Grado en Ingeniería de Tecnologías de Telecomunicación

Escuela Politécnica Superior de Linares

Universidad de Jaén

Escuela Politécnica Superior de Linares

Trabajo Fin de Grado

DESARROLLO DE UN SISTEMA DE MICROLOCALIZACIÓN Y SEGUIMIENTO EN INTERIORES CON SENSORES BLUETOOTH LE

Alumno: Juan Ramón Camero Chillón

Tutor: Prof. D. Fernando Parra Rodríguez

Depto.: Ingeniería de. Telecomunicación

Febrero, 2018

DESARROLLO DE UN SISTEMA DE MICROLOCALIZACIÓN Y SEGUIMIENTO EN INTERIORES CON SENSORES BLUETOOTH LE

Alumno: Juan Ramón Camero Chillón

Tutor: Prof. D. Fernando Parra Rodríguez

Depto.: Ingeniería de. Telecomunicación

Febrero, 2018

FIRMA AUTOR

FIRMA TUTOR

INDICE

1 AGRADECIMIENTOS 7

2 RESUMEN 7

3 INTRODUCCION 8

4 MOTIVACIÓN 9

5 OBJETIVOS 10

6 ANTECEDENTES 10

7 UBICACIÓN 11

8 RECURSOS UTILIZADOS 11

9 MÉTODOS 12

9.1 ANÁLISIS DE LA SEÑAL 12

9.2 PROCESADO DE LA SEÑAL 13

9.2.1 FILTRO BUTTERWORTH 14

9.2.2 FILTRO CHEBYSHEV 16

9.2.3 FILTRO DE KALMAN 18

9.2.4 CONCLUSIÓN DEL PROCESADO DE LA SEÑAL 22

9.3 ANÁLISIS DE LAS MUESTRAS FILTRADAS 23

9.3.1 Tabla mediciones en [dBm] nodo 1 para el plano polar 24

9.4 DIAGRAMA RADIACCIÓN 26

9.5 REDES NEURONALES 27

9.5.1 DEFINICIÓN RED NEURONAL 27

9.5.2 VENTAJAS DE LAS REDES NEURONALES ARTIFICIALES 28

9.5.3 ELEMENTOS BASICOS DE UNA RED NEURONAL 28

9.5.4 FUNCIÓN DE ACTIVACION 29

9.5.5 ARQUITECTURA RED NEURONAL ARTIFICIAL 32

9.5.6 ENTRENAMIENTO DE LA RED NEURONAL 33

9.5.7 PARÁMETROS PREDEFINIDOS PARA DETENER EL ENTRENAMIENTO DE LA RED NEURONAL ARTIFICIAL 35

9.5.8 NÚMERO DE CAPAS OCULTAS Y NEURONAS DE LA RED NEURONAL 35

9.6 INTERPOLACIÓN DE LA DISTANCIA MEDIANTE REDES NEURONALES 36

9.6.1 CONCLUSIÓN DE LAS REDES NEURONALES PARA INTERPOLAR LA DISTANCIA 39

9.7 TRILATERACIÓN 40

9.7.1 PRUEBA REAL TRILATERACION 41
9.7.2 CONCLUSIONES DE LA TRILATERACION 43
9.8 MÉTODO DE POSICIONAMIENTO CARTESIANO 44
9.8.1 ANÁLISIS MUESTRAS FILTRADAS 45
9.8.2 PRUEBAS MÉTODO CARTESIANO 46
9.8.3 CONCLUSIÓN USO DEL MÉTODO DE POSICIONAMIENTO CARTESIANO 57
10 CONCLUSIONES 58
11 ANEXOS 60
11.1 PROGRAMAS REALIZADOS 60
11.1.1 Filtro butterwort 60
11.1.2 Filtro chebyshev 61
11.1.3 Filtro de kalman 62
11.1.4 Redes neuronales para trilateración 63
11.1.5 Método de posicionamiento trilateración 65
11.1.6 Método de posicionamiento cartesiano 66
11.1.7 Programa para visualizar el método cartesiano 68
11.2 MEDIDAS TOMADAS 69
11.2.1 Medidas plano polar 69
11.2.2 Medidas plano cartesiano 71
12 BIBLIOGRAFIA 73

INDICE DE FIGURAS

Figura 1: Señal emitida por el ibeacon 13
Figura 2: Respuesta en frecuencia filtro Butterworth 14
Figura 3: Respuesta en fase filtro Butterworth 15
Figura 4: señal filtrada filtro Butterworth 15
Figura 5: respuesta en frecuencia filtro chevysheb 16
Figura 6: respuesta en fase filtro chevyshe 17
Figura 7: señal filtrada filtro chevysheb 17
Figura 8: Esquema fases y ecuaciones filtro de kalman 19
Figura 9: Señal filtra por el filtro de kalman 21

Figura 10: comparativa señal filtrada por filtro de kalman y señal original *21*

Figura 11: Ruido filtrado por filtro de kalman *22*

Figura 12: Medida con el ibeacons plano *25*

Figura 13: Medida con el ibeacons inclinado *26*

Figura 14: diagrama de radiación *27*

Figura 15: esquema red neuronal *29*

Figura 16: Función de activación escalon *30*

Figura 17: función de activación lineal *31*

Figura 18: función de activación sigmoidal *31*

Figura 19: función de activación tangente sigmoidal *32*

Figura 20: Función de activación gausiana *32*

Figura 21: Interpolación de la distancia con el entrenamiento de rezago *37*

Figura 22: interpolación de la distancia con el entrenamiento de descenso de gradiente por lotes con momentumn *38*

Figura 23: Comprobación que la red ofrece una interpolación correcta *39*

Figura 24: prueba 1 de trilateración *42*

Figura 25: prueba 2 de trilateración *43*

Figura 26: prueba 1 posicionamiento cartesiano con 2 receptores *48*

Figura 27: prueba 2 posicionamiento cartesiano con 2 receptores *49*

Figura 28: prueba 3 posicionamiento cartesiano con 2 receptores *50*

Figura 29: prueba 4 posicionamiento cartesiano con 2 receptores *51*

Figura 30: prueba 5 posicionamiento cartesiano con 2 receptores *52*

Figura 31: prueba 1 posicionamiento cartesiano con 3 receptores *53*

Figura 32: prueba 2 posicionamiento cartesiano con 3 receptores *54*

Figura 33: prueba 3 posicionamiento cartesiano con 3 receptores *55*

Figura 34: prueba 4 posicionamiento cartesiano con 3 receptores *56*

Figura 35: prueba 5 posicionamiento cartesiano con 3 receptores *57*

INDICE DE TABLAS

Tabla 1: Recursos Utilizados..... 12

Tabla 2: mediciones en [dBm] nodo 1 para el plano polar..... 24

Tabla 3: Error En Distancia Para La Prueba De -66.28 dBm..... 38

Tabla 4: Error En Distancia Prueba 1 Trilateración..... 42

Tabla 5: Error En Distancia Prueba 2 Trilateración..... 43

Tabla 6: Error Medio En Distancia Trilateración..... 44

Tabla 7: Nivel De Potencia En dBm Del Nodo 1 Método Trilateración..... 45

Tabla 8: Error Medio En Distancia Con Posicionamiento Cartesiano Para 2 Receptores..... 47

Tabla 9: Error Medio En Distancia Con Posicionamiento Cartesiano Para 3 Receptores..... 47

Tabla 10: Error En Distancia Con Posicionamiento Cartesiano Prueba 1 Para 2 Receptores..... 48

Tabla 11: Error En Distancia Con Posicionamiento Cartesiano Prueba 2 Para 2 Receptores..... 49

Tabla 12: Error En Distancia Con Posicionamiento Cartesiano Prueba 3 Para 2 Receptores..... 50

Tabla 13: Error En Distancia Con Posicionamiento Cartesiano Prueba 4 Para 2 Receptores..... 51

Tabla 14: Error En Distancia Con Posicionamiento Cartesiano Prueba 5 Para 2 Receptores..... 52

Tabla 15: Error en distancia con posicionamiento cartesiano prueba 1 para 3 receptores..... 53

Tabla 16: Error En Distancia Con Posicionamiento Cartesiano Prueba 2 Para 3 Receptores..... 54

Tabla 17: Error En Distancia Con Posicionamiento Cartesiano Prueba 3 Para 3 Receptores..... 55

Tabla 18: Error En Distancia Con Posicionamiento Cartesiano Prueba 4 Para 3 Receptores..... 56

Tabla 19: Error En Distancia Con Posicionamiento Cartesiano Prueba 5 Para 3 Receptores..... 57

1 AGRADECIMIENTOS

Durante la etapa de mi formación académica, he compartido momentos con muchas personas de las que estaré, por muchas razones, eternamente agradecido.

En primer lugar, a todos los familiares y personas allegadas que, no solo me han apoyado económicamente, sino que también me han apoyado durante los momentos tan duros que un estudiante de Ingeniería está obligado a atravesar.

Me gustaría agradecer, además, a todos mis compañeros que ahora ya los considero amigos y, por supuesto, a todos los profesores que han logrado que esta etapa sea mucho más fructífera gracias a sus conocimientos y aportaciones a mi formación.

Por último, me gustaría resaltar y hacer una mención especial a la aportación de mi tutor del Trabajo de Fin de Grado, Don Fernando Parra Rodríguez que me ha guiado y ha hecho posible la realización de este trabajo.

2 RESUMEN

El planteamiento de este proyecto nace de la necesidad de encontrar una solución para la localización en interiores, donde es imposible utilizar el sistema GPS debido a la degradación de la señal emitida por los satélites.

En este Trabajo de Fin de Grado se ha propuesto una solución basada en la tecnología Bluetooth low energy en un dispositivo emisor y diversos dispositivos receptores (Raspberry pi) distribuidos en un espacio cerrado de pruebas. Para ello, se definen celdas de resolución (tanto en coordenadas polares como cartesianas) para generar un mapa de calor. Este mapa se utilizará para entrenar una red neuronal con el objetivo de interpolar las posiciones intermedias.

3 INTRODUCCION

Este proyecto tiene la finalidad de crear una solución para el posicionamiento en interiores con la tecnología inalámbrica *Bluetooth le.*

En este apartado se desarrolla una breve introducción para centrar el tema tratado en la Memoria, siendo estos los siguientes:

Motivación: Expresa que ha sido, lo que nos ha impulsado a la realización de este proyecto.

Objetivos: Expresa los objetivos que este proyecto necesita cubrir.

Antecedentes: Se explica una breve historia de cómo ha evolucionado el posicionamiento desde sus inicios hasta hoy en día

Ubicación: Queda expuesto donde se ha realizado dicho proyecto y los motivos de la elección de dicho habitáculo.

Recursos utilizados: Se muestran todos los elementos utilizados para la realización de este proyecto, tanto elementos hardware como a nivel software.

Métodos: En este apartado se muestra todos los apartados que se han realizados para el correcto funcionamiento de este proyecto, dicho apartado consta de los siguientes apartados:

1) **Análisis de la señal:** Se recogieron muestras de la señal emitida por el beacon para determinar si la señal recibida era apta para la realización de este trabajo fin de grado.
2) **Procesado de la señal:** Se realiza un estudio de procesado de la señal, donde se estudia una variedad de filtrados, concluyendo con el filtrado óptimo.
3) **Análisis de las muestras filtradas:** En este apartado, se lleva a cabo una serie de mediciones de potencia para posteriormente entrenar una red neuronal con el objetivo de estimar la distancia del emisor a cada uno de los receptores en función de la potencia recibida en los diversos receptores.
4) **Diagramas de radiación:** En este apartado, con las mediciones anteriormente tomadas se muestra el diagrama de radiación para posteriormente comprobar si los diagramas interpolados por la red neuronal son acertados.
5) **Redes neuronales:** En este apartado se estudia todos los conceptos necesarios para la perfecta creación de una red neuronal.

6) **Diagrama de radiación en distancia:** En este apartado se puede ver el resultado de la red neuronal, donde se estima la distancia del emisor al receptor.
7) **Método de trilateración para el posicionamiento:** Una vez estimada la distancia que nos encontramos queda determinar la posición exacta, para eso se realiza un estudio del método y se muestran unas pruebas reales
8) **Método de posicionamiento cartesiano:** En este apartado se busca mejorar la técnica de posicionamiento mediante un método cartesiano, obteniendo resultados claramente más satisfactorios.

Conclusiones: Se hace un análisis de todos los apartados, las ventajas, las desventajas y se llega a la conclusión de cuál de los métodos de posicionamiento cumple con los requisitos de este proyecto.

Anexos: se muestran tanto los programas informáticos realizados para la realización del proyecto como las medidas de potencia obtenidas

Bibliografía: se muestran las fuentes de las cuales nos hemos ayudado para la realización de este proyecto.

4 MOTIVACIÓN

Conforme va avanzando la tecnología en la época actual, van creciendo las necesidades de la sociedad. Y, por tanto, va incrementando, a la vez, la necesidad de crear nuevas técnicas que satisfagan dichas necesidades. Gracias a la tecnología *ibeacons* es posible la localización en espacios cerrados.

En este Trabajo de Fin de Grado hemos realizado un sistema de posicionamiento en interiores basado en la tecnología antes enunciada, *ibeacons* e inteligencia artificial.

Ibeacons utiliza la tecnología *Bluetooth* de baja energía como tecnología inalámbrica, para la comunicación entre nuestros *raspberry pi* y el *ibeacons.*

Entre las ventajas que caracterizan esta tecnología *Bluetooth* destaca el bajo consumo y la distancia que puede abarcar. La distancia aproximada abarcada por dicha tecnología es de, aproximadamente, cincuenta metros que es una distancia relativamente extensa si tenemos en cuenta que se trata de un espacio interior.

Lo que verdaderamente llama la atención de este Trabajo de Fin de Grado es que hay una extensa variedad de posibilidades a la hora de poder realizar este proyecto, desde el uso de las múltiples tecnologías inalámbricas hasta el método de posicionamiento. Son muchas las formas pero también es importante destacar la complejidad, puesto que en el interior de una vivienda, el medio es cambiante y existen excesivas reflexiones, haciendo así el objetivo de conseguir un método eficiente y nuevo es complejo.

5 OBJETIVOS

El objetivo fundamental de este Trabajo de Fin de Grado es encontrar una tecnología con un bajo consumo y, a la vez, un método acertado para la visualización en tiempo real de la posición de una persona u objeto en espacio cerrado, todo esto, con la ayuda de inteligencia artificial.

El uso de redes neuronales artificiales en este proyecto lo que busca es agilizar y facilitar el proceso de posicionamiento e intentar solventar el problema que otros medios de posicionamiento no tienen en cuenta, esto es, la reflexión.

6 ANTECEDENTES

El posicionamiento empezó sus primeros pasos en 1965 con un sistema norteamericano llamado *transit,* con el que podían obtener una posición con un error de unos 100 metros, con la ayuda de 6 satélites.

Posteriormente, le siguió el sistema *Timation* y en el año 1973 llego *Navstar*, que gracias al desarrollo de los relojes atómicos, se diseñó una constelación de satélites, portando cada uno de ellos uno de estos relojes y estando todos sincronizados con base en una referencia de tiempo determinado. De este último es del que conocemos como hoy en día el sistema de posicionamiento GPS.

Por último, a principios del siglo XXI se puso en marcha un estudio por la Unión Europea denominado *Galileo* con el cual la precisión puede llegar a ser de centímetros.

Como queda claro a nivel histórico, el posicionamiento se creó para posicionamientos terrestres, no para interiores. Pero con el avance de las tecnologías y la llegada de nuevas necesidades, las tecnologías como WiFi, Bluetooth, RFID, infrarrojos,

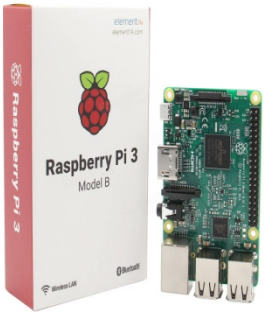
NFC, ZigBee, ultrasonido han empezado a aprovecharse para el posicionamiento en interiores puesto que su señal no decae dentro del edificio como le pasa a la señal del satélite.

7 UBICACIÓN

La ubicación elegida para las pruebas realizadas ha sido la Escuela Politécnica Superior de Linares. Esto ha sido así por razones de ubicación y por la facilidad para poder hacer uso de los materiales.

La sala elegida, en concreto, para las pruebas ha sido el laboratorio de radiocomunicaciones y microondas que, debido a su forma, facilita las labores de mediciones y el trabajo, puesto que es un laboratorio conocido y el que cumple con los requisitos que el trabajo demanda.

8 RECURSOS UTILIZADOS

SOFTWARE	FORMA	CARACTERÍSTICAS
Raspberry pi 3		• CPU Quad Core 1.2GHz Broadcom BCM2837 de 64 bits • 1GB de RAM • BCM43438 LAN inalámbrica y Bluetooth de baja energía (BLE) a bordo • GPIO extendido de 40 pines • 4 puertos USB 2 • Salida estéreo de 4 polos y puerto de video compuesto • HDMI de tamaño completo • Puerto de cámara CSI para conectar una cámara Raspberry Pi • Puerto de pantalla DSI para conectar una pantalla táctil Raspberry Pi • Puerto Micro SD para cargar su sistema operativo y almacenar datos • Fuente de alimentación micro USB conmutada actualizada de hasta 2.5 A
Ibeacons Bluetooth leibks105		• Duración de la batería 30-40 meses (dependiendo de la potencia de Tx en el intervalo de 1s) 3-4 meses (dependiendo de la potencia de Tx en el intervalo de 100 ms) • Protocolos: Beacon Eddystone: UID, URL, TLM y EID Actualización de firmware en el aire Eddystone Configuration GATT Service • Alcance en situaciones optimas de 70m

Telemetro láser		• Distancia máxima de medida: 50 metros • Precisión: ± 2 mm • Medidas aéreas y de volumen • Cálculo matemático de Pitágoras 1+2 • Cálculos de suma y resta • Almacenaje de medidas
Medidor de ángulos láser		• Rango angular de 0 a 180º • Grosor de línea <3mm a 4m • Capacidad de trabajo > 20 Horas
SOFTWARE	USO	
Matlab	• Programa o herramienta matemática, que consta con un lenguaje de programamion propio llamado "lenguaje M" del que nos hemos ayudado para programar todos los archivos necesarios para la creacion de la aplicación.	
Putty	• PuTTY es un cliente SSH y Telnet que ofrece la posibilidad de conectarnos a servidores remotos para ejecutar los comandos que deseemos. • Hemos utilizado este programa para acceder al raspberry, configurarlo, y poder recoger las medidas de potencia.	

Tabla 1: Recursos Utilizados

9 MÉTODOS

9.1 ANÁLISIS DE LA SEÑAL

El punto de partida de la realización de este proyecto es el análisis de la potencia de la señal *Bluetooth* emitida por el ibeacon.

Para el análisis enunciado, hemos usado un raspberry pi. Este servirá como receptor para recoger la señal y procesarla y para saber si la señal obtenida por nuestro receptor es ruidosa o, en cambio, es apta para seguir con el proyecto.

Tras recoger la señal, esta se observa en la pantalla, como se puede ver en la imagen que se muestra a continuación

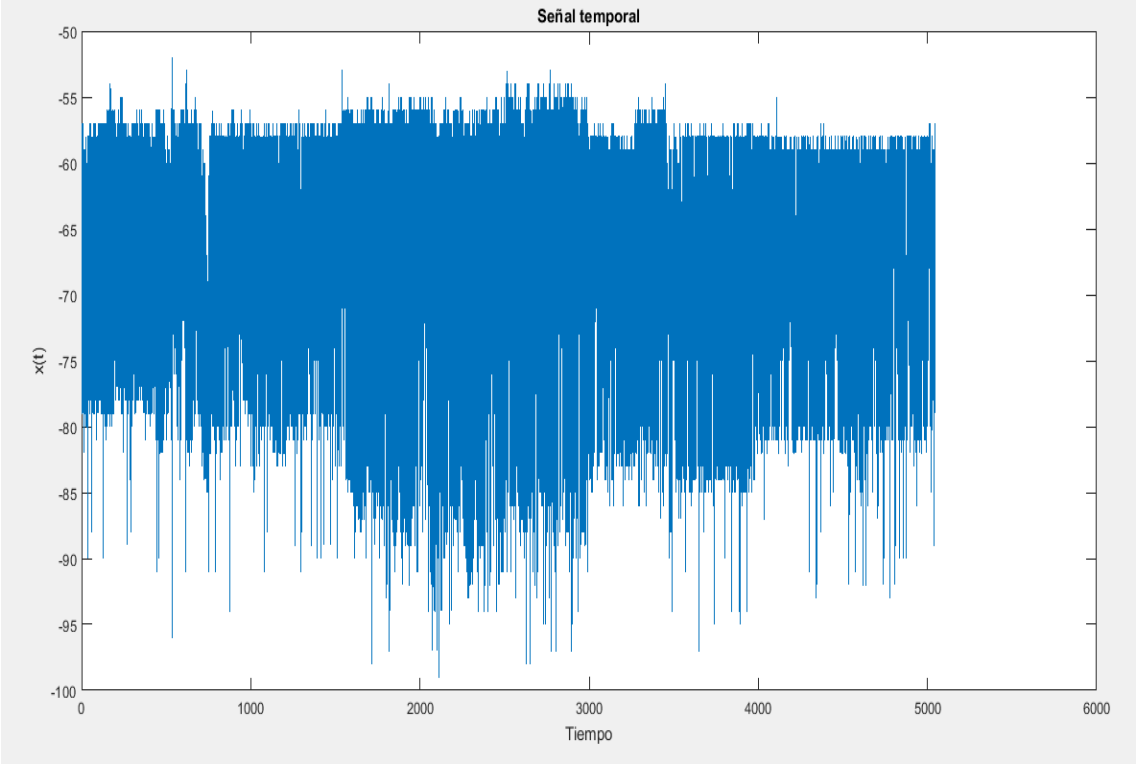


Figura 1: Señal emitida por el ibeacon

9.2 PROCESADO DE LA SEÑAL

Tras analizar la señal emitida por el *ibeacon*, nos damos cuenta que la señal es muy ruidosa, lo cual era de esperar puesto que la banda del *bluetooth* es la de 2.4 Ghz y es una banda ruidosa.

