de verdad se estaban añadiendo mejoras o no.

En el capítulo 6 se recogen los resultados en los que nos hemos ido basando para ver como mejoraba el procesado.

En el Anexo II se encuentra el código Matlab del procesado.

## 5.3 Procesado electromiografía

### 5.3.1 Procesado señal

La forma de onda de una señal EMG parece a simple vista mucho más aleatoria que la de una señal ECG. Los estímulos que reciben cualquiera de los músculos analizados (bíceps, antebrazo y pectoral) se producen más debidos a la voluntad de la persona que los que recibe el corazón. Esto se manifiesta en la periodicidad de la señal producida.

Se aprecia cierta periodicidad en la señal EMG mostrada en la figura 5.17, pero lejos de lo evidente que se apreciaba en la figura ECG. Si se observase la señal sin haber realizado un zoom como en el caso de la figura 5.15, se podría decir que se trata de una señal relativamente aleatoria lista para ser analizada. Pero obviamente se debe valorar la aleatoriedad de manera objetiva, motivo por el cual se usa la herramienta Nist Test Suite. Por lo tanto antes de analizar la señal se deben realizar ciertos procesados que tienen como objetivo hacer que la secuencia supere los test de la herramienta.

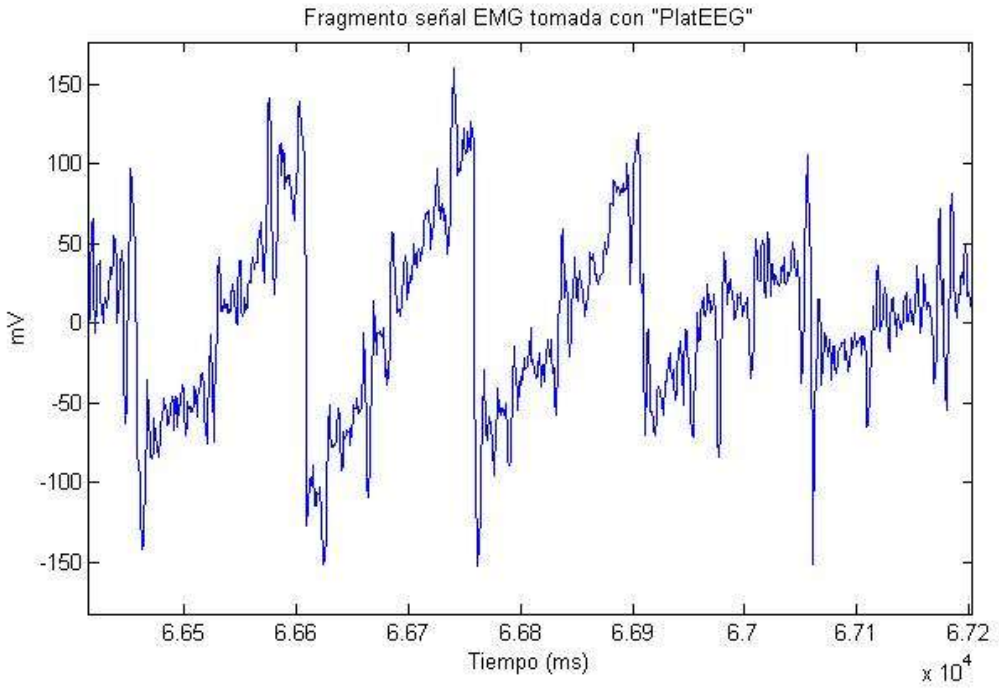


*Figura 5.17. Fragmento señal EMG tomada con Plat EEG.*

La sencillez y el bajo coste computacional es uno de los principales objetivos de este trabajo. Este procesado es el más simple de los realizados en el trabajo en cuanto al uso de operaciones o funciones.

Si anteriormente se comentó que los bucles “Eliminar media por tramos” y “Recortar la señal” eran imprescindibles en cualquier algoritmo que busque generar aleatoriedad con una señal biométrica, aquí se confirma. Estos dos procesados van a ser la única transformación que va a sufrir nuestra señal EMG.

En las figuras 5.18, 5.19 y 5.20, se puede ver la forma de onda de una señal EMG antes y después de ser procesada.

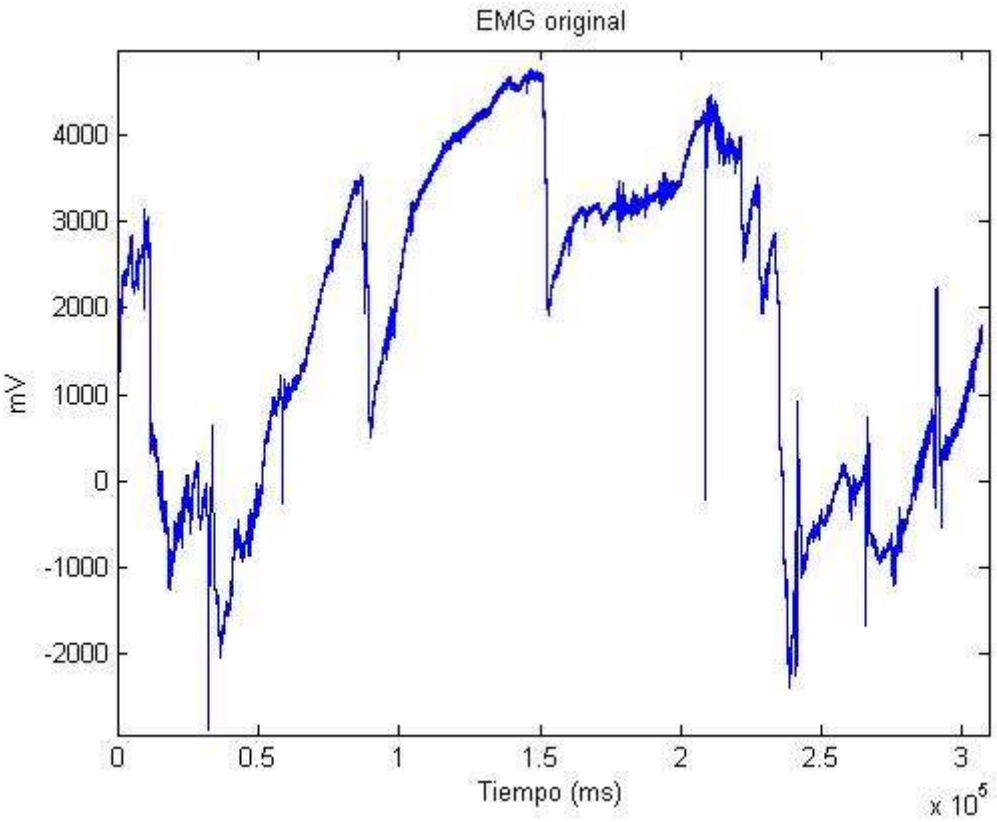


*Figura 5.18. EMG obtenida con PlatEEG.*

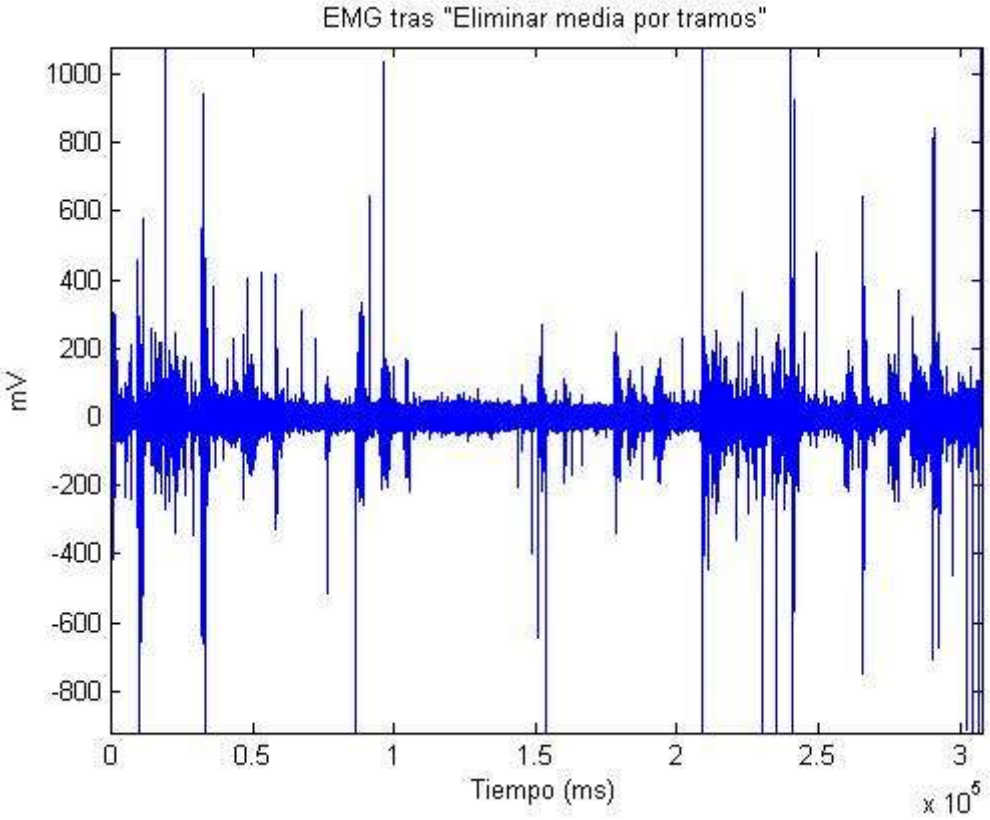


*Figura 5.19. Señal EMG tras eliminar media por tramos.*

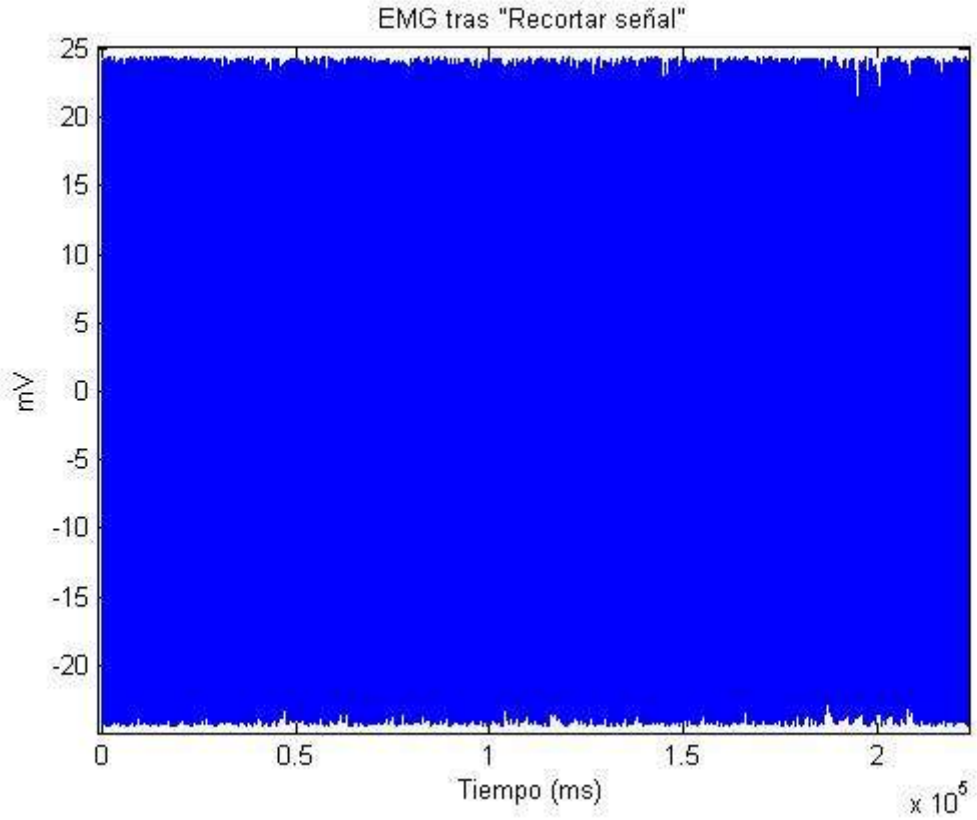


*Figura 5.20. Señal EMG tras acotar en amplitud.*

En este caso se trata de una de las señales medidas experimentalmente con *PlatEEG*. El aspecto de la señal original puede resultar extraño, pero hay que aclarar que la señal tal y cómo se muestra en la figura 5.17, se podría ver simplemente haciendo "zoom". Las componentes de baja frecuencia que le hacen tener ese aspecto se deben a acoplamientos por el cargador del portátil y demás dispositivos presentes en la toma de medidas. Estas componentes podrían eliminarse con un filtrado, pero nuestro procesado "Eliminar media por tramos" ya ofrece un buen resultado.

En el Anexo III se encuentra el código Matlab del procesado.

### 5.3.2 Procesado ruido

Otro de los objetivos de este trabajo es que un mismo procesado proporcione resultados óptimos con distintas señales biométricas. Exactamente el mismo procesado utilizado para las señales ECG, es utilizado con las señales EMG. En el capítulo 6, se analizarán los resultados obtenidos. Se muestra en las figuras 5.21 y 5.22, los cambios sufridos por la señal en el procesado.

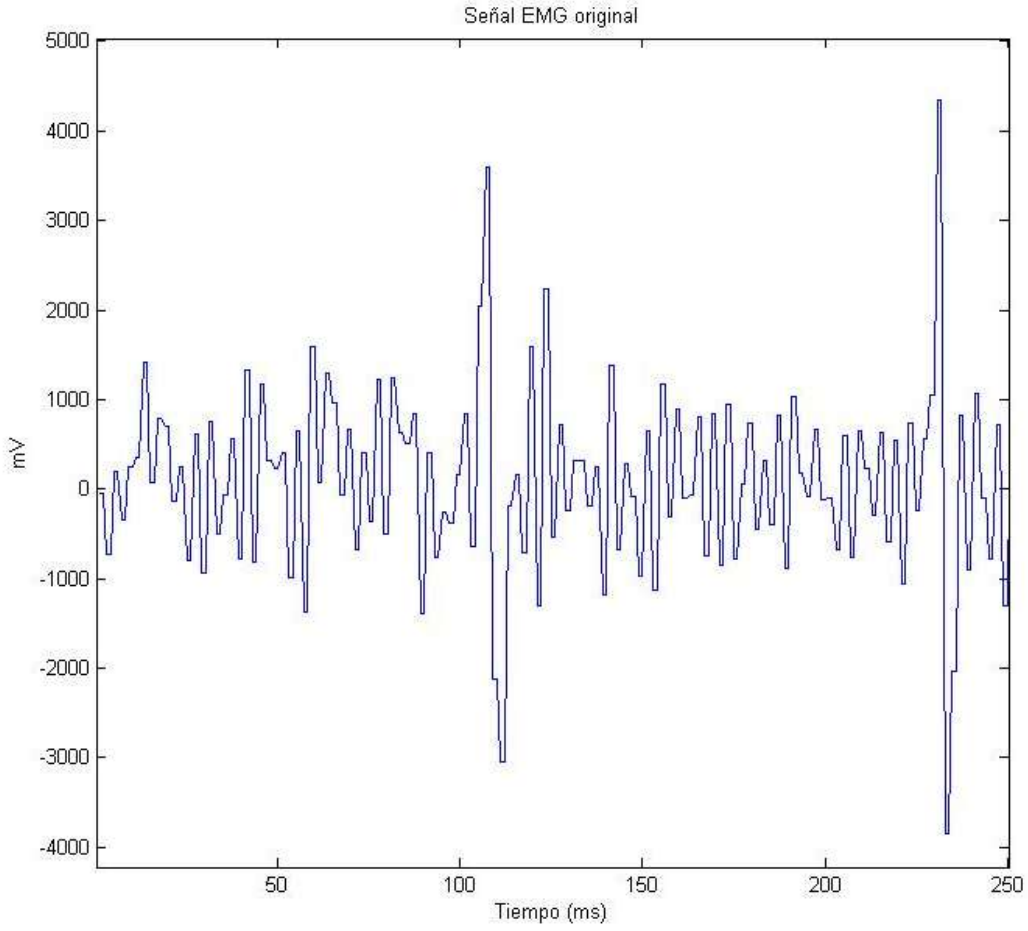


*Figura 5.21. Señal EMG original obtenida de Physionet.*

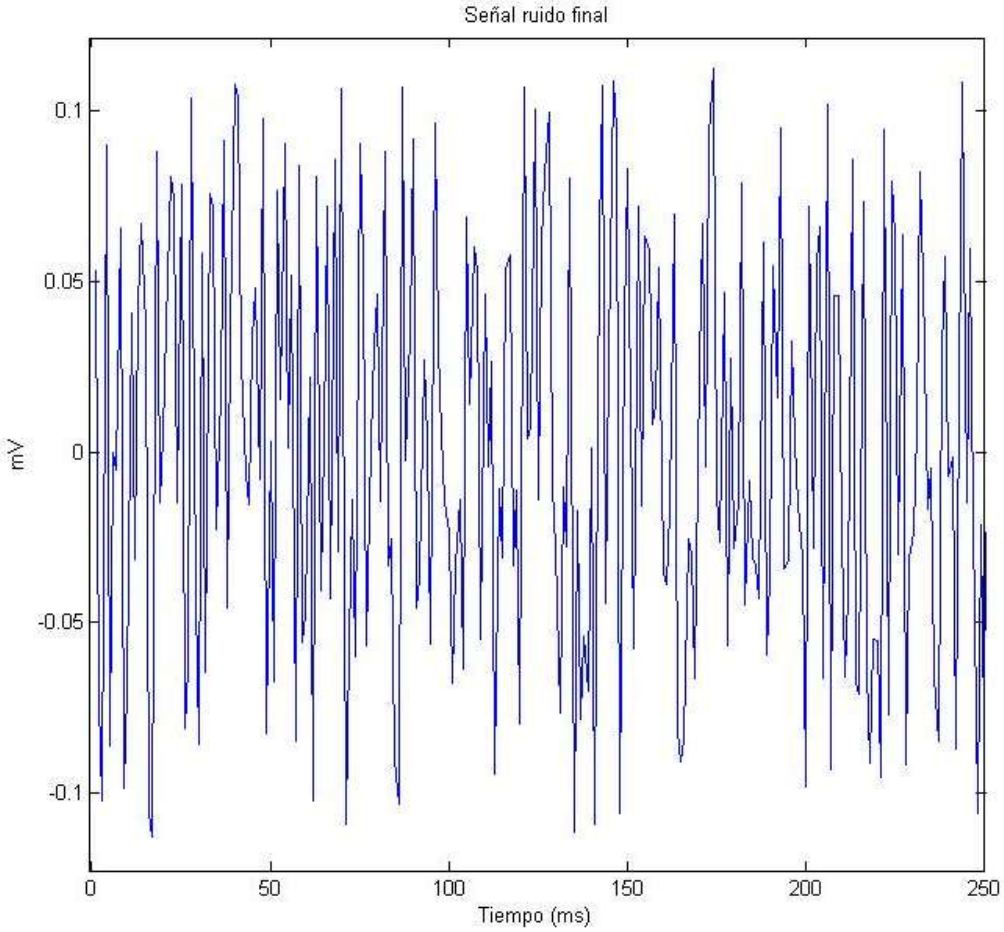


*Figura 5.22. Señal ruido de EMG.*

En este caso se utiliza una de las señales EMG obtenidas de *Physionet.org*, de ahí la diferencia que se observa si comparamos la figura 5.21 con la 5.17, que se trataba de una obtenida con *PlatEEG*. También se representan ambas señales con "zoom", ya que anteriormente se han representado "sin zoom" y merece la pena ver la forma onda así también. Se aprecia así cómo es la señal que antes parecía un rectángulo azul debido a estar muy "alejada".

