CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS
AVANZADOS DEL INSTITUTO POLITÉCNICO NACIONAL
UNIDAD ZACATENCO

DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

SECCIÓN DE ELECTRÓNICA DEL ESTADO SÓLIDO

“SISTEMA INALÁMBRICO NO INVASIVO BASADO EN SENSORES MEMS PARA EL SEGUIMIENTO Y EVALUACIÓN DE ACTIVIDAD PERSONAL”

T E S I S

Que presenta

M. EN C. LUIS SÁNCHEZ MÁRQUEZ

Para obtener el grado de

DOCTOR EN CIENCIAS

EN LA ESPECIALIDAD DE INGENIERÍA ELÉCTRICA

Directores de la tesis:

DR. MARIO ALFREDO REYES BARRANCA
DRA. GRISELDA STEPHANY ABARCA JIMÉNEZ

Ciudad de México Octubre, 2024

A Luis Martín Flores Nava

AGRADECIMIENTOS

Quiero agradecer principalmente a mis padres por el apoyo brindado en el transcurso de mi vida. A mis asesores de tesis: Dr. Mario Alfredo Reyes Barranca y Dra. Griselda Sthepany Abarca Jiménez por las revisiones, asesorías y consejos. Al CINVESTAV y al Conahcyt por los recursos y apoyo brindados para continuar con mi formación académica. A Andrea López Tapia, nadie me ha apoyado como ella, además de motivarme a dar lo mejor de mí. Al profesor Luis Martín Flores Nava, por brindarme sus conocimientos y consejos. Y a mis hermanos Hugo, Raúl y Alexandro, gracias por acompañarme en la vida.

CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

Este trabajo presenta la propuesta de una red inalámbrica de sensores inteligentes para obtener las vibraciones en un piso dispersivo y amortiguado producidas por el impacto de las pisadas de una persona. Las vibraciones se utilizan para localizar los impactos por medio de un algoritmo heurístico que permite una baja tasa de transmisión/recepción de datos. Usualmente, la localización se realiza sin dar mayor importancia al número de sensores y su ubicación dentro de la habitación; sin embargo, en este trabajo se muestra que, si se realiza un estudio de ambos parámetros, el algoritmo de localización puede tener un mejor desempeño. En este trabajo, el número de sensores, así como su ubicación son optimizados utilizando metaheurísticas bio-inspiradas para minimizar el error de localización de pisadas a lo largo de la habitación. Luego de comparar diferentes algoritmos de optimización, se selecciona el método con el menor error de estimación para la habitación completa. Las pruebas experimentales muestran que la optimización mejora el desempeño del algoritmo de localización para diferentes trayectorias. De igual forma, se muestra que con la optimización se puede reducir el costo del sistema.

Palabras Clave: Red de sensores, Seguimiento de ocupantes, Localización de impactos, Acelerómetro, Optimización, Metaheurísticas bio-inspiradas

ABSTRACT

This work presents the proposal of a wireless network of intelligent sensors to obtain vibrations in a dispersive and damped floor produced by the impact of a person's footsteps. The vibrations are used to locate the impacts through a heuristic algorithm that allows for a low data transmission/reception rate. Traditionally, the placement of sensors within a room has not been given significant consideration in the context of localization; however, our findings indicate that a detailed analysis of both the number and location of sensors can substantially enhance the accuracy of footstep localization. In this study, we have optimized the arrangement and quantity of sensors through the application of bio-inspired metaheuristics, aiming to minimize localization errors across the room. Upon evaluating various bio-inspired metaheuristic optimization algorithms, we identified the one that yielded the lowest estimation errors for the room as a whole. Our experimental tests demonstrate that such optimization significantly enhances the efficacy of the localization algorithm for different trajectories. It is also shown that optimization can reduce the cost of the system.

Keywords: Sensor network, Occupant tracking, Impact localization, Accelerometer, Optimization, Bio-inspired metaheuristics

OBJETIVOS

Objetivo general:

Diseñar un sistema inalámbrico de bajo costo y no invasivo, basado en acelerómetros, para adquirir las señales generadas por los impactos de los pies de una persona sobre un piso de concreto y determinar su ubicación en el interior de una habitación.

Objetivos particulares:

- Identificar un acelerómetro de bajo costo que cumpla con los propósitos de detección de señales producidas por los impactos de los pies de una persona.
- Proponer una red inalámbrica de sensores no invasiva para obtener las vibraciones en el piso de concreto de una habitación, producidas por los impactos de los pies de una persona.
- Implementar un algoritmo que permita identificar la trayectoria seguida por el sujeto bajo estudio en un entorno cotidiano.
- Realizar un estudio del error de localización en función del número de sensores utilizados.
- Optimizar la ubicación de los sensores dentro de una habitación para minimizar el error de localización de las pisadas hechas por un individuo.