Antes de seguir con la realización del proyecto deberemos filtrar la señal para obtener una señal lo más fiable posible para una correcta interpretación de los datos recogidos. Usaremos una serie de filtros para analizar cual nos conviene usar

9.2.1 FILTRO BUTTERWORTH

El filtro de *Butterworth* es uno de los filtros más usados y básicos, la característica de este filtro es que ofrece una respuesta lo más plana posible hasta la frecuencia de corte. Seguidamente a la frecuencia de corte la respuesta empieza a decaer a razón de "20n [dB]" por década. Siendo "n" el número de polos de nuestro filtro

Tras la realización de filtro de *Butterworth* y tras su análisis podemos obtener lo siguiente:

9.2.1.1 Respuesta en frecuencia filtro Butterworth:

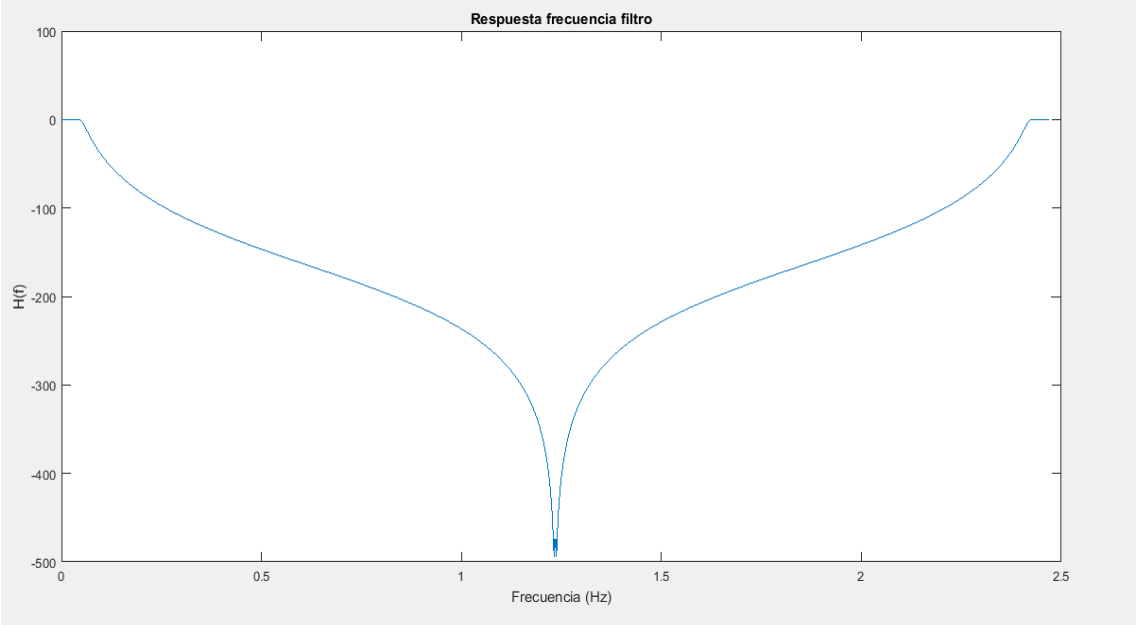


Figura 2: Respuesta en frecuencia filtro Butterworth

9.2.1.2 Respuesta en fase filtro Butterworth

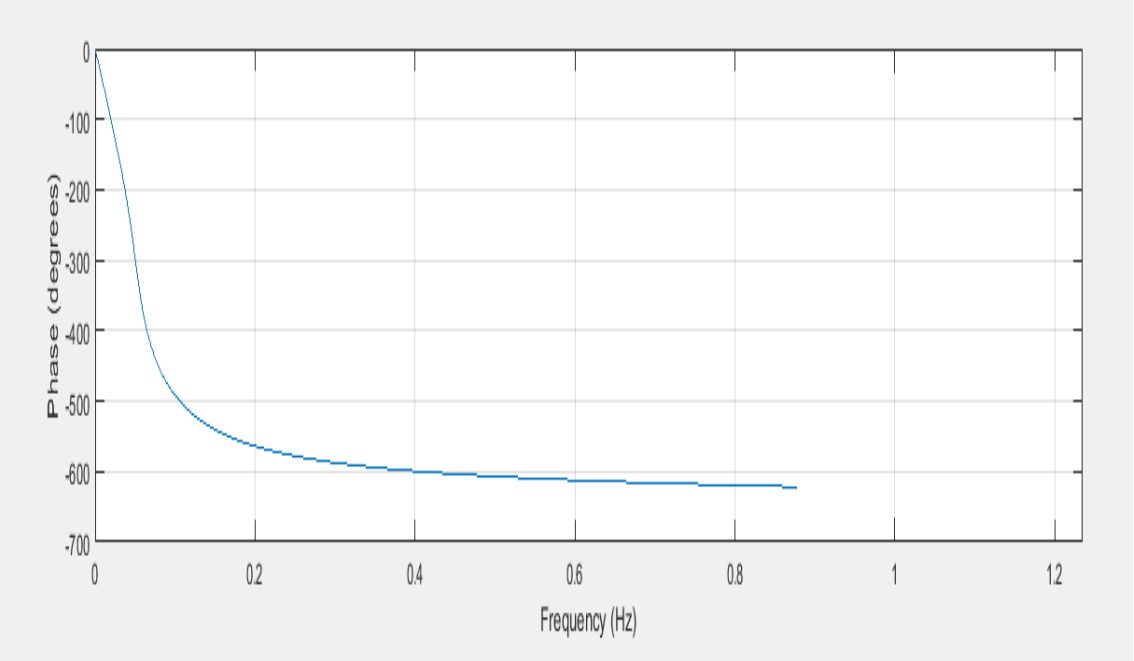


Figura 3: Respuesta en fase filtro Butterworth

9.2.1.3 Señal filtrada filtro Butterworth

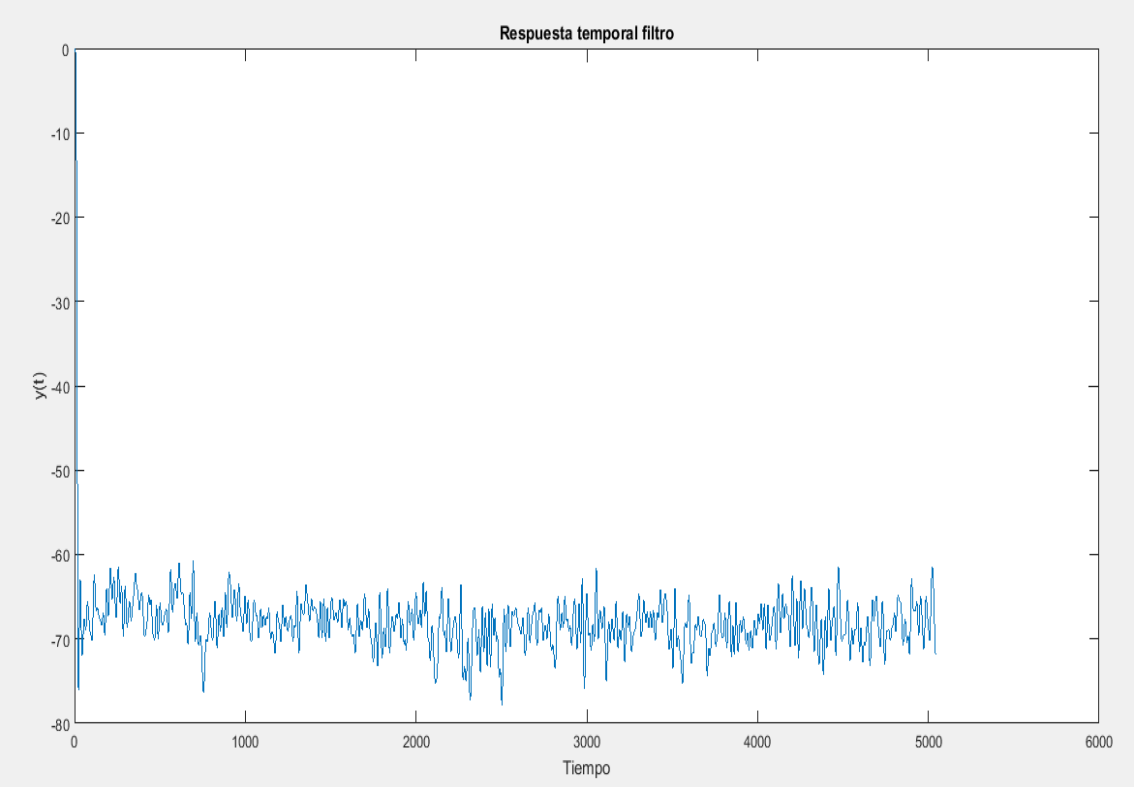


Figura 4: señal filtrada filtro Butterworth

A pesar de que la señal de salida respecto a la señal de entrada ha mejorado sustancialmente, todavía se puede percibir que en la salida del filtro un poco de ruido y, por tanto, la medida que se obtendría no sería del todo fiable.

9.2.2 FILTRO CHEBYSHEV

El filtro de *Chebyshev* ofrece una caída más abrupta a partir de la frecuencia de corte con un número de orden menor.

Para que el orden sea menor y la caída más abrupta, este filtro requiere un pequeño rizado en la banda pasante y un ligero empeoramiento del comportamiento en fase, como se podrá observar a continuación.

9.2.2.1 Respuesta en frecuencia filtro chevysheb

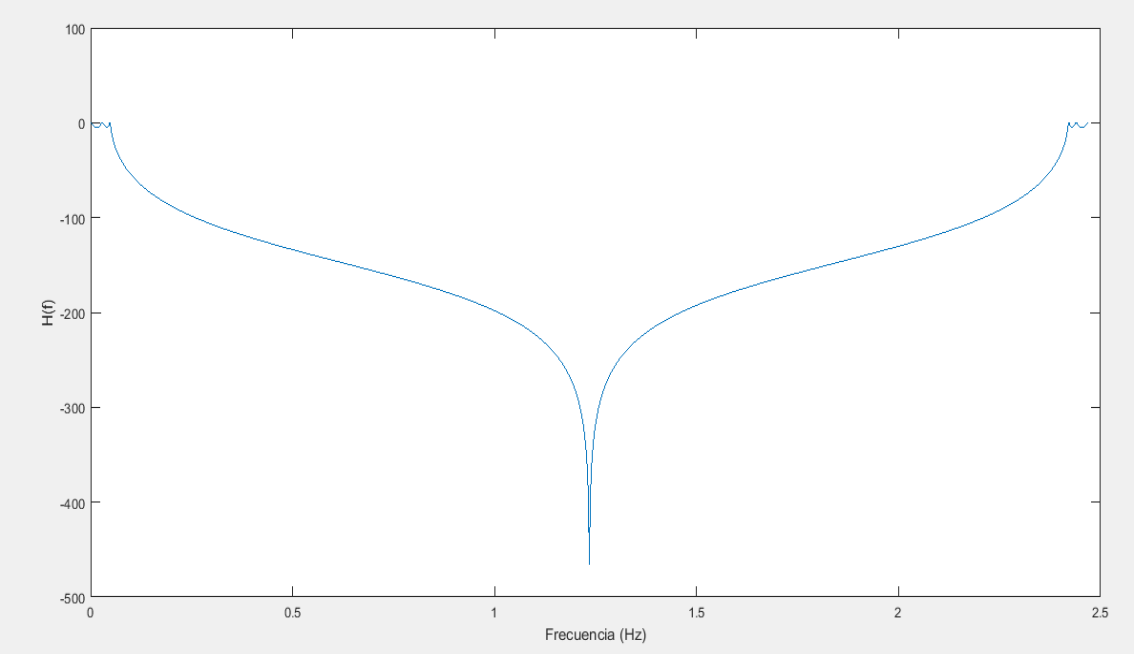


Figura 5: respuesta en frecuencia filtro chevysheb

9.2.2.2 Respuesta en fase filtro chevysheb

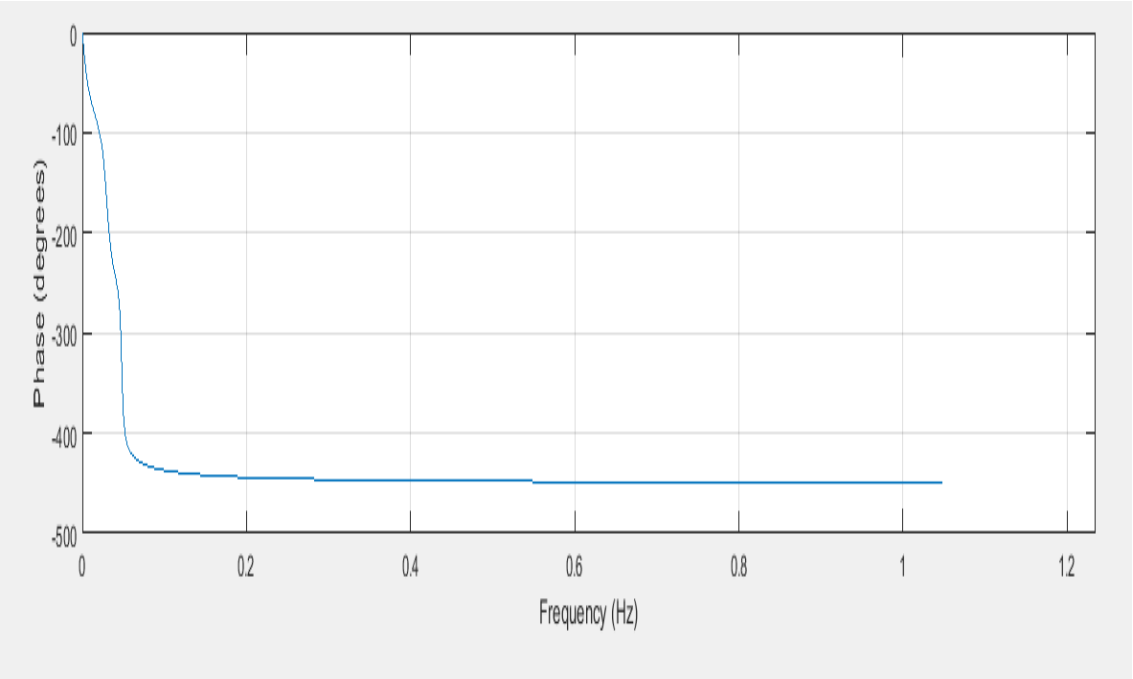


Figura 6: respuesta en fase filtro chevyshe

9.2.2.3 Señal filtrada filtro chevysheb

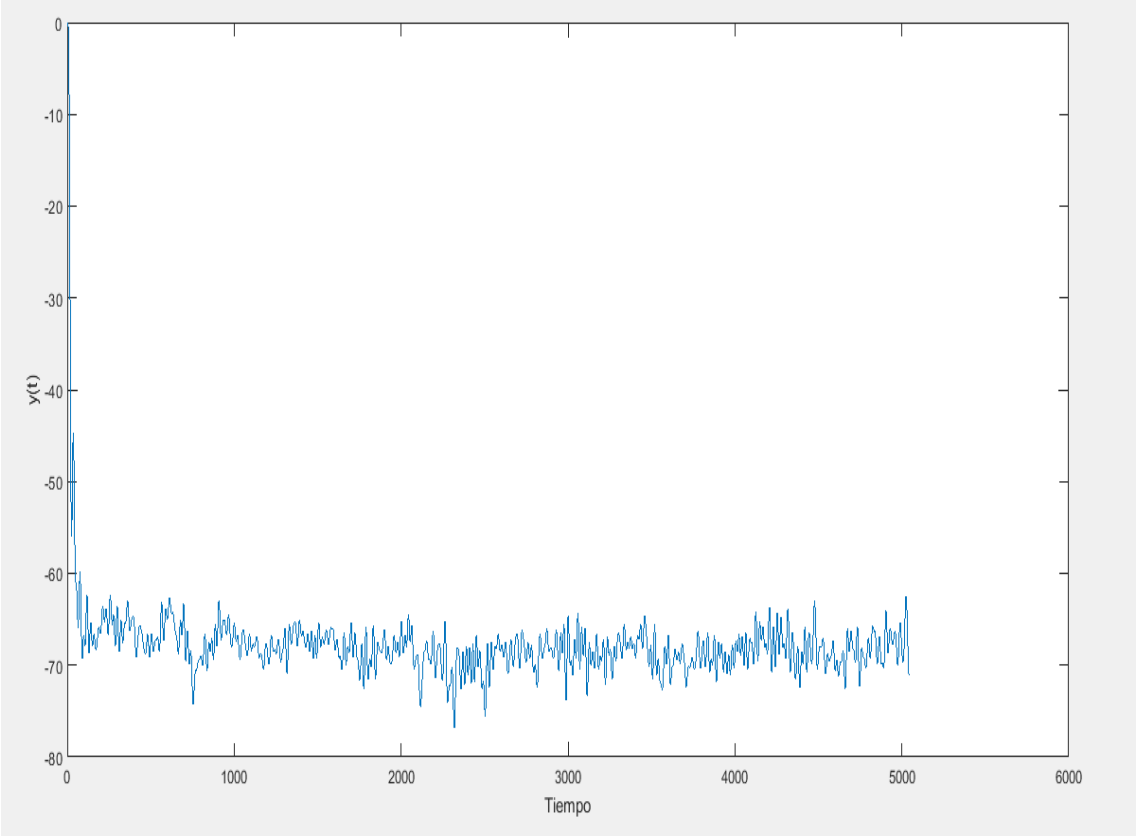


Figura 7: señal filtrada filtro chevysheb

A pesar de que la señal de salida ha mejorado respecto a la señal de entrada, todavía se puede apreciar, en la salida del filtro, un poco de ruido y, por tanto, la medida obtenida no sería del todo fiable.

Tras estas pruebas podemos observar que los filtros convencionales no son la mejor de las opciones, por ello vamos a realizar un estudio de un nuevo filtrado llamo filtro de *Kalman*, con el que esperamos obtener una salida del todo fiable.

9.2.3 FILTRO DE KALMAN

El filtro de Kalman en su conjunto una serie de ecuaciones matemáticas que desarrollan un estimador tipo predictor–corrector, con el único objetivo de minimizar el error estimado de la covarianza.

La peculiaridad con otros filtros, es que este no necesita una frecuencia de corte, consiguiendo con esto poder filtrar en toda la banda del espectro. Además de ser un filtro que ahorra mucha memoria puesto que solo depende de muestras anteriores.

En primer lugar hay que saber que las dos ecuaciones del filtro de Kalman son:

$$X_k = Ax_{k1} + Bu_k + W_{k-1} \quad (1)$$

$$Z_k = Hx_k + V_k \quad (2)$$

"A" es una matriz de forma que hace referencia al estado anterior k-1 con el estado actual. Para nuestro caso en concreto, esta matriz toma valor constate, y su valor es "1" puesto que sabemos que tendremos que tener una señal constate en la que la muestra posterior debe de valor lo mismo que la anterior.

"B" es una matriz de forma, que hace referencia con la entrada de control uk. En nuestro caso, esta parte no entra en juego puesto que no tendremos señal de control uk

"H" es una matriz, que relaciona el estado con la medición. Para nuestro caso en concreto este valor valdrá "1", puesto que esta relacionando el estado con la medición y

sabemos que la medición, es la ecuación de estados más algo de ruido.

Las variables W_k y V_k hacen referencia al ruido del proceso y el ruido de la medición, con una varianza Q y R respectivamente.

A continuación podrá ver un esquema de las etapas y ecuaciones que sigue el filtro de kalman.

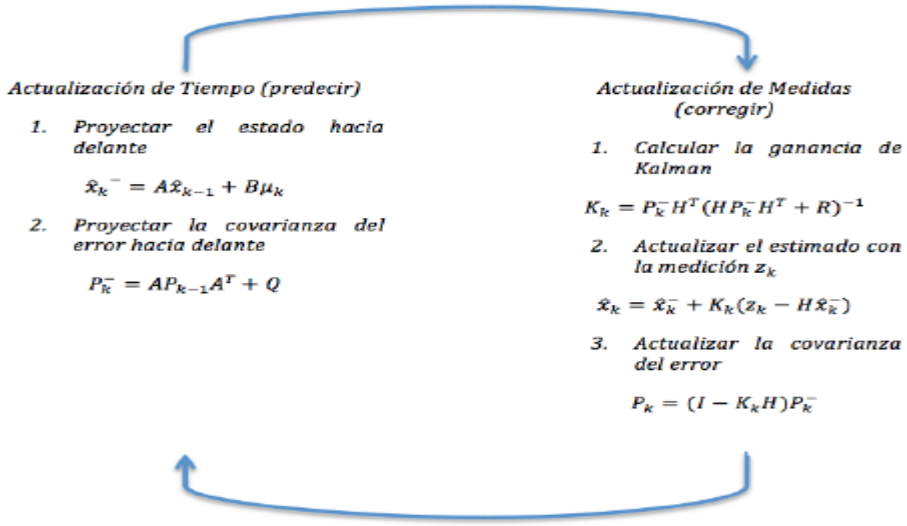


Figura 8: Esquema fases y ecuaciones filtro de kalman

Las ecuaciones de predicción obtienen una estimación de la covarianza del error y el estado actual del sistema en el tiempo respecto del anterior (k-1).

$$\hat{x}^- = A\hat{x}_{k-1} + B\mu_k \quad (3)$$

$$P^- = AP_{k-1}A^T + Q \quad (4)$$

Siendo (3) la estimación a priori y (4) covarianza del error a priori de la estimación.

La otra parte es la parte de corrección donde estas se basan en una nueva información, que es la de las mediciones para llegar a conseguir una mejor estimación.

$$K_k = P^{-}H^T (HP^{-}H^T + R)^{-1} \quad (5)$$

$$\hat{x}_k = \hat{x}^{-} + K_k(Z_k - H\hat{x}^{-}) \quad (6)$$

$$P_k = (I - K_k H)P^{-} \quad (7)$$

En la que (5) es la ganancia del filtro de Kalman y se calcula de forma que se minimice la covarianza del error.

Seguidamente nos encontramos con (6) que es la actualización del valor estimado con la medición con el fin de mejorar la estimación de la fase de predicción.

Por último, nos encontramos con (7) que es la actualización de covarianza del error.

Seguidos estos pasos anteriormente descritos, creamos nuestro filtro de Kalman y se le introduce una señal de entrada obtenida por uno de nuestros receptores, obteniendo a su salida:

9.2.3.1 Señal filtrada filtro de kalman

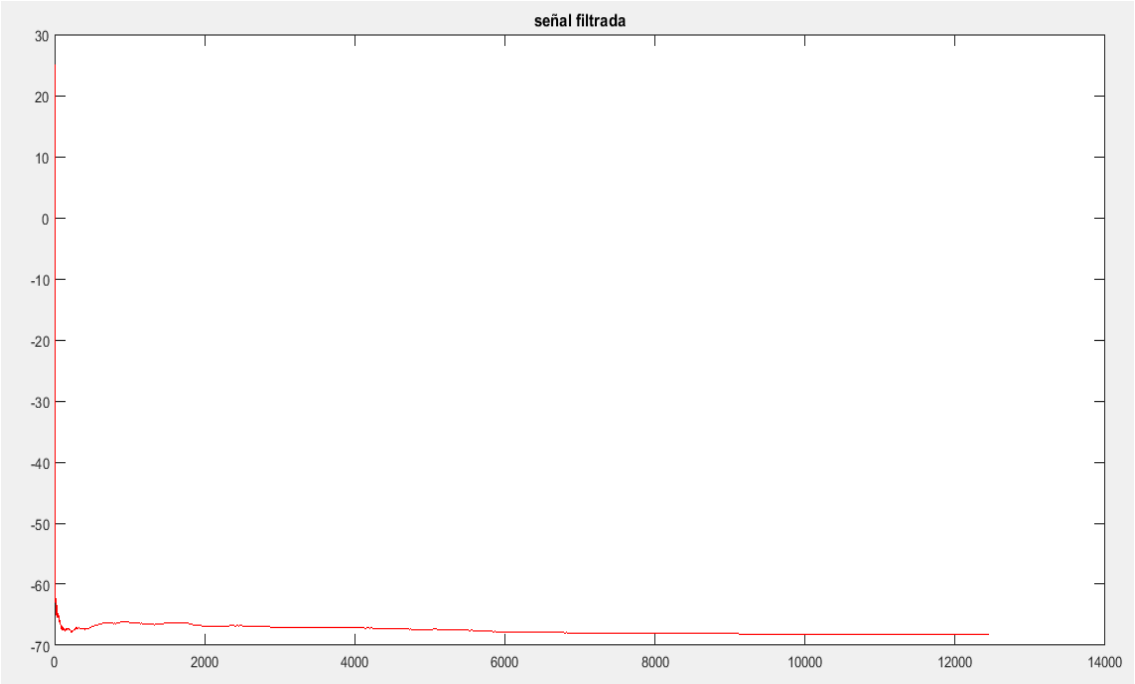


Figura 9: Señal filtra por el filtro de kalman

Si se analiza la señal de salida de nuestro filtro se ve a simple vista que la señal ha sido mejorada gratamente, y ahora las medidas que tomemos serán del todo fiables, con el ruido eliminado.

Si comparamos con la señal de entrada se puede ver una gran diferencia que a continuación se mostrara en la siguiente figura.