En el Anexo IV se encuentra el código Matlab del procesado.

## 5.4 Procesado electroencefalograma

Es necesario en este caso tener en cuenta que cada vez que se realiza una medida del electroencefalograma, se realizan en realidad 64 medidas distintas. El "casco" utilizado para ello se compone de 64 sensores, por lo que cada uno captará una señal distinta. Se divide por lo tanto este procesado en dos casos distintos.

### 5.4.1 Procesado de la señal completa

Se toma en este primer caso como señal de entrada la señal completa. Es decir, se combinan las 64 señales de las que se dispone, una detrás de otra formando una única señal. Será por tanto una señal con una longitud de 64 veces la longitud de cada señal individual.

Una vez se tiene la señal que se va a utilizar como señal de entrada, se procede exactamente igual que en varios de los anteriores casos. El procesado consiste en la combinación de "Eliminar media por tramos" y "Recortar señal". Los últimos pasos de nuevo son normalizar la señal para finalmente generar la secuencia de bits a analizar.

Se muestra en las figuras 5.23 y 5.24 la forma de onda de la señal EEG de entrada y la señal final tras todo el procesado que generará la secuencia de bits.

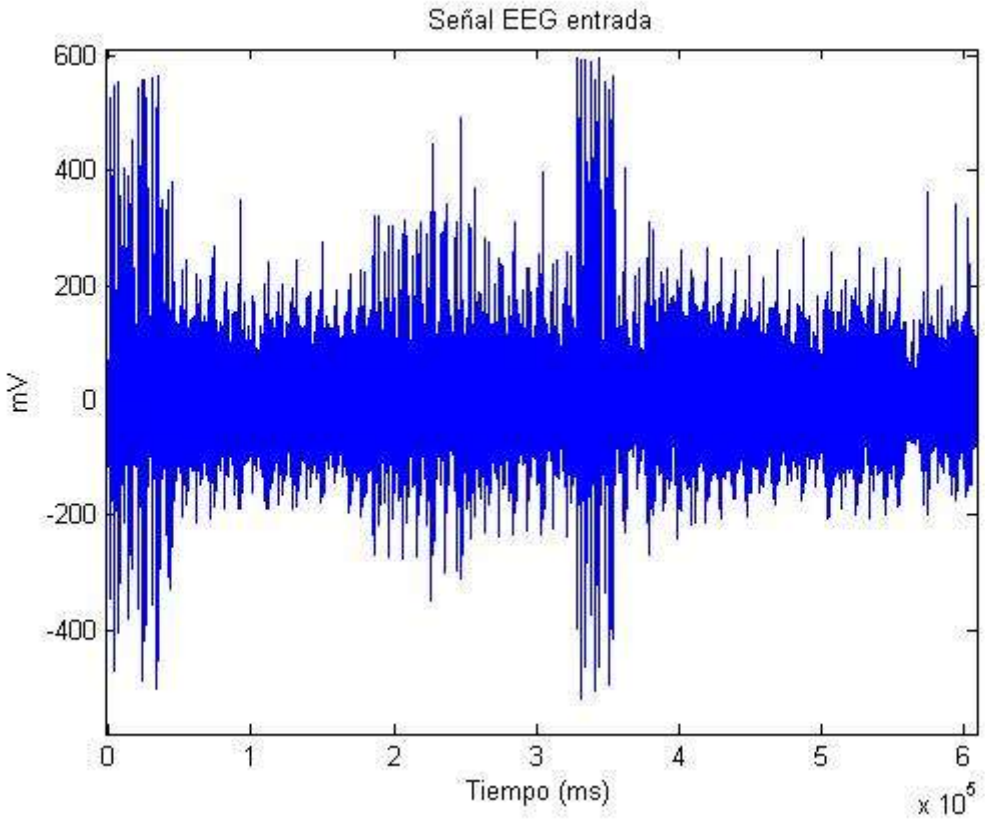


*Figura 5.23. Señal EEG utilizada como entrada.*

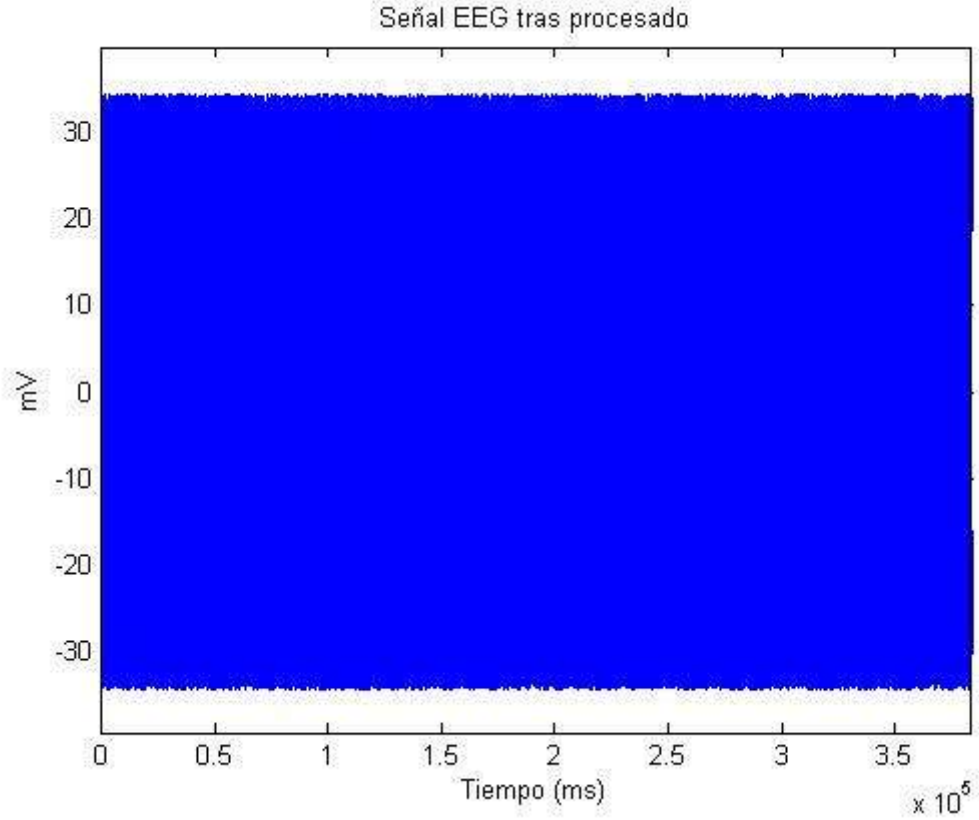


*Figura 5.24. Señal EEG tras ser procesada.*

De nuevo el resultado, en cuanto a forma de onda, es el esperado. En este caso la señal EEG no es periódica en absoluto, ya que este tipo de estímulos son propios de la actividad cerebral en cada momento pero no son algo que la persona controle fácilmente. Sin embargo de nuevo para superar los test se necesita un procesado básico.

En el Anexo V se encuentra el código Matlab del procesado.

### 5.4.2 Procesado de la señal por partes

En este segundo caso, la señal que se va a tomar como señal de entrada va a ser cada una de las 64 señales captadas por separado. Es decir se van a analizar las mismas 64 señales que en el apartado anterior se combinaron, pero ahora cada una individualmente. Con esto lo que se pretende es, a parte de comprobar cómo funcionan las señales EEG como RNG, ver si estas 64 señales están correlacionadas entre sí. Si los resultados en este caso fuesen mejores que en el apartado anterior, tendríamos indicios de que las señales están correlacionadas, pero habría que investigar más a fondo para poder afirmarlo.

Las figuras 5.25 y 5.26 representan la señal inicial y tras ser procesada respectivamente. Se muestra la señal “af3”, una de las 64 elegida totalmente al azar y sin ningún propósito concreto.

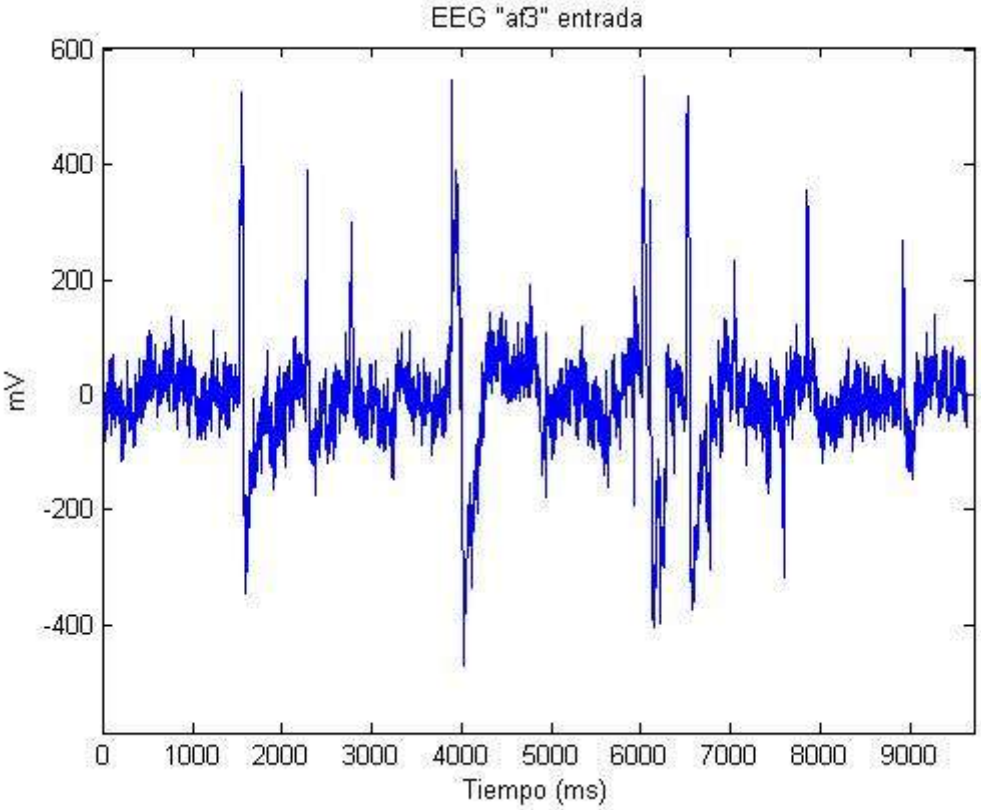


*Figura 5.25. Señal EEG sensor af3.*

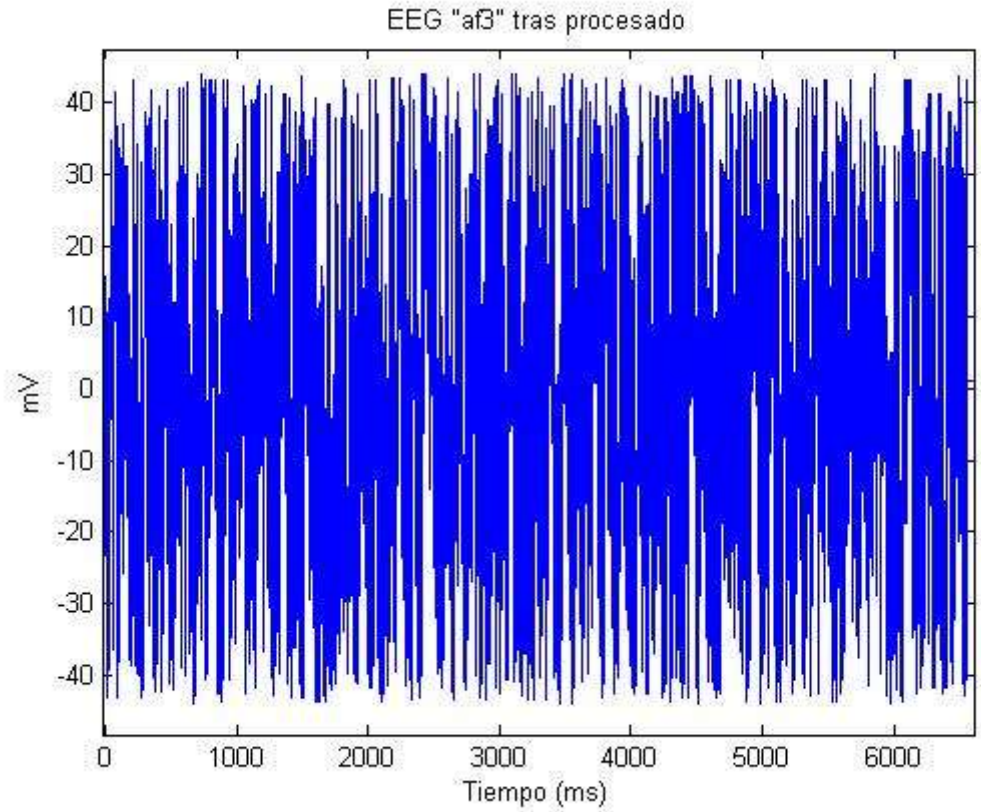


*Figura 5.26. Señal EEG sensor af3 procesada.*

Hay que aclarar tras observar la figura 5.26 que el hecho de que la forma de onda en este caso tenga un aspecto distinto a las anteriores se debe simplemente a que en este caso la señal es más corta. Se recuerda que la señal inicial para este procesado es 64 veces más corta que en el apartado que toma la señal completa. Por lo tanto no quiere decir que haya perdido ‘‘calidad’’ en cuanto a la aleatoriedad.

### 5.4.3 Procesado de ruido

Se sigue manteniendo el objetivo de un mismo procesado útil para todas las señales estudiadas, por lo tanto se vuelven a realizar pruebas con el procesado de ruido. Procesado exactamente igual al utilizado con las señales ECG y EMG. Lo único que cambia es la señal utilizada como entrada.