- Establecer una base sólida mediante el diseño de la red para facilitar futuras investigaciones centradas en la medición y análisis de caminatas reales.

JUSTIFICACIÓN

El rastreo de una persona en el interior de un edificio tiene múltiples aplicaciones. Por ejemplo, al seguir la trayectoria de una persona se puede rastrear a un intruso. Durante un escenario de rescate/evacuación de emergencia, los socorristas pueden localizar a los ocupantes en peligro. En casas inteligentes, si se conoce la ubicación de una persona, se puede reducir el consumo energético de lámparas, aire acondicionado, etc. [1]. Y en el área de la salud, monitorear la actividad diaria que realiza una persona al caminar puede ayudar a estimar el gasto del nivel energético que ésta tiene durante el día y con esto prevenir, retrasar o controlar algunas enfermedades crónicas (diabetes, padecimientos cardiovasculares y respiratorios, obesidad) [2].

El grupo de investigación del laboratorio de VLSI de la SEES Cinvestav-Zacatenco tiene experiencia trabajando con sistemas micro electro mecánicos (MEMS) empleados como sensores y actuadores; sin embargo, no se ha trabajado en aplicaciones específicas para este tipo de dispositivos. Por esta razón, esta tesis pretende utilizar dispositivos MEMS para la localización de impactos de pisadas. Debido a que esta tarea requiere acelerómetros con características distintas a las de nuestros diseños, se optará por dispositivos comerciales.

Los sistemas de rastreo de personas en interiores ya han sido desarrollados; sin embargo, suelen ser invasivos o muy costosos. Por lo tanto, este proyecto busca diseñar un sistema de bajo costo y no invasivo, utilizando sensores MEMS para captar las vibraciones producidas por los impactos de los pies sobre un piso de concreto y determinar su ubicación mediante un algoritmo de localización de pisadas.

Los conocimientos obtenidos del sistema propuesto proporcionarán una base sólida para futuros desarrollos. Una vez validado el sistema, se podrá extender el enfoque a la medición y análisis de caminatas reales para el rastreo de personas en el interior de un edificio.

1 INTRODUCCIÓN

En el capítulo actual se presenta una breve introducción a los métodos de localización de personas, sus aplicaciones, las consideraciones que se deben tener para localizar personas en el interior de un edificio (a partir de ahora llamado interiores) y el estado del arte.

De igual forma se presentan los métodos de optimización y cómo estos podrían ayudar a mejorar el desempeño de los algoritmos de localización.

En el capítulo 2 se presentan los aspectos teóricos necesarios para diseñar y optimizar una red de sensores inteligentes. En el capítulo 3 se presenta el diseño y optimización de la red de sensores inteligentes. En el capítulo 4 se presentan los resultados experimentales de dicha red. Finalmente, en los capítulos 5, 6 y 7, se presenta una discusión de los resultados, las conclusiones generales y el trabajo a futuro, respectivamente.

1.1 Métodos clásicos de localización de objetivos

Para localizar y seguir la trayectoria de una persona u objeto en movimiento (también conocidos como fuente u objetivo), se emplea una red de sensores (también llamados nodos) capaces de medir alguna excitación del medio causada por el objetivo.

Dentro de los métodos tradicionales para localizar objetivos, se encuentran los que se basan en el tiempo de llegada de la señal (TOA, por sus siglas en inglés “Time Of Arrival”); los que se basan en la diferencia de tiempos

de llegada de la señal (TDOA, por sus siglas en inglés “Time Difference Of Arrival”); los que se basan en el ángulo de llegada (AOA, por sus siglas en inglés “Angle Of Arrival”); y los que se basan en la fuerza de la señal recibida (RSS, por sus siglas en inglés “Received Signal Strength”) [3]. Estos métodos clásicos de localización fueron diseñados principalmente para sistemas de radio frecuencia (RF) o Wi-Fi, donde las señales a estudiar se propagan por el aire y se considera que la velocidad de propagación c es constante.

La Figura 1-1 resume estos métodos clásicos de localización de objetivos. Conociendo las ubicaciones $\boldsymbol{s}_1, \boldsymbol{s}_2, \boldsymbol{s}_3$ de los sensores, los métodos basados en TOA y RSS estiman la ubicación mediante la intersección de tres círculos, cuyos radios R_1, R_2, R_3 están dados en función del tiempo de llegada (donde El TOA se define como el tiempo en el que la excitación del medio llega desde el objetivo hasta un sensor) o la potencia de la señal recibida. El método basado en TDOA estima la ubicación mediante la intersección de dos hipérbolas formadas por las diferencias de tiempo de llegada de la señal t_{ij} (donde i, j son un par de nodos), tomando como referencia a uno de los nodos (en la Figura 1-1b el nodo de referencia es $\boldsymbol{s}_1$). Finalmente, el método basado en AOA, estima la ubicación del objetivo mediante la intersección de dos líneas, cuyos ángulos se obtienen con las antenas de los nodos. En el Anexo A se presenta con más detalle estos métodos.