9.2.3.2 Comparación señal filtrada con señal real

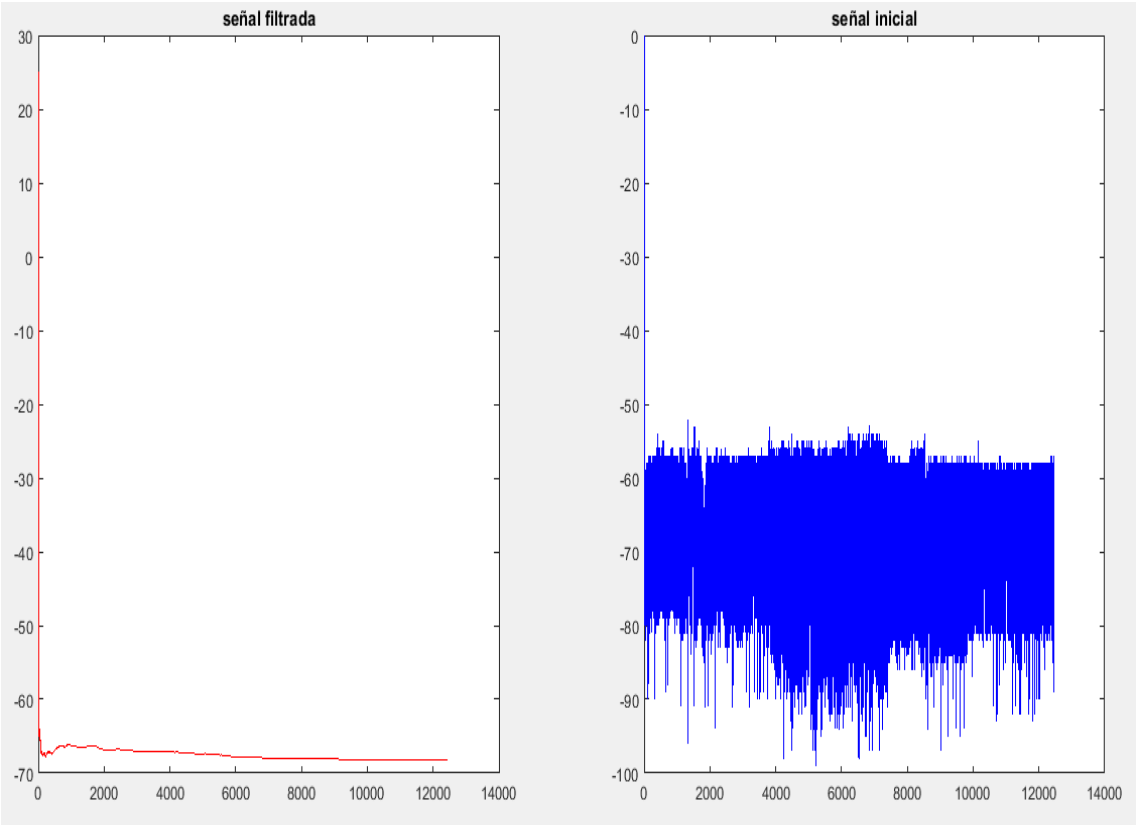


Figura 10: comparativa señal filtrada por filtro de kalman y señal original

Para una mejor comparación de cómo se comporta nuestro filtro de *Kalman* recurrimos a quitarle a nuestra señal la media sabiendo que entono a esa cifra debe estar nuestra medida y filtramos el ruido sabiendo que a la salida del filtro deberíamos obtener lo más parecido a cero posible.

Después de filtrar el ruido obtenemos el resultado mostrado en la siguiente figura.

9.2.3.3 Ruido filtrado

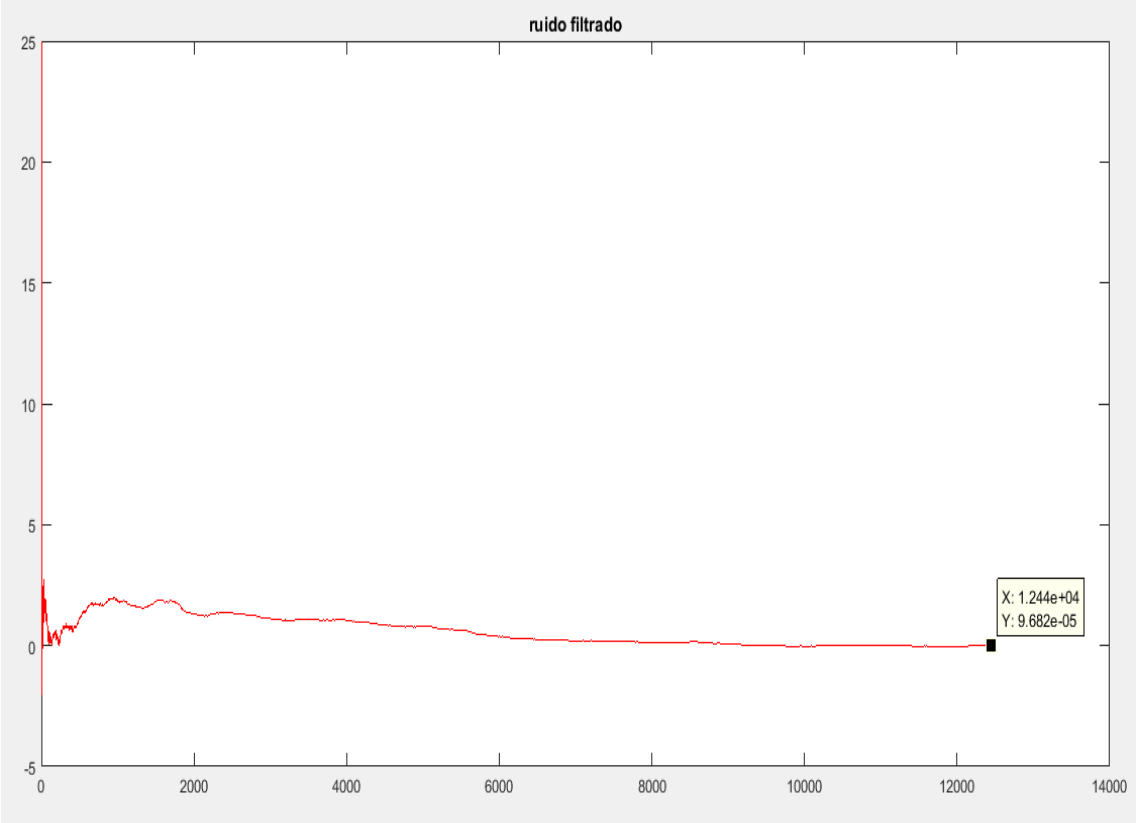


Figura 11: Ruido filtrado por filtro de kalman

9.2.4 CONCLUSIÓN DEL PROCESADO DE LA SEÑAL

Si comparamos el filtro *Chebyshev* con el filtro *Butterworth* lo primero que observamos es que ninguno de los dos filtrados es adecuadi para esta aplicación. Esto no significa que no sean buenos filtros, sino que no son aptos para esta aplicación. Es destacable que, para un comportamiento semejante entre el filtro *Butterworth* y el *Chebyshev*, el filtro de *Butterworth* necesita un orden de 7 mientras el *Chebyshev* cumple la misma función con un orden 5 pero, con algo de rizado en la banda de paso y un peor comportamiento en fase. Para un orden fijado, el filtro de *Butterworth*, el ancho de región de transición es mayor, pero, sin embargo, contará con un mejor comportamiento en fase y no tendrá rizado en la banda de paso.

En lo relativo al filtrado de *Kalman*, esta es una aplicación totalmente óptima para el uso de filtrado de señal puesto que elimina la señal no deseada por completo. Es una herramienta matemática excelente puesto que posee dos etapas: la de predicción y la de corrección, que lo hacen idóneo para una buena estimación. También cuenta con otra

diferencia en torno a otros tipos de filtros ya que no tiene frecuencia de corte puesto que se basa en la característica del ruido, teniendo este la ventaja de poder filtrar en todas las bandas de frecuencias sin tener que definir una u otra.

9.3 ANÁLISIS DE LAS MUESTRAS FILTRADAS

Una vez elegido el filtro, vamos a determinar que nuestro plano de trabajo se corresponde con un plano en coordenadas polares, puesto que el método que usaremos para el posicionamiento será el de trilateración.

Posteriormente, tomaremos las mediciones cada 0.5 metros de distancia y cada 10 grados respecto cada uno de los receptores, para así, con posterioridad, poder hacer una buena estimación de la distancia.

Para la realización de este análisis nos hemos servido de la ayuda de algunos de la información anteriormente citada, en concreto, en el apartado de requisitos *hardware*. Estos son:

- Telemetro láser para saber la distancia a la que se toma la medición.
- Medidor de ángulos láser para saber en qué ángulo en el que se toma la medición.
- *Ibeacons* para emitir una señal *bluetooth*.
- *Raspberry pi* para la captación de la señal.

Una vez que hemos recogido todas las muestras una por una y han sido analizadas por nuestro filtro de *Kalman*, obtendremos una tabla de mediciones que, por motivo de ahorro de espacio, no se mostrarán todos los datos ni las muestras obtenidas antes de filtrar, puesto que son muy extensas y existen varios receptores.

9.3.1 Tabla mediciones en [dBm] nodo 1 para el plano polar

Distancia / grados	0.5m	1m	1.5m	2m	2.5m	3m	3.5m	4m	4.5m	5m
90g	-58.9	-57.81	-60.17	-64.71	-61.2	-73.4	-71.6	-73.71	-71.54	-86.57
100g	-56.03	-61.21	-65.4	-67.09	-63.22	-70.2	-67.63	-72.45	-74.18	-67.12
110g	-51.09	-60.54	-61.77	-71.1	-69.03	-65.17	-76.98	-69.41	-67.54	-70.39
120g	-55.13	-57.31	-58.76	-69.88	-67.09	-68.62	-63.34	-66.61	-64.66	-75.3
130g	-54.39	-53.34	-61.03	-68.28	-65.42	-63.79	-61.41	-61.09	-65.88	-72.54
140g	-52.25	-59.44	-67.98	-66.46	-64.5	-68.87	-64.11	-64.91	-67.49	
150g	-57.42	-61.37	-64.1	-63.31	-69.24	-64.06	-63.28	-66.48		
160g	-59.56	-65.1	-68.6	-65.62	-64.26	-63.07	-67.65			
170g	-52.81	-69.22	-63.2	-68.93	-66.84	-62.52	-68.72			
180g	-53.5	-64.22	-69.16	-63.36	-60.84	-63.92				

Tabla 2: mediciones en [dBm] nodo 1 para el plano polar

A pesar de que lo normal sería que la potencia decayera con la distancia, no sucede así en este caso, al menos en algunas de las mediciones. Esto es debido al medio donde tomamos las mediciones, es una sala con muchos elementos que afectan a la propagación de la onda como, por ejemplo:

- **Reflexión**: La reflexión sucede cuando una onda cambia de dirección al entrar en contacto con el elemento de separación entre dos medios cambiantes, regresa al punto donde se originó.
- **Refracción**: La refracción es una modificación de dirección y velocidad de la onda cuando pasa de un medio de índice de refracción n1 a otro con índice refractivo n2, siendo n1 distinto a n2.
- **Difracción**: La difracción es un fenómeno característico de las ondas, que es observable cuando se atraviesa un obstáculo del tamaño de su longitud de onda. El caso más característico es cuando una onda atraviesa una rejilla la onda se ve distorsionada y se propaga para cualquier dirección.
- **Multitrayecto**: El multitrayecto es un fenómeno consistente en la propagación de una onda por varios caminos diferentes que finalmente llegan al mismo destino.

Ello se debe a los fenómenos de reflexión y de difracción y puede llegar a ser un fenómeno perjudicial o beneficioso.

También se ha podido observar que las mediciones dependen, en gran medida, de la ubicación de la habitación, ya sea por el movimiento de personal humano, como de objetos. Otro aspecto que hemos podido observar es la variación de la potencia de la medición según el ángulo de inclinación del ibeacons. En nuestro caso, esto no va a ser una molestia, puesto que las medidas se han recogido dejando el ibeacons en el suelo, pero para futuras investigaciones, sería de gran interés tenerlo en cuenta como se podrá ver en las siguientes imágenes, donde se ha recogido la señal en el mismo punto, salvo moviendo la inclinación de ibeacons.

Ibeacons plano

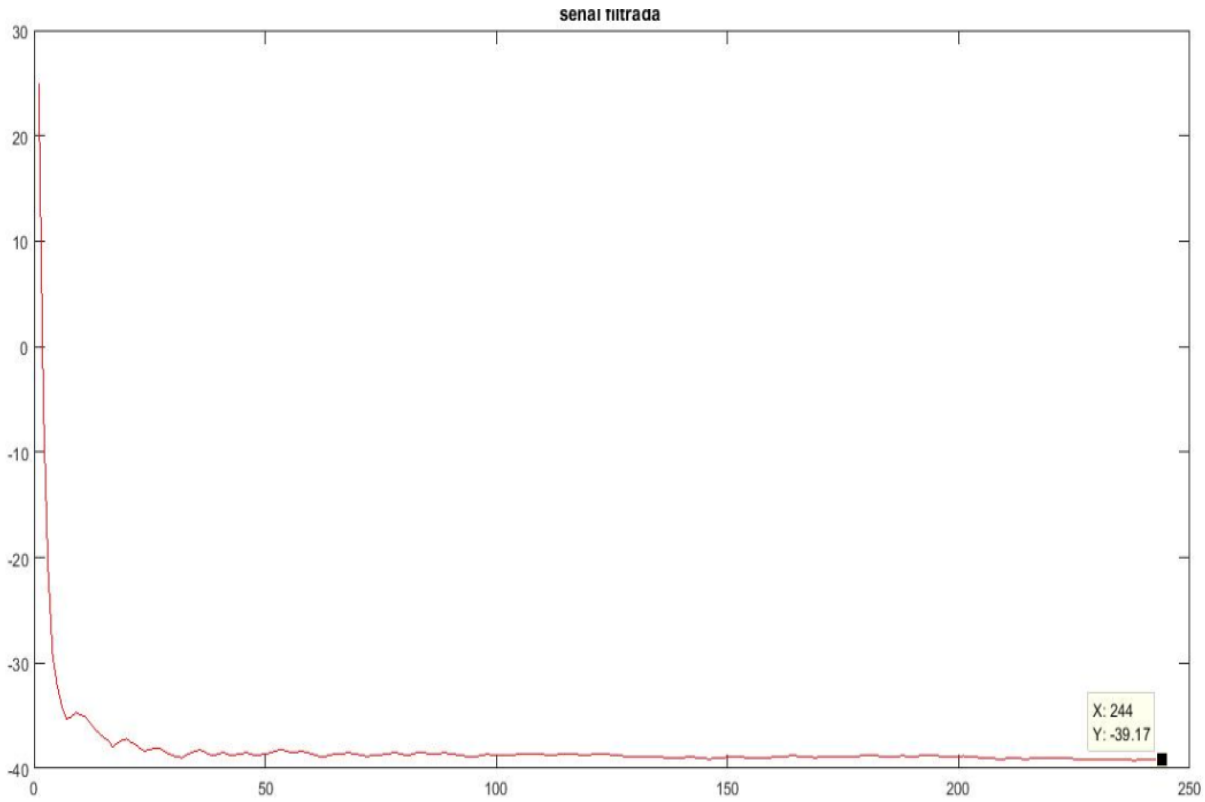


Figura 12: Medida con el ibeacons plano

Ibeacons inclinado

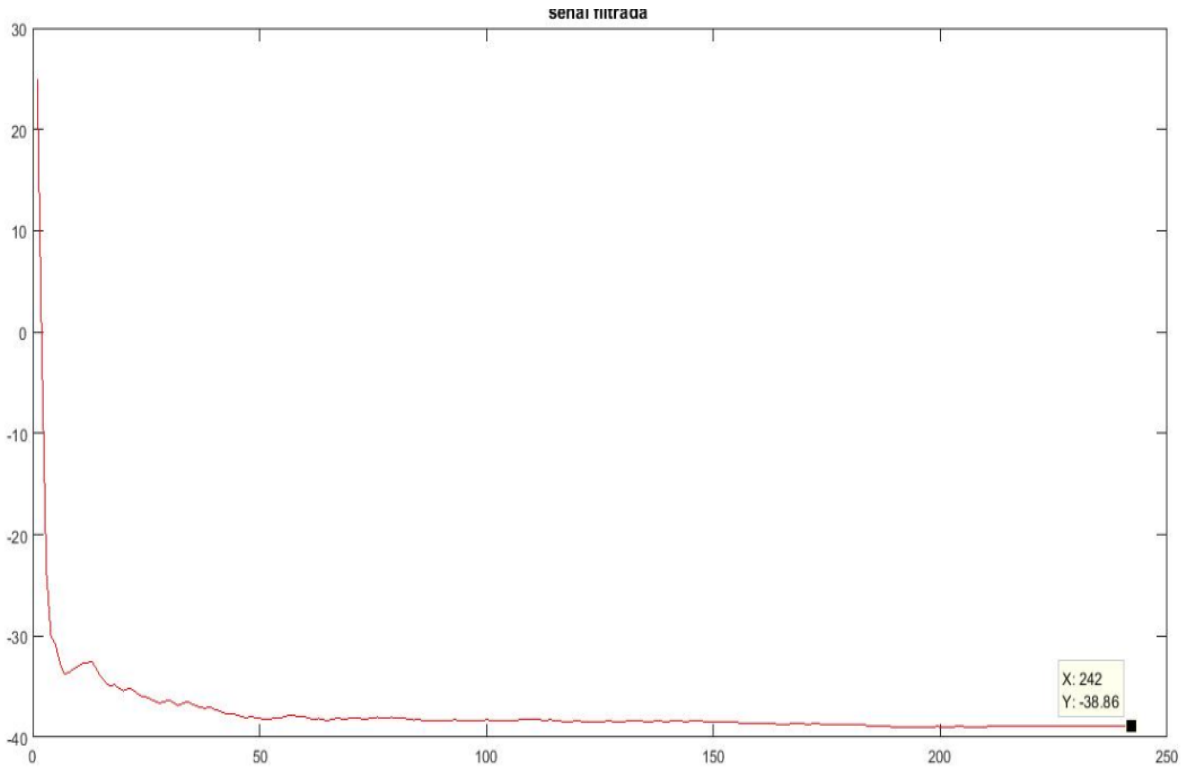


Figura 13: Medida con el ibeacons inclinado

Como se puede observar no es una variación totalmente brusca, la medida cambia en 1 dBm, pero también es cierto que hay ángulos que pueden afectar mucho más a la medida, esto es una simple prueba para corroborar que la inclinación afecta.

9.4 DIAGRAMA RADIACCIÓN

Un aspecto importante que caracteriza a una antena es su diagrama de radiación. Un diagrama de radiación no deja de ser si no una representación grafica de sus características de radiación en función del ángulo.

En nuestro caso, es importante conocer el diagrama de radiación para saber si las medidas que hemos tomado están bien o ha habido algún fallo. También nos servirá pasar saber si la red neuronal nos da una salida acertada, puesto que no va a ser muy distinto al diagrama de radiación, salvo que este estará en función de la distancia en vez de la potencia.

De antemano sabemos que nuestro diagrama de radiación debería de ser omnidireccional, esto no ocurrirá por que como hemos visto en las mediciones hay algunos factores que altera el medio de trasmisión, haciendo que pueda haber alguna variación.

Para saber si es esta todo correcto, hemos tomado mediciones a lo largo a 1 metro de distancia respecto un ángulo de 90º obteniéndose los siguientes resultados:

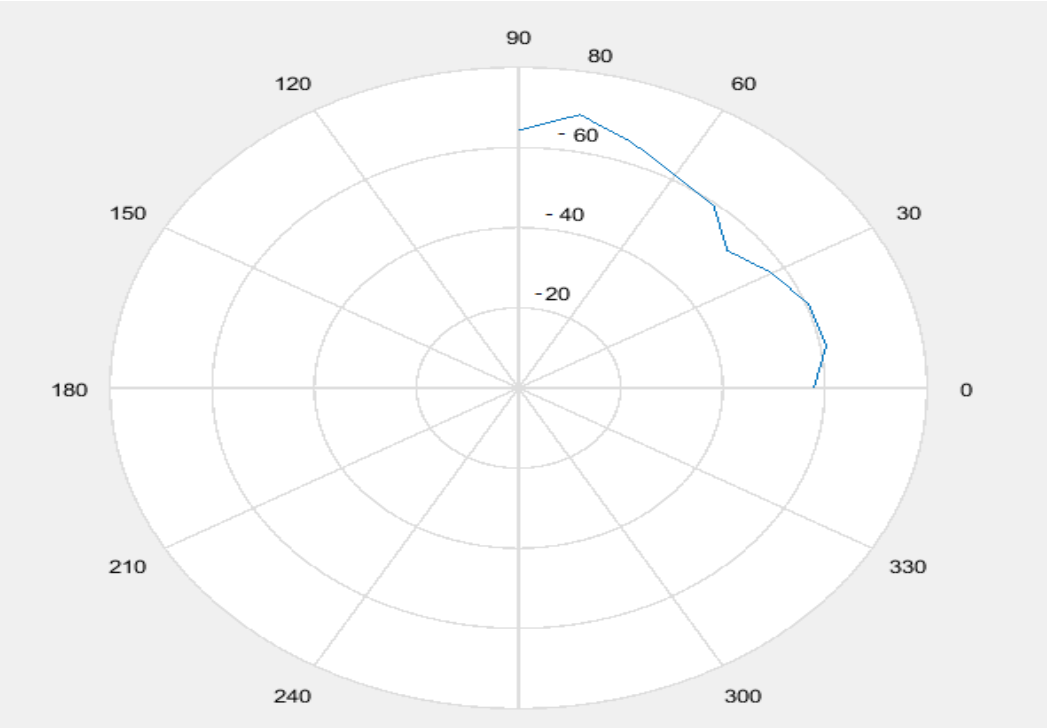


Figura 14: diagrama de radiación

9.5 REDES NEURONALES

9.5.1 DEFINICIÓN RED NEURONAL

"Las redes neuronales artificiales son redes interconectadas masivamente en paralelo de elementos simples (usualmente adaptativos) y con organización jerárquica, las cuales intentan interactuar con los objetos del mundo real del mismo modo que lo hace el sistema nervioso biológico". [21].

9.5.2 VENTAJAS DE LAS REDES NEURONALES ARTIFICIALES

Debido a su constitución y a sus fundamentos, las redes neuronales artificiales presentan un gran número de características semejantes a las del cerebro. Por ejemplo, son capaces de aprender de la experiencia, de generalizar de casos anteriores a nuevos casos, de abstraer características esenciales a partir de entradas que representan información irrelevante, etc. Esto hace que ofrezcan numerosas ventajas y que este tipo de tecnología se esté aplicando en múltiples áreas. Entre las diversas ventajas, cabe destacar [4]:

- **Aprendizaje adaptativo**. Las redes neuronales tienen la capacidad de auto aprender basándose en un entrenamiento.
- **Autoorganización**. Las redes neuronales pueden organizarse por sí solas mediante la información que estas perciben del aprendizaje.
- **Tolerancia a fallos**. Las redes neuronales pueden llegar a ofrecer una respuesta aceptables incluso si se daña la red parcialmente.
- **Flexibilidad:** Las redes neuronales puede llegar a funcionar bien incluso variando parcialmente información de entrada, como señales con ruido u otros cambios en la entrada
- **Operación en tiempo real**.
- **Fácil inserción dentro de la tecnología existente**.

9.5.3 ELEMENTOS BASICOS DE UNA RED NEURONAL

Una red neuronal artificial la constituyen 3 capas que son: la de entrada, la oculta y la de salida.

- **Capa de entrada**: Esta capa está compuesta por aquellas neuronas que reciben los datos de entrada de nuestra red. En nuestro caso, esta capa contendrá la potencia que reciben los nodos.
- **Capa de salida**: Esta capa es aquella cuyas neuronas proporcionan la salida de la red neuronal. En nuestro caso será la distancia de la baliza.
- **Capa oculta**: Esta capa está compuesta por aquellas neuronas que no tienen una conexión directa con el entorno. Esta capa ofrece la posibilidad de contener más de una capa oculta.

En la siguiente imagen se puede apreciar un esquema de lo que puede ser una red neuronal artificial.