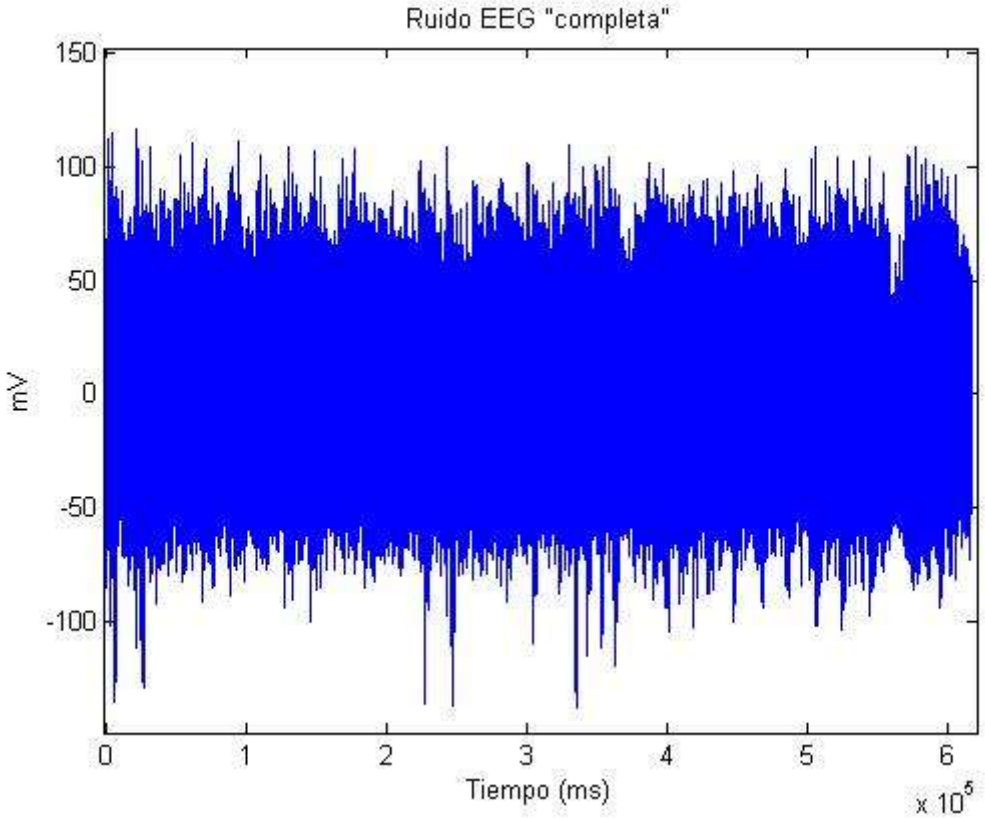


*Figura 5.27. Señal EEG ruido.*

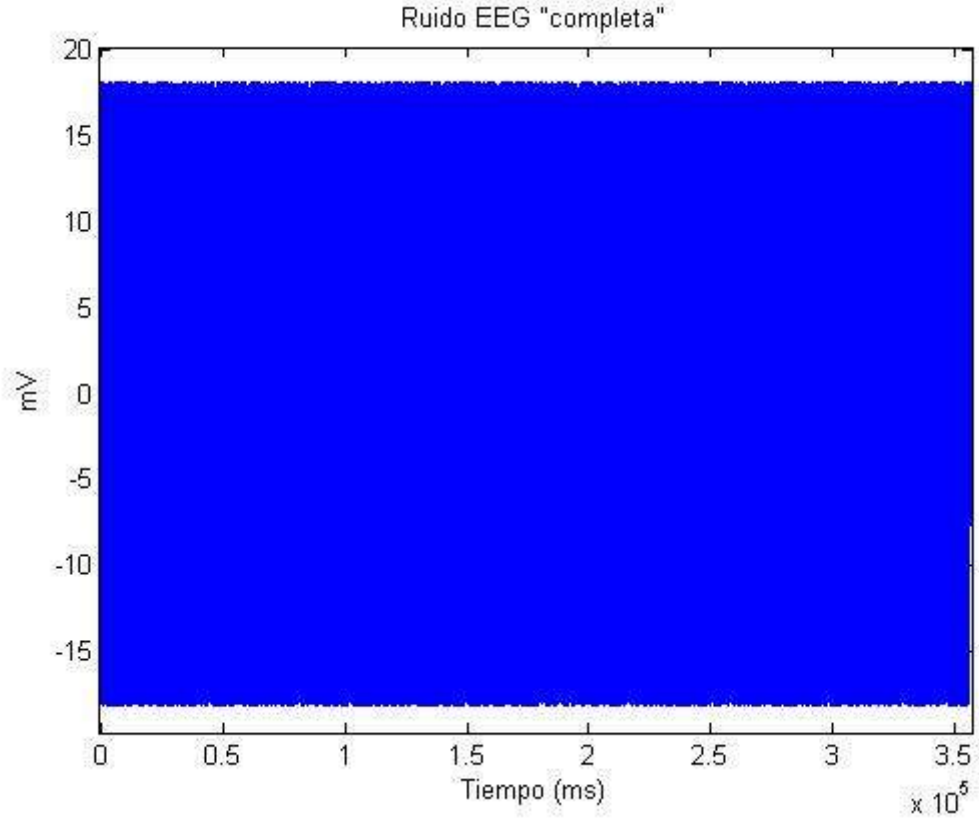


*Figura 5.28. Señal ruido EEG acotada en amplitud.*

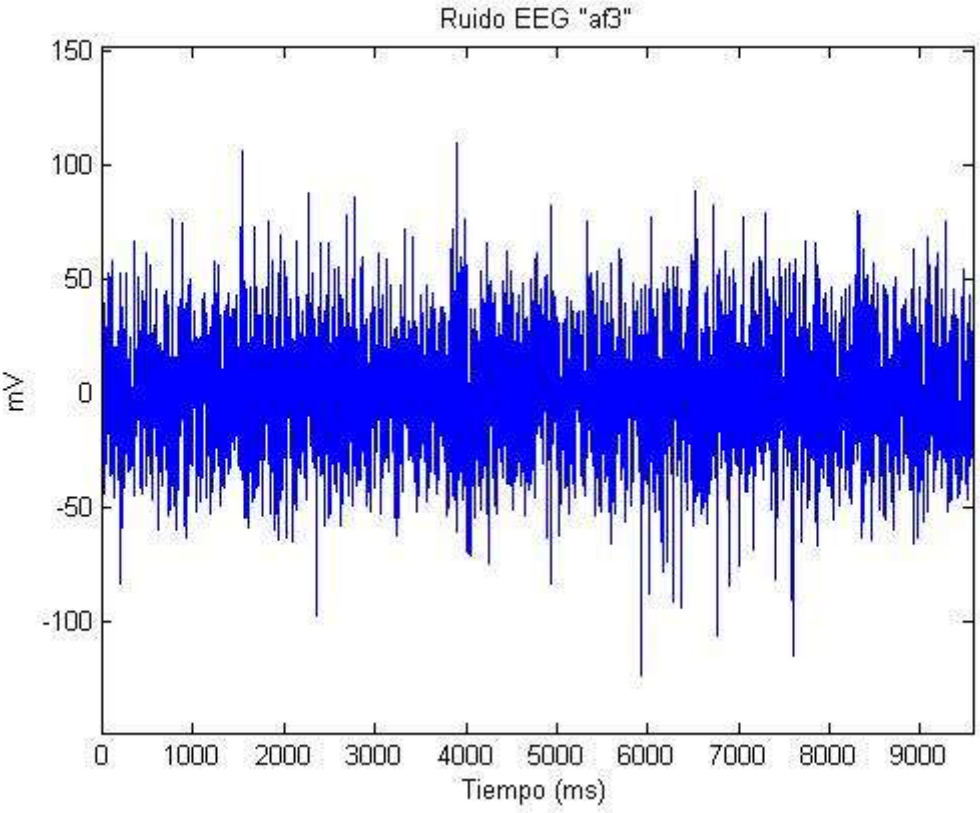


*Figura 5.29. Señal ruido EEG sensor af3 acotada en amplitud.*

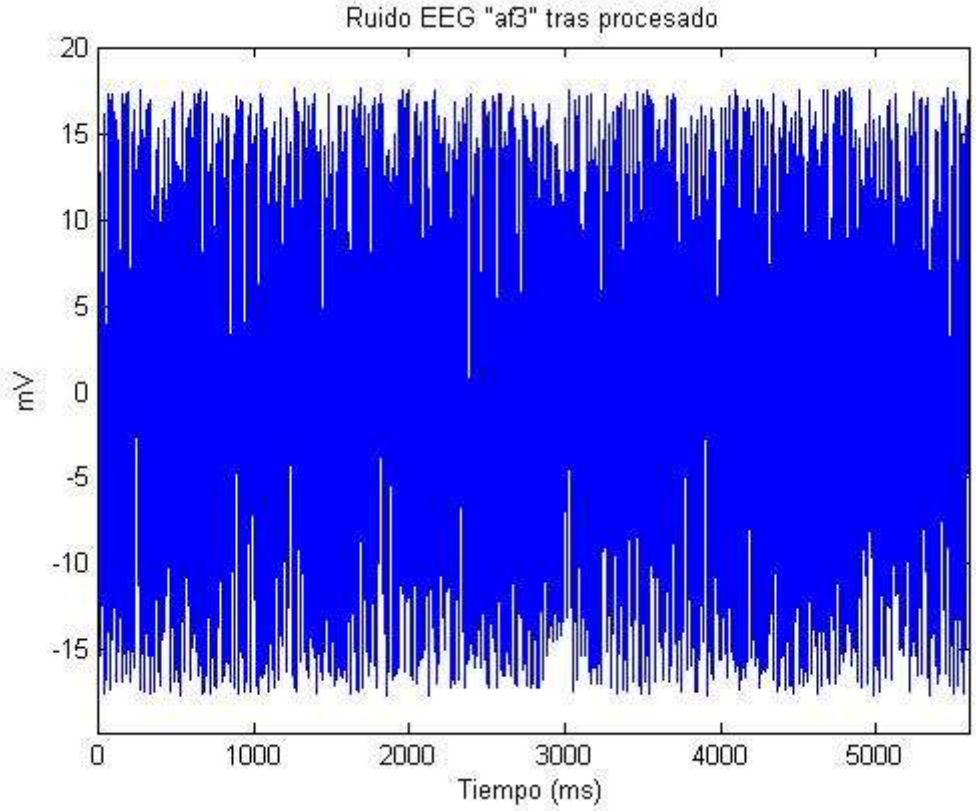


*Figura 5.30. Señal ruido EEG af3procesada.*

Se muestra de nuevo en las figuras 5.27 y 5.28 la forma de onda para la señal EEG combinación de las 64 señales captadas, antes y después el procesado. En las figuras 5.29 y 5.30, se vuelve a representar la señal "af3" en representación de las 64 posibles.

En el Anexo VI se encuentra el código Matlab del procesado.

# Capítulo 6

# Resultados obtenidos

## 6.1 Introducción

En el próximo capítulo se presentan los resultados obtenidos para los procesados expuestos en el Capítulo 7. Al igual que en el Capítulo 7, los resultados se van a clasificar según el tipo de señal y el tipo de procesado.

Se pretende que los resultados sean lo más representativos posible, por lo que se utilizan, en la medida de lo posible, gráficos en lugar de tablas.

La toma de decisiones para incluir novedades en el procesado se ha realizado en base a evaluaciones con la herramienta Nist Test Suite. En cada punto de inflexión se ha evaluado la aleatoriedad de la señal antes y después de incluir cada novedad en el procesado. Algunas de las novedades incluían mejoras y otras muchas no. Son los puntos de inflexión en los que se ha mejorado la señal los que se van a mencionar. Recordar que cuando se habla de "mejorar" la señal, significa que la señal va a generar una secuencia más aleatoria que la anterior.

En el Capítulo 7 se han evaluado las mejoras en cada parte del procesado simplemente observando las formas de onda de la señal. En este capítulo se muestran los resultados obtenidos para los test de la herramienta Nist Test Suite para evaluar de manera objetiva estas mejoras hasta llegar a la solución final.

## 6.2 Resultados electrocardiograma

### 6.2.1 Procesado señal

Se pretende evaluar la señal en todos los puntos de inflexión que se han tenido. Para este procesado el primer punto es el momento en el que se decide insertar la

amplitud cuando se generan las señales RP, RQ, RR, RS y RT. Sin embargo aunque en el Capítulo 7 se ha visto que hay mejoras, los test de Nist muestran unos valores de 0/100 en ambos casos. Esto se debe a que aún es el primer paso del procesado. Por lo tanto no se muestra ningún gráfico ya que no tiene sentido.

El siguiente punto a evaluar es la parte del procesado en la que se elimina la media por tramos. El objetivo de este procesado es eliminar acumulaciones que tenga la señal. Si por ejemplo la señal tiene un tramo en el que los valores se acumulan en un valor muy alto, el número de unos de la secuencia va a ser mucho mayor.

La figura 6.1 muestra un gráfico que compara la señal antes y después de eliminar la media por tramos. Como el objetivo principal es que el número de unos y ceros se iguale, los test que principalmente van a mejorar como se observa en la figura 6.1, son los de frecuencia.

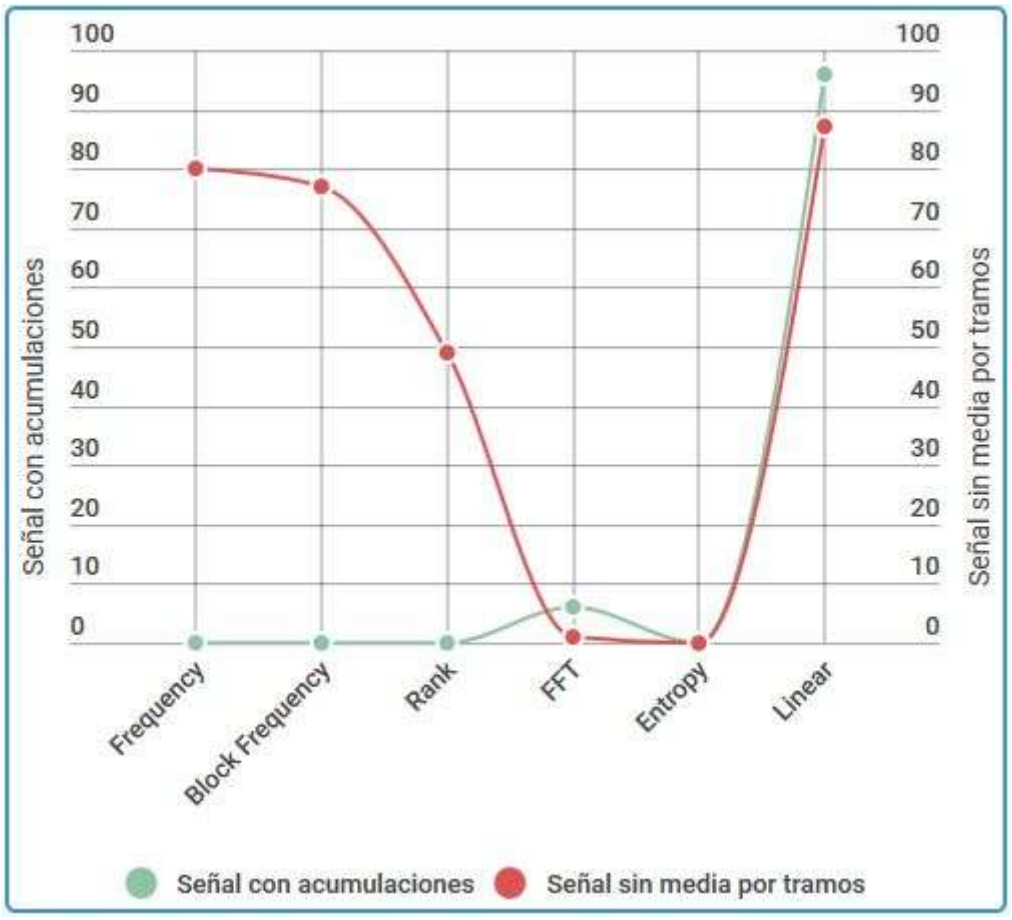


*Figura 6.1. Gráfico comparación resultados antes y después de eliminar media por tramos.*

La siguiente decisión importante realizada es acotar la señal en amplitud. Con esto se pretende mejorar la distribución de las muestras para que todos los valores del rango tengan la misma probabilidad de aparecer. Como se observa en la figura 6.2, la mejora se produce en prácticamente todos los test.

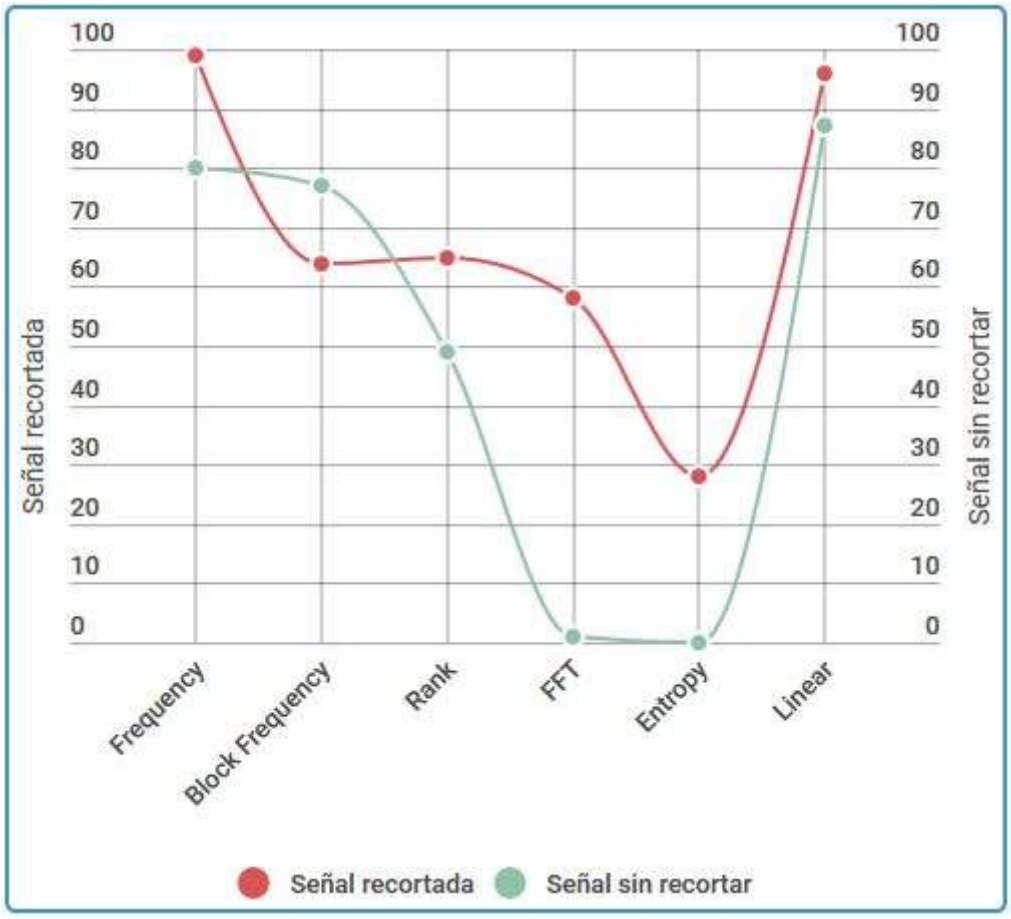


*Figura 6.2. Gráfico comparación resultados antes y después de acotar en amplitud.*

La última parte del procesado que tiene cambios realmente significativos es la parte en la que cada cierto número variable de bits de la secuencia, se realiza un conteo y se intenta equilibrar el valor de unos y ceros. Deberían mejorar los test de frecuencia, pero en la figura 6.3 se observa que no sólo mejoran estos test sino que lo hacen todos ellos. Esto se debe a que para obtener buenos resultados en los test Rank, FFT y Entropy, los test de frecuencia deben tener unos resultados prácticamente perfectos. Que el número de unos y ceros sea el mismo es uno de los requisitos principales para la aleatoriedad. Estos buenos resultados de frecuencia posibilitan que el resto de test pueda superarse con éxito.

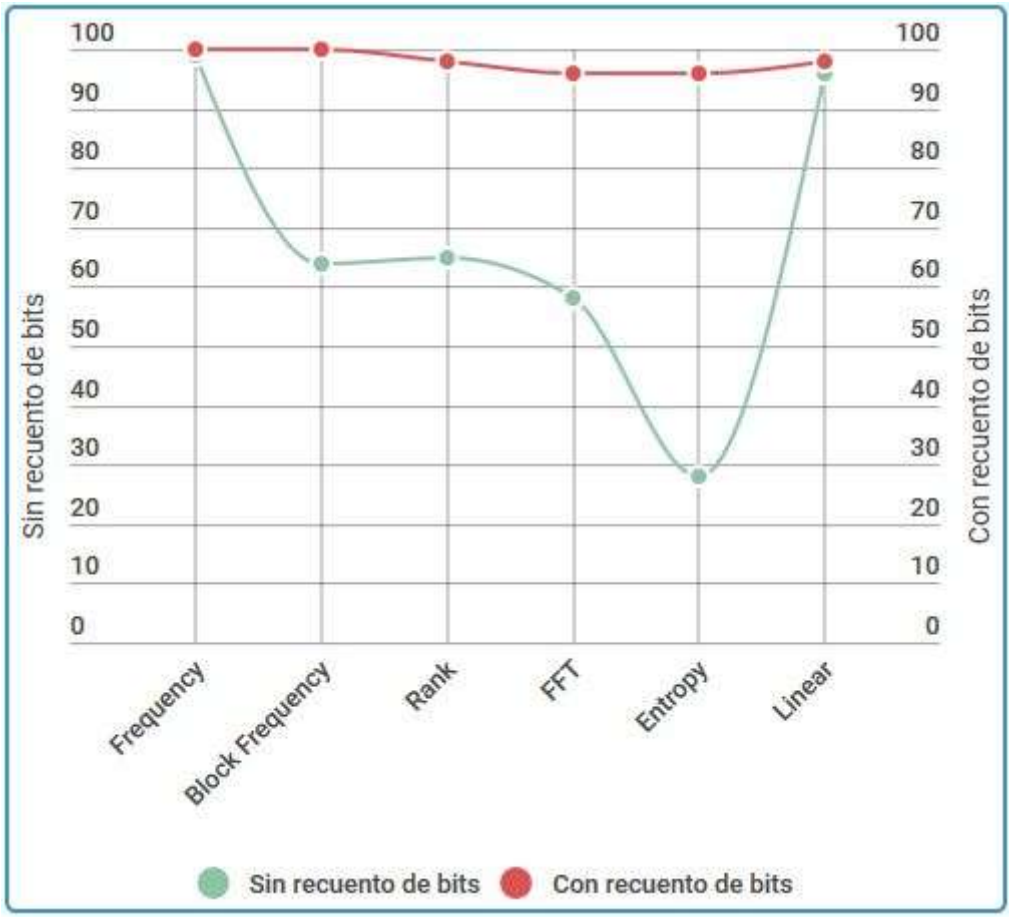


*Figura 6.3. Gráfico comparación resultados antes y después de eliminar bits más significativos.*

Por último para terminar de evaluar el procesado queda mostrar los resultados obtenidos para un gran número de señales ECG con el objetivo de dar validez estadística a los resultados. Los gráficos mostrados para terminar muestran el resultado tras realizar el procesado al completo, no resultados parciales. Un total de 86 señales ECG fueron evaluadas.

Se puede ver en la figura 6.4, el valor medio de todos los resultados obtenidos para cada test con las señales obtenidas de la base de datos Physionet.org. Resultados algo más bajos a lo esperado debido a la alta periodicidad que muestran estas señales aquí recogidas. Periodicidad que, aunque presente también, es mucho menor en las muestras tomadas experimentalmente.