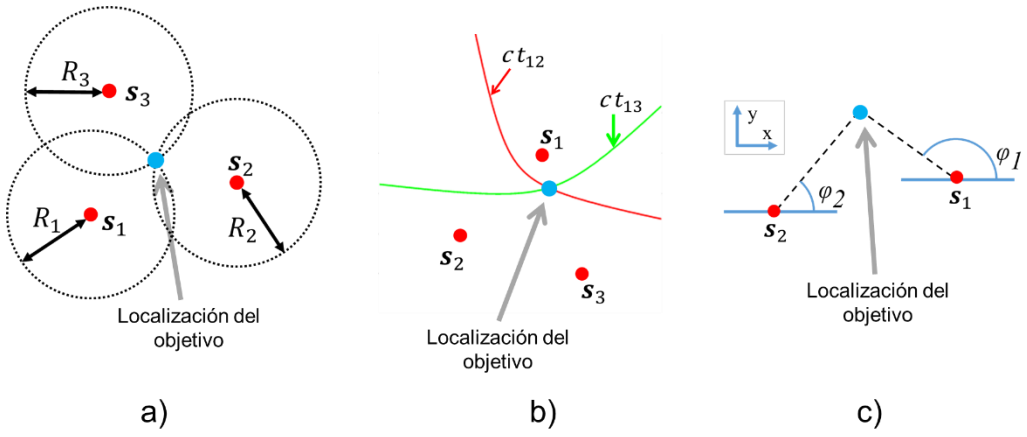


Figura 1-1. Resumen de los métodos clásicos de localización de objetivos. a) TOA y RSS se basan en círculos que se intersectan. b) TDOA se basa en hipérbolas que se intersectan. c) AOA se basa en rectas que se intersectan

1.1.1 Aplicaciones de los métodos clásicos de localización

Los métodos antes mencionados (TOA, TDOA, AOA, RSS), se pueden aplicar a diferentes tecnologías [3]. Por ejemplo, el Sistema de Posicionamiento Global (GPS, por sus siglas en inglés “Global Positioning System”) usa un conjunto de satélites (nodos dinámicos) y se basa en la estimación del TOA para determinar la ubicación del usuario.

El sistema de identificación por radio frecuencia (RFID, por sus siglas en inglés “Radio-Frequency Identification”) es un sistema inalámbrico que identifica etiquetas (“tags”, en inglés) adheridas al objeto de interés. Este sistema consta de lectores colocados a lo largo del lugar de prueba y etiquetas RFID que se colocan en el objeto a localizar. Para hacer la localización, se utiliza el método RSS.

En sistemas móviles, se miden las señales de radio que viajan entre una estación móvil y un conjunto de estaciones fijas. La ubicación de la estación móvil se puede obtener recopilando la información de radiolocalización del RSS, TOA, TDOA o sus combinaciones.

Entonces, dependiendo de la tecnología y los métodos utilizados, la tarea de localización de objetivos tendrá diferentes niveles de complejidad y requerimientos de infraestructura. La Figura 1-2 ilustra las aplicaciones antes mencionadas.