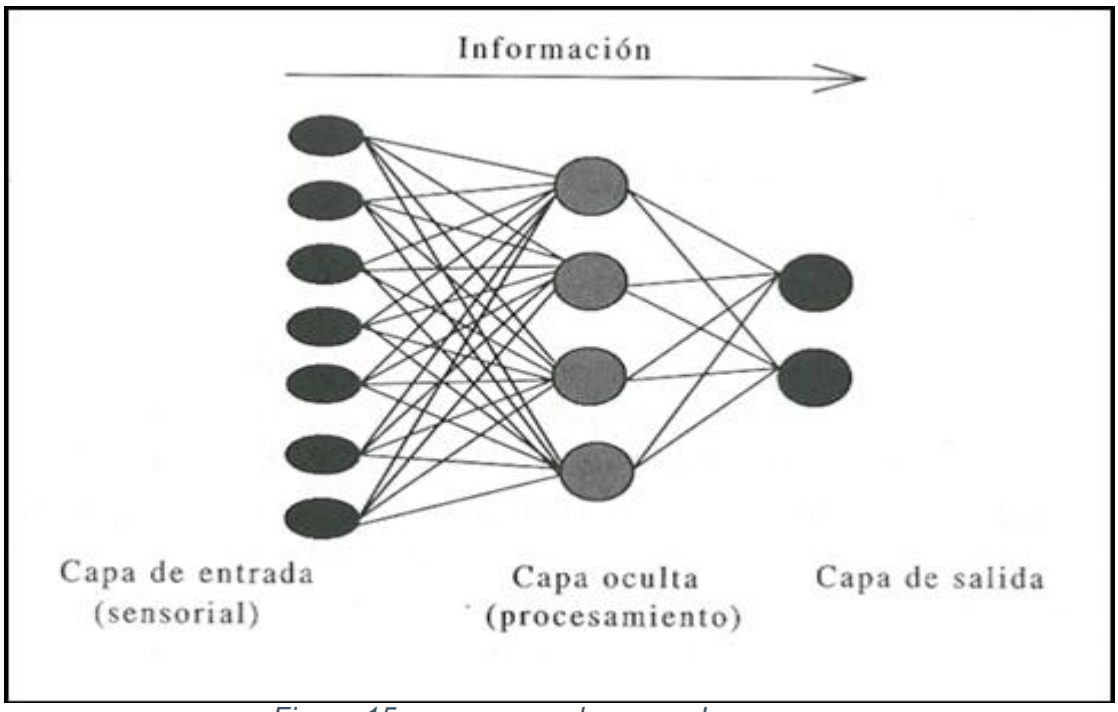


Figura 15: esquema red neuronal

9.5.4 FUNCIÓN DE ACTIVACION

Entre las unidades o neuronas que forman una red neural artificial existe un conjunto de conexiones que unen unas a otras. Cada unidad transmite señales a aquellas que están conectadas con su salida. Asociada con cada unidad Ui hay una función de salida fi(ai(t)), la cual transforma el estado actual de activación ai(t) en una señal de salida yi(t), es decir: [3]

$$yi(t) = fi(ai(t)) \tag{8}$$

El vector que contiene las salidas de todas las neuronas en un instante t es:

$$Y(t) = (f1(a1(t)), f2(a2(t)),..., fi(ai(t)),..., fN(aN(t))) \tag{9}$$

En algunos modelos, esta salida es igual al nivel de activación de la unidad, en cuyo caso la función fi es la función identidad, fi(ai(t)) = ai(t). A menudo fi es de tipo sigmoidal y suele ser, por lo general, la misma para todas las unidades. [3]

Las cuatro funciones de transferencia más usadas son:

- Función escalón.
- Función lineal.
- Función Sigmoidal.
- Función Gaussiana.

9.5.4.1 FUNCIÓN ESCALÓN

Es una función simple, puesto que contamos con neuronas binarias. Lo que significa que cuando el valor de las entradas está por encima de un umbral, se activa el 1; si es menor se activa 0 (ó -1).

$$f(neta) = \begin{cases} 1 & neta \geq 0 \\ 0 & neta < 0 \end{cases}$$

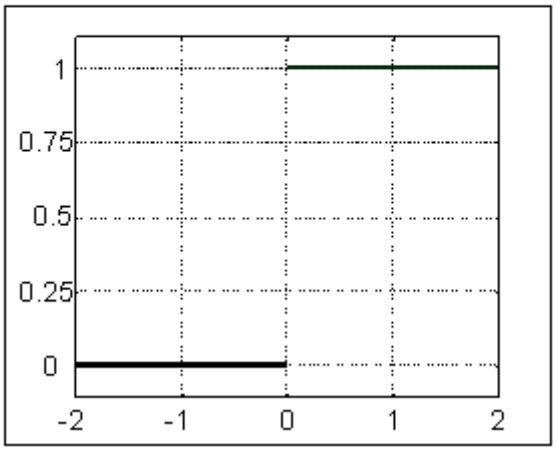


Figura 16: Función de activación escalon

9.5.4.2 FUNCIÓN LINEAL

La función lineal tiene una expresión tal que: f(x) = x. En las neuronas que usan función lineal, si entre la suma de las señales de entrada es esta por debajo de un límite inferior, la activación es 0 (ó -1). Por otra parte si la suma de las entradas es mayor que un umbral superior, entonces la activación es 1. Por otra parte si la suma de las entradas nos entrega un valor entre medias de esos umbrales superior e inferior la activación se define como una función lineal de la suma de las señales de entrada. [3]

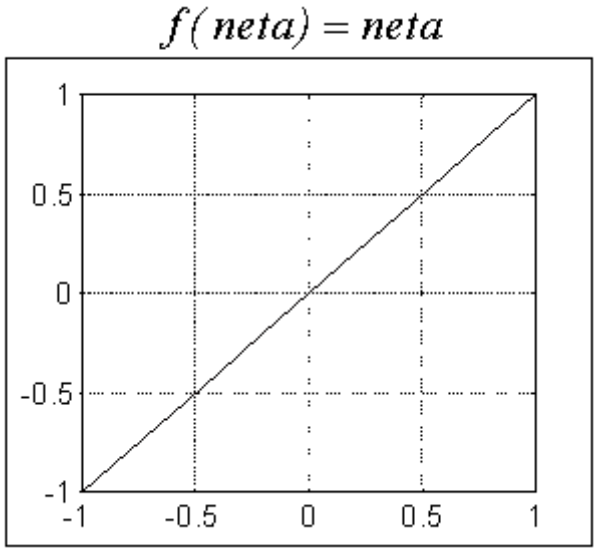


Figura 17: función de activación lineal

9.5.4.3 FUNCIÓN SIGMOIDAL

Es una función con límites superiores e inferiores con un incremento, en la que la mayoría de los valores de estimulación de entrada, ofrecerá a la salida de la función un valor que se encontrara en alguna de las zonas bajas o altas de la sigmoide.

Sigmoidal:

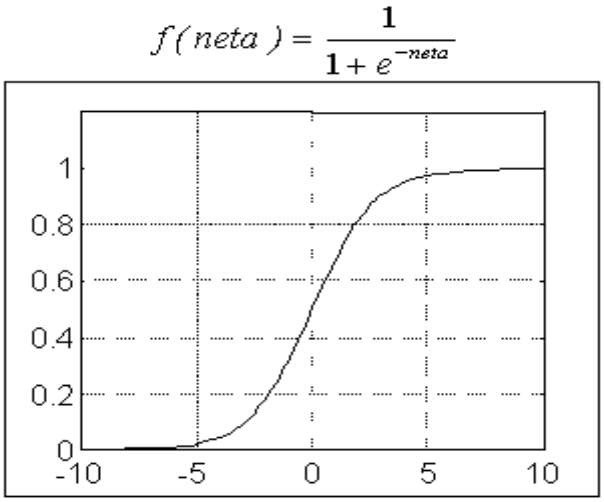


Figura 18: función de activación sigmoidal

$$f(neta) = \frac{2}{1+e^{-neta}} - 1$$

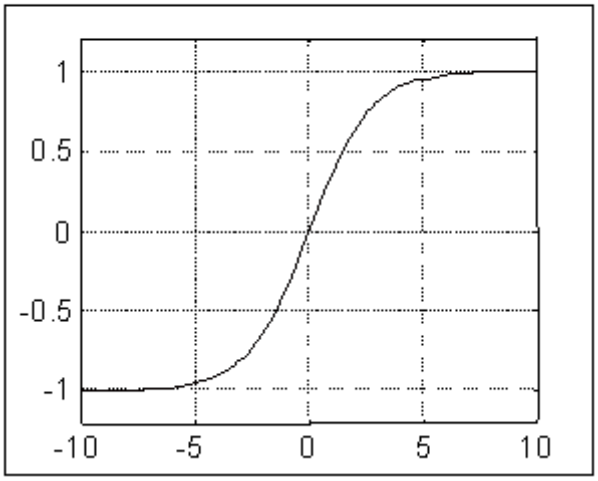


Figura 19: función de activación tangente sigmoidal

9.5.4.4 FUNCIÓN GAUSIANA

La nota caracterizadora de esta función de activación es que los centros y anchura pueden ser adaptados.

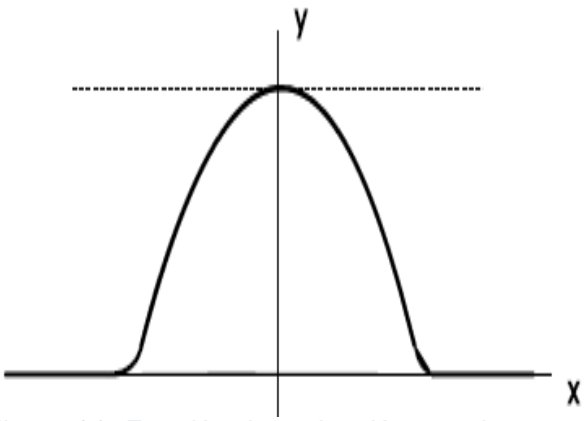


Figura 20: Función de activación gausiana

9.5.5 ARQUITECTURA RED NEURONAL ARTIFICIAL

La arquitectura en redes neuronales artificiales, que es la más comúnmente usada, es el algoritmo del *backpropagation*. Este ha sido el método utilizado para este trabajo.

En la red *backpropagation* cada neurona de una capa, a expensas de la capa de entrada, recibe entradas de todas las neuronas de la capa anterior y, a la vez, esta envía a su salida a todas las neuronas de la capa posterior (a expensas de la de salida).

El algoritmo *backpropagation* posee dos fases: una hacia adelante y otra hacia atrás. Durante la primera fase la entrada es enseñada a la red y enviada atreves de toda la red hasta la salida. Cuando se obtienen todos los valores de salida de la red empieza la segunda fase, y se comparan dichos valores con el vector de salida deseada. Se ajustan los pesos de la salida según el error calculado. Se pasa a la capa anterior con una retropropagación del error (*backpropagation*), ajustándose los pesos y siguiendo así hasta llegar a la primera capa. Así se han cambiado los pesos para cada ejemplo de aprendizaje del problema que, en principio, contábamos con el valor de sus entradas y el de las salidas con las que enseñamos a la red. [3]

9.5.6 ENTRENAMIENTO DE LA RED NEURONAL

Son muchos los tipos de entrenamientos con los que puedes entrenar a la red neuronal, nosotros nos hemos basado en tres tipos de entrenamiento que son:

9.5.6.1 Algoritmo Descenso de gradiente por lotes con momentumn

Este método de entrenamiento usa un entrenamiento por lotes. En el modo por lote se actualizan los pesos cuando termina el entrenamiento entero. Además este método de entrenamiento es rápido puesto que cuenta con el momentumn, esto lo que hace es ignorar los pequeños errores

9.5.6.2 Algoritmo de Levenberg-Marquardt

El algoritmo de Levenberg-Marquardt, también conocido como método de mínimos cuadrados amortiguado, ha sido diseñado para trabajar específicamente con funciones de error que se expresan como suma de errores cuadráticos. [22]

Este algotitmo hace uso del vector de gradiente y de la matriz Jacobiana , lo que le permite funcionar sin tener que calcular la matriz hessiana de manera totalmente exacta.,

Donde se puede expresar la matriz hessiana de la forma:

$$H=J^T*J \tag{8}$$

El vector gradiente se puede calcular de la siguiente forma:

$$G= J^T *e \tag{9}$$

Siendo J la matriz de jacobianos y el vector de errores de la red

A partir de las igualdades anteriores se define el proceso de mejora de parámetros con el algoritmo de Levenberg-Marquardt como:

$$X_{k+1}=X_k-[H+\mu*I]^{-1}*G \tag{10}$$

Cuando el escalar μ es cero, esta ecuación se comporta como el método de Newton. Cuando μ es alto, se comporta como el descenso de gradiente.

El objetivo es cambiar hacia el método de Newton lo más rápidamente posible, puesto que converge más rápidamente. Así, μ se empieza con un valor alto, pareciéndose al descenso de gradiente y si en alguna iteración se genera un fallo entonces μ asciende pero si no hay error μ desciende de manera que se parecerá al método de Newton.

9.5.6.3 Algoritmo de rezago

El objetivo del algoritmo de rezago es eliminar efectos de las magnitudes de las derivadas parciales. El signo de la derivada se usa para saber la dirección del ajuste de los pesos. La magnitud de la derivada no tiene efecto en la actualización del peso. El cambio de los pesos se calcula a través de un valor de actualización separado. El valor de actualización de cada peso y umbral se ve incrementado por un valor Δi siempre que la derivada de la función de error con respecto a ese peso tenga el mismo signo durante dos iteraciones sucesivas. El valor de actualización se ve mermado por un valor Δd cuando la derivada respecto a dicho peso cambia de signo respecto a la iteración previa. Si esta derivada tiene el valor cero, el valor de actualización no varía. Cuando los pesos comienzan a oscilar, su variación se reduce. Si los pesos se modifican en la misma dirección durante varias iteraciones, su variación se incrementa. [23]

El algoritmo de rezago suele tener la característica de ser más rápido que el descenso de gradiente estándar. Este algoritmo requiere de poca memoria de almacenamiento, porque sólo se necesitan almacenar los valores de actualización para cada peso y umbral.

9.5.7 PARÁMETROS PREDEFINIDOS PARA DETENER EL ENTRENAMIENTO DE LA RED NEURONAL ARTIFICIAL

Antes de empezar con un entrenamiento en la red, es de gran importancia predefinir con antelación, cuáles serán los motivos por los que la red deberá de detenerse, algunos de estos motivos suelen ser:

- Hemos llegado a un error que consideramos que ya es lo suficientemente pequeño para el correcto uso de la aplicación.
- Hemos llegado a una iteración máxima predefinida.
- Se ha llegado al descenso de gradiente deseado para esta aplicación.

9.5.8 NÚMERO DE CAPAS OCULTAS Y NEURONAS DE LA RED NEURONAL

Es difícilmente determinable el número de capas ocultas que necesitaremos con posterioridad para un correcto uso de la red, pero si bien es cierto que la mayoría a las redes les basta con tres capas:

- Una capa de entrada.
- Una capa de salida.
- Una capa oculta.

Añadir mas capas ocultas de las necesarias puede ralentizar el aprendizaje y no suponer que la red aprenda mejor. Sin embargo, esto depende de nuestra aplicación puesto que se han dado hechos en los que usando dos capas ocultas la aplicación aprende mejor que con una.

En cambio, determinar el número de neuronas necesarias para nuestra red neuronal, no suele ser una tarea fácil puesto que, en esta parte, interviene la eficacia del aprendizaje.

La manera más óptima es probar la combinación que más satisfaga nuestras necesidades con el menor número de neuronas y capas ocultas posible. Esto es porque usar más neuronas y capas ocultas de las necesarias tiene un coste computacional que incrementa el tiempo de respuesta de la red neuronal.

9.6 INTERPOLACIÓN DE LA DISTANCIA MEDIANTE REDES NEURONALES

En la realización del siguiente apartado se ha contado con las mediciones anteriormente tomadas en el laboratorio de microondas de la Universidad de Jaén en la Escuela Politécnica Superior de Linares.

Se ha creado una red neuronal artificial *backpropagation* de la que cabe destacar que se le han introducido dos entradas y se le precisa una salida. Estas entradas son la potencia recogida y el ángulo respecto de un punto de origen del receptor y se le precisa a la salida la distancia. Así se obtiene interpolación de la distancia para una determinada potencia en los grados que deseemos que nuestra red neuronal nos muestre.

Para la realización de este apartado se ha contado con dos métodos de entrenamiento para la red neuronal, primero se ha realizado con el entrenamiento de rezago y segundo con el de descenso de gradiente por lotes con momentumn

Como ejemplo, vamos a plantear la siguiente situación: una prueba sería precisar el diagrama de radiación en distancia a nuestra red neuronal, en la que contará con: una entrada de -66.28 dBm y el ángulo en el que queremos que nos muestre el diagrama. En nuestro caso, este ángulo oscila entre 0º hasta 90º, en saltos de 10º. Con esto obtendríamos lo siguiente:

- **Interpolación de la distancia con el método de entrenamiento de rezago**

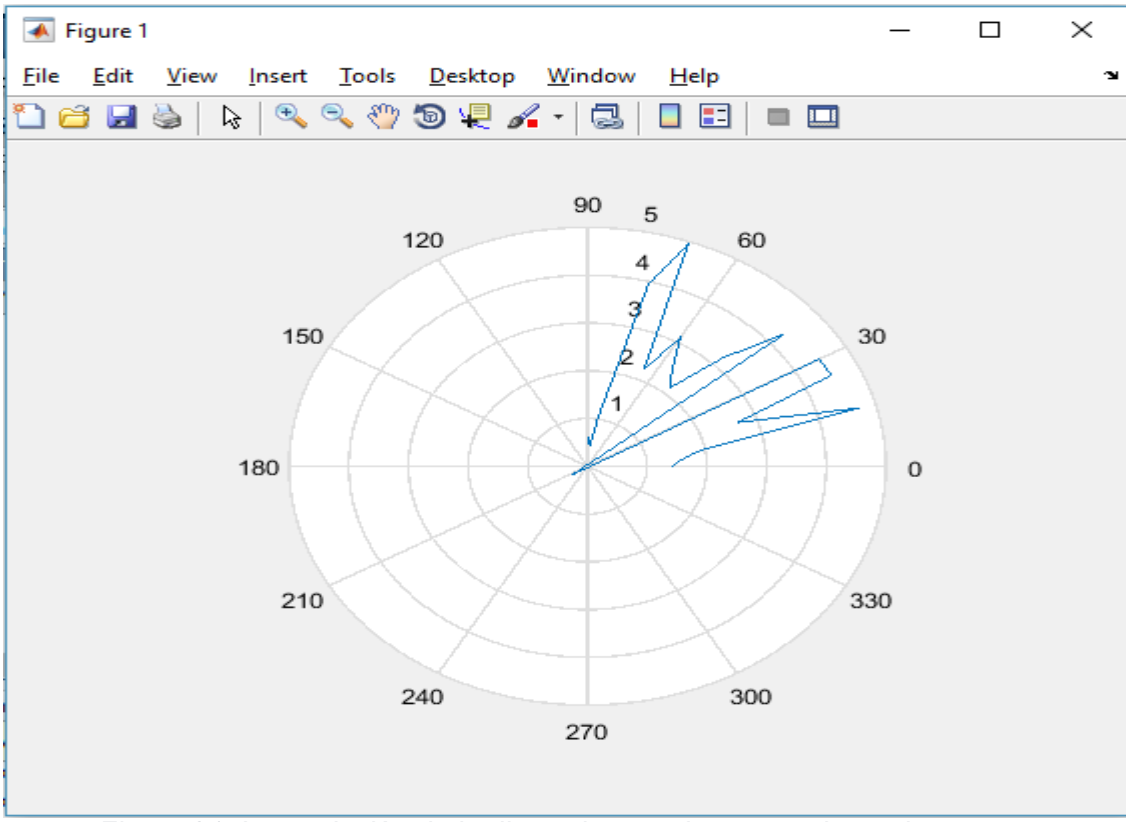


Figura 21: Interpolación de la distancia con el entrenamiento de rezago

En la realización de este entrenamiento salta a la vista que el error cometido por este tipo de entrenamiento es muy elevado, puesto que se realizo un apartado llamado diagrama de radiación para saber si lo que nos calculaba la red neuronal era comprensible o no. Además no nos serviría de mucho para el método de posicionamiento que se va a usar que es el de trilateración, puesto que necesita de dos puntos de corte y como se puede observar con tal diagrama se podrían conseguir varios puntos de corte.

- **Interpolación de la distancia con el método de entrenamiento de descenso de gradiente por lotes con momentumn**

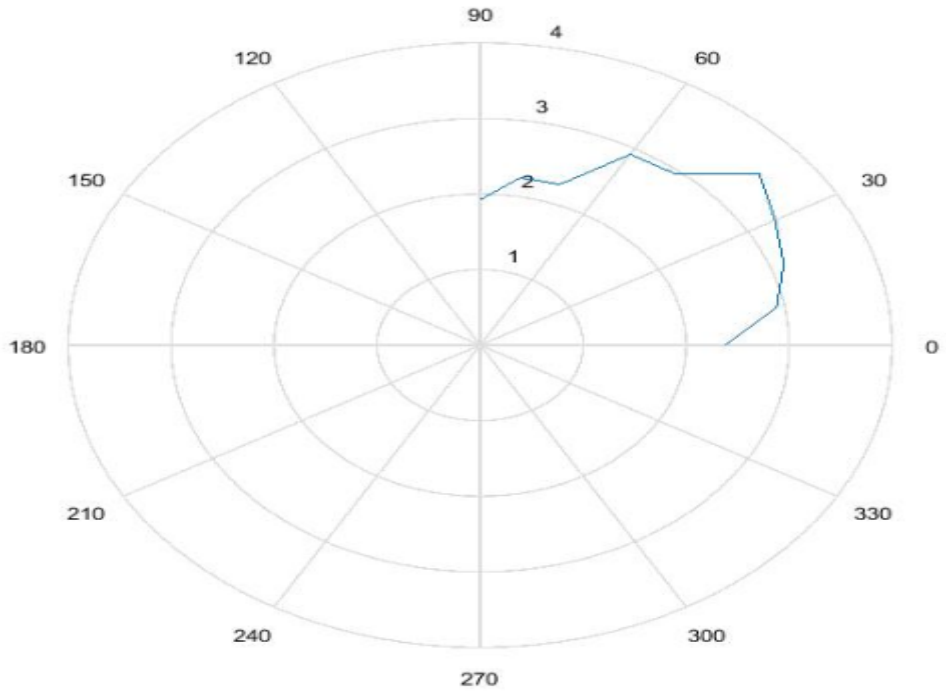


Figura 22: interpolación de la distancia con el entrenamiento de descenso de gradiente por lotes con momentumn

Como se puede observar la figura se parece más a lo que se calculó primeramente en el apartado de diagrama de radiación, lo cual nos informa de que posiblemente sea más acertado que el anterior. No obstante, se va a comprobar que el resultado con este método de entrenamiento es más acertado.

Para saber si es correcta la solución que nos ha entregado la red neuronal, nos hemos posicionado en 80º del receptor y con una potencia de -66.28 dBm. Mostraremos el resultado la curva calculada por la red neuronal para dicho caso, el punto calculado por nuestra red neuronal y el punto real para hacer una comparativa.

Error En Distancia Para La Prueba De -66.28 dB

Posición real	1.66m
Posición calculada	2.25m

Tabla 3: Error En Distancia Para La Prueba De -66.28 dBm

Por tanto, el error cometido por la aplicación sería de 0.59m

Para que pueda apreciarse visualmente la diferencia entre lo que muestra la red y el punto real de donde nos encontramos, se muestra en la siguiente imagen el “o”, que corresponde con el punto real donde nos encontramos, y la “x”, que corresponde con donde la red neuronal calcula que estamos.