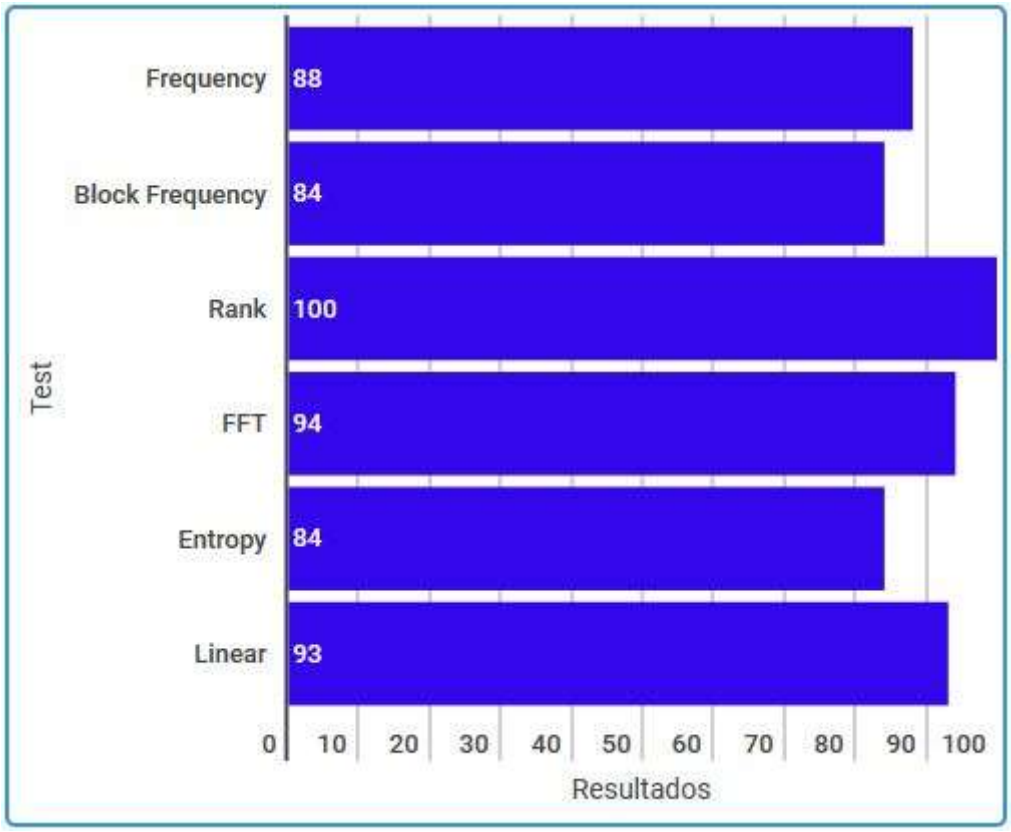


*Figura 6.4. Resultados para las señales de Physionet.*

Llegado este punto, se hace uso de las señales tomadas experimentalmente. Uno de los modos utilizado para capturar señales ECG es un sensor basado en pletismografía programado en Arduino. Otro dispositivo al que se tuvo acceso y con el que se tomaron muestras ECG es la *Plat EEG*. La figura 6.5 muestra los resultados obtenidos con el sensor de pletismografía. En este caso un total de 2 señales fueron tomadas ya que se dispuso del sensor por un tiempo limitado y se tuvieron algunas dificultades relacionadas con la luz que llegaba del exterior. De nuevo dos señales ECG de alrededor de 5 minutos cada una fueron tomadas con la *Plat EEG*. No se tuvo ningún inconveniente con esta toma de datos, sin embargo el dispositivo se utilizó un limitado tiempo.

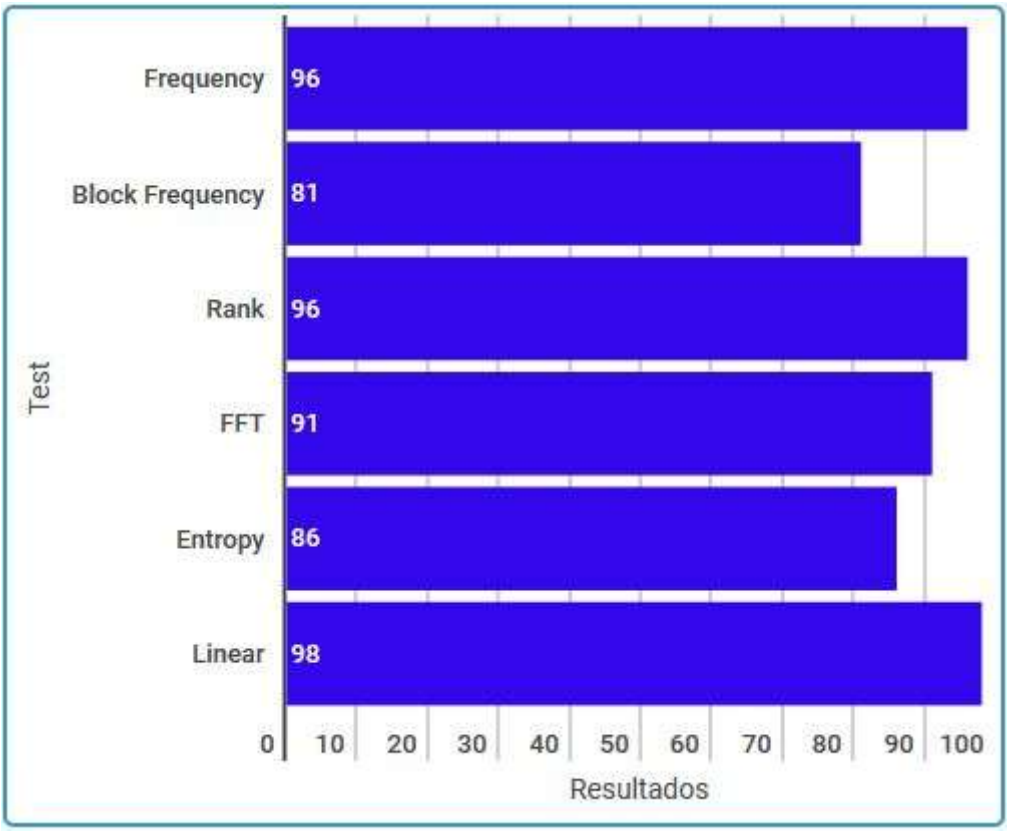


*Figura 6.5.Resultados para las señales del sensor de pletismografía.*

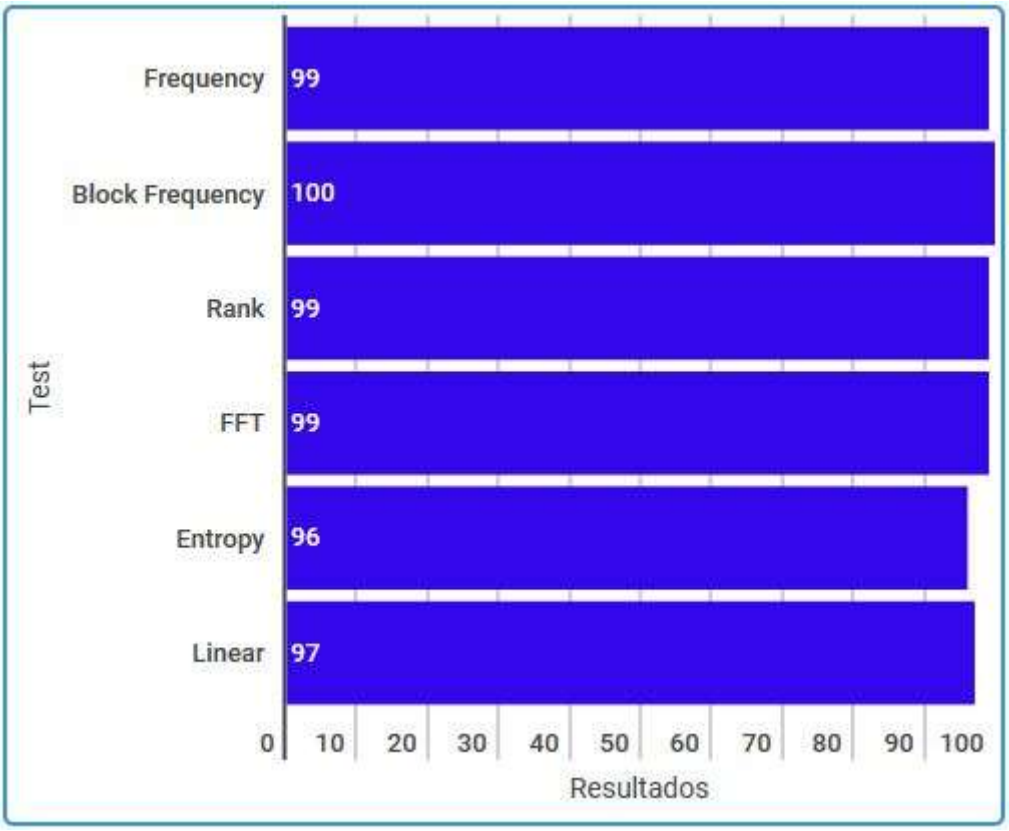


*Figura 6.6 Resultados para señales obtenidas con Plat EEG.*

Con esto, quedan expuestos todos los resultados y pruebas realizadas para evaluar este procesado. Resultados satisfactorios en la mayoría de los casos, aunque no en todos. Señalar que aunque los resultados no estén por encima del 95% en todos los casos se debe a que este es el procesado más complejo de todos los realizados en el trabajo. La complejidad se debe a que se utiliza una característica periódica de la señal para conseguir la aleatoriedad. Se observa también cómo esa "extrema" periodicidad que muestran las señales de la base de datos de *Physionet.org* afecta a los resultados, pero en la realidad esos pulsos no son tan periódicos por lo que los resultados son mejores.

### 6.2.2 Procesado ruido

Se procede a mostrar en este apartado los resultados para cada uno de los puntos donde se han producido cambios sustanciales en la aleatoriedad de la señal. Se han producido principalmente en tres puntos. La figura 6.7 recoge los resultados de los test para la señal ruido original y tras cada uno de los tres puntos mencionados.

El primer cambio importante viene cuando se elimina la media por tramos de la señal ruido. De nuevo se vuelve a equilibrar la cantidad de unos y ceros. Por lo tanto se observa en la figura 6.7 cómo mejoran los resultados con el ruido tras eliminarle la media por tramos (rojo) con respecto a la señal original (verde). Al equilibrar el número de unos y ceros los cambios más notorios se producen en los test de frecuencia.

Otra de las partes básicas de todos los procesados realizados es recortar la señal en amplitud. Se calcula un valor "límite" a partir del cual vamos a descartar las muestras que superen ese valor. Es totalmente normal que alguna irregularidad en la toma de datos haga que la señal tenga un pico de amplitud aislado. Al normalizar la señal para después llevar todos los valores entre 0-255 antes de transformar la secuencia a bits, el valor que se corresponde con 255 va a ser el de ese pico aislado ya que la señal se divide entre *max(signal).* Esto provoca que al resto de muestras, que realmente son las que importan, se les reste "importancia". La información que aportan esas muestras que realmente importan va a ser menor si el máximo es ese punto aislado. Si disminuye la información que aportan, va a disminuir la entropía. Por lo tanto uno de los objetivos principales de esta parte del procesado es hacer mejorar el test de entropía. En la figura 6.7 se observa como en efecto la entropía es uno de los test que mejora. Sin embargo no es el único que mejora, también lo hacen el test Rank y FFT. Estos dos test evalúan la dependencia linear y la periodicidad principalmente. Al eliminar muestras de la señal con este procesado también se van a romper tanto rangos de periodicidad como bloques dependientes linealmente entre sí.

Por último se evalúa cómo mejora la aleatoriedad del ruido al tomar 8 bits o solamente 4 bits para generar la secuencia. La mejora se produce en todos los test, ya que si se toman los 4 bits menos significativos, las tendencias y las acumulaciones de muestras dejarán de ocasionar problemas. En la figura 6.7 se observan estas mejoras.

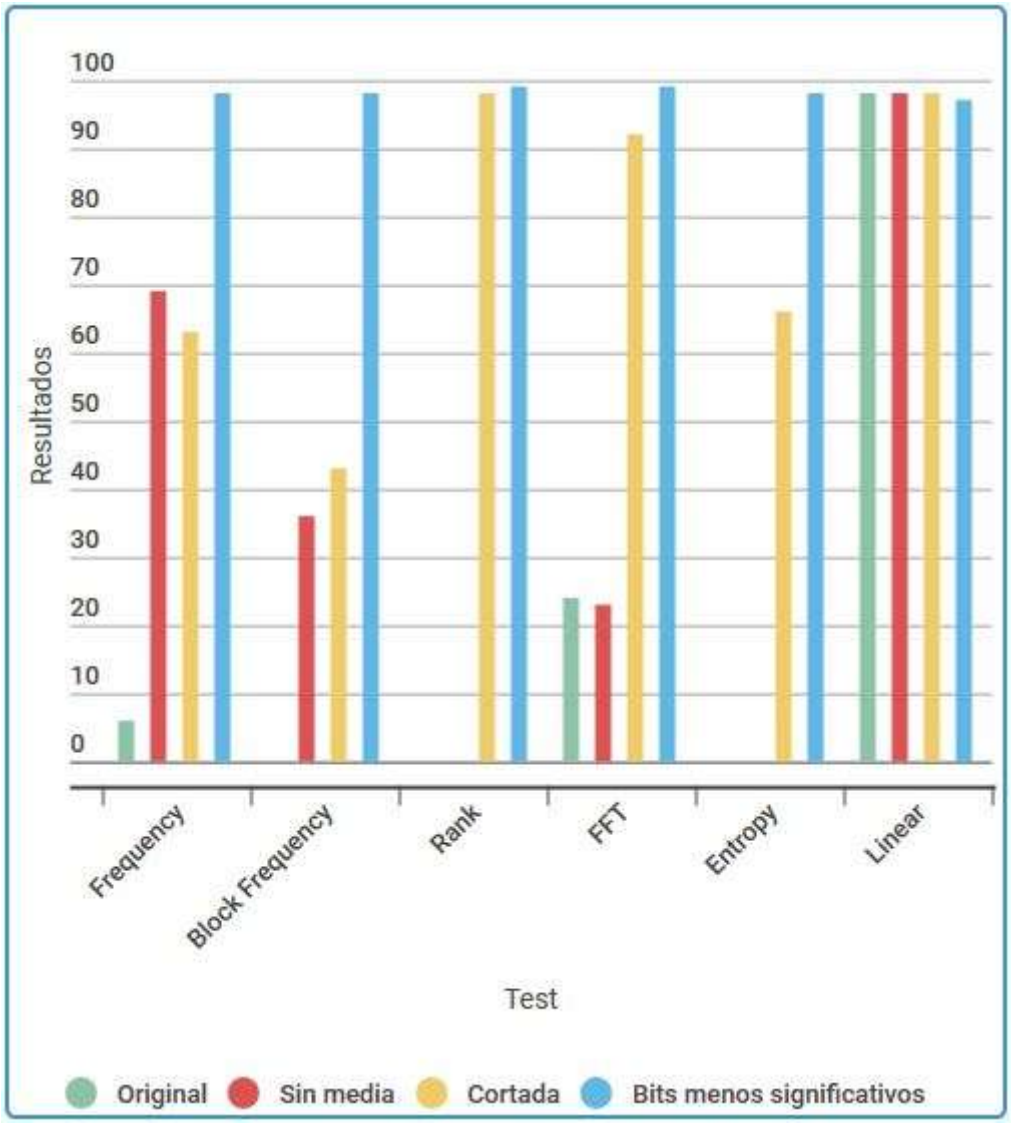


*Figura 6.7.Gráfico comparativo resultados Procesado de ruido.*

Los resultados correspondientes al color azul de la figura 6.7, muestran lo que serían los resultados finales para este procesado. De nuevo un total de 86 señales ECG fueron evaluadas. Resultados por encima del 95% para todos los test validan este procesado para este tipo de señales.

Por último, se vuelven a mostrar la evaluación del procesado con las señales tomadas experimentalmente. En la figura 6.8 se pueden observar los resultados obtenidos para las 2 señales captadas con el sensor de pletismografía. En la figura 6.9, los obtenidos con las señales captadas por el dispositivo *Plat EEG*. En ambos casos resultados totalmente exitosos que confirman los obtenidos con las señales de la base de datos *Physionet.org.*

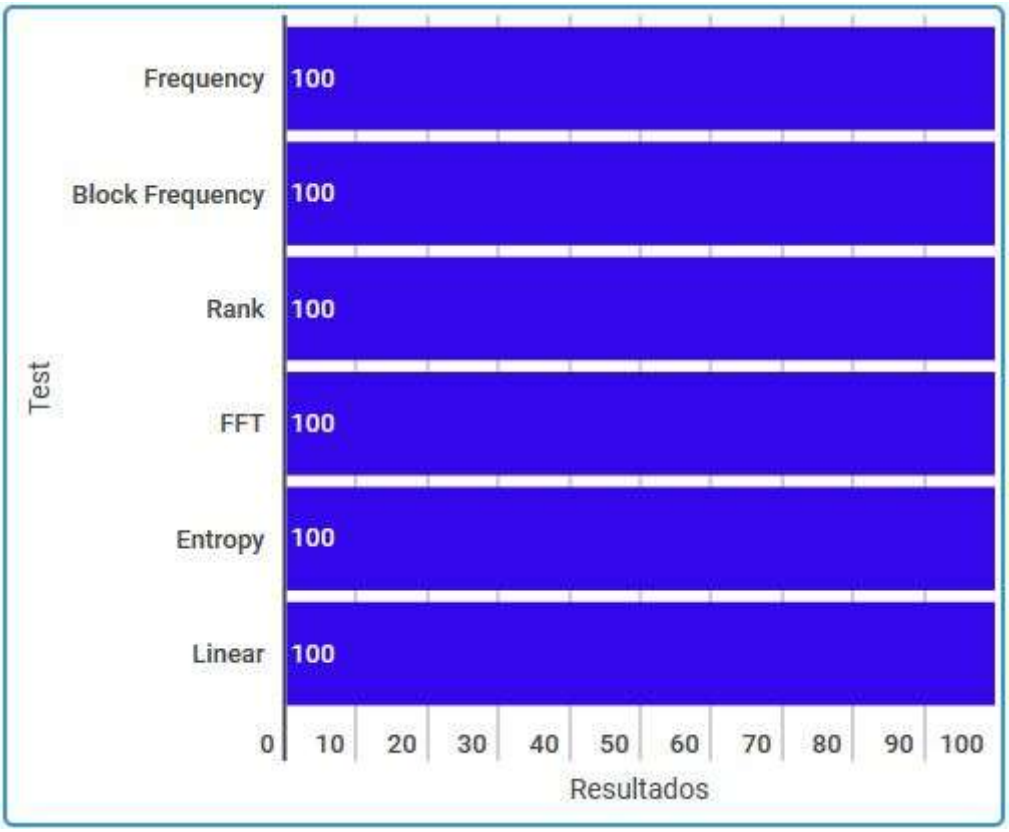


*Figura 6.8. Resultados obtenidos con señales del sensor de pletismografía.*

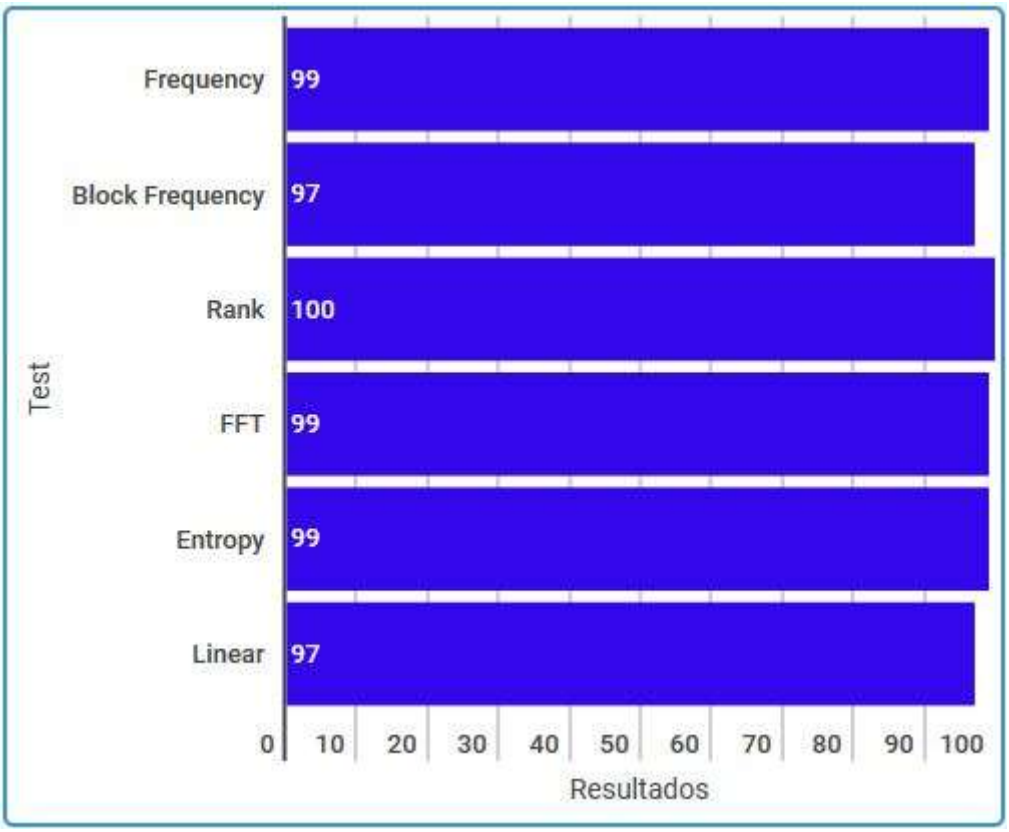


*Figura 6.9. Resultados obtenidos con señales de Plat EEG.*

# 6.3 Resultados electromiografía

## 6.3.1 Procesado señal

Se trata en este apartado la evaluación de los resultados obtenidos para las señales EMG. Las señales EMG pueden cambiar según el músculo en el que se hayan tomado las muestras y sobre todo de la actividad que se esté realizando. Las señales EMG de la base de datos *Physionet.org* son señales tomadas a personas mientras dormían. Por lo tanto son señales que cambian muy lentamente y dejan de ser útiles para este trabajo. Se toman distintas señales EMG con la *Plat EEG* tanto en bíceps como en los músculos extensores del antebrazo en dos sujetos distintos realizando algunos movimientos.