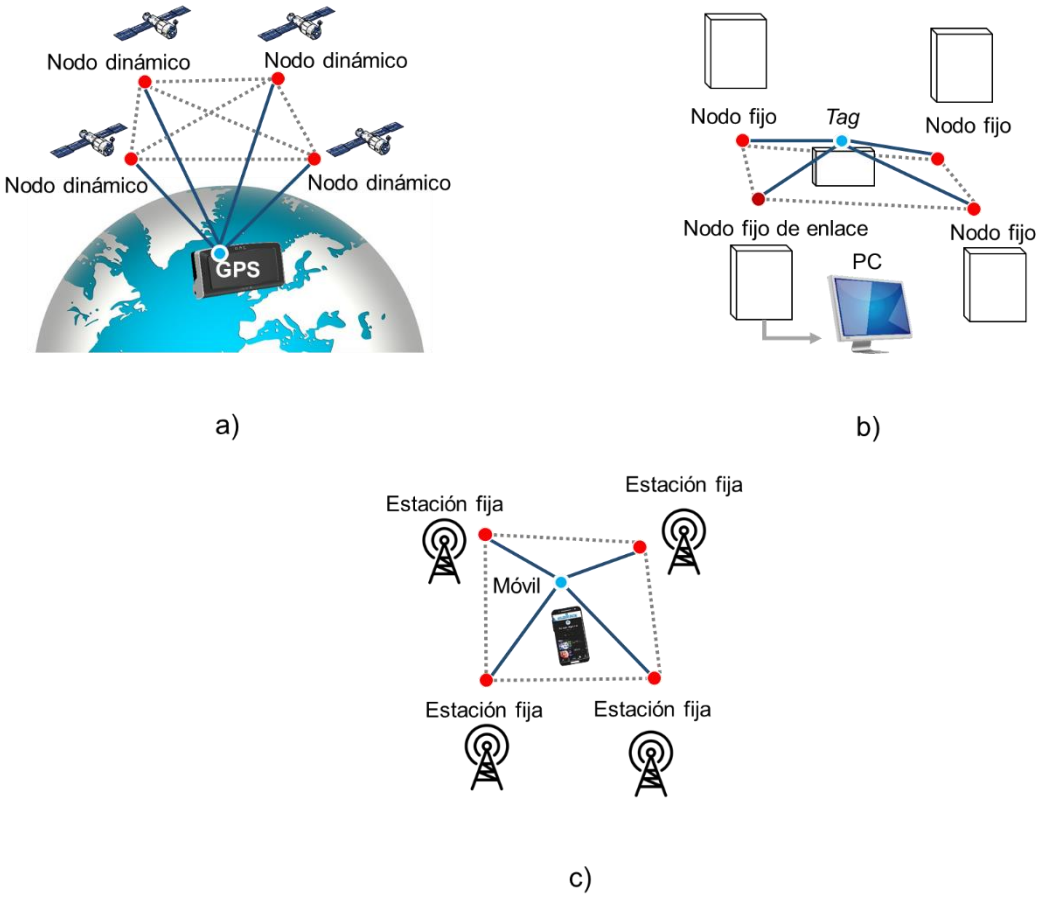


Figura 1-2. Aplicaciones de los métodos de localización de objetivos. a) GPS, b) RFID, c) Móvil. Los puntos rojos son los nodos de la red, el punto azul es el objetivo a localizar. Las líneas sólidas representan las conexiones entre el objetivo y los nodos de la red. Las líneas punteadas representan las conexiones entre los nodos de la red

1.2 Localización de objetivos en interiores

Para localizar objetivos en interiores se pueden utilizar sistemas de radiofrecuencia o WiFi, sistemas de grabación de audio/video y sistemas con sensores inerciales. La Figura 1-3 resume las características de los sistemas antes mencionados.

Figura 1-3. Sistemas de localización de objetivos en interiores y sus requerimientos

Como vimos en la sección anterior, los sistemas de radiofrecuencia o WiFi requieren de un conjunto de estaciones que se comuniquen con un móvil. Esto, además del costo de la infraestructura, requiere que el usuario lleve consigo dispositivos conectados permanentemente. Es decir, para hacer la localización de la persona, se requiere que el dispositivo móvil esté encendido, conectado a la red y que el usuario decida llevarlo consigo [4].

Por su parte, los sistemas de grabación de audio y video suponen un problema para la intimidad de los ocupantes debido a su carácter invasivo. Por ejemplo, las cámaras y micrófonos en entornos de oficina pueden influir en el comportamiento de los ocupantes [1], [2], [4].

En cuanto a los sistemas con dispositivos inerciales, se pueden clasificar en dos: invasivos y no invasivos. Los invasivos utilizan acelerómetros y giroscopios colocados en el cuerpo de la persona [5], por lo que se tienen los mismos inconvenientes que los mencionados en los sistemas RF o WiFi. Los no invasivos, utilizan acelerómetros o sensores de vibración colocados sobre el piso de la habitación para medir las vibraciones producidas por las pisadas al caminar [1], [6], [7], [8], [9], [10]. Debido a las características del piso, en estos sistemas es necesario modificar los métodos tradicionales de localización de objetivos (TOA, TDOA, RSS, AOA).

En este trabajo se estudia la localización de objetivos con sistemas inerciales no invasivos, ya que el usuario no necesita llevar consigo algún dispositivo y la infraestructura requerida puede ser de bajo costo.

1.2.1 Acelerómetros MEMS

Un acelerómetro es un dispositivo microelectromecánico (MEMS) que transducen la aceleración en una señal eléctrica, detectando el desplazamiento de una masa respecto a unos electrodos fijos. La unidad de medida común utilizada para describir las características de los acelerómetros es g, que corresponde a la aceleración de caída libre debida al campo gravitatorio de la Tierra a nivel