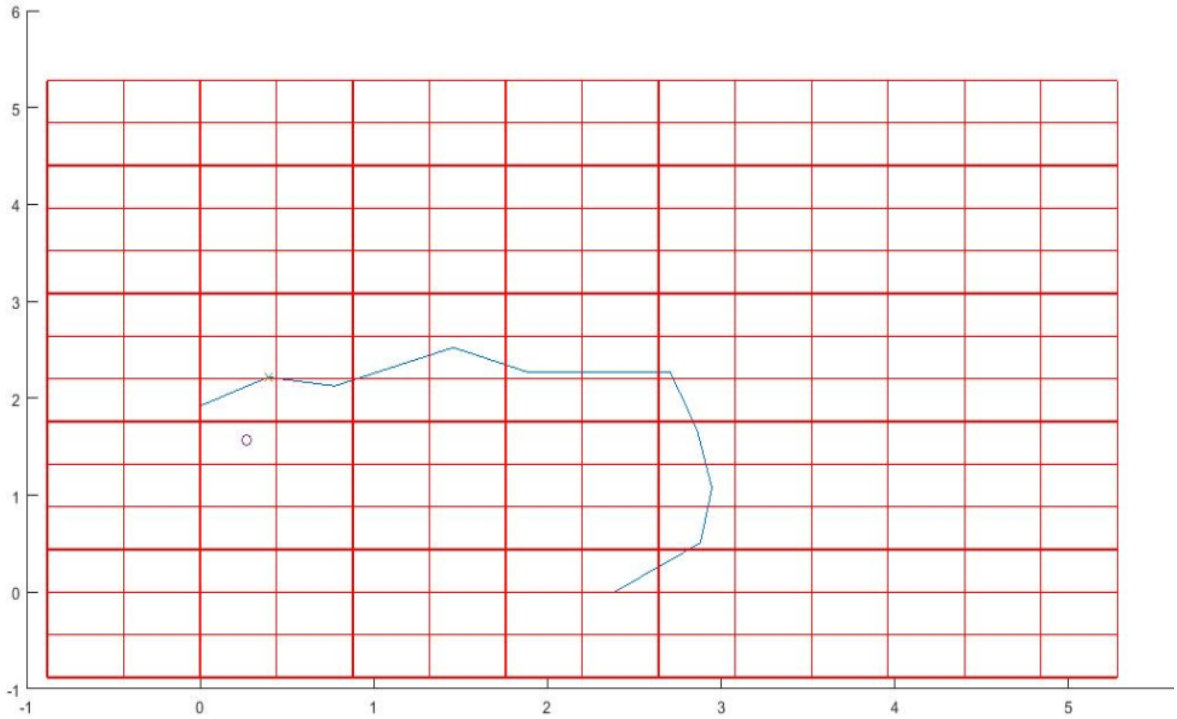


Figura 23: Comprobación que la red ofrece una interpolación correcta

9.6.1 CONCLUSIÓN DE LAS REDES NEURONALES PARA INTERPOLAR LA DISTANCIA

Queda demostrada la certeza de haber usado las redes neuronales para saber la distancia a razón de la potencia recibida siempre que la elección del entrenamiento, las neuronas y las capas ocultas sean las acertadas. Como se puede apreciar en la figura 23, el error es muy bajo y más sabiendo las dificultades que muestra una habitación cerrada, con elementos cambiantes y con personas que entran y salen que pueden afectar significativamente a la señal de recepción.

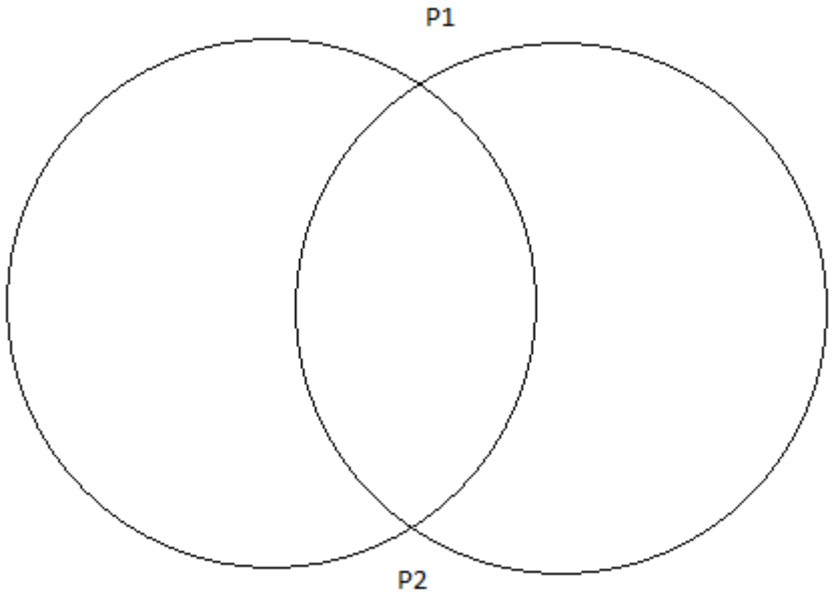
La diferencia es mínima respecto a una habitación y más mínima aún con respecto a un bloque donde existen varias habitaciones donde verdaderamente importante no es un acierto de milímetros, sino de habitación.

Una vez concluida la interpolación de distancia a razón de la potencia, nos queda localizar el punto exacto ya que existen diversos puntos de distancia, exactamente un punto por cada grado que introduces a la red. Para ello nos vamos a ayudar de una técnica llamada trilateración que es una técnica conocida y usada para el posicionamiento en exteriores.

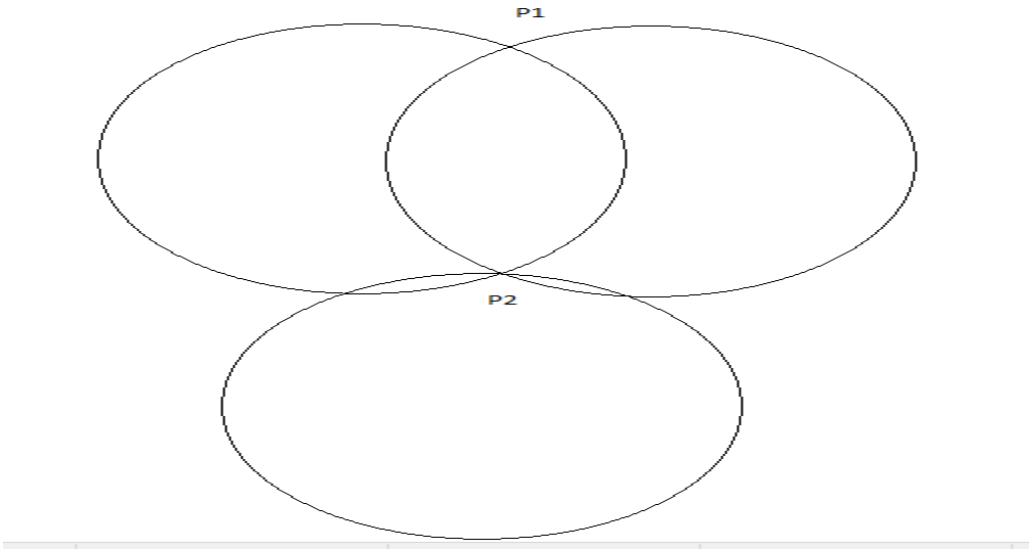
9.7 TRILATERACIÓN

El método de trilateración es un método de posicionamiento muy usado en telecomunicaciones por su sencillez y exactitud, explicamos su funcionamiento a continuación.

1. El primer paso que debemos de dar es la elección de los dos receptores más cercanos entre ellos
2. Mostramos en el mapa los diagramas de los dos receptores.
3. Nos quedamos con los dos puntos más próximos entre sí de las dos circunferencias. Tenemos dos puntos de corte que serían P1 y P2. Con lo cual aún no sabemos dónde estaremos porque podemos estar entre esos dos puntos.



4. Para solucionar la disputa anterior elegimos un nuevo tercer receptor que será el más cercano de los que tengamos en caso de tener más de tres y volvemos a hacer el anterior paso pero con las tres circunferencias.
5. Y, finalmente, el punto más cercano al tercer receptor sería el punto P2 y ya no quedaría duda en el punto que nos encontraríamos.



9.7.1 PRUEBA REAL TRILATERACION

El caso anteriormente explicado, es para un medio ideal en el que nada perturba la señal. En nuestro caso nos encontramos en el interior de una vivienda, lo cual la reflexión hará que no tengamos diagramas de radiación tan ideales.

Una vez colocados los 3 receptores, iniciamos una prueba y pedimos que la red neuronal nos obtenga los diagramas de radiación para cada nodo.

Tras varias pruebas en el entorno de trabajo, modificando la posición del *ibeacon* podemos obtener los siguientes resultados:

9.7.1.1 PRUEBA 1

Para un aprecio visual del error puede ver la siguiente imagen donde se muestra el punto real con una “o” y el punto calculado con una “x”

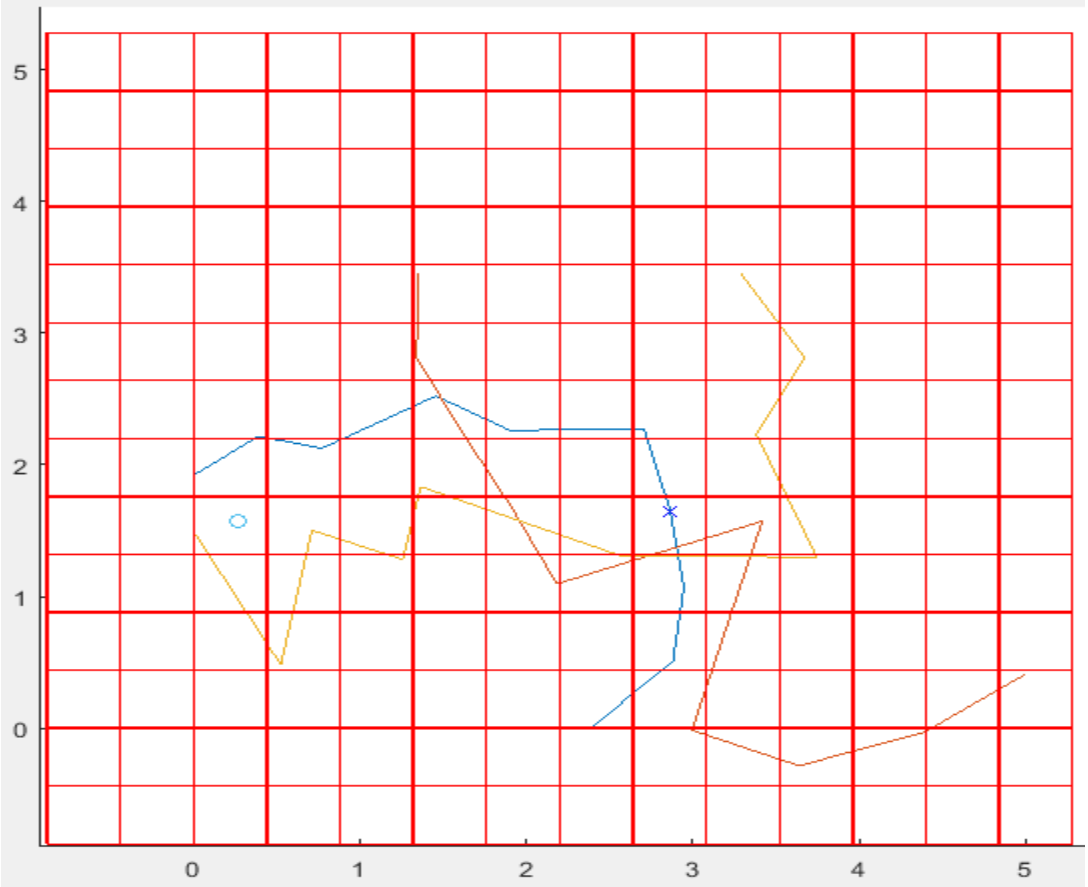


Figura 24: prueba 1 de trilateración

De aquí en adelante, la comparación la vamos a hacer en coordenadas cartesianas para poder hacer una mejor comparación con el siguiente método de posicionamiento.

Error En Distancia Prueba 1 Trilateración:

	Posición x	Posición y
Posición real	0.27m	1.57m
Posición calculada	2.864m	1.653m
Error cometido	2.59m	0.083m

Tabla 4: Error En Distancia Prueba 1 Trilateración

9.7.1.2 PRUEBA 2

Para un aprecio visual del error puede ver la siguiente imagen donde se muestra el punto real con una “o” y el punto calculado con una “x”

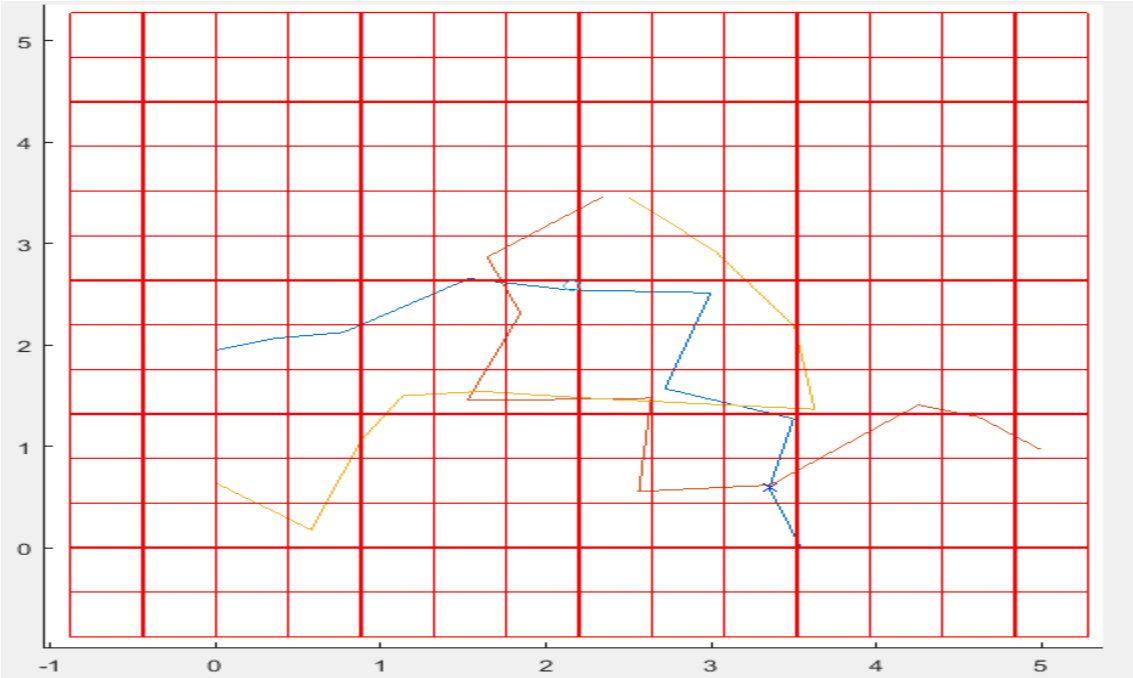


Figura 25: prueba 2 de trilateración

Error en distancia prueba 2 trilateracion:

	Posición x	Posición y
Posición real	2.16m	2.58m
Posición calculada	3.351m	0.591m
Error cometido	1.191m	1.989m

Tabla 5: Error En Distancia Prueba 2 Trilateración

9.7.2 CONCLUSIONES DE LA TRILATERACION

Aunque el método es bueno y efectivo en algunos ámbitos, para nuestro caso en particular deja que desear, puesto que para la trilateración nos valemos de:

- Los puntos cercanos a los de corte entre los diagramas de los dos primeros nodos
- Distancia que hay entre el punto de corte y el tercer nodo

Esto hace que con pequeños cambios en la salida de una red neuronal dichos puntos puedan cambiar sustancialmente. Este pequeño error que pueda tener la red

neuronal se ve afectado por tres veces, puesto que para la realización de este método precisamos de 3 redes neuronales, una por cada nodo que tengamos.

Además, otro posible caso que nos puede afectar es el entorno, haciendo una propagación un tanto difusa, provocando que en vez de dos puntos de corte haya como tres o cuatro puntos y nuestro programa no sabría dónde ubicarlo.

Tras una gran variedad de pruebas que se han realizado sobre el método de trilateración, se ha realizado un análisis de todas ellas para hacer una valoración del error cuadrático medio, para saber cuánto de fiable es este método para nuestra aplicación

Con dicho medio de posicionamiento obtenemos el siguiente error cuadrático medio:

	Posición x	Posición y
Error cometido	1.191m	1.989m

Tabla 6: Error Medio En Distancia Trilateración

9.8 MÉTODO DE POSICIONAMIENTO CARTESIANO

Como se ha podido comprobar en el anterior capítulo, el método de trilateración no es adecuado para nuestra aplicación, lo cual, nos vemos en la necesidad de encontrar con la solución adecuada para la realización de dicho trabajo fin de grado.

El método con el que vamos a intentar solventar dicha ambigüedad de posicionamiento va a ser mediante la realización de un plano cartesiano. Donde vamos a realizar medidas cada variación de 0.44 metros tanto en la coordenada “x” como en la coordenada “y”.

Para la realización de este apartado nos hemos servido de la ayuda de algunos de los aparatos anteriormente citados en el apartado de requisitos hardware. Estos son:

- Telemetro láser para saber a la distancia a la que se toma la medición

- Ibeacons para emitir una señal bluetooth
- Raspberry pi para la captación de la señal

9.8.1 ANÁLISIS MUESTRAS FILTRADAS

Una vez que hemos recogido todas las muestras una por una, procesamos dichas medidas por nuestro filtro de Kalman, donde podremos obtener una tabla de mediciones de potencia real para cada punto, que, por motivo de ahorro de espacio, no se mostrarán todos los datos ni las muestras obtenidas antes de filtrar, puesto que son muy extensas y existen varios receptores.

Nivel De Potencia En dBm Del Nodo 1 Método Trilateración:

X[m] / y[m]	0	0.44	0.88	1.32	1.76	2.2	2.64	3.08
0	-35.12	-63.33	-66.35	-61.96	-73.63	-70.65	-67.63	-74.19
0.44	-52.32	-52.95	-65.65	-64.15	-67.37	-73.89	-65.59	-63.71
0.88	-56.07	-56.85	-63.76	-64.02	-68.95	-70.04	-65.78	-64.4
1.32	-64.46	-69.2	-62.8	-65.67	-71.5	62.6	-68.23	-77.05
1.76	-68.6	-62.76	-69.81	-73.4	-66.54	-65.41	-71.46	-70.54
2.2	-63.99	-72.5	-71.25	-67.68	-72.08	-75.31	-72.6	-69.23
2.64	-66.69	-72.87	-72.63	-66.14	-71.44	-66.39	-69.03	-76.17
3.08	-73.46	-72.81	-66.88	-70.67	-68.42	-71.6	-69.88	-76.33

Tabla 7: Nivel De Potencia En dBm Del Nodo 1 Método Trilateración

Como vimos en las mediciones anteriores, para el plano polar, lo normal sería que la potencia decayera con la distancia, no sucede así en este caso, al menos en algunas de las mediciones. Esto es debido, como dijimos anteriormente, al medio donde tomamos las mediciones, es una sala con muchos elementos que afectan a la propagación de la onda, pudiendo surgir algún efecto que afecte a la propagación, como por ejemplo:

- Reflexión

- Refracción
- Difracción
- Multitrayecto

Estas medidas nos van a servir para alimentar una red neuronal artificial, donde recibiendo por la entrada la potencia captada por los raspberry pi, se le va a precisar 2 datos de salida, que serán las coordenadas cartesianas "X" e "Y" del punto donde nos encontremos.

9.8.2 PRUEBAS MÉTODO CARTESIANO

Una vez tenemos todas nuestras muestras procesadas, con ayuda del software matlab, que posee una maravillosa tolbox para redes neuronales, nos creamos una red neuronal backpropagation como en apartados anteriores pudimos explicar más detalladamente.

En el curso de este apartado, hemos realizado un análisis detallado de este método de posicionamiento, analizando la mejoría en el método según el entrenamiento usado como en el número de receptores empleados.

Para comprobar el funcionamiento del nuevo método de posicionamiento, hemos realizado unas pruebas reales, donde nos hemos posicionado en varias posiciones al azar para comprobar el resultado real con el calculado por la red neuronal.

9.8.2.1 SEGÚN EL ENTRENAMIENTO EMPLEADO

En primer lugar, vamos a analizar el tipo de entrenamiento y el número de receptores que se adapta más a la naturaleza de este trabajo fin de grado.

Hemos usado 3 tipos de entrenamiento para el aprendizaje de esta red que han sido:

- Algoritmo descenso de gradiente por lotes con momentumn
- Algoritmo rápido de rezago
- Algoritmo Levenberg-Marquardt

Para cada caso podemos obtener un error medio de:

Error Medio En Distancia Con Posicionamiento Cartesiano Para 2 Receptores:

	Error medio cometido en la coordenada x	Error medio cometido en la coordenada y
Descenso de gradiente por lotes con momentumn	0.79m	0.69m
Rezago	1.5m	0.51m
Levenberg-Marquardt	1.40m	0.66m

Tabla 8: Error Medio En Distancia Con Posicionamiento Cartesiano Para 2 Receptores

Error Medio En Distancia Con Posicionamiento Cartesiano Para 3 Receptores:

	Porcentaje de acierto en la coordenada x	Porcentaje de acierto en la coordenada y
Descenso de gradiente por lotes con momentumn	0.50m	0.54m
Rezago	0.68m	0.86m
Levenberg-Marquardt	0.77m	0.60m

Tabla 9: Error Medio En Distancia Con Posicionamiento Cartesiano Para 3 Receptores

En la realización de las siguientes pruebas se llega a la conclusión que el método más acertado es el algoritmo de descenso de gradiente por lotes con momentumn puesto que tanto para el uso de 2 receptores como el de 3 receptores es el que mejor comportamiento presenta.

9.8.2.2 SEGÚN EL NUMERO DE RECEPTORES EMPLEADOS

9.8.2.2.1 PRUEBAS METODO CARTESIANO 2 RECEPTORES

Para un aprecio visual del error, en las consecutivas imágenes se mostrara, el punto real con una “o” y el punto calculado con una “x”.