Estas señales tienen los resultados mostrados en la figura 6.10, en la que se recogen los resultados de cada test para cada una de las 4 señales distintas. Los sujetos son *s1 o s2*, mientras que el músculo es *bíceps o antebrazo*.

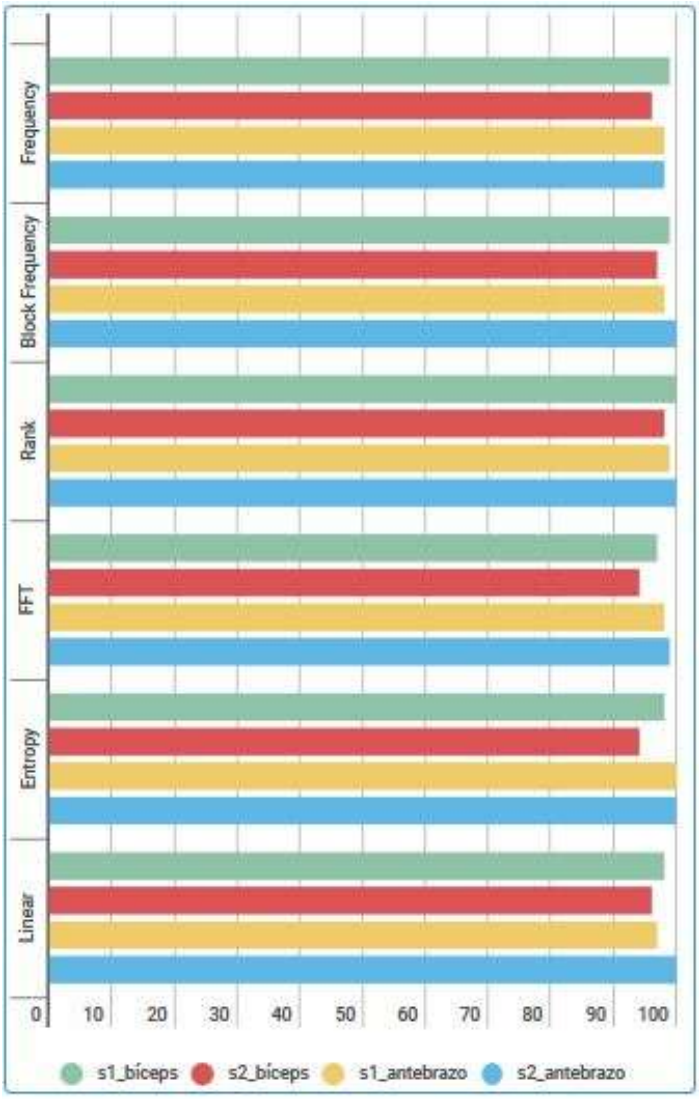


*Figura 6.10. Gráfico comparativo resultados EMG Plat EEG.*

Al observar la forma de onda de estas 4 señales no se observa ninguna diferencia importante, salvo por algún movimiento fuera de lo normal realizado. Los resultados demuestran que en efecto las diferencias en cuanto a aleatoriedad son mínimas. Resultados exitosos en todos los test para todas las señales.

### 6.3.2 Procesado ruido

En este caso, de nuevo se muestran los resultados para las señales tanto de bíceps como de antebrazo para ambos sujetos medidas experimentalmente. La diferencia es que en este caso si son válidas las señales de la base de datos *Physionet.org*, ya que sea cual sea la señal, el ruido tanto eléctrico como por interferencias que se introduce va a ser similar. Este procesado utiliza el ruido como se ha explicado, por lo tanto son señales útiles.

La figura 6.11 muestra los resultados de las señales tomadas con el dispositivo *Plat EEG*. Resultados que de nuevo muestran que las diferencias entre músculos o individuos no importan ya que en todos los casos se superan los test exitosamente. Resultados muy similares los recogidos en la figura 6.12, que resume la evaluación del procesado realizada con un total de 30 señales EMG obtenidas de *Physionet.org.*

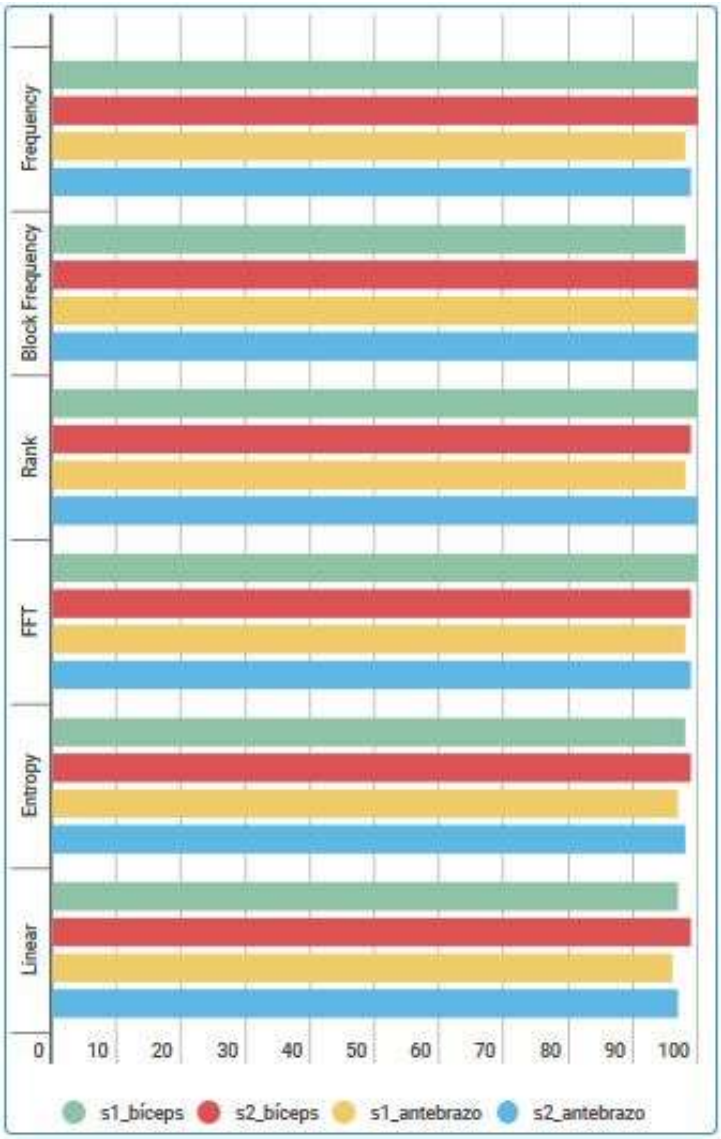


*Figura 6.11. Gráfico comparativo resultados ruido EMG Plat EEG.*

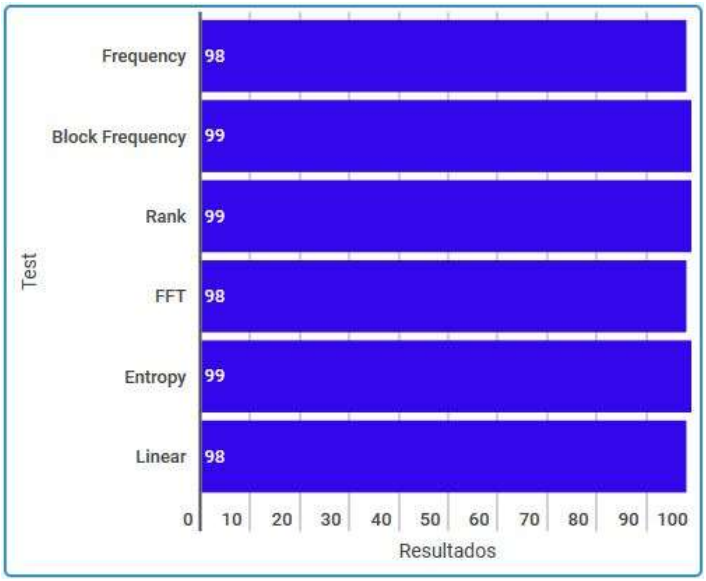


*Figura 6.12. Resultados ruido EMG.*

## 6.4 Resultados electroencefalograma

### 6.4.1 Procesado señal

Se realizó una excepción en cuanto a la línea de trabajo seguida y para este procesado se divide en dos: uno que procesa las 64 señales individualmente y un segundo que procesa una señal que combina las 64 señales posibles.

El objetivo de evaluar las 64 señales que captura el dispositivo utilizado por *Physionet.org* era ver de qué manera cambiaban los resultados según la zona. La figura 6.13 muestra los resultados por zonas. Se han utilizado gráficos radiales en los que cada color se corresponde con un test distinto. Para adaptarlo al *Sistema 10/20* se ha calculado la media por zonas para que sea más fácil observar los resultados. De la misma forma, para saber de qué zona se trata, se calcula la media para los puntos simétricos y se muestran los resultados solamente en media cabeza.

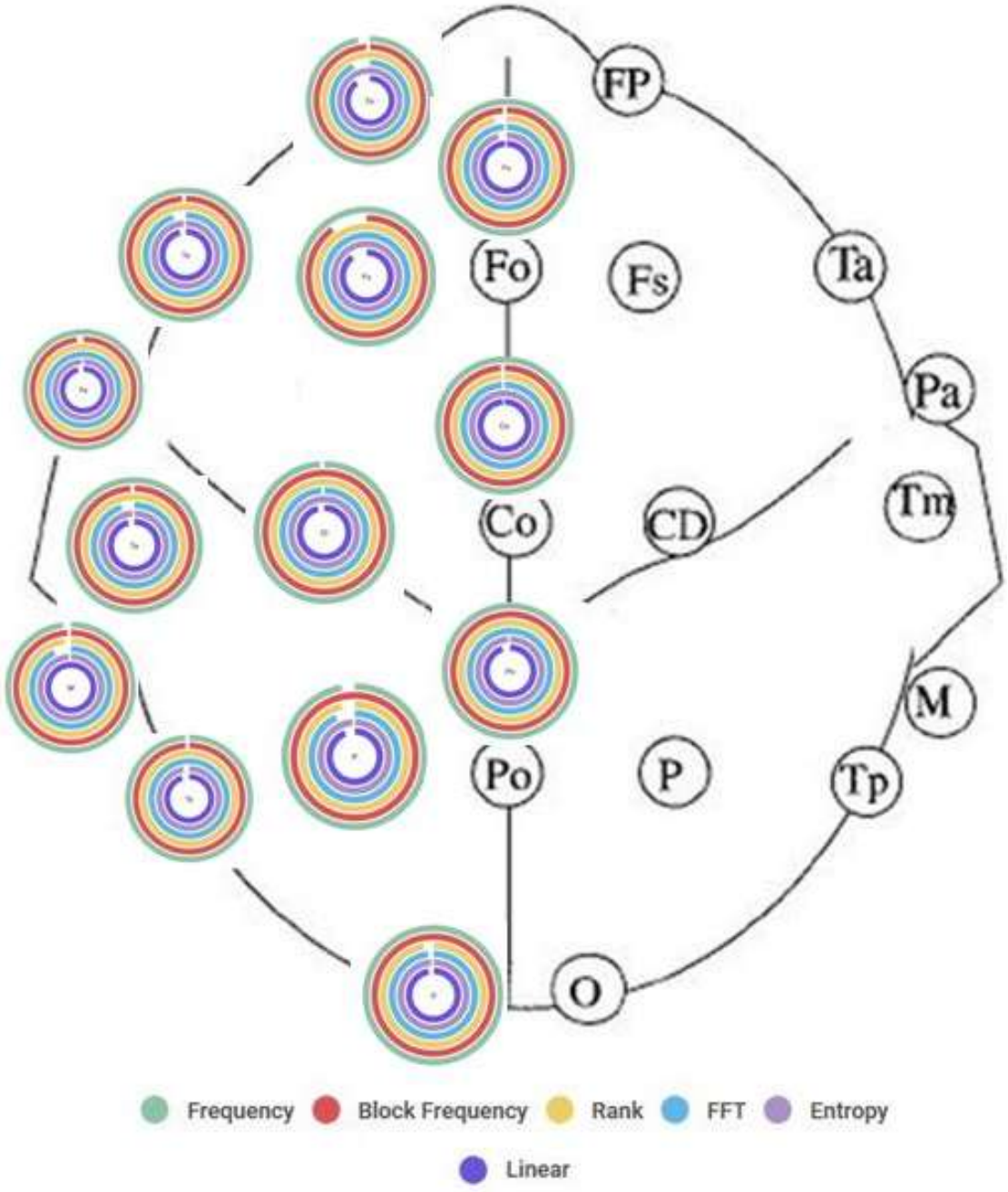


*Figura 6.13. Resultados EEG dividido por zonas.*

Los resultados son satisfactorios por igual, por lo que para el trabajo las señales son tan útiles sea cual sea la zona donde se hayan medido. Se realizan por lo tanto pruebas con una señal compuesta por las 64 individuales que contiene cada señal tomada de *Physionet.org.* Se toman un total de 30 señales de esta base de datos para dar validez estadística a las pruebas. La figura 6.14 recoge los resultados obtenidos para cada test.

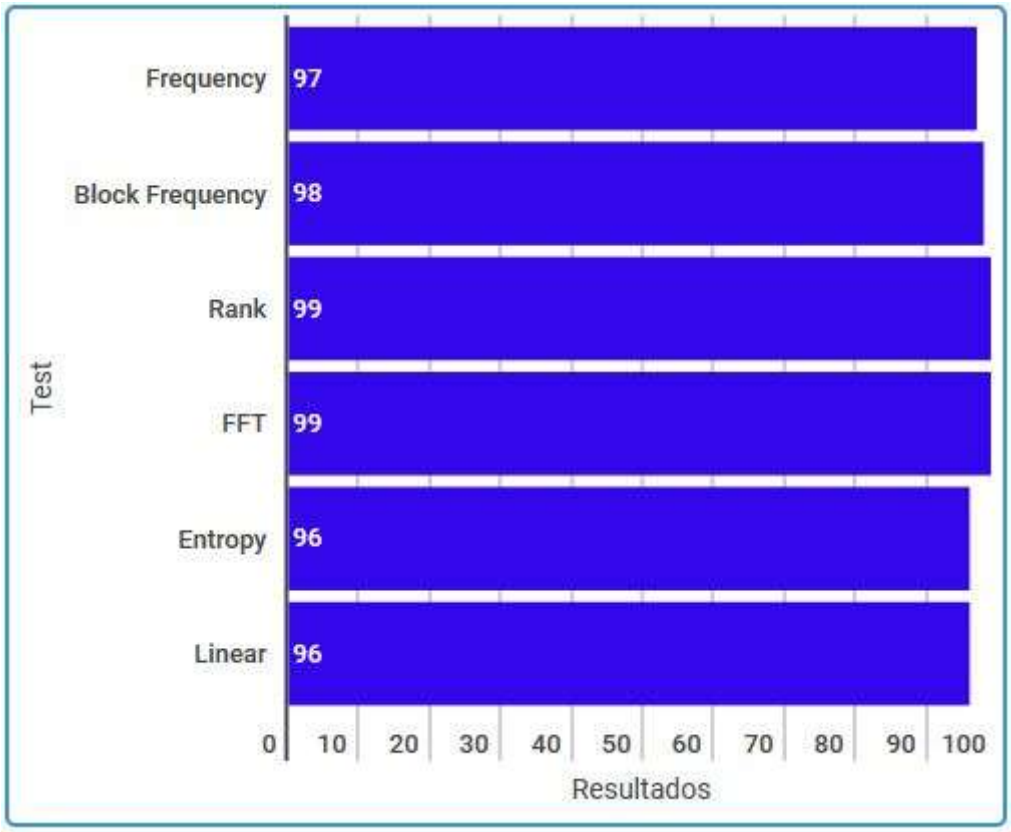


*Figura 6.14.Resultados EEG completa.*

Resultados que satisfacen el objetivo ya que superan todos los test con éxito. Tanto tomando las señales individuales como combinándolas en una los resultados son los deseados. Sin embargo evaluar las señales por zonas ha sido simplemente otra forma de ver los resultados ya que una solución más óptima será la completa. Una señal que va tomando muestras a la vez de los 64 sensores va a generar una clave con una velocidad 64 veces mayor.

### 6.4.2 Procesado ruido

Se ha llegado a la conclusión de que utilizar la señal combinación de las 64 individuales es la solución más óptima, por lo que en este apartado se evaluará este caso solamente. La figura 6.15 muestra los resultados de las pruebas realizadas en la que se observa que todos los test superan exitosamente los test propuestos.