9.8.2.2.1.1 Prueba 1

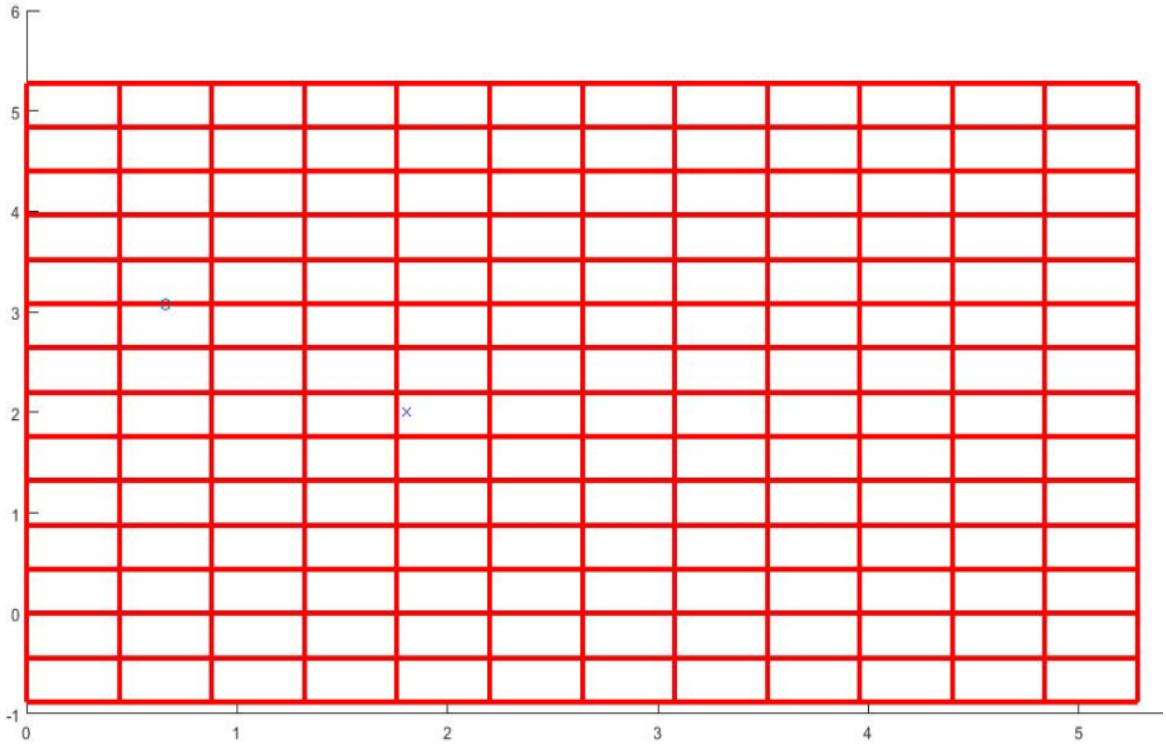


Figura 26: prueba 1 posicionamiento cartesiano con 2 receptores

Error cometido por la prueba 1:

	Posición x	Posición y
Punto real	0.66m	3.08m
Punto calculado	1.8061m	1.9985m
Error cometido	1.1461m	1.0815m

Tabla 10: Error En Distancia Con Posicionamiento Cartesiano Prueba 1 Para 2 Receptores

9.8.2.2.1.2 Prueba 2

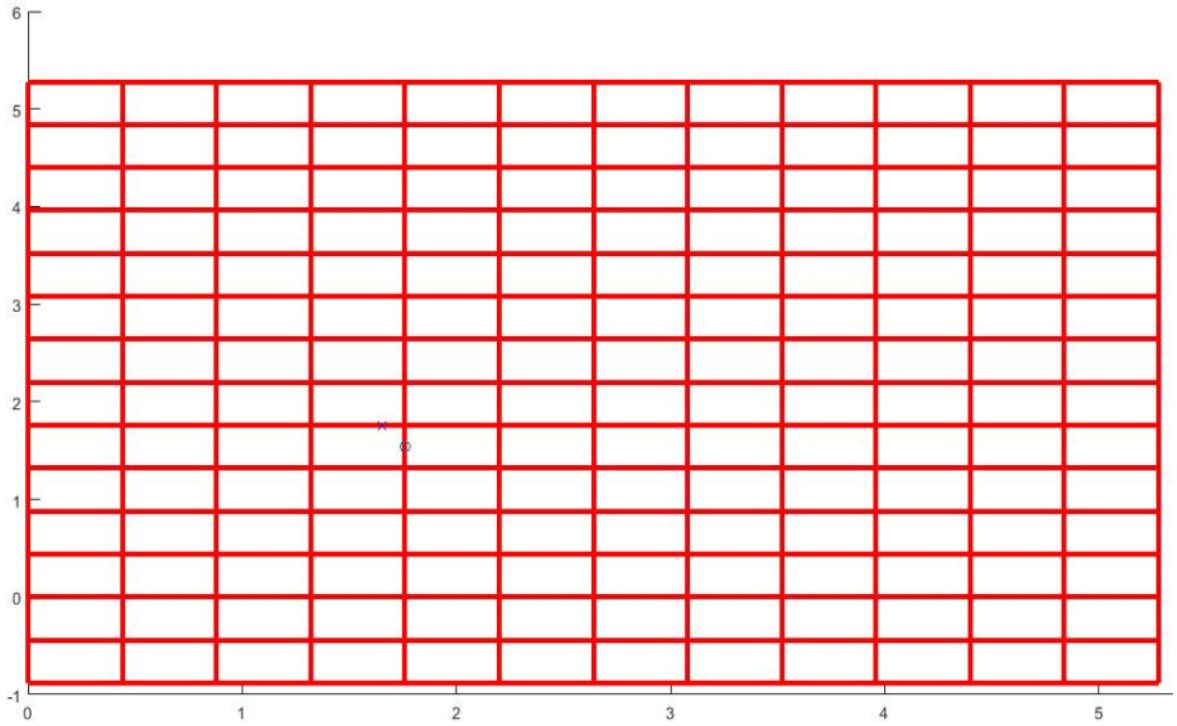


Figura 27: prueba 2 posicionamiento cartesiano con 2 receptores

Error cometido por la prueba 2:

	Posición x	Posición y
Punto real	1.76m	1.54m
Punto calculado	1.6526m	1.7511m
Error cometido	0.1074m	0.211m

Tabla 11: Error En Distancia Con Posicionamiento Cartesiano Prueba 2 Para 2 Receptores

9.8.2.2.1.3 Prueba 3

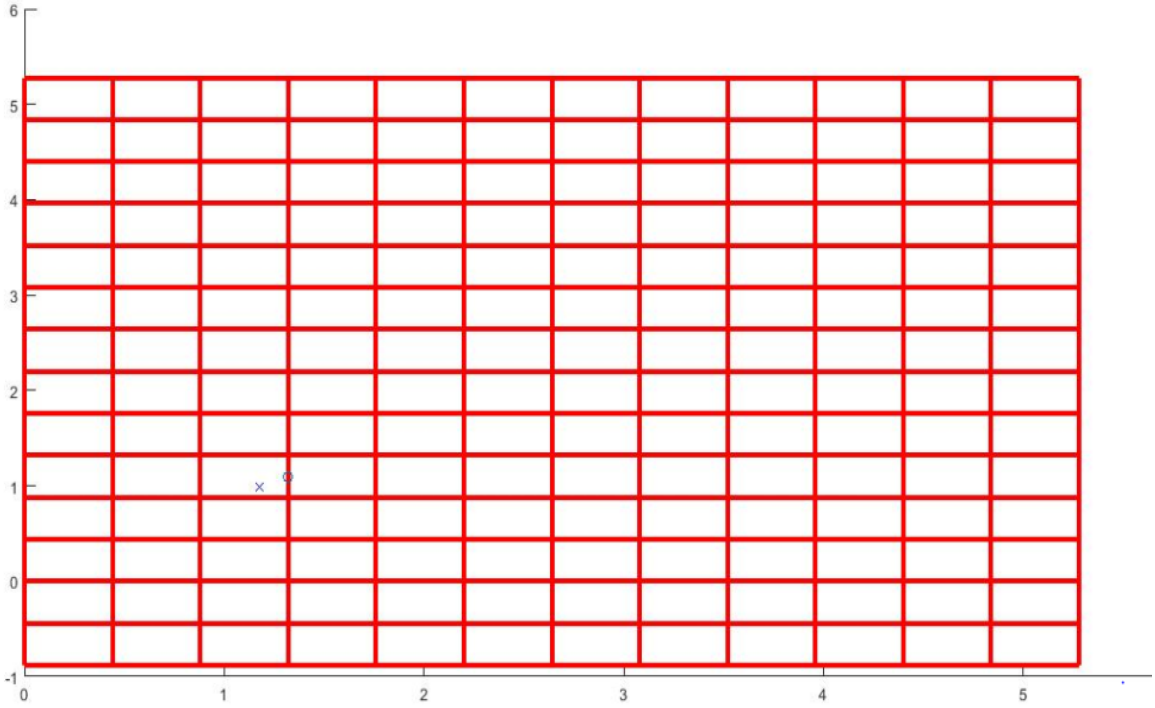


Figura 28: prueba 3 posicionamiento cartesiano con 2 receptores

Error cometido en la prueba 3

	Posición x	Posición y
Punto real	1.32m	1.1m
Punto calculado	1.1763m	0.9831m
Error cometido	0.1437m	0.1169m

Tabla 12: Error En Distancia Con Posicionamiento Cartesiano Prueba 3 Para 2 Receptores

9.8.2.2.1.4 Prueba 4

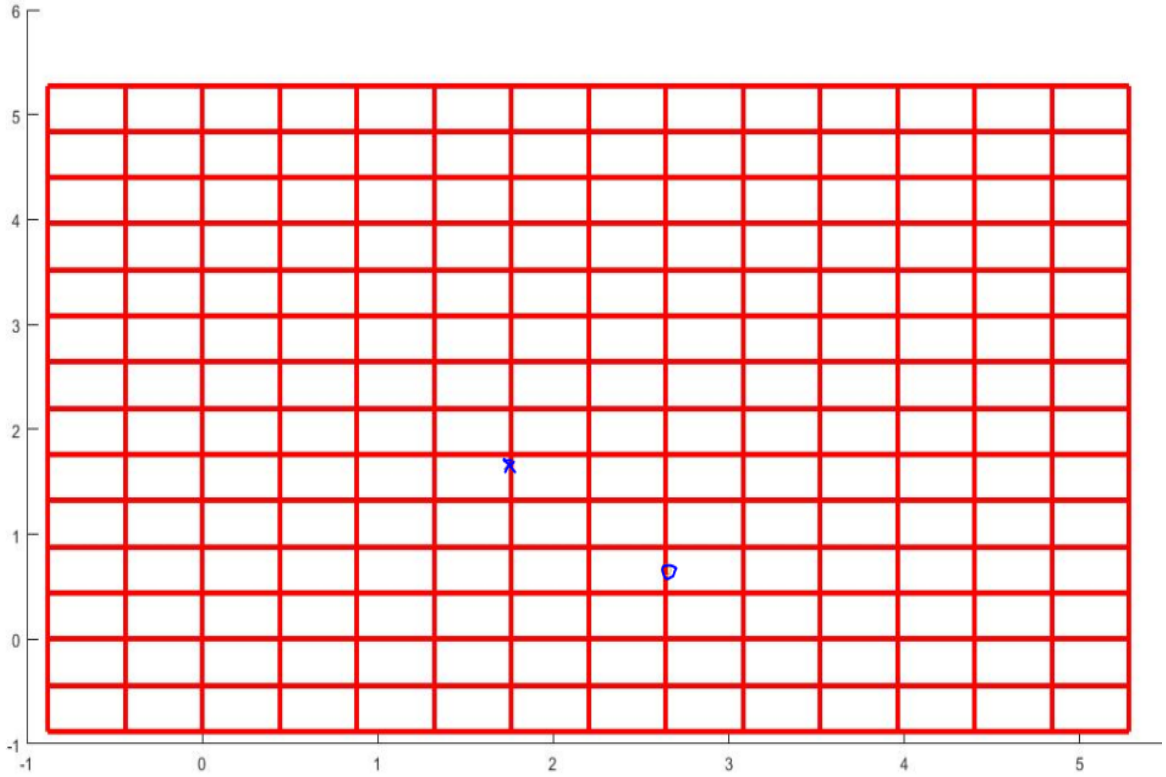


Figura 29: prueba 4 posicionamiento cartesiano con 2 receptores

Error cometido en la prueba 4:

	Posición x	Posición y
Punto real	2.64m	0.66m
Punto calculado	1.7401m	1.6667m
Error cometido	0.8999	1.0067

Tabla 13: Error En Distancia Con Posicionamiento Cartesiano Prueba 4 Para 2 Receptores

9.8.2.2.1.5 Prueba 5

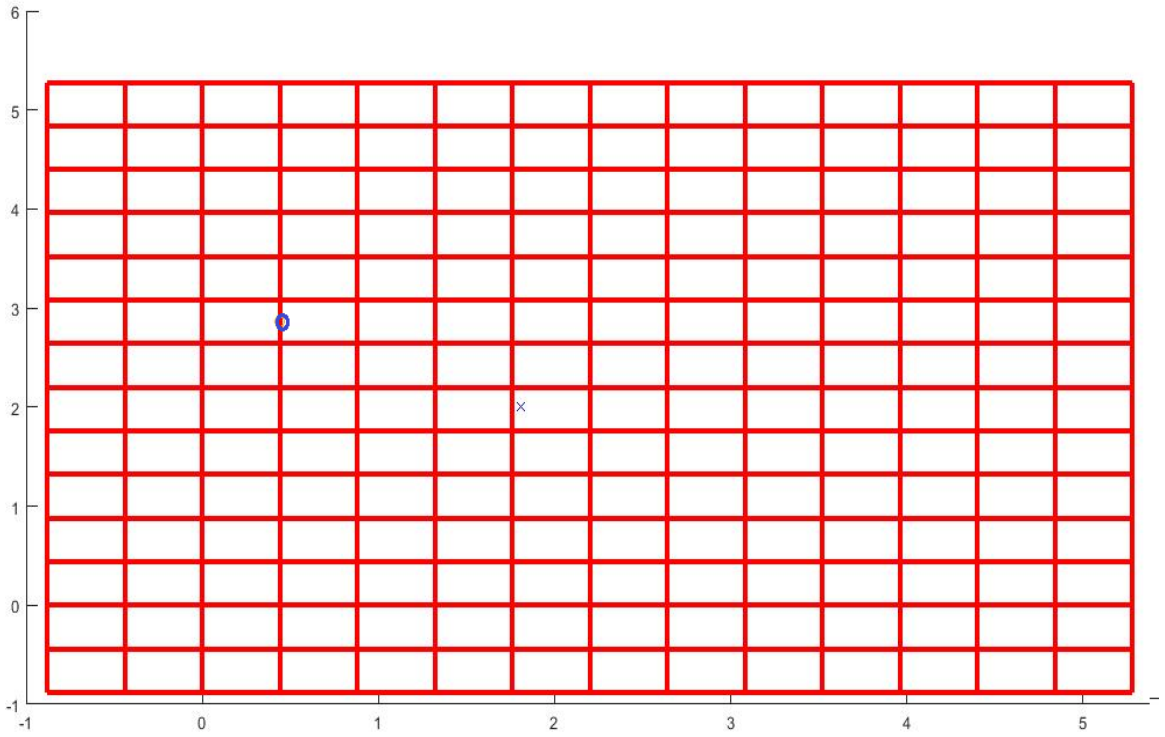


Figura 30: prueba 5 posicionamiento cartesiano con 2 receptores

Error cometido en la prueba 5:

	Posición x	Posición y
Punto real	0.44m	2.86m
Punto calculado	1.8061m	1.9986m
Error cometido	1.406m	0.8614m

Tabla 14: Error En Distancia Con Posicionamiento Cartesiano Prueba 5 Para 2 Receptores

9.8.2.2.2 PRUEBAS METODO CARTESIANO 3 RECEPTORES

9.8.2.2.2.1 Prueba 1

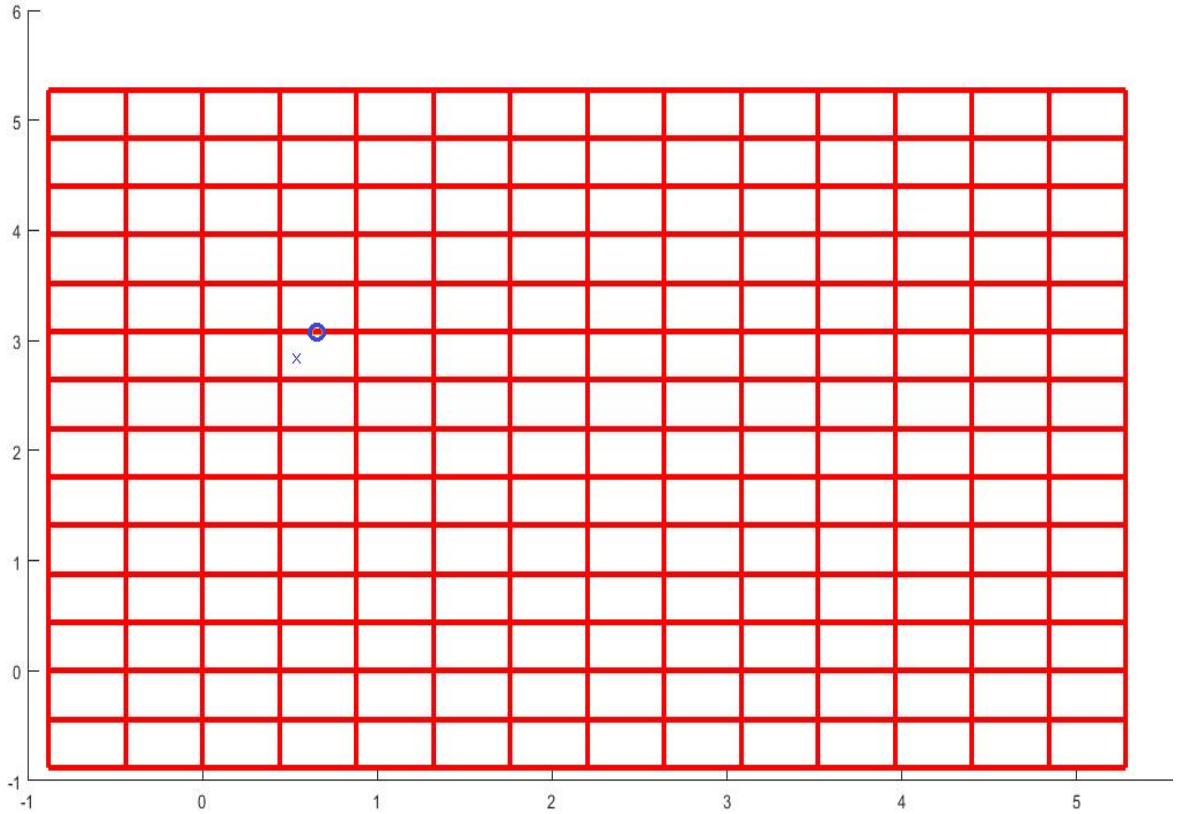


Figura 31: prueba 1 posicionamiento cartesiano con 3 receptores

Error cometido en la prueba 1:

	Posición x	Posición y
Punto real	0.66m	3.08m
Punto calculado	0.5462m	2.8433m
Error cometido	0.1138m	0.2367m

Tabla 15: Error en distancia con posicionamiento cartesiano prueba 1 para 3 receptores

9.8.2.2.2.2 Prueba 2

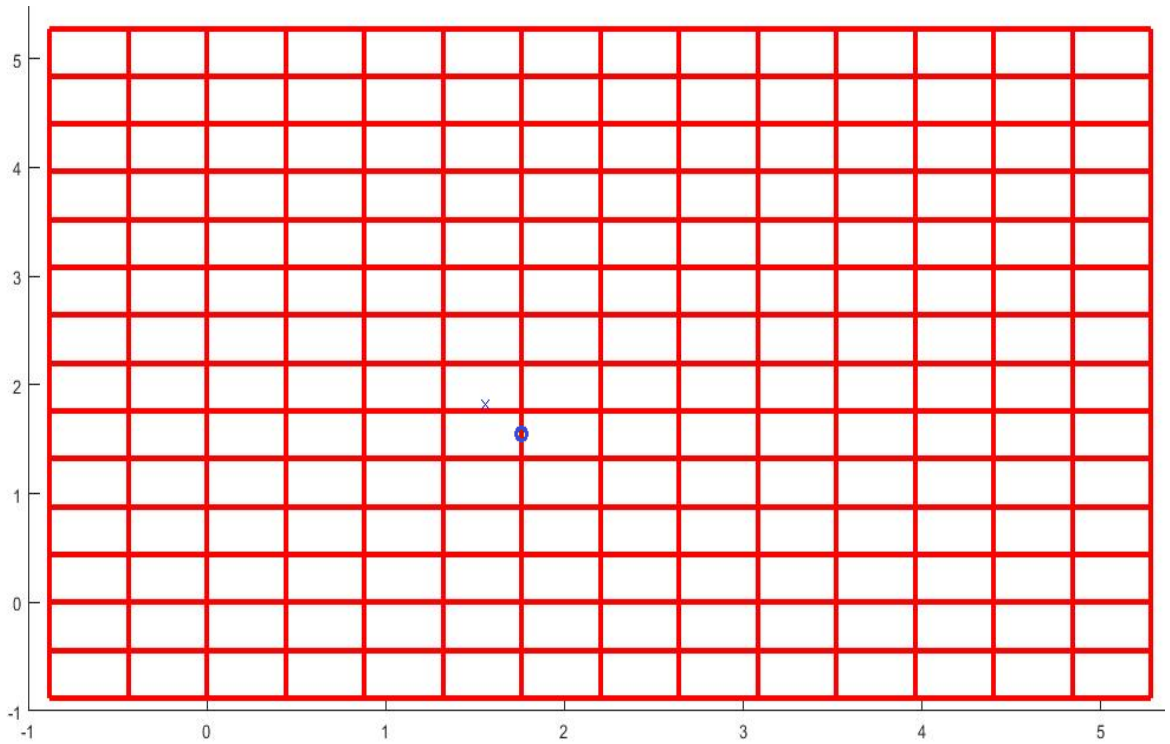


Figura 32: prueba 2 posicionamiento cartesiano con 3 receptores

Error cometido en la prueba 2:

	Posición x	Posición y
Punto real	1.76m	1.54m
Punto calculado	1.5598m	1.8177m
Error cometido	0.2002m	0.2777m

Tabla 16: Error En Distancia Con Posicionamiento Cartesiano Prueba 2 Para 3 Receptores

9.8.2.2.2.3 Prueba 3

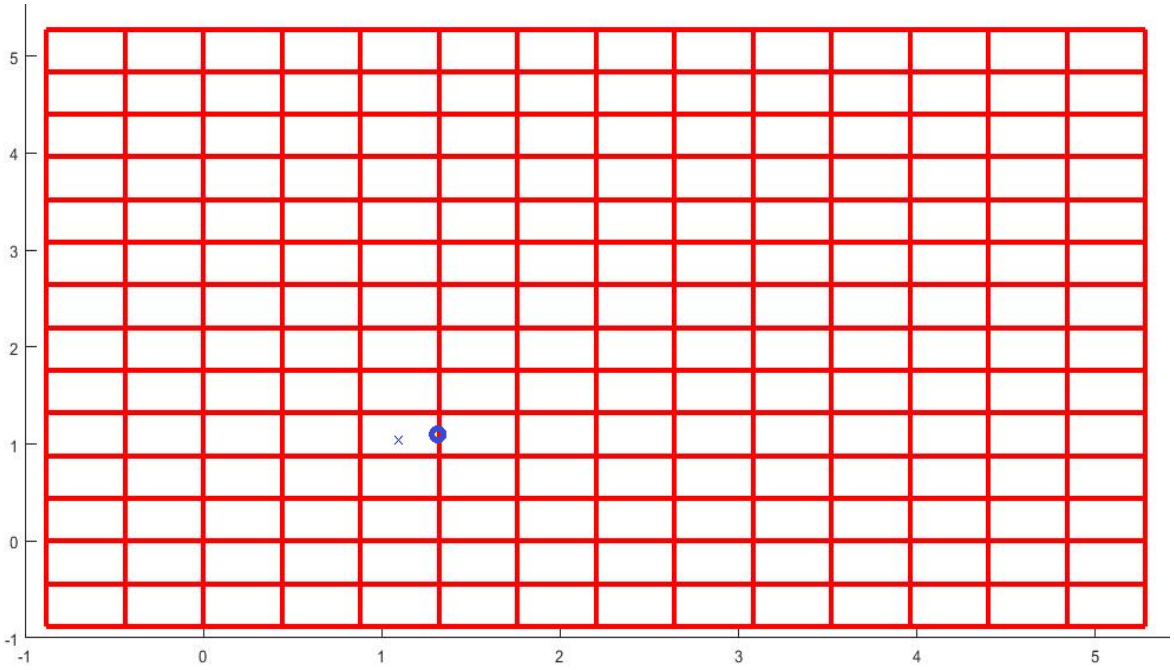


Figura 33: prueba 3 posicionamiento cartesiano con 3 receptores

Error cometido en la prueba 3:

	Posición x	Posición y
Punto real	1.32m	1.1m
Punto calculado	1.0972m	1.0364m
Error cometido	0.2228m	0.0636m

Tabla 17: Error En Distancia Con Posicionamiento Cartesiano Prueba 3 Para 3 Receptores

9.8.2.2.2.4 Prueba 4

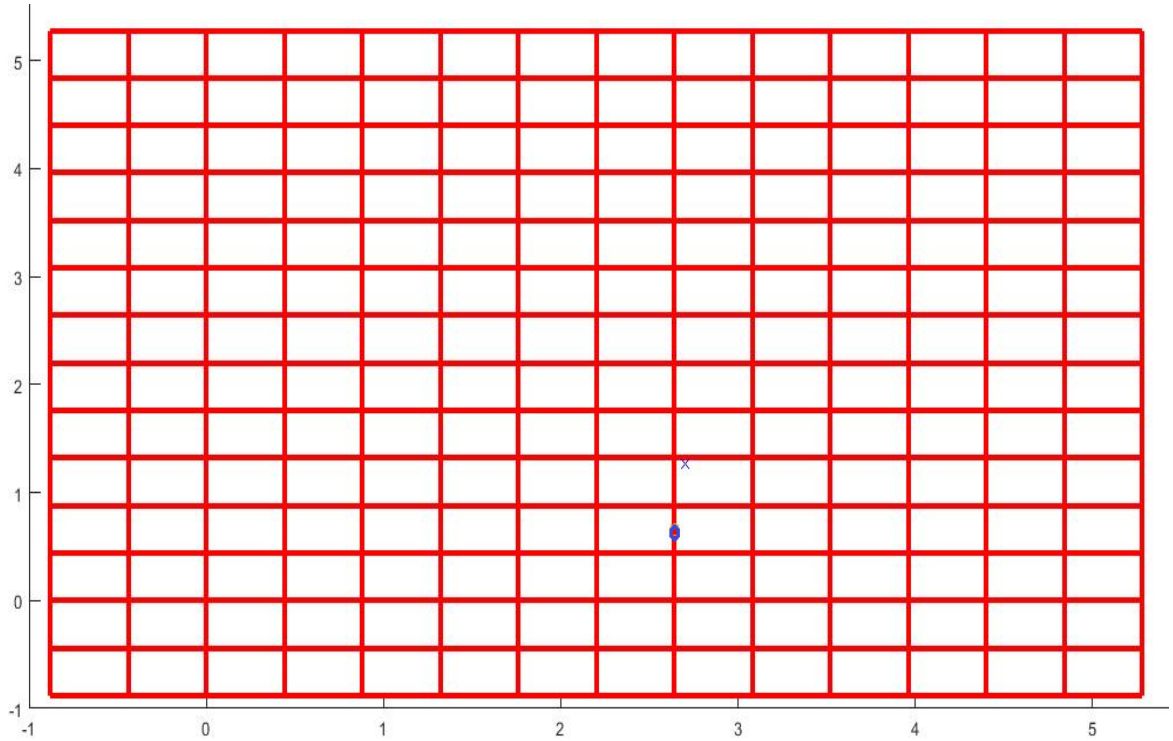


Figura 34: prueba 4 posicionamiento cartesiano con 3 receptores

Error cometido en la prueba 4:

	Posición x	Posición y
Punto real	2.64m	0.66m
Punto calculado	2.6987m	1.2635m
Error cometido	0.0587m	0.6035m

Tabla 18: Error En Distancia Con Posicionamiento Cartesiano Prueba 4 Para 3 Receptores

9.8.2.2.2.5 Prueba 5

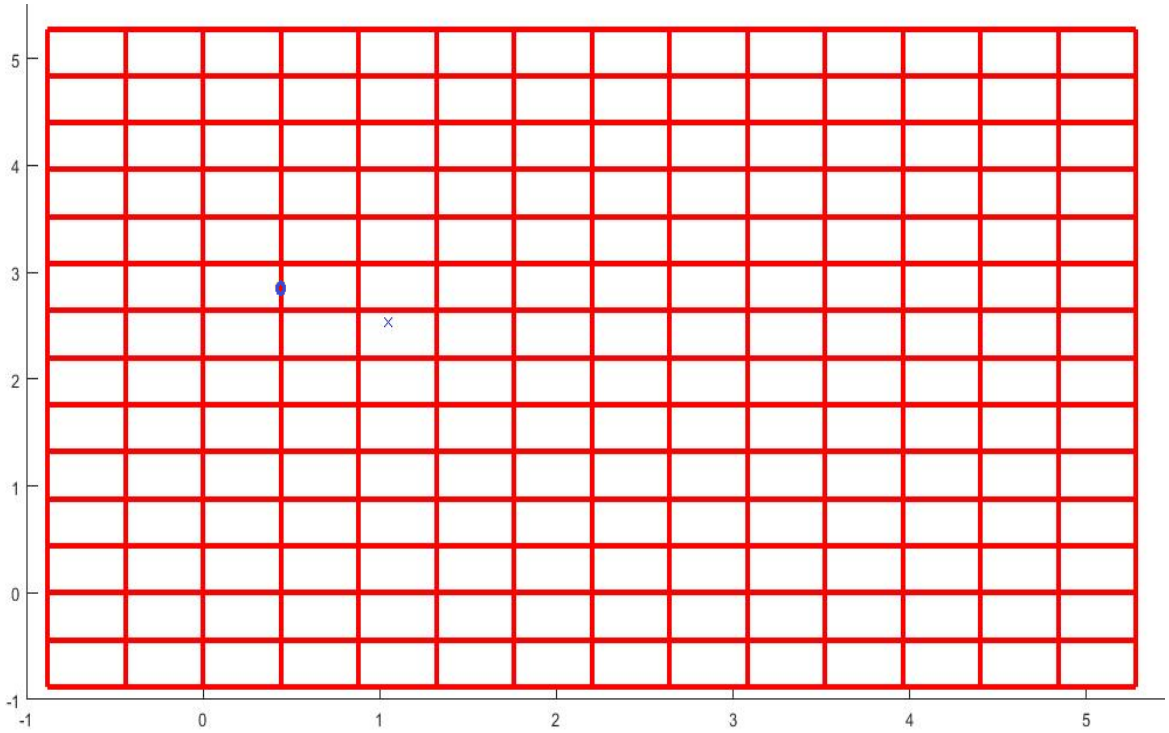


Figura 35: prueba 5 posicionamiento cartesiano con 3 receptores

Error cometido en la prueba 5:

	Posición x	Posición y
Punto real	0.44m	2.86m
Punto calculado	1.0498m	2.5351m
Error cometido	0.6498m	0.3249m

Tabla 19: Error En Distancia Con Posicionamiento Cartesiano Prueba 5 Para 3 Receptores

9.8.3 CONCLUSIÓN USO DEL MÉTODO DE POSICIONAMIENTO CARTESIANO

Se ha estudiado un método de posicionamiento en coordenadas cartesianas con ayuda de redes neuronales artificiales. En este se ha estudiado como afecta al posicionamiento según el método de entrenamiento usado y el número de receptores usados.

Tras largas pruebas en el entorno de trabajo, se ha llegado a la conclusión de que el método de entrenamiento más efectivo es el algoritmo de gradiente descendiente por lotes con momentumn, puesto que ofrece un error de cálculo de posición mucho mejor que los demás métodos. Además se ha estudiado como afecta el número de receptores empleados para el método, llegando a la conclusión que aumentando el número de receptores la calidad de cálculo de la posición se ve mejorado gratamente.

10 CONCLUSIONES

Este proyecto ha cumplido con los requisitos de encontrar un método de posicionamiento para interiores. La realización de este proyecto cubre todas bases establecidas con el método de posicionamiento cartesiano.

Lo primero que se quiere destacar es que se dio con una solución francamente buena para la eliminación de impurezas de la señal, siendo esta técnica finalmente el filtrado de kalman, que tiene una gran diferencia con los métodos anteriormente usados.

Si analizamos las medidas tomadas, francamente depende en gran medida del medio y de la posición del ibeacons. Se han recogido medidas a diferentes horas del día donde el nivel en el mismo punto donde se puede observar una diferencia en la medida. Por otro lado, cabe destacar el posicionamiento de la baliza bluetooth, puesto que según la inclinación que le des a la baliza, la diferencia puede afectar significativamente en la medida.

Lo que verdaderamente es novedoso en la realización de este proyecto es el uso de redes neuronales para determinar la distancia. Este apartado sorprende gratamente, puesto que la red neuronal es capaz de adivinar la distancia a la que nos encontramos con un error bastante mínimo, teniendo en cuenta que estamos en un espacio cerrado y los problemas que pueden afectar a la propagación son diversos.

Si comparamos entre los dos métodos de posicionamientos usados en la realización de este proyecto, el método de posicionamiento cartesiano ofrece una clara ventaja frente al tradicional método por trilateración. Esta ventaja ya no solo es porque el método cartesiano ofrezca un mejor resultado, sino también por los siguientes aspectos:

- Necesidad de saber exactamente la localización de los receptores, mientras que en el método cartesiano no, ya que te da solución respecto un punto de origen.
- El método de trilateración requiere, como mínimo, de 3 receptores, con lo que en el método cartesiano con 2 receptores ya ofrecía mejores resultados.
- En nuestro caso hemos usado un plano en dos dimensiones, pero si hubiésemos usado un plano en 3 dimensiones, seria infinitamente más sencillo llevarlo a cabo con el método cartesiano. Tanto por la hora de realizar las mediciones como por la complejidad de la solución.
- Mientras que el método cartesiano usa una sola red neuronal, para el caso en el de la trilateración precisamos de 3 redes neuronales, esto hará que el error sea tres veces mayor.

11 ANEXOS

11.1 PROGRAMAS REALIZADOS

11.1.1 Filtro butterwort

```
%Borrado de memoria
clearvars ?global
%Lectura de datos de la primera columna de la hoja de cálculo que es la
%potencia recibida
x = xlsread('muestras.xlsx','Limpio','A1:A12454');
%Tasponemos el vector para convertirlo en una matriz fila
x = x.';
%Leemos los valores temporales que se encuentran en la segunda columna
tiempos = xlsread('muestras.xlsx','Limpio','B1:B12454');
%Obtenemos la media de tiempos que será el periodo de muestreo
Ts = median(tiempos)/1000;
%La longitud del vector de señal
L = length(x);
%La media de la señal
Media = median(x);
%Restamos la media para que sólo quede el ruido
%x = x-Media;
%Generamos el vector de tiempos
t = (0:(L-1))*Ts;
%Declaración de la frecuencia de muestreo
fs=1/Ts;
%Transformada rápida de Fourier de la señal
X=fft(x,L);
%Módulo de la transformada
X_abs = abs(X/L);
%Cálculo de la densidad espectral de potencia Gx.
Gx=X_abs.^2;
%Definición del vector de frecuencias
f=-fs/2:fs/(length(X)-1):fs/2;

%Cáculo de un filtro de Butterwood
%Límite de la frecuencia de paso
fp=0.05;
%Límite de la frecuencia de corte
fc=0.1;
%Rizado en la banda de paso en dB
Rp=3;
%Rizado en la bada de rechazo en dB
Rs=40;
%Límite de la frecuencia de paso normalizada a la frecuencia de Nyquist
%fs/2
Wp=fp/(fs/2);
%Límite inferior de la banda de rechazo normalizada
Ws=fc/(fs/2);
[n, Wn] = buttord(Wp, Ws, Rp,Rs);
[B,A]=butter(n,Wn);
[h,w] = freqz(B,A,length(x),'whole',fs);
%filtrado del ruido de entrada de entrada como prueba
y=filter(B,A,x);
subplot(321);plot(t,x);title('Señal
temporal');xlabel('Tiempo');ylabel('x(t)');
```

```
%Módulo de la transformada de Fourier
subplot(322);plot(f,fftshift(X_abs));title('Módulo
TF');xlabel('Frecuencia (Hz)');ylabel('|X(f)|');
%Densidad espectral
subplot(323);plot(f,fftshift(Gx));title('Densidad espectral de
potencia');xlabel('Frecuencia (Hz)');ylabel('Gx(f)');
%Respuesta frecuencial del filtro
subplot(324);plot(w,20*log10(abs(h)));title('Respuesta frecuencia
filtro');xlabel('Frecuencia (Hz)');ylabel('H(f)');
%Respuesta temporal del filtro
subplot(325);plot(t,y);title('Respuesta temporal
filtro');xlabel('Tiempo');ylabel('y(t)');
```

11.1.2 Filtro chebyshev

```
%Borrado de memoria
clearvars ?global
%Lectura de datos de la primera columna de la hoja de cálculo que es la
%potencia recibida
x = xlsread('muestras.xlsx','Limpio','A1:A12454');
%Leemos los valores temporales que se encuentran en la segunda columna
tiempos = xlsread('muestras.xlsx','Limpio','B1:B12454');
%Obtenemos la media de tiempos que será el periodo de muestreo
Ts = median(tiempos)/1000;
%La longitud del vector de señal
L = length(x);
%La media de la señal
Media = median(x);
%x = x-Media;
%Generamos el vector de tiempos
t = (0:(L-1))*Ts;
%Declaración de la frecuencia de muestreo
fs=1/Ts;
%Transformada rápida de Fourier de la señal
X=fft(x,L);
%Módulo de la transformada
X_abs = abs(X/L);
%Cálculo de la densidad espectral de potencia Gx.
Gx=X_abs.^2;
%Definición del vector de frecuencias
f=-fs/2:fs/(length(X)-1):fs/2;


%Cáculo de un filtro de chebyshev
%Límite de la frecuencia de paso
fp=0.05;
%Límite de la frecuencia de corte
fc=0.07;
%Rizado en la banda de paso en dB
Rp=5;
%Rizado en la bada de rechazo en dB
Rs=35;
%Límite de la frecuencia de paso normalizada a la frecuencia de Nyquist
%fs/2
Wp=fp/(fs/2);
%Límite inferior de la banda de rechazo normalizada
Ws=fc/(fs/2);
[N,Wn] =cheb1ord(Wp, Ws, Rp,Rs);
[B,A]=cheby1(N,Rp,Wn);
[h,w] = freqz(B,A,length(x),'whole',fs);
%filtrado
```

```
y=filter(B,A,x);

%Impresión de resultados
%Señal temporal
figure(2)
subplot(321);plot(t,x);xlabel('Tiempo');ylabel('x(t)');
%Módulo de la transformada de Fourier
subplot(322);plot(f,fftshift(X_abs));xlabel('Frecuencia
(Hz)');ylabel('|X(f)|');
%Densidad espectral
subplot(323);plot(f,fftshift(Gx));xlabel('Frecuencia
(Hz)');ylabel('Gx(f)');
%Respuesta frecuencial del filtro
subplot(324);plot(w,20*log10(abs(h)));xlabel('Frecuencia
(Hz)');ylabel('H(f)');
%Respuesta temporal del filtro
subplot(325);plot(t,y);xlabel('Tiempo');ylabel('y(t)');
```

11.1.3 Filtro de kalman

```
clear all
clc
tic
X(1)=25;
e = xlsread('nodo1_130g_6m.xlsx','nodo1_130g_6m','A1:A150');
e=e';
%e=e-mean(e);%ruido

var_obs=var(e);%varianza del ruido de medicion
var_estado=0;   %varianza estado
P_despues(1)=var(e); %covarianza error a priori

L = length(e);
%La media de la señal
Media = median(e);
%Restamos la media para que sólo quede el ruido
%x = x-Media;
%Generamos el vector de tiempos
%t = (0:(L-1))*Ts;

for k=1:length(e)
    %estapa prediccion
   %predicion del estado
   X_antes(k+1)=X(k);
   %prediccion de la varianza
   P_antes(k+1)=P_despues(k)+var_estado;

   %estapa correccion
   ganancia(k+1)=P_antes(k+1)/(P_antes(k+1)+var_obs);%ganancia
   Y(k+1)=e(k);%observacion
   X(k+1)=X_antes(k+1)+ganancia(k+1)*(Y(k+1)-X_antes(k+1));%estimacion
del estado
   P_despues(k+1)=(1-ganancia(k))*P_antes(k+1);
end
toc
figure(1);
 subplot(121);plot(X,'r');title('señal filtrada del ruido')
subplot(122); plot(Y,'b');title('el ruido')
```

11.1.4 Redes neuronales para trilateración

11.1.4.1 Red neuronal 1

```
clc, clear all, close all
%la primera fila de p seria nuestra potencia y la segunda fila seria el
angulo
pya1=[-58.9 -56.03 -51.09 -55.13 -54.39 -52.25 -57.42 -59.56 -52.81 -53.5
-57.81 -61.21 -60.54 -57.31 -53.34 -59.44 -61.37 -65.1 -69.22 -64.22 -
60.17 -65.4 -61.77 -58.76 -61.03 -67.98 -64.1 -68.6 -63.2 -69.16 -64.71 -
67.09 -71.1 -69.88 -68.28 -66.46 -63.31 -65.62 -68.93 -63.36 -61.2 -63.22
-69.03 -67.09 -65.42 -64.5 -69.24 -64.26 -66.84 -60.84 -73.4 -70.2 -65.17
-68.62 -63.79 -68.87 -64.06 -63.07 -62.52 -63.92 -71.6 -67.63 -76.98 -
63.34 -61.41 -64.11 -63.28 -67.65 -68.72 -73.71 -72.45 -69.41 -66.61 -
61.09 -64.91 -66.48 -71.54 -74.18 -67.54 -64.66 -65.88 -67.49 -86.57 -
67.12 -70.39 -75.3 -72.54;0 10 20 30 40 50 60 70 80 90 0 10 20 30 40 50
60 70 80 90 0 10 20 30 40 50 60 70 80 90 0 10 20 30 40 50 60 70 80 90 0
10 20 30 40 50 60 70 80 90 0 10 20 30 40 50 60 70 80 0 10 20 30 40 50 60
60 70 80 0 10 20 30 40 50 60 0 10 20 30 40 50 0 10 20 30 40];

%el vector t seria nuestra salida ideal
t=  [0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 1 1 1 1 1 1 1 1 1 1 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 2 2 2 2 2 2 2 2 2 2 2.5 2.5 2.5 2.5 2.5
2.5 2.5 2.5 2.5 2.5 3 3 3 3 3 3 3 3 3 3 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5
3.5 4 4 4 4 4 4 4 4.5 4.5 4.5 4.5 4.5 4.5 5 5 5 5 5];

%se crea la red neuronal
net1=newff(minmax(pya1),[200 1],{'tansig','purelin'},'traingdm');

%se le impone una parada
net1.trainParam.epochs = 800;

%entreno a la red
[net1,tr]=train(net1,pya1,t);

%aqui simulo la red
distancia1=sim(net1,pya1)

%pruebas de simulacion
p1_nodo1=sim(net1,[[-66.28 -66.28 -66.28 -66.28 -66.28 -66.28 -66.28 -
66.28 -66.28 -66.28];[0:10:90]])
p2_nodo1=sim(net1,[[-69.92 -69.92 -69.92 -69.92 -69.92 -69.92 -69.92 -
69.92 -69.92 -69.92];[0:10:90]])
p3_nodo1=sim(net1,[[-66.03 -66.03 -66.03 -66.03 -66.03 -66.03 -66.03 -
66.03 -66.03 -66.03];[0:10:90]])
```

11.1.4.2 Red neuronal 2

```
clc, clear all, close all
%la primera fila de p seria nuestra potencia y la segunda fila seria el
angulo
pya2=[-59.3 -52.2 -50.61 -52.39 -62.13 -54.92 -53.04 -50.95 -65.53 -53.23
-58.07 -62.26 -56.43 -57.06 -54.85 -56.75 -59.82 -68.03 -61.32 -60.66 -
63.39 -71.22 -61.77 -66.62 -65.6 -73.18 -68.16 -62.13 -65.52 -63.04 -
68.33 -61.44 -59.81 -66.42 -65.72 -63.83 -61.89 -62.58 -69.36 -76.66 -
72.29 -69.16 -69.15 -69.75 -73.49 -64.19 -65.94 -67.77 -68.72 -69.35 -
68.81 -72.64 -69.76 -64.79 -64.95 -64.23 -62.64 -63.41 -70.83 -74.11 -
78.42 -71.45 -70.19 -67.86 -65.61 -62.9 -61.04 -65.18 -70.47 -72.3 -70.23
-66.94 -66.98 -65.45 -74.71 -68.52 -74.49 -69.08 -67.88 -69.99 -66.1 -
70.8 -77.56 -65.21 -67.02 -68.85 -74.58;0 10 20 30 40 50 60 70 80 90 0 10
20 30 40 50 60 70 80 90 0 10 20 30 40 50 60 70 80 90 0 10 20 30 40 50 60
```

```
70 80 90 0 10 20 30 40 50 60 70 80 90 0 10 20 30 40 50 60 70 80 90 0 10
20 30 40 50 60 70 80 0 10 20 30 40 50 60 0 10 20 30 40 50 0 10 20 30 40];

%el vector t seria nuestra salida ideal
t=[0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 1 1 1 1 1 1 1 1 1 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 2 2 2 2 2 2 2 2 2 2.5 2.5 2.5 2.5 2.5
2.5 2.5 2.5 2.5 2.5 3 3 3 3 3 3 3 3 3 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5
3.5 4 4 4 4 4 4 4 4.5 4.5 4.5 4.5 4.5 4.5 5 5 5 5 5];

%se crea la red neuronal
net2=newff(minmax(pya2),[200 1],{'tansig','purelin'},'traingdm');
%se le impone una parada
net2.trainParam.epochs = 800;
%entreno a la red
[net2,tr]=train(net2,pya2,t);
%aqui simulo la red
distancia2=sim(net2,pya2)

 %pruebas de simulacion
p1_nodo2=sim(net2,[[-73.25 -73.25 -73.25 -73.25 -73.25 -73.25 -73.25 -
73.25 -73.25 -73.25];[0:10:90]])

p2_nodo2=sim(net2,[[-69.05 -69.05 -69.05 -69.05 -69.05 -69.05 -69.05 -
69.05 -69.05 -69.05];[0:10:90]])

p3_nodo2=sim(net2,[[-70.95 -70.95 -70.95 -70.95 -70.95 -70.95 -70.95 -
70.95 -70.95 -70.95];[0:10:90]])
```

11.1.4.3 Red neuronal 3

```
clc, clear all, close all
%la primera fila de p seria nuestra potencia y la segunda fila seria el
angulo
pya3=[-51.3 -55.58 -55.98 -52.97 -52.82 -56.51 -51.94 -54.56 -64.53 -52.8
-56.72 -61.55 -67.19 -69.52 -60.06 -57.46 -58.88 -57.84 -61.29 -64.27 -
67.12 -72.04 -68.35 -69.87 -60.72 -57.13 -58.39 -68.63 -61.39 -52.14 -
69.83 -63.56 -65.76 -66.54 -65.96 -68.62 -61.45 -65.99 -59.81 -64.39 -
63.36 -60.83 -64.42 -67.91 -71.35 -64.82 -70.29 -63.16 -65.73 -61.76 -
66.2 -66.92 -66.14 -61.78 -63.56 -66.34 -70.14 -77.53 -77.2 -65.03 -65.39
-66.49 -70.94 -64.98 -65.21 -65.62 -69.79 -63.4 -76.88 -68.01 -71.72 -
66.96 -67.28 -66.23 -72.01 -71.51 -72.15 -69.27 -66.33 -66.59 -72.45 -
73.55 -74.45 -66.14 -67.54 -66.01 -84.27;0 10 20 30 40 50 60 70 80 90 0
10 20 30 40 50 60 70 80 90 0 10 20 30 40 50 60 70 80 90 0 10 20 30 40 50
60 70 80 90 0 10 20 30 40 50 60 70 80 90 10 20 30 40 50 60 70 80 90 30
40 50 60 70 80 90 40 50 60 70 80 90 50 60 70 80 90]


%el vector t seria nuestra salida ideal
t=  [0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 1 1 1 1 1 1 1 1 1 1.5 1.5
1.5 1.5 1.5 1.5 1.5 1.5 1.5 1.5 2 2 2 2 2 2 2 2 2 2.5 2.5 2.5 2.5 2.5
2.5 2.5 2.5 2.5 2.5 3 3 3 3 3 3 3 3 3 3.5 3.5 3.5 3.5 3.5 3.5 3.5 3.5
3.5 4 4 4 4 4 4 4 4.5 4.5 4.5 4.5 4.5 4.5 5 5 5 5 5];

%se crea la red
net3=newff(minmax(pya3),[200 1],{'tansig','purelin'},'traingdm');

%se le impone una parada
net3.trainParam.epochs = 800;

%entreno a la red
```

```
[net3,tr]=train(net3,pya3,t);

%aqui simulo la red
distancia3=sim(net3,pya3)

%se simulan pruebas
p1_nodo3=sim(net3,[[-67.02 -67.02 -67.02 -67.02 -67.02 -67.02 -67.02 -
67.02 -67.02 -67.02];[0:10:90]])
p2_nodo3=sim(net3,[[-65.75 -65.75 -65.75 -65.75 -65.75 -65.75 -65.75 -
65.75 -65.75 -65.75];[0:10:90]])
p3_nodo3=sim(net3,[[-64 -64 -64 -64 -64 -64 -64 -64 -64 -64];[0:10:90]])
```