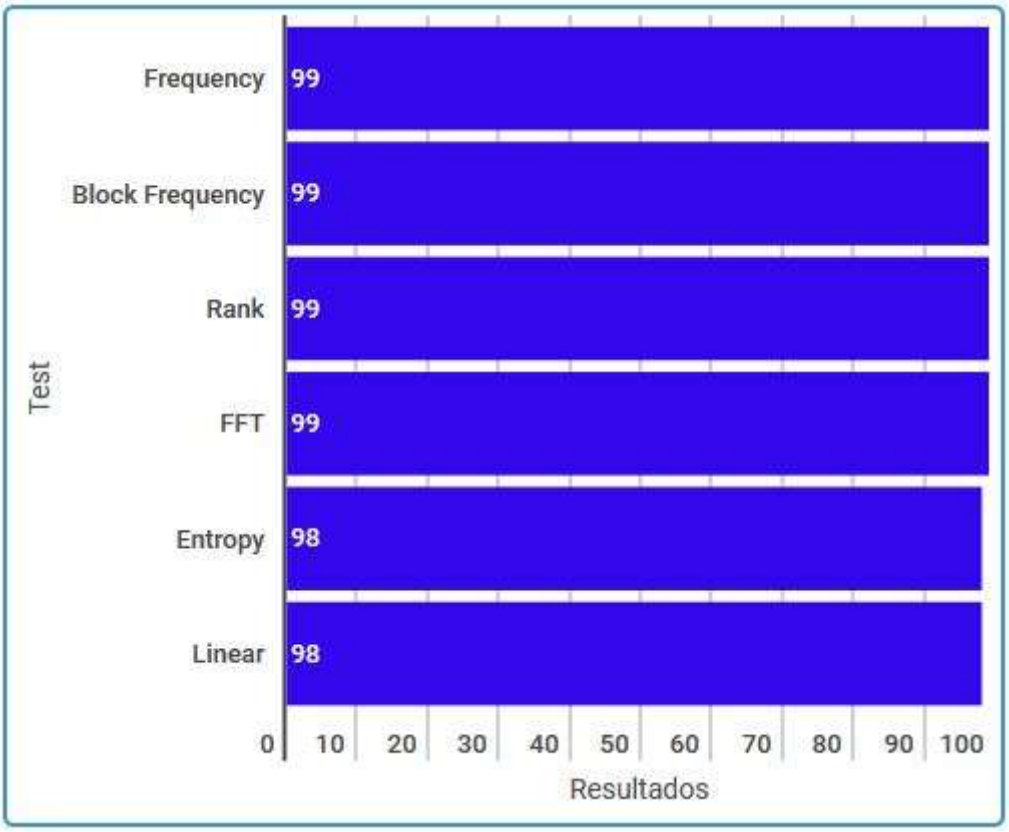


*Figura 6.15.Resultados ruido EEG completa.*

## 6.5 Resultados a tener en cuenta para el test Entropy

Sin lugar a dudas, el test *Entropy* es el más difícil de superar. En muchos casos se han obtenido resultados del 100% para los test *Frequency, Block Frequency, Rank, FFT y Linear*, y para el test *Entropy* un resultado del 0%. Por lo tanto se ha estudiado especialmente a fondo este test y merece la pena dedicar un apartado a mostrar lo obtenido.

### 6.5.1 ¿Por qué se descartan los 4 bits más significativos?

El hecho de no tener en cuenta los 4 bits más significativos supone una pérdida importante ya que se descartan la mitad de los bits de la secuencia. Sin embargo los resultados mejoran lo suficiente como para afrontar esta pérdida.

El test *Entropy* analiza el número de veces que se repiten *m* bits consecutivos dentro de una secuencia de longitud *M*. El valor de *m* es 8 por defecto, aunque se puede modificar. Por lo tanto el test va a contar el número de veces que se repite el mismo fragmento de 8 bits. Si se insertan en la secuencia 8 bits por cada muestra (los 4 más significativos más los 4 menos significativos), esto se traduce en que el test va a contar el número de veces que se repite una muestra de la señal. Sin embargo, si por cada

muestra sólo se toman los 4 bits menos significativos, el test sigue analizando la secuencia con *m=8.* Significa que el test cuenta el número de veces que se repiten dos muestras consecutivas de la señal. Esta situación es mucho menos probable por lo que los resultados van a mejorar considerablemente.

Se estudió la posibilidad de utilizar solamente los 2 bits menos significativos. Sin embargo como se observa en la figura 6.16, los resultados son lo suficientemente buenos con 4 bits, y el cambio de 4 a 2 no es suficiente para compensar la pérdida de bits que supone. Los resultados que recoge la figura 6.16, se han obtenido con la *señal ruido* para señales ECG. Se estudió para las señales ECG y a partir de ese momento se utilizaron 4 bits para el resto de señales estudiadas.

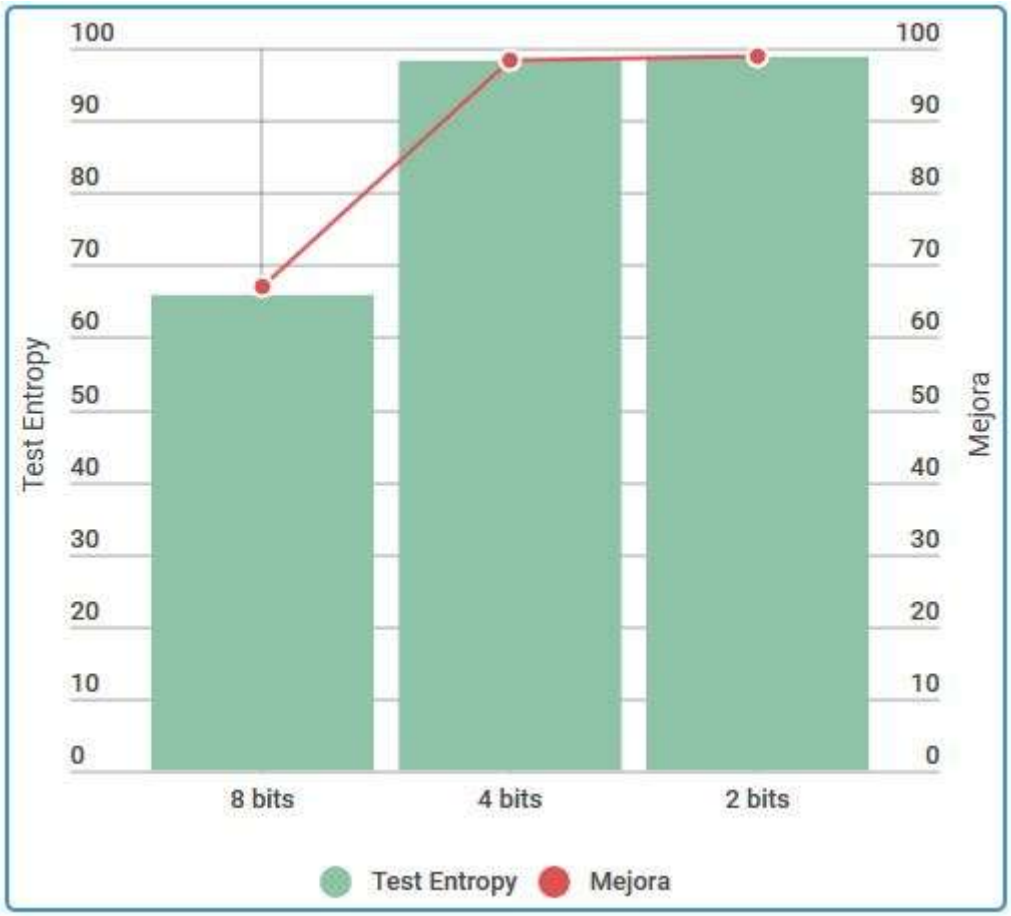


*Figura 6.16. Gráfico comparativo Test Entropy según el número de bits tomados.*

## 6.5.2 ¿Por qué no aumentar el valor de *m*?

Se pensó en la posibilidad de cambiar el valor de *m*. Si se aumentase *m*, el número de veces que se repite una determinada secuencia va a ser menor ya que es menos probable. Se incrementó y efectivamente los resultados mejoraban. Sin embargo, al indagar en los archivos que genera Nist Test Suite como resultado, se observó que si se

aumenta el valor de *m* por encima de cierto valor los resultados dejan de ser válidos. Este valor máximo para *m* cambia según el tamaño de secuencia a analizar. Aumentar el valor de *m* deja de ser una solución.

El valor de *m* es inversamente proporcional al resultado obtenido. Si se consigue que el número de repeticiones que detecta el test se mantenga para valores de *m* cada vez más pequeños los resultados van a mejorar. Se pretende evaluar el valor para *m* más óptimo en cuanto a resultado del test *Entropy*. Se observa en la figura 6.17 el resultado para el test *Entropy* para todos los posibles valores de *m.*

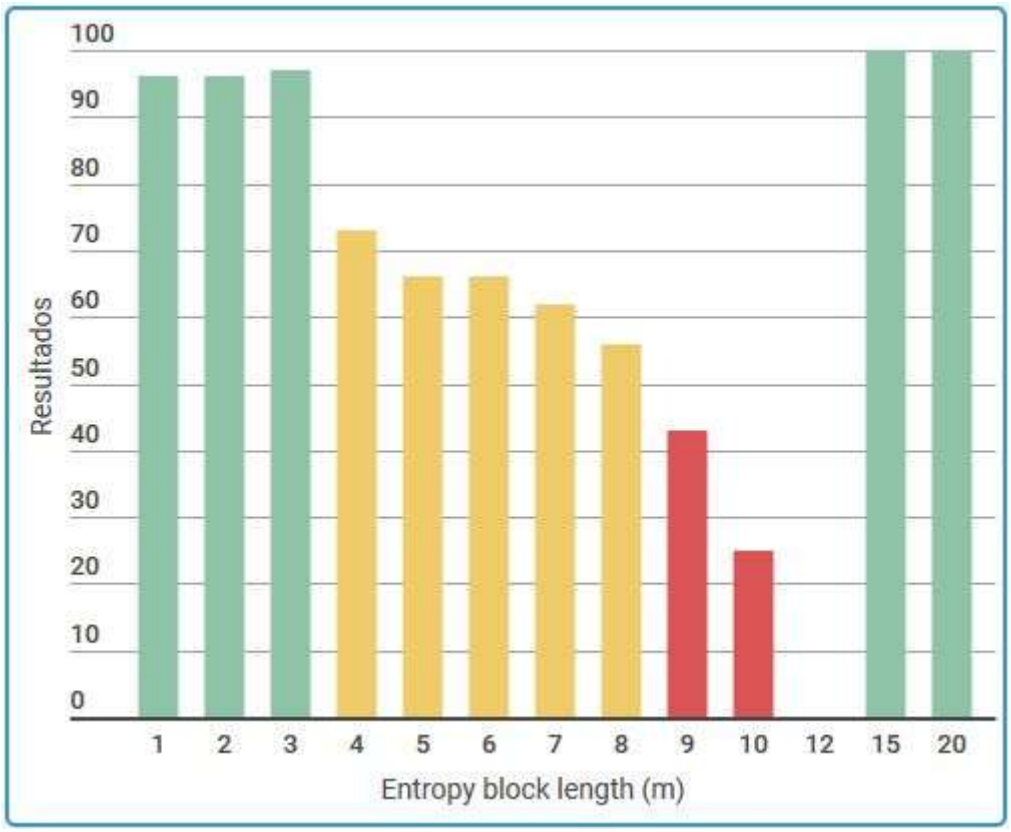


*Figura 6.17. Gráfico comparativo para distintos Entropy block length.*

Un valor de *m* superior a 8 nos proporciona resultados no válidos, por lo que se descartan los valores de 15 y 20. Efectivamente se observa que para valores pequeños de *m* los resultados también mejoran. Se toma por tanto *m=3* para el resto de evaluaciones.

# Capítulo 7

# Planificación y costes

## 7.1 Planificación temporal

### 7.1.1 Búsqueda de información

Esta primera fase del proyecto consiste en una primera toma de contacto con el tema, la situación actual, las líneas de investigación realizadas hasta el momento, etc. Parte de esta información inicial fue proporcionada por el tutor, para así poder partir de un punto donde realizar una búsqueda de información adicional. El objetivo era tanto distribuir el tiempo necesario como planificar las necesidades software, hardware y humanas que se iban a necesitar durante la realización del trabajo.

### 7.1.2 Familiarización software

La primera necesidad para la realización del trabajo era una primera toma de contacto con el software y las herramientas necesarias. Las principales han sido Matlab y Nist Test Suite. En Matlab se iba a realizar toda la parte de visualización, tratamiento y procesado de las señales usadas, además de la implementación de la solución final. Durante todo el grado, en varias asignaturas se ha utilizado este software, por lo que prácticamente se podía empezar a trabajar directamente. No pasaba lo mismo con la herramienta Nist Test Suite, para la que se necesitó más tiempo del esperado en su instalación y puesta a punto para poder empezar a realizar las pruebas pertinentes. Sin embargo el uso de esta herramienta era indiscutible ya que se adaptaba perfectamente a las necesidades.

### 7.1.3 Recolección de datos

Una vez se tenían disponibles las herramientas necesarias para empezar a trabajar, lo siguiente fue recolectar las señales con las que poder empezar a trabajar. Había múltiples posibilidades aquí, por lo que se acordó que la línea de trabajo consistiría primero en implementar un procesado válido para una de las señales tomada de una base de datos. Una vez se consiguiese una solución aceptable realizar pruebas con el

resto de señales de la base de datos. Finalmente cuando se tuviese una solución robusta acudir al hardware que nos fuese posible para realizar pruebas a nuestro procesado con muestras reales tomadas experimentalmente.

### 7.1.4 Solución propuesta

Ya con todo lo necesario, comienza la parte más compleja del trabajo tanto para el alumno como para el profesor. Se lleva a cabo el diseño de varios procesados que diesen una solución aceptable para el problema propuesto. La mayor parte del tiempo total se emplea en esta parte. El alumno ha trabajado en ir mejorando el procesado llevando a cabo ideas tanto propias como proporcionadas por el tutor en las tutorias semanales que se llevaron a cabo. En estas el alumno comentaba sus ideas, avances y problemas para que el tutor lo guiase en el camino para llegar a la solución buscada.

### 7.1.5 Evaluación de los primeros resultados

Tras las primeros procesados que funcionaban con la señal tomada de la base de datos, se procede a realizar múltiples pruebas con la herramienta Nist Test Suite. Se usan un gran número de señales con el objetivo de dar validez estadística a los resultados y poder pasar así a pruebas con muestras tomadas personalmente.

### 7.1.6 Toma de muestras reales con el hardware disponible

En esta fase del trabajo se hace uso de hardware proporcionado por otros investigadores o docentes del centro. Con este hardware se toman algunas muestras de varias señales con las que trabajar y sobre las que evaluar los procesados. El objetivo de esto es dar validez a todo el trabajo realizado haciendo ver que es aplicable a la realidad.

### 7.1.7 Elaboración de la memoria del proyecto

Última fase, y otra de las más costosas del proyecto. Se procede a realizar una memoria con la que intentar explicar toda la línea de trabajo e investigación realizada y presentar los resultados de una manera comprensible para cualquier lector que no sea experto en la materia.

La figura 7.1 muestra el diagrama de Gantt del trabajo.


*Figura 7.1.Diagrama de Gantt.*

# 7.2 Fase y horas de trabajo

Para cada una de las fases de trabajo descritas, se estima el número de horas empleadas para su realización. La tabla 7.1 recoge el número de horas para cada una.

| Fase de trabajo | Horas de trabajo (h) |
|---|---|
| Búsqueda de información | 7 |
| Familiarización software | 20 |
| Recolección de datos | 5 |
| Solución propuesta | 350 |
| Evaluación de resultados | 70 |
| Toma de muestras | 5 |
| Memoria del proyecto | 70 |
| **TOTAL** | **527** |

*Tabla 7.1. Fases y horas de trabajo.*

# 7.3 Estimación de costes

## 7.3.1 Recursos humanos

Los recursos humanos constan del trabajo realizado por el tutor/profesor junto con el trabajo realizado por el alumno. Se realiza una estimación de ambos ya que no se puede calcular con exactitud el tiempo empleado.

La parte del tutor consta de las horas dedicadas para las tutorías y búsqueda de información interesante, más el tiempo empleado en la corrección del trabajo. Se han realizado tutorias semanales durante unos 6 meses de alrededor de 90 minutos cada una, lo que supone unas 50 horas. El tiempo estimado para la corrección es de 7 horas, por lo que el total de horas dedicadas por el tutor se estima que son 57 horas.

El sueldo de un profesor/doctor gira en torno a los 50 €/hora, y el de un alumno recién graduado se estima que sea de 20 €/hora. Todo esto suma un total de:

$$Coste_{Recursos\ humanos} = 50€/h * 57h + 20€/h * 527h = 13.390€$$

### 7.3.2 Recursos hardware

La herramienta hardware básica con la que se ha trabajado es un ordenador portátil. El precio del portátil es de 600€, con una vida útil de 6 años aproximadamente. Esto hace que cada año de uso de este portátil esté valorado en 100€. Para este trabajo se ha utilizado durante 7 meses a tiempo parcial (en torno al 50% del uso se ha dedicado al trabajo), lo que hace que el coste sea de 30€.

Se pretende realizar un presupuesto mínimo, por lo que no se va a incluir el precio de los sensores utilizados para la toma de medidas real, ya que no se trata de algo imprescindible para el trabajo. Sin embargo se utilizó ya que se tuvo acceso gratuito a ellos.

### 7.3.3 Recursos software

Como software se ha utilizado principalmente Matlab y Nist Test Suite. Nist Test Suite es un software gratuito por lo que su uso no supone ningún coste. Matlab si requiere de licencia, que tiene un precio de 35€.

También se requiere para el trabajo conexión a internet, que supone un precio mensual de 21€. Para los 7 meses estimados de uso para el trabajo a tiempo parcial tiene un coste de 147€.

### 7.3.4 Coste total

Por lo tanto, una vez se tiene el coste estimado para cada parte del proyecto, el coste total estimado para el proyecto será de:

$$Coste\ total = 13.390 + 30 + 147 = 13567€$$

# Capítulo 8

# Conclusión y líneas futuras de trabajo

## 8.1 Introducción

Se han realizado todas las pruebas necesarias se han evaluado todas las soluciones propuestas para este trabajo. Se procede en este capítulo a exponer las conclusiones obtenidas tanto de los resultados como de la validez que tienen en realidad. Se expondrán además algunas líneas de trabajo en las que se podría seguir investigando.

## 8.2 Conclusión

Los objetivos planteados al comienzo del trabajo se han cumplido prácticamente en todos los aspectos. Uno de los principales objetivos era dar una solución que fuese válida para todas las señales elegidas para el trabajo. El procesado que basa la generación de la secuencia aleatoria en el ruido de la señal cumple este objetivo. El mismo procesado ha sido utilizado para los 3 tipos de señales elegidas con resultados exitosos en los 3 casos. Otro de los principales objetivos fue la generación de una secuencia aleatoria utilizando la señal limpia, y no sólo el ruido como en el primer caso. Esta solución a parte de parecer más elegante elimina el posible riesgo comentado en su momento de que se pudiese obtener la señal de ruido utilizando como entrada una señal distinta a la original. Esto supondría una amenaza a la seguridad, y aunque se vio que las probabilidades eran mínimas se quiso llevar a cabo. Uno de los “retos” planteados fue utilizar una característica periódica de la señal para obtener la secuencia aleatoria. Se trata del caso de la señal ECG. Los resultados obtenidos en este caso aunque válidos para la mayoría de los test, son los peores dentro de todos los obtenidos. Sin embargo se considera que son totalmente útiles y que quizá con algún que otra mejora se podrían usar.

Hay que tener presente la finalidad de los objetivos planteados. Desde un principio se ha dejado claro que el principal uso que se le quiere dar a las secuencias aleatorias es la generación de claves privadas para encriptación. Existen varios métodos de encriptación que requieren claves de distinta longitud en cada caso. Es por tanto necesario tener en cuenta la velocidad en bits por segundo a la que se va a generar la secuencia. La velocidad está estrechamente relacionada con la señal utilizada y con la

frecuencia de muestreo utilizada.

En el procesado del apartado 5.2.1 se utilizan los pulsos de la señal ECG para generar la secuencia. De cada pulso PQRST se obtienen un total de 5 muestras para la señal de salida (RP, RQ, RR, RS y RT). Al acotar la señal en amplitud se descartan varias muestras que suponen una pérdida de alrededor del 30% de la longitud total de la señal. Quedan por tanto unas 3’5 muestras/pulso. Cada muestra en valor decimal contribuye con 4 bits a la secuencia de bits generada (se toman los 4 bits menos significativos). Se tienen por tanto 14 bits/pulso. El ritmo cardiaco medio es de 1-1’7 pulsos/segundo (60-100 pulsaciones/minuto). Por lo tanto se generan entre 14-24 bits/segundo con este procesado. El bucle utilizado para equilibrar la cantidad de unos y ceros incluye algunos bits adicionales, pero no es un valor fijo y los cambios para la velocidad son mínimos. Por lo tanto no se tiene en cuenta.

Para el resto de procesados, las muestras utilizadas para la señal de salida con la que se genera la secuencia aleatoria coinciden con la frecuencia de muestreo. Las señales tomadas de la base de datos *Physionet.org* tienen una frecuencia de muestreo de 250 muestras/segundo. Las señales tomadas experimentalmente tienen una frecuencia de muestreo de 500 muestras/segundo tanto con el dispositivo *Plat EEG* como con el sensor de Arduino. En todos estos procesados se acota la señal en amplitud por lo que se siguen perdiendo una media del 30% del total de las muestras. De nuevo se toman solamente los 4 bits menos significativos en todos los procesados. Se consigue por tanto una velocidad de 700 bits/segundo si se muestrea a 250 muestras/segundo y 1400 bits/segundo si se muestrea a 500 muestras/segundo. Estas velocidades son aplicables tanto a los procesados que utilizan la señal limpia como a los que utilizan el ruido, ya que cada muestra de la señal tiene parte de señal limpia y parte de ruido.

Cada algoritmo de encriptación requiere una clave de longitud distinta y unos requisitos distintos. La tabla 8.1 recoge la longitud de clave utilizada por algunos de los principales algoritmos de encriptación, así como el tiempo empleado para la generación de esta clave por los distintos procesados.

| Algoritmo de encriptación | Longitud de clave | Tiempo empleado con algoritmo ECG (señal limpia) | Tiempo empleado con algoritmos de ruido (f=250 m/s) | Tiempo empleado con algoritmos de ruido (f=500 m/s) |
|---|---|---|---|---|
| RSA | 15360 | 1097-640 s | 22 s | 11 s |
| DH-DSA | 15360 | 1097-640 s | 22 s | 11s |
| ECDH | 512 | 36-21 s | 0’73 s | 0’36 s |
| AES | 256 | 18-10 s | 0’36 s | 0’18 s |

*Tabla 8.1. Tiempo empleado en generar clave para distintos algoritmos* [20].

## 8.3 Líneas futuras de trabajo

La solución propuesta en este trabajo se ha visto ofrece buenos resultados en cuanto a aleatoriedad, y se adapta a los requisitos que exigen los distintos algoritmos en los que puede ser usado. Cumple las expectativas deseadas, por lo que lejos de concluir

el estudio e investigación en este ámbito, abre un gran abanico de posibilidades.

La figura 8.1 muestra el impacto económico del IoT en los próximos años. La variedad dentro de las tecnologías IoT es muy extensa y como se observa en la figura 7.1, la inversión para su desarrollo es igual de importante. Se comentó en los capítulos 1 y 2 la importancia de la seguridad necesaria para todo este tipo de dispositivos.

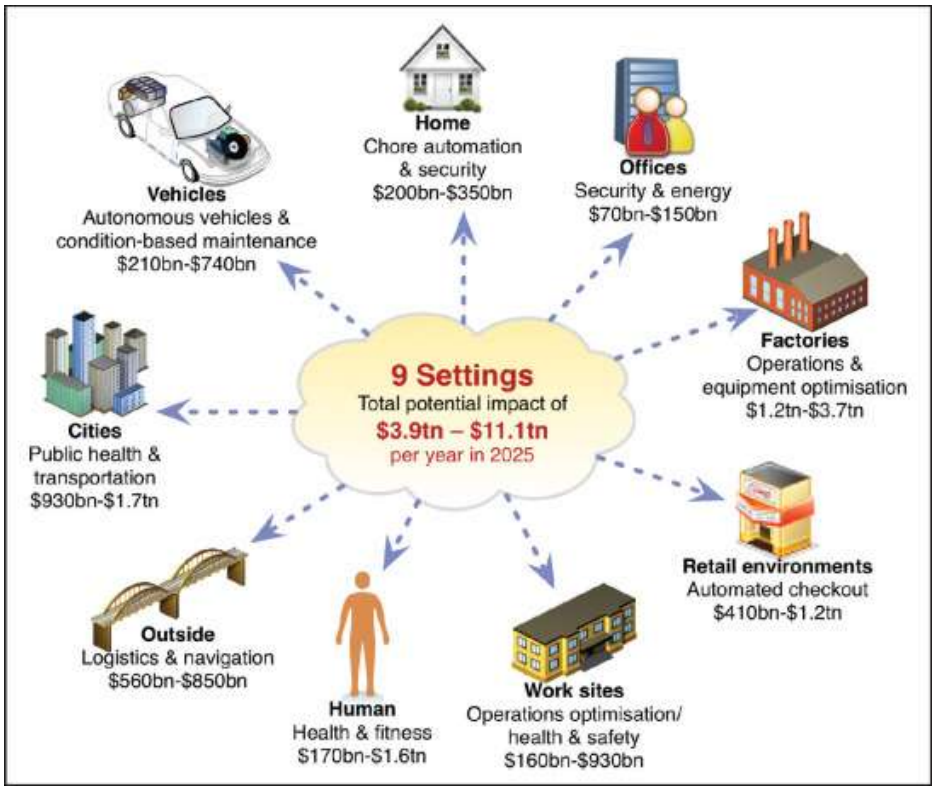


*Figura 8.1. Impacto económico IoT. Imagen tomada de [3].*

La idea estudiada en este trabajo es aplicable a todo este tipo de dispositivos, ya que la mayoría de ellos incluye sensores que pueden funcionar como fuente de aleatoriedad. Aunque estos dispositivos no tengan interacción directa con el cuerpo humano, van a tener interacción directa con la naturaleza o factores medioambientales que puedan ser fuentes de aleatoriedad.

No sólo es necesaria la aleatoriedad en casos de encriptación. En algunos casos para la autenticación de dispositivos es necesario generar un número aleatorio. Por ejemplo en las redes inalámbricas móviles, a la hora de que una estación autentique un terminal móvil es necesario que el AuC (Authentication Center), genere un número aleatorio (RAND) que será uno de los elementos de la "Tripleta de Autenticación". Por tanto la autenticación de dispositivos en redes inalámbricas es otro de los ámbitos donde se puede aplicar esta idea de trabajo y donde se pueden realizar varios trabajos de investigación.

# ANEXO I: CODIGO PROCESADO SEÑAL ECG

Código Matlab generado para procesar señales ECG y generar secuencias aleatorias con la señal limpia.

```
function ProcesadoECG1(Name)

%Ruido de la señal
s1=Name';
s2=smooth(s1,10000);
ecgsmooth=s1-s2;

%Filtrado de la señal
[C,L]=wavedec(ecgsmooth,8,'db4');
[d1,d2,d3,d4,d5,d6,d7,d8]=detcoef(C,L,[1,2,3,4,5,6,7,8]);
[thr,sorh,keepapp]=ddencmp('den','wv',ecgsmooth);
cleanecg=wdencmp('gbl',C,L,'db4',8,thr,sorh,keepapp);

%Obtener señal ruido
ruido = Name' - cleanecg - s2;

%Localizar picos R en la señal
[ondasRECG, nR] = findpeaks(ecgsmooth,'MinpeakDistance',150);
nR=nR(2:(length(nR)-1));

%Localizar picos Q en la señal
nQ=zeros(1,length(nR));
for k = 1:(length(nR))
    n1 = nR(k) - 50;
    n2 = nR(k);
    ecgPart = cleanecg(n1:n2);
    A = ecgPart;
    [fila, columnaQ] = findpeaks (-A,'MinpeakDistance',50);
    if length(columnaQ)>1
        columnaQ=columnaQ(length(columnaQ));
    end
    if length(columnaQ)==0
        columnaQ=40;
    end
    nQ(k) = [nR(k) - length(A) + columnaQ];
end
nQ = nQ';

%Localizar picos S en la señal
nS=zeros(1,length(nR));
for j = 1:length(nR)
    m1 = nR(j) + 50;
    m2 = nR(j);
```

```
    ecgPart = cleanecg(m2:m1);
    B = ecgPart;
    [fila, columnaS] = findpeaks (-B,'MinpeakDistance',50);
    if length(columnaS)>1
        columnaS=columnaS(length(columnaS));
    end
    if length(columnaS)==0
        columnaS=10;
    end
    nS(j) = [nR(j) + columnaS];
end
nS = nS';

%Localizar picos T en la señal
nT=zeros(1,length(nR));
for i = 1:length(nR)
    n1 = nR(i) + 100 + 20;
    if n1>length(cleanecg)
        n1=length(cleanecg);
    end
    n2 = nR(i)+20;
    ecgPart = cleanecg(n2:n1);
    B = ecgPart;
    [fila, columnaT] = findpeaks (B,'MinpeakDistance',75);
    if length(columnaT) > 1
        columnaT=columnaT(length(columnaT));
    end
    if length(columnaT)==0
        columnaT=10;
    end
   nT(i)= [nR(i) + 20 + columnaT];
end
nT = nT';

%Localizar picos P en la señal
nP=zeros(1,length(nR));
for l = 1:length(nR)
    n1 = nR(l) - 75 - 20;
    n2 = nR(l)-20;
    ecgPart = cleanecg(n1:n2);
    A = ecgPart;
    [fila, columnaP] = findpeaks (A,'MinpeakDistance',75);
    if length(columnaP)>1
        columnaP=columnaP(length(columnaP));
    end
    if length(columnaP)==0
        columnaP=10;
    end
    nP(l) = [nR(l) - 20 - length(A) + columnaP];
end
nP = nP';

%GENERAR SECUENCIA RR RQ RS RP RT ETC
    rr = zeros(1, (length(nR)-1));
    rq = zeros(1, (length(nR)-1));
    rs = zeros(1, (length(nR)-1));
    rp = zeros(1, (length(nR)-1));
    rt = zeros(1, (length(nR)-1));
```

```
for z = 1:(length(nR)-1)
    rr(z) = nR(z+1) - nR(z) + cleanecg(nR(z));
    rq(z) = nR(z) - nQ(z) + cleanecg(nQ(z));
    rs(z) = nS(z) - nR(z) + cleanecg(nS(z));
    rp(z) = nR(z) - nP(z) + cleanecg(nP(z));
    rt(z) = nT(z) - nR(z) + cleanecg(nT(z));

end


%ELIMINAR MEDIA POR TRAMOS DE CADA SEÑAL Y ACOTAR EN AMPLITUD
rr_nm=rr;
l0=length(rr_nm)
l=l0-1;

while (l>100)

    for i=1:l:l0
        if(i+l)<l0
            tramo = rr_nm(i:i+l);
            rr_nm(i:i+l) = tramo - mean(tramo);
        end
    end
    l = l/2;
end

l=l*2;
rr_nm(l0-l:l0)=rr_nm(l0-l:l0)-mean(rr_nm(l0-l:l0));

n1=1;
n2=1;
positivos=0;
negativos=0;
for k=1:l0
    if rr_nm(k)>0
        positivos(n1) = rr_nm(k);
        n1 = n1+1;
    else
        negativos(n2) = rr_nm(k);
        n2 = n2+1;
    end
end
mean(positivos)
mean(negativos)
corte = (mean(positivos)+abs(mean(negativos)))/2;

k=1;
for i=1:length(rr_nm)
    if abs(rr_nm(i))<corte
        rr_nmN(k)=rr_nm(i);
        k=k+1;
    end
end

rq_nm=rq;
```

```
l0=length(rq_nm);
l=l0-1;

while (l>100)

    for i=1:l:l0
        if(i+l)<l0
            tramo = rq_nm(i:i+l);
            rq_nm(i:i+l) = tramo - mean(tramo);
        end
    end
    l = l/2;
end

l=l*2;
rq_nm(l0-l:l0)=rq_nm(l0-l:l0)-mean(rq_nm(l0-l:l0));

n1=1;
n2=1;
positivos=0;
negativos=0;
for k=1:l0
    if rq_nm(k)>0
        positivos(n1) = rq_nm(k);
        n1 = n1+1;
    else
        negativos(n2) = rq_nm(k);
        n2 = n2+1;
    end
end
mean(positivos)
mean(negativos)
corte = (mean(positivos)+abs(mean(negativos)))/2;

k=1;
for i=1:length(rq_nm)
    if abs(rq_nm(i))<corte
        rq_nmN(k)=rq_nm(i);
        k=k+1;
    end
end


rs_nm=rs;
l0=length(rs_nm);
l=l0-1;

while (l>100)

    for i=1:l:l0
        if(i+l)<l0
            tramo = rs_nm(i:i+l);
            rs_nm(i:i+l) = tramo - mean(tramo);
        end
    end
    l = l/2;
end
```

```
l=l*2;
rs_nm(l0-l:l0)=rs_nm(l0-l:l0)-mean(rs_nm(l0-l:l0));

n1=1;
n2=1;
positivos=0;
negativos=0;
for k=1:l0
    if rs_nm(k)>0
        positivos(n1) = rs_nm(k);
        n1 = n1+1;
    else
        negativos(n2) = rs_nm(k);
        n2 = n2+1;
    end
end
mean(positivos)
mean(negativos)
corte = (mean(positivos)+abs(mean(negativos)))/2;

k=1;
for i=1:length(rs_nm)
    if abs(rs_nm(i))<corte
        rs_nmN(k)=rs_nm(i);
        k=k+1;
    end
end


rp_nm=rp;
l0=length(rp_nm);
l=l0-1;

while (l>100)

    for i=1:l:l0
        if(i+l)<l0
            tramo = rp_nm(i:i+l);
            rp_nm(i:i+l) = tramo - mean(tramo);
        end
    end
    l = l/2;
end

l=l*2;
rp_nm(l0-l:l0)=rp_nm(l0-l:l0)-mean(rp_nm(l0-l:l0));

n1=1;
n2=1;
positivos=0;
negativos=0;
for k=1:l0
    if rp_nm(k)>0
        positivos(n1) = rp_nm(k);
        n1 = n1+1;
    else
```

```
        negativos(n2) = rp_nm(k);
        n2 = n2+1;
    end
end
mean(positivos)
mean(negativos)
corte = (mean(positivos)+abs(mean(negativos)))/2;

k=1;
for i=1:length(rp_nm)
    if abs(rp_nm(i))<corte
        rp_nmN(k)=rp_nm(i);
        k=k+1;
    end
end



rt_nm=rt;
l0=length(rt_nm);
l=l0-1;

while (l>100)

    for i=1:l:l0
        if(i+l)<l0
            tramo = rt_nm(i:i+l);
            rt_nm(i:i+l) = tramo - mean(tramo);
        end
    end
    l = l/2;
end

l=l*2;
rt_nm(l0-l:l0)=rt_nm(l0-l:l0)-mean(rt_nm(l0-l:l0));

n1=1;
n2=1;
positivos=0;
negativos=0;
for k=1:l0
    if rt_nm(k)>0
        positivos(n1) = rt_nm(k);
        n1 = n1+1;
    else
        negativos(n2) = rt_nm(k);
        n2 = n2+1;
    end
end
mean(positivos)
mean(negativos)
corte = (mean(positivos)+abs(mean(negativos)))/2;

k=1;
for i=1:length(rt_nm)
    if abs(rt_nm(i))<corte
        rt_nmN(k)=rt_nm(i);
        k=k+1;
    end
```

```
end

%NORMALIZAR SEÑALES

rpNor=rp_nmN/max(abs(rp_nmN));
rp256=127.5*rpNor+127.5;

rrNor=rr_nmN/max(abs(rr_nmN));
rr256=127.5*rrNor+127.5;

rqNor=rq_nmN/max(abs(rq_nmN));
rq256=127.5*rqNor+127.5;

rsNor=rs_nmN/max(abs(rs_nmN));
rs256=127.5*rsNor+127.5;

rtNor=rt_nmN/max(abs(rt_nmN));
rt256=127.5*rtNor+127.5;


%OBTENER RUIDO ENTRE 1-128 Y 1-5

k=1;
for i=1:length(ruido)
    if abs(ruido(i))<12
        ruidoN(k)=ruido(i);
        k=k+1;
    end
end

ruido1=ruidoN/max(abs(ruidoN));
ruido128=63*ruido1+64;
ruido128=fix(ruido128);

ruido09=2.5*ruido1+3.5;
ruido09=fix(ruido09);


%CONSTRUIR SEÑAL SALIDA COMBINANDO RP RQ RR RS RT

salida=[];
c=1;

while(length(salida)<1000000)

    inicio=ruido128(c);
    fin=inicio+ruido128(c+1);

    if(ruido09(c)==1)&&(fin<(length(rp256)))
        trozo=rp256(inicio:fin);
    end
    if(ruido09(c)==2)&&(fin<(length(rq256)))
        trozo=rq256(inicio:fin);
    end
    if(ruido09(c)==3)&&(fin<(length(rr256)))
        trozo=rr256(inicio:fin);
```

```
    end
    if(ruido09(c)==4)&&(fin<(length(rs256)))
         trozo=rs256(inicio:fin);
    end
    if(ruido09(c)==5)&&(fin<(length(rt256)))
         trozo=rt256(inicio:fin);
    end

    salida=[salida trozo];
    c=c+1;

end

%ELIMINAR MEDIA POR TRAMOS Y ACOTAR EN AMPLITUD A LA SEÑAL DE SALIDA.
salida_nm=salida;
l0=length(salida_nm);
l=l0-1;

while (l>100)

    for i=1:l:l0
        if(i+l)<l0
            tramo = salida_nm(i:i+l);
            salida_nm(i:i+l) = tramo - mean(tramo);
        end
    end
    l = l/2;
end

l=l*2;
salida_nm(l0-l:l0)=salida_nm(l0-l:l0)-mean(salida_nm(l0-l:l0));

n1=1;
n2=1;
positivos=0;
negativos=0;
for k=1:l0
    if salida_nm(k)>0
        positivos(n1) = salida_nm(k);
        n1 = n1+1;
    else
        negativos(n2) = salida_nm(k);
        n2 = n2+1;
    end
end
mean(positivos)
mean(negativos)
corte = (mean(positivos)+abs(mean(negativos)))/2;

k=1;
for i=1:length(salida_nm)
    if abs(salida_nm(i))<corte
        salida_nmN(k)=salida_nm(i);
        k=k+1;
    end
end
```

```
%NORMALIZAR Y PASAR A BINARIO
outNor=salida_nmN/max(abs(salida_nmN));
out256=127.5*outNor+127.5;
outDec=dec2bin(out256,8);

%GENERAR SECUENCIA DE SALIDA DE BITS
%BUCLE EQUILIBRAR 1 Y 0
cont_zeros=0;
cont_unos=0;
ctramo1=0;
ctramo0=0;
insertados=0;

insert=fix(ruido128(1)*2);
cr=1;

for i=1:size(outDec,1)

  for j=1:4
    if outDec(i,j+4)=='1'
        cont_unos=cont_unos+1;
        ctramo1=ctramo1+1;
        insert=insert-1;
        outFinal(4*(i-1)+j+insertados)='1';
    else
        cont_zeros=cont_zeros+1;
        ctramo0=ctramo0+1;
        insert=insert-1;
        outFinal(4*(i-1)+j+insertados)='0';

    end
    if insert==0
            if ctramo1>ctramo0
                outFinal(4*(i-1)+j+1+insertados)='0';
                insertados=insertados+1;
                cont_zeros=cont_zeros+1;
            else
                outFinal(4*(i-1)+j+1+insertados)='1';
                insertados=insertados+1;
                cont_unos=cont_unos+1;
            end
        cr=cr+1;
        insert=fix(ruido128(cr)*2);
        ctramo0=0;
        ctramo1=0;
    end
  end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%ESCRIBIR SALIDA EN FICHERO
fileID=fopen('filename.e','w');
fprintf(fileID,outFinal);
fclose(fileID);
```

```
end
```