11.1.5 Método de posicionamiento trilateración

```
clear all
tic

%a continuación se muestran las distancia obtenías por las redes
neuronales en el anterior apartado
%Nodo1
d1=[2.3844    2.9241    3.1355    3.3066    3.5340    2.9537    2.9140
2.2626    2.2507    1.9211]
%d1 =[3.5442    3.4032    3.7235    3.1410    3.9123    3.3222    3.0672
2.2607    2.0997    1.9480]
%d1=[2.3449    2.8356    3.0009    3.3891    3.5210    2.9642    2.8542
2.2661    2.2493    1.9038]

%nodo2
d2=[3.6464    3.7167    3.5938    3.5671    3.6772    2.4641    4.0128
3.9857    3.5490    3.0474]
%d2 =[2.6612    3.4077    3.3563    4.0152    3.0857    3.7920    3.2807
2.1836    2.2019    2.4949]
%d2=[3.0649    3.4104    3.5182    3.8962    3.6099    3.8178    3.6981
2.8196    3.2712    2.6454]


%Nodo3
d3=[1.9660    3.0233    2.0824    2.5158    2.1224    3.3412    4.3239
3.5988    3.7290    3.2845]
%d3 =[2.8215    3.3351    2.5634    2.2640    2.5036    3.1024    4.1876
3.7288    3.0701    2.4998]
%d3=[2.5517    2.4934    2.6000    2.3272    3.1671    2.6734    3.2285
3.0224    2.2564    2.0055]

%lo primero que deberiamos hacer es posicionar los receptores

 t1=0:pi/18:pi/2;
 x1=d1.*(cos(t1));
 y1=d1.*(sin(t1));
 hold on
 plot(x1,y1);axis 'square'


 t2=pi:pi/18:(3*pi)/2;
 x2=5+(d2.*cos(t2));
 y2=3.46+(d2.*sin(t2));
 plot(x2,y2);axis 'square'
```

```
t3=(3*pi)/2:pi/18:2*pi;
x3=(cos(t3)).*d3;
y3=3.46+(d3.*sin(t3));
plot(x3,y3);axis 'square'



for i=1:length(x1)
  x1(i);
  y1(i);

    for j=1:length(x2)

        dx1(j)=sqrt((x2(j)-x1(i))^2+(y2(j)-y1(i))^2);

    end
   di(i)=min(dx1);
end



 %obtencion del primer punto de corte
di2=sort(di);%ordenamos de menor a mayor para quedarme con las dos
posiciones mas cercanas
X=x1(find(di==di2(1)));%punto1
Y=y1(find(di==di2(1)));

%obtencion del segundo punto de corte
X2=x1(find(di==di2(2)));%punto2
Y2=y1(find(di==di2(2)));

%obtencion de cual de los dos puntos es el final
P31=abs(Y-x3); %en referencia a punto 1
P32=abs(Y2-x3);% en referencia al punto 2

%se compara cual es la minima separacion en y de los dos puntos para
saber
%donde estamos

if min(P31)<min(P32)
    disp(['nos encontramos en el pinto (x),(y): ',num2str([X Y])] )

else
    disp(['nos encontramos en el pinto (x),(y): ',num2str([X2 Y2])] )

end

toc
```

11.1.6 Método de posicionamiento cartesiano

```
clc, clear all, close all

%la primera fila de p seria nuestra potencia del nodo1
%la segunda fila de p seria nuestra potencia del nodo2
%la tercera fila de p seria nuestra potencia del nodo3
```

```
p1=[-35.12  -63.33  -66.35  -61.96  -73.63  -70.65  -67.63  -74.19 -52.32
-52.95  -65.65  -64.15  -67.37  -73.89  -65.59  -63.71 -56.07   -56.85  -
63.76  -64.02  -68.95  -70.04  -65.78  -64.4 -64.46 -69.2 -62.8 -65.67 -
71.5 -62.6 -68.23 -77.05 -68.6 -62.76 -69.81 -73.4 -66.54 -65.41 -71.46 -
70.54 -63.99 -72.5 -71.25 -67.68 -72.08 -75.31 -72.6 -69.23 -66.69 -72.87
-72.63 -66.14 -71.44 -66.39 -69.03 -76.17 -73.46 -72.81 -66.88 -70.67 -
68.42 -71.6 -69.88 -76.33;-67.82 -69.27 -72.9 -70.76 -71.55 -71.14 -67.52
-26.2 -67.88 -65.56 -68.89 -68.32  -67.73  -64.32  -59.42  -63.01 -68.13
-67.82  -66.89  -66.48  -65.48  -69.24  -75.13  -65.85 -72.03   -70.84  -
75.11  -65.65  -68.96  -73.36  -67.92  -70.71 -75.88   -74.36  -66.49  -
72.54  -68.98  -69.22  -64.52  -67.15 -69.63   -71.22  -75.11  -77.99  -
76.7   -71.47  -71.24  -73.05 -64.65   -69.21  -65.89  -70.73  -72.6   -
66.09  -73.53  -73.58 -68.7 -66.76 -69.82  -70.79  -75.35  -74.74  -72.5
-72.94 ;-71.65  -69.03  -72.73  -78.19  -67.69  -74.43  -71.32  -63.46 -
70.32   -82.08  -73.77  -72.42  -67.74  -76.06  -75.72  -71.28 -68.06   -
69.28  -72.71  -65.83  -71 -79.54  -70.46  -69.14 -69.31   -70.11  -72.69
-72.11  -70.9   -76.45  -74.97  -70.74 -66.39   -69.34  -69.77  -71.67  -
74 -69.96  -78.41  -69.17 -62.54   -68.63  -74.27  -74.72  -72.14  -69.62
-70.93  -66.23 -60.89   -70.39  -69.23  -70.77  -73.59  -65.23  -76.79  -
67.81 -39.72   -50.81  -60.72  -63.21  -67.41  -66.6   -73.91  -69.15];

%el vector t seria nuestra salida ideal en nuestro caso yo lo he tomado
por x e y respectivamente
t=[0 0.44 0.88 1.32 1.76 2.2 2.64 3.08 0 0.44 0.88 1.32 1.76 2.2 2.64
3.08 0 0.44 0.88 1.32 1.76 2.2 2.64 3.08 0 0.44 0.88 1.32 1.76 2.2 2.64
3.08 0 0.44 0.88 1.32 1.76 2.2 2.64 3.08 0 0.44 0.88 1.32 1.76 2.2 2.64
3.08 0 0.44 0.88 1.32 1.76 2.2 2.64 3.08 0 0.44 0.88 1.32 1.76 2.2 2.64
3.08;0 0 0 0 0 0 0 0 0.44 0.44 0.44 0.44 0.44 0.44 0.44 0.44 0.88 0.88
0.88 0.88 0.88 0.88 0.88 0.88 0.88 1.32 1.32 1.32 1.32 1.32 1.32 1.32 1.32
1.76 1.76 1.76 1.76 1.76 1.76 1.76 1.76 2.2 2.2 2.2 2.2 2.2 2.2 2.2 2.2 2.2
2.64 2.64 2.64 2.64 2.64 2.64 2.64 2.64 3.08 3.08 3.08 3.08 3.08 3.08
3.08 3.08];

%se crea la red
net1=newff(minmax(p1),[200 2],{'tansig','purelin'},'traingdm');
%se le impone una parada
net1.trainParam.epochs =800;

%entreno a la red
[net1,tr]=train(net1,p1,t);
%aqui simulo la red
distancia=sim(net1,p1)

%se simulan las pruebas

a=sim(net1,[-54.33;-72.96;-70.38])
b=sim(net1,[-77.44;-75.79;-52.02])
c=sim(net1,[-66.25;-68.61;-68.86])
d=sim(net1,[-71.17;-76.32;-77.26])
e=sim(net1,[-64.94;-72.31;-72.92])
f=sim(net1,[-33.82;-64.89;-67.81])
g=sim(net1,[-69.32;-63.56;-71.97])
h=sim(net1,[-66.61;-64.71;-70.59])
i=sim(net1,[-75.54;-76.66;-62.73])

%se crea un vector con la posición “x” y otro con la posición “y”

x=[a(1) b(1) c(1) d(1) e(1) f(1) g(1) h(1) i(1)];
y=[a(2) b(2) c(2) d(2) e(2) f(2) g(2) h(2) i(2)];
```

```
%error en x para cada simulación, el número es el punto real
errorxp1=abs(x(1)-0.66)
errorxp2=abs(x(2)-0.66)
errorxp3=abs(x(3)-1.76)
errorxp4=abs(x(4)-2.86)
errorxp5=abs(x(5)-1.32)
errorxp6=abs(x(6)-0)
errorxp7=abs(x(7)-2.64)
errorxp8=abs(x(8)-1.1)
errorxp9=abs(x(9)-0.4)

%se crea un vector con los errores de las simulaciones para la coordenada
"x"
vector_errorx=[errorxp1 errorxp2 errorxp3 errorxp4 errorxp5 errorxp6
errorxp7 errorxp8 errorxp9]

%%error en y para cada simulación, el numero es el punto real

erroryp1=abs(y(1)-0)
erroryp2=abs(y(2)-3.08)
erroryp3=abs(y(3)-1.54)
erroryp4=abs(y(4)-2.86)
erroryp5=abs(y(5)-1.1)
erroryp6=abs(y(6)-0.22)
erroryp7=abs(y(7)-0.66)
erroryp8=abs(y(8)-1.98)
erroryp9=abs(y(9)-2.86)

%se crea un vector con los errores de las simulaciones para la coordenada
"y"
 vector_errory=[erroryp1 erroryp2 erroryp3 erroryp4 erroryp5 erroryp6
erroryp7 erroryp8 erroryp9]
L=length(vector_errorx);
%error cuadratico medio en x
error_cuad_x = sqrt(1/L*sum(vector_errorx.^2))
%error cuadratico medio en y
error_cuad_y = sqrt(1/L*sum(vector_errory.^2
```

11.1.7 Programa para visualizar el método cartesiano

```
%se crea la habitación, se dibuja una cuadricular para simular el suelo
en baldosas
X=-0.88:0.44:5.28
Y=-0.88:0.44:5.28
hold on
[X,Y]=meshgrid(X,Y)
plot(X,Y,X',Y','linewidth',1,'color', [1 0 0])
%punto real donde nos encontramos
x=0.27;
y=1.57;
%Punto que nos calcula la red neuronal
x1=0.3908;
y1=2.217;
%se grafica el punto donde estamos
plot(x,y,'o')

%se grafica el punto calculado por la red neuronal
plot(x1,y1,'b x')
```

11.2 MEDIDAS TOMADAS

Para no extender la memoria no se pondrá una muestra de la captación del raspberry antes de ser filtrado, sino que directamente se pondrá las medidas filtradas por el filtro de kalman para el caso del plano polar y en el caso del plano cartesiano.

11.2.1 Medidas plano polar

11.2.1.1 Nodo1

	0.5m	1m	1.5m	2m	2.5m	3m	3.5m	4m	4.5m	5m
0g	-58.9	-57.81	-60.17	-64.71	-61.2	-73.4	-71.6	-73.71	-71.54	-86.57
10g	-56.03	-61.21	-65.4	-67.09	-63.22	-70.2	-67.63	-72.45	-74.18	-67.12
20g	-51.09	-60.54	-61.77	-71.1	-69.03	-65.17	-76.98	-69.41	-67.54	-70.39
30g	-55.13	-57.31	-58.76	-69.88	-67.09	-68.62	-63.34	-66.61	-64.66	-75.3
40g	-54.39	-53.34	-61.03	-68.28	-65.42	-63.79	-61.41	-61.09	-65.88	-72.54
50g	-52.25	-59.44	-67.98	-66.46	-64.5	-68.87	-64.11	-64.91	-67.49	
60g	-57.42	-61.37	-64.1	-63.31	-69.24	-64.06	-63.28	-66.48		
70g	-59.56	-65.1	-68.6	-65.62	-64.26	-63.07	-67.65			
80g	-52.81	-69.22	-63.2	-68.93	-66.84	-62.52	-68.72			
90g	-53.5	-64.22	-69.16	-63.36	-60.84	-63.92				

prueba1	
1_pprueba_8g_1.6m	-66.28
prueba 2	
1_pprueba_50g_3.37m	-69.92
prueba3	
1_prueba__50g_2.4m	-66.03

11.2.1.2 Nodo 2

	0.5m	1m	1.5m	2m	2.5m	3m	3.5m	4m	4.5m	5m
0g	-59.3	-58.07	-63.39	-68.33	-72.29	-68.81	-78.42	-72.3	-74.49	-77.56
10g	-52.2	-62.26	-71.22	-61.44	-69.16	-72.64	-71.45	-70.23	-69.08	-65.21
20g	-50.61	-56.43	-61.77	-59.81	-69.15	-69.76	-70.19	-66.94	-67.88	-67.02

30g	-52.39	-57.06	-66.62	-66.42	-69.75	-64.79	-67.86	-66.98	-69.99	-68.85
40g	-62.13	-54.85	-65.6	-65.72	-73.49	-64.95	-65.61	-65.45	-66.1	-74.58
50g	-54.92	-56.75	-73.18	-63.83	-64.19	-64.23	-62.9	-74.71	-70.8	
60g	-53.04	-59.82	-68.16	-61.89	-65.94	-62.64	-61.04	-68.52		
70g	-50.95	-68.03	-62.13	-62.58	-67.77	-63.41	-65.18			
80g	-65.53	-61.32	-65.52	-69.36	-68.72	-70.83	-70.47			
90g	-53.23	-60.66	-63.04	-76.66	-69.35	-74.11				

prueba 1	
2_pprueba_25g_5m	-73.25
prueba 2	
2_pprueba_60g_2.70m	-69.05
prueba3	
2_prueba__30g_3.76m	-70.95

11.2.1.3 Nodo 3

	0.5	1	1.5	2	2.5	3	3.5	4	4.5	5
0g	-51.3	-56.72	-67.12	-69.83	-63.36	-66.2				
10g	-55.58	-61.55	-72.04	-63.56	-60.83	-66.92	-65.39			
20g	-55.98	-67.19	-68.35	-65.76	-64.42	-66.14	-66.49			
30g	-52.97	-69.52	-69.87	-66.54	-67.91	-61.78	-70.94	-68.01		
40g	-52.82	-60.06	-60.72	-65.96	-71.35	-63.56	-64.98	-71.72	-72.15	
50g	-56.51	-57.46	-57.13	-68.62	-64.82	-66.34	-65.21	-66.96	-69.27	-74.45
60g	-51.94	-58.88	-58.39	-61.45	-70.29	-70.14	-65.62	-67.28	-66.33	-66.14
70g	-54.56	-57.84	-68.63	-65.99	-63.16	-77.53	-69.79	-66.23	-66.59	-67.54
80g	-64.53	-61.29	-61.39	-59.81	-65.73	-77.2	-63.4	-72.01	-72.45	-66.01
90g	-52.8	-64.27	-52.14	-64.39	-61.76	-65.03	-76.88	-71.51	-73.55	-84.27

prueba 1	
3_pprueba_9g_2.111m	-67.02
prueba 2	
3_pprueba_59g_3.41m	-65.75
prueba3	-64

3_prueba__40g_2.32m	

11.2.2 Medidas plano cartesiano

11.2.2.1 Nodo 1

x	0	0.44	0.88	1.32	1.76	2.2	2.64	3.08
y								
0	-35.12	-63.33	-66.35	-61.96	-73.63	-70.65	-67.63	-74.19
0.44	-52.32	-52.95	-65.65	-64.15	-67.37	-73.89	-65.59	-63.71
0.88	-56.07	-56.85	-63.76	-64.02	-68.95	-70.04	-65.78	-64.4
1.32	-64.46	-69.2	-62.8	-65.67	-71.5	62.6	-68.23	-77.05
1.76	-68.6	-62.76	-69.81	-73.4	-66.54	-65.41	-71.46	-70.54
2.2	-63.99	-72.5	-71.25	-67.68	-72.08	-75.31	-72.6	-69.23
2.64	-66.69	-72.87	-72.63	-66.14	-71.44	-66.39	-69.03	-76.17
3.08	-73.46	-72.81	-66.88	-70.67	-68.42	-71.6	-69.88	-76.33

1_prueba1_x0.66_y0	-54.33
1_prueba2_x0.66_y3.08	-77.44
1_prueba3_x1.76_y1.54	-66.25
1_prueba4_x2.86_y2.86	-71.17
1_prueba5_x1.32_y1.1	-64.94
1_prueba6_x0_y0.22	-33.82
1_prueba7_x2.64_y0.66	-69.32
1_prueba8_x1.1_y1.86	-66.61
1_prueba9_x0.44_y2.86	-75.54

11.2.2.2 Nodo 2

x	0	0.44	0.88	1.32	1.76	2.2	2.64	3.08
y								
0	-67.82	-69.27	-72.9	-70.76	-71.55	-71.14	-67.52	-26.2
0.44	-67.88	-65.56	-68.89	-68.32	-67.73	-64.32	-59.42	-63.01
0.88	-68.13	-67.82	-66.89	-66.48	-65.48	-69.24	-75.13	-65.85
1.32	-72.03	-70.84	-75.11	-65.65	-68.96	-73.36	-67.92	-70.71
1.76	-75.88	-74.36	-66.49	-72.54	-68.98	-69.22	-64.52	-67.15
2.2	-69.63	-71.22	-75.11	-77.99	-76.7	-71.47	-71.24	-73.05
2.64	-64.65	-69.21	-65.89	-70.73	-72.6	-66.09	-73.53	-73.58
3.08	-68.7	-66.76	-69.82	-70.79	-75.35	-74.74	-72.5	-72.94

2_prueba1_0.66_y0	-72.96
2_prueba_x0.66_y3.08	-75.79
2_prueba_x1.76_y1.54	-68.61
2_prueba_x2.86_y2.86	-76.32
2_prueba_x1.32_y1.1	-72.31
2_prueba_x0_y0.22	-64.89
2_prueba_x2.64_y0.66	-63.56
2_prueba_x1.1_y1.86	-64.71
2_prueba_x0.44_y2.86	-76.66

11.2.2.3 Nodo 3

x	0	0.44	0.88	1.32	1.76	2.2	2.64	3.08
y								
0	-71.65	-69.03	-72.73	-78.19	-67.69	-74.43	-71.32	-63.46
0.44	-70.32	-82.08	-73.77	-72.42	-67.74	-76.06	-75.72	-71.28
0.88	-68.06	-69.28	-72.71	-65.83	-71	-79.54	-70.46	-69.14
1.32	-69.31	-70.11	-72.69	-72.11	-70.9	-76.45	-74.97	-70.74
1.76	-66.39	-69.34	-69.77	-71.67	-74	-69.96	-78.41	-69.17
2.2	-62.54	-68.63	-74.27	-74.72	-72.14	-69.62	-70.93	-66.23
2.64	-60.89	-70.39	-69.23	-70.77	-73.59	-65.23	-76.79	-67.81
3.08	-39.72	-50.81	-60.72	-63.21	-67.41	-66.6	-73.91	-69.15

3_prueba_x0.66_y0	-70.38
3_prueba_x0.66_y3.08	-52.02
3_prueba_x1.76_y1.54	-68.86
3_prueba_x2.86_y2.86	-77.26
3_prueba_x1.32_y1.1	-72.92
3_prueba_x0_y0.22	-67.81
3_prueba_x2.64_y0.66	-71.97
3_prueba_x1.1_y1.86	-70.59
3_prueba_x0.44_y2.86	-62.73

12 BIBLIOGRAFIA

[1]Acquatella, P. (15 de marzo de 2012). Tutorial de Backpropagation - Un algoritmo de entrenamiento para redes neuronales. Recuperado el 27 de diciembre de 2017, de https://es.mathworks.com/matlabcentral/fileexchange/35659-tutorial-de-backpropagation-un-algoritmo-de-entrenamiento-para-redes-neuronales

[2]Flórez, R., & Fernández, J. M. (2008). Las Redes Neuronales Artificiales. La Coruña: Netbiblo Sl.

[3]Gonzalez, J. R., & Hernando, V. j. (1995). Redes neuronales artificiales : fundamentos, modelos y aplicaciones. Madrid: Ra-Ma editorial.

[4]Harston, C. T., Maren, A. J., & Pap, R. M. (1990). Handbook of Neural Computing Applications. California: Academic Press.

[5]lopez, J. a. (03 de marzo de 2000). Redes Neuronales. Recuperado el 07 de 12 de 2017, de http://members.tripod.com/jesus_alfonso_lopez/RnaIntro2.html

[6]Matich, D. J. (marzo de 2001). Redes Neuronales: Conceptos Básicos y. Recuperado el 26 de diciembre de 2017, de https://www.frro.utn.edu.ar/repositorio/catedras/quimica/5_anio/orientadora1/monograias/matich-redesneuronales.pdf

[7]"Redes Neuronales Artificiales". XIII Escuela de Verano de Informática, Isla de A Toxa.Asociación Española de Informática y Automática (AEIA), 1991.

[8]E. Castillo Ron, Á. Cobo Ortega, J. M. Gutiérrez Llorente, R. E. Pruneda González, “Introducción a la Redes Funcionales con Aplicaciones”, Paraninfo, 1999

[9]Greg Welch, Gary Bishop, "Introducción al filtro de Kalman", Universidad de Carolina del Norte en el Departamento de Ciencias de la Computación de Chapel Hill, 2001

[10]Ortiz, B. A., Nieto, A. M., & Quintero, S. E. (19 de mayo de 2013). Metodología para la Estimación de Parámetros en Tiempo Real mediante Filtros de Kalman y Mínimos Cuadrados.

[11]Molina, S. R. (s.f.). Bases del filtro de kalman. Recuperado el 16 de octubre de 2017,de http://www.bccr.fi.cr/investigacioneseconomicas/metodoscuantitativos/Filtro_de_Kalman.pdf

[12]Inelmatic. (octubre de 2015). El filtro de kalman. Recuperado el enero de 2018, de http://www.inelmatic.com/web/wp-content/uploads/2015/11/filtrodekalman1.pdf

[13]Lazaro, M. (2003). Tema 4.6 el filtro de kalman. Recuperado el 16 de Enero de 2018, de http://www.tsc.uc3m.es/~mlazaro/Docencia/Doctorado/FiltAdapt/Kalman.pdf

[14]Rodríguez, D., & Alamo, T. (s.f.). Tema 9. Espacio de Estados. Recuperado el 20 de Octubre de 2017, de http://www.control-class.com/Tema_9/Slides/Tema_9_Espacio_de_Estados.pdf

[15]Moreno, V. I. (s.f.). Filtrado. Recuperado el 7 de Diciembre de 2017, de http://www.unet.edu.ve/~ielectro/6-Filtrado.pdf

[16]Cogollos, B. S. (2016). Fundamentos de la Teoria de Filtros. Valencia: Universidad Politécnica de Valencia.

[17]Fernández, G., Martínez, W., Ávalos, V., & Ramírez, J. C. (6 de diciembre de 2010). Filtros Butterworth. Recuperado el 17 de octubre de 2017, de http://filtrosbutterworthw.blogspot.com.es/

[18]Fuentes Conde, M. Tema 6. Filtros activos con amplificador operacional Introducción.

[19]Mercedes, F. A. (s.f.). Tema 9: Tringulacion y trilateracion. Recuperado el 16 de Enero de 2018, de http://ocw.upm.es/ingenieria-cartografica-geodesica-y-fotogrametria/topografia-ii/contenidos/Mis_documentos/Tema-9-Triangulacion-y-Trilateracion/Teoria_Triang_Tema_9.pdf

[20]Vela, C. (6 de agosto de 2014). “Beacons bluetooth”: como un GPS de interiores. Recuperado el 14 de octubre de 2017, de https://aunclicdelastic.blogthinkbig.com/beacons-bluetooth-como-un-gps-de-interiores/

[21]T.Kohonen. "An Introduction to Neural Computing". Neural Networks, Vol.1, pp. 3-16,1988.

[22]Sancho, F. (23 de abril de 2017). Entrenamiento de Redes Neuronales: mejorando el Gradiente Descendiente. Recuperado el 4 de enero de 2018, de http://www.cs.us.es/~fsancho/?e=165.

[23]Mateo, F. (junio de 2012). Redes neuronales y preprocesado de variables. Recuperado el 4 de enero de 2018, de https://riunet.upv.es/bitstream/handle/10251/16702/tesisUPV3874.pdf