# ANEXO II: CODIGO PROCESADO RUIDO ECG

Código Matlab generado para procesar señales ECG y generar secuencias aleatorias con el ruido.

```
function ProcesadoECG2(Name)

%Eliminar ruido de la señal
s1=Name';
s2=smooth(s1,10000);
ecgsmooth=s1-s2;

%Filtrar la señal
[C,L]=wavedec(ecgsmooth,8,'db4');
[d1,d2,d3,d4,d5,d6,d7,d8]=detcoef(C,L,[1,2,3,4,5,6,7,8]);
[thr,sorh,keepapp]=ddencmp('den','wv',ecgsmooth);
cleanecg=wdencmp('gbl',C,L,'db4',8,thr,sorh,keepapp);

%Obtener el ruido restandole la señal limpia
ruido = Name' - cleanecg - s2;

%ELIMINO MEDIA POR TRAMOS DE LA SEÑAL RUIDO

ruido_nm=ruido;
l0=length(ruido_nm);
l=l0-1;

while (l>100)

    for i=1:l:l0
        if(i+l)<l0
            tramo = ruido_nm(i:i+l);
            ruido_nm(i:i+l) = tramo - mean(tramo);
        end
    end
    l = l/2;
end

n1=1;
n2=1;
positivos=0;
negativos=0;
for k=1:l0
    if ruido_nm(k)>0
        positivos(n1) = ruido_nm(k);
        n1 = n1+1;
    else
        negativos(n2) = ruido_nm(k);
        n2 = n2+1;
```

```
    end
end
mean(positivos)
mean(negativos)
corte = (mean(positivos)+abs(mean(negativos)))/2;

%ACOTAR EN AMPLITUD LA SEÑAL RUIDO
k=1;
for i=1:length(ruido_nm)
    if abs(ruido_nm(i))<corte
        ruido_nmN(k)=ruido_nm(i);
        k=k+1;
    end
end

%OBTENER RUIDO ENTRE 1-128
k=1;
for i=1:length(ruido)
    if abs(ruido(i))<12
        ruidoN(k)=ruido(i);
        k=k+1;
    end
end

ruido1=ruidoN/max(abs(ruidoN));
ruido128=63*ruido1+64;
ruido128=fix(ruido128);

%Normalizar y pasar a binario
outNor=ruido_nmN/max(abs(ruido_nmN));
out256=127.5*outNor+127.5;
outDec=dec2bin(out256,8);

%Generar secuencia de bits
cont_zeros=0;
cont_unos=0;
for i=1:size(outDec,1)
  for j=1:4
    if outDec(i,j+4)=='1'
        cont_unos=cont_unos+1
        outFinal(4*(i-1)+j)='1';
    else
        cont_zeros=cont_zeros+1
        outFinal(4*(i-1)+j)='0';
    end
  end
end


%ESCRIBIR SALIDA EN FICHERO
fileID=fopen('filename.e','w');
fprintf(fileID,outFinal);
fclose(fileID);

end
```

# ANEXO III: CODIGO PROCESADO SEÑAL EMG

Código Matlab generado para procesar señales EMG y generar secuencias aleatorias con la señal limpia.

```
function ProcesadoEMG1(signal)

%ELIMINAR MEDIA POR TRAMOS DE LA SEÑAL
signal_nm=signal;
l0=length(signal_nm);
l=l0-1;

while (l>100)

    for i=1:l:l0
        if(i+l)<l0
            tramo = signal_nm(i:i+l);
            signal_nm(i:i+l) = tramo - mean(tramo);
        end
    end
    l = l/2;
end

l=l*2;
signal_nm(l0-l:l0)=signal_nm(l0-l:l0)-mean(signal_nm(l0-l:l0));

n1=1;
n2=1;
positivos=0;
negativos=0;
for k=1:l0
    if signal_nm(k)>0
        positivos(n1) = signal_nm(k);
        n1 = n1+1;
    else
        negativos(n2) = signal_nm(k);
        n2 = n2+1;
    end
end
mean(positivos)
mean(negativos)
corte = (mean(positivos)+abs(mean(negativos)))/2;

%ACOTAR EN AMPLITUD LA SEÑAL
k=1;
for i=1:length(signal_nm)
    if abs(signal_nm(i))<corte
        signal_nmN(k)=signal_nm(i);
        k=k+1;
    end
```

```
end

%Normalizar y paso a binario
outNor=signal_nmN/max(abs(signal_nmN));
out256=127.5*outNor+127.5;
outDec=dec2bin(out256,8);

%GENERAR SECUENCIA DE BITS
cont_zeros=0;
cont_unos=0;
for i=1:size(outDec,1)
  for j=1:4
    if outDec(i,j+4)=='1'
        cont_unos=cont_unos+1;
        outFinal(4*(i-1)+j)='1';
    else
        cont_zeros=cont_zeros+1;
        outFinal(4*(i-1)+j)='0';
    end
  end
end

%ESCRIBO SALIDA EN FICHERO
fileID=fopen('filename.e','w');
fprintf(fileID,outFinal);
fclose(fileID);

end
```

# ANEXO IV: CODIGO PROCESADO EMG RUIDO

Código Matlab generado para procesar señales EMG y generar secuencias aleatorias con el ruido.

```
function ProcesadoEMG2(Name)

%Eliminar ruido de la señal
s1=Name';
s2=smooth(s1,10000);
emgsmooth=s1-s2;

%Filtrar la señal
[C,L]=wavedec(emgsmooth,8,'db4');
[d1,d2,d3,d4,d5,d6,d7,d8]=detcoef(C,L,[1,2,3,4,5,6,7,8]);
[thr,sorh,keepapp]=ddencmp('den','wv',emgsmooth);
cleanemg=wdencmp('gbl',C,L,'db4',8,thr,sorh,keepapp);

%Obtener el ruido restandole la señal limpia
ruido = Name' - cleanemg - s2;

%ELIMINAR MEDIA POR TRAMOS DE LA SEÑAL RUIDO

ruido_nm=ruido;
l0=length(ruido_nm);
l=l0-1;

while (l>100)

    for i=1:l:l0
        if(i+l)<l0
            tramo = ruido_nm(i:i+l);
            ruido_nm(i:i+l) = tramo - mean(tramo);
        end
    end
    l = l/2;
end

n1=1;
n2=1;
positivos=0;
negativos=0;
for k=1:l0
    if ruido_nm(k)>0
        positivos(n1) = ruido_nm(k);
        n1 = n1+1;
    else
        negativos(n2) = ruido_nm(k);
        n2 = n2+1;
    end
end
```

```
mean(positivos)
mean(negativos)
corte = (mean(positivos)+abs(mean(negativos)))/2;

%ACOTAR EN AMPLITUD LA SEÑAL RUIDO
k=1;
for i=1:length(ruido_nm)
    if abs(ruido_nm(i))<corte
        ruido_nmN(k)=ruido_nm(i);
        k=k+1;
    end
end

%OBTENER RUIDO ENTRE 1-128 PARA METER 1 Y 0

k=1;
for i=1:length(ruido)
    if abs(ruido(i))<12
        ruidoN(k)=ruido(i);
        k=k+1;
    end
end

ruido1=ruidoN/max(abs(ruidoN));
ruido128=63*ruido1+64;
ruido128=fix(ruido128);

%Normalizar y pasar a binario
outNor=ruido_nmN/max(abs(ruido_nmN));
out256=127.5*outNor+127.5;
outDec=dec2bin(out256,8);

%GENERAR SECUENCIA DE BITS
cont_zeros=0;
cont_unos=0;
for i=1:size(outDec,1)
  for j=1:4
    if outDec(i,j+4)=='1'
        cont_unos=cont_unos+1;
        outFinal(4*(i-1)+j)='1';
    else
        cont_zeros=cont_zeros+1;
        outFinal(4*(i-1)+j)='0';
    end
  end
end

%ESCRIBO SALIDA EN FICHERO
fileID=fopen('filename.e','w');
fprintf(fileID,outFinal);
fclose(fileID);

end
```

# ANEXO V: CODIGO PROCESADO SEÑAL EEG

Código Matlab generado para procesar señales EEG y generar secuencias aleatorias con la señal limpia.

```
function ProcesadoEEG1(signal)

%ELIMINAR MEDIA POR TRAMOS DE LA SEÑAL
signal_nm=signal;
l0=length(signal_nm);
l=l0-1;

while (l>100)

    for i=1:l:l0
        if(i+l)<l0
            tramo = signal_nm(i:i+l);
            signal_nm(i:i+l) = tramo - mean(tramo);
        end
    end
    l = l/2;
end

l=l*2;
signal_nm(l0-l:l0)=signal_nm(l0-l:l0)-mean(signal_nm(l0-l:l0));

n1=1;
n2=1;
positivos=0;
negativos=0;
for k=1:l0
    if signal_nm(k)>0
        positivos(n1) = signal_nm(k);
        n1 = n1+1;
    else
        negativos(n2) = signal_nm(k);
        n2 = n2+1;
    end
end
mean(positivos)
mean(negativos)
corte = (mean(positivos)+abs(mean(negativos)))/2;

%ACOTAR EN AMPLITUD LA SEÑAL
k=1;
for i=1:length(signal_nm)
    if abs(signal_nm(i))<corte
        signal_nmN(k)=signal_nm(i);
        k=k+1;
    end
end
```

```
%Normalizar y paso a binario
outNor=signal_nmN/max(abs(signal_nmN));
out256=127.5*outNor+127.5;
outDec=dec2bin(out256,8);

%GENERAR SECUENCIA DE BITS
cont_zeros=0;
cont_unos=0;
for i=1:size(outDec,1)
  for j=1:4
    if outDec(i,j+4)=='1'
        cont_unos=cont_unos+1;
        outFinal(4*(i-1)+j)='1';
    else
        cont_zeros=cont_zeros+1;
        outFinal(4*(i-1)+j)='0';
    end
  end
end

%ESCRIBO SALIDA EN FICHERO
fileID=fopen('filename.e','w');
fprintf(fileID,outFinal);
fclose(fileID);

end
```

# ANEXO IV: CODIGO PROCESADO EEG RUIDO

Código Matlab generado para procesar señales EEG y generar secuencias aleatorias con la el rudio.

```
function ProcesadoEEG2(Name)

%Eliminar ruido de la señal
s1=Name';
s2=smooth(s1,10000);
eegsmooth=s1-s2;

%Filtrar la señal
[C,L]=wavedec(eegsmooth,8,'db4');
[d1,d2,d3,d4,d5,d6,d7,d8]=detcoef(C,L,[1,2,3,4,5,6,7,8]);
[thr,sorh,keepapp]=ddencmp('den','wv',eegsmooth);
cleaneeg=wdencmp('gbl',C,L,'db4',8,thr,sorh,keepapp);

%Obtener el ruido restandole la señal limpia
ruido = Name' - cleaneeg - s2;

%ELIMINAR MEDIA POR TRAMOS DE LA SEÑAL RUIDO

ruido_nm=ruido;
l0=length(ruido_nm);
l=l0-1;

while (l>100)

    for i=1:l:l0
        if(i+l)<l0
            tramo = ruido_nm(i:i+l);
            ruido_nm(i:i+l) = tramo - mean(tramo);
        end
    end
    l = l/2;
end

n1=1;
n2=1;
positivos=0;
negativos=0;
for k=1:l0
    if ruido_nm(k)>0
        positivos(n1) = ruido_nm(k);
        n1 = n1+1;
    else
        negativos(n2) = ruido_nm(k);
        n2 = n2+1;
    end
end
```

```
mean(positivos)
mean(negativos)
corte = (mean(positivos)+abs(mean(negativos)))/2;

%ACOTAR EN AMPLITUD LA SEÑAL RUIDO
k=1;
for i=1:length(ruido_nm)
    if abs(ruido_nm(i))<corte
        ruido_nmN(k)=ruido_nm(i);
        k=k+1;
    end
end

%OBTENER RUIDO ENTRE 1-128 PARA METER 1 Y 0

k=1;
for i=1:length(ruido)
    if abs(ruido(i))<12
        ruidoN(k)=ruido(i);
        k=k+1;
    end
end

ruido1=ruidoN/max(abs(ruidoN));
ruido128=63*ruido1+64;
ruido128=fix(ruido128);

%Normalizar y pasar a binario
outNor=ruido_nmN/max(abs(ruido_nmN));
out256=127.5*outNor+127.5;
outDec=dec2bin(out256,8);

%GENERAR SECUENCIA DE BITS
cont_zeros=0;
cont_unos=0;
for i=1:size(outDec,1)
  for j=1:4
    if outDec(i,j+4)=='1'
        cont_unos=cont_unos+1;
        outFinal(4*(i-1)+j)='1';
    else
        cont_zeros=cont_zeros+1;
        outFinal(4*(i-1)+j)='0';
    end
  end
end

%ESCRIBO SALIDA EN FICHERO
fileID=fopen('filename.e','w');
fprintf(fileID,outFinal);
fclose(fileID);

end
```

# Bibliografía


[1] C. C. Nacional, "Informe de Actividades," 2014.

[2] S. L. H. a. C. Liu, "Sensor-Based Random Number Generator Seeding," *IEEE Access,* 2015.

[3] CCN-CERT, "Ciberamenazas y Tendencias," 2017.

[4] F.-M. M. J.-F. V. y. A. Z. Rafael Álvarez, "Extensión y parametrización de un generador pseudoaleatorio matricial," Alicante, 2012.

[5] M. B. A. M. S. L. BLUM, A SIMPLE UNPREDICTABLE PSEUDO-RANDOM NUMBER GENERATOR, 1986.

[6] E. M. A. Díaz, *Intercambio público de claves usando matrices sobre anillos de grupo,* Murcia: Trabajo Fin de Grado, 2015.

[7] A. M. Mancilla Herrera, Números aleatorios. Historia, teoría y aplicaciones, Barranquilla, Colombia: Universidad del norte, 2000.

[8] M. G. d. l. Fuente, *Generación de numeros aleatorios físicos en dispositivos electrónicos.,* Universidad de Valladolid: Escuela Técnica Superior de Ingenieros de Telecomunicación., 2016.

[9] J. S. i. Casals, "http://www.investigacionyciencia.es," [Online]. Available: http://www.investigacionyciencia.es/blogs/tecnologia/20/posts/biometra-aplicada-a-la-salud-y-a-la-seguridad-11242.

[10] D. G. P. Garza, "El electrocardiograma y su tecnología," *Avances.*

[11] E. D. C. J. R. P. OSCAR EDUARDO VERA, "EXTRACCIÓN DE CARACTERÍSTICAS DE LA SEÑAL ELECTROCARDIOGRÁFICA MEDIANTE SOFTWARE DE ANÁLISIS MATEMÁTICO," *Scientia et Technica,* 2006.

[12] M. M. F. E. C. D. R. F. J. Á. Alberto OCHOA, *Sistema de Adquisición y Procesamiento de Señales Electrocardiográficas,* Universidad de Colima, México., 2010.

[13] M. C. P. Duque, *Análisis de señal del impulso cardíaco para el mejoramiento del diagnóstico de patologías del corazón.,* Universidad Técnica de Pereira.: Tesis de maestría., 2011.

[14] L. B. Borau, *Estudio de la señal de electromiograma para el tratamiento de la incontinencia urinaria,* Catalunya: Proyecto final de carrera, 2016.

[15] C. A. A. CORAS, *PROCESAMIENTO DE SEÑALES DE ELECTROMIOGRAFÍA SUPERFICIAL PARA LA DETECCIÓN DE MOVIMIENTO DE DOS DEDOS DE LA MANO",* Lima, Perú: TESIS PARA OPTAR EL TÍTULO PROFESIONAL DE INGENIERO ELECTRÓNICO, 2012.

[16] D. D. M. Lucas, *Evaluación de la terapia multi-sensorial mediante el análisis de electroencefalogramas,* Valladolid: Trabajo Fin de Grado, 2014.

[17] R. B. Navarro, *Instrumentación Biomédica,* Universidad de Alcalá: Departamento Electrónica. Universidad Alcalá..

[18] D. E. S. Vázquez, *DISEÑO Y DESARROLLO DE UNA APLICACIÓN PARA CONTROLAR UN TELÉFONO MÓVIL MEDIANTE SISTEMAS BRAIN COMPUTER INTERFACE (BCI) ORIENTADA A PERSONAS CON GRAVE DISCAPACIDAD*, Valladolid: Trabajo Fin de Grado., 2016.

[19] J. S. L. V. AndrewRukhin, A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications, National Institute of Standars and Technology, 2010.

[20] E. Barker, W. Barker, WBurrW.Polk and M. Smid, Recommendation for Key Management 800-57, 2007.

[21] PhysioNet, "The research resource for complex physiologic signals.," [Online]. Available: https://www.physionet.org/.

[22] C. M. F. P. S. M. M. Á. L.-G. Jesús Minguillón, "MÓDULOS PLAT-EEG PARA MEDIDAS LAPLACIANAS CON ELECTRODO SECO," Granada, 2016.

[23] N. Koblitz, A Course in Number Theory and Cryptography. Springer Graduate Texts in Mathematics, 1994.

[24] A. T. B. y. J. M. V. Romero, Generación de Numeros Aleatorios, 2014.

[25] S. M. I. G. F. R. S. M. I. Guanglou Zheng, "Multiple ECG Fiducial Points based Random Binary Sequence Generation for Securing Wireless Body Area Networks," *IEE Access*, 2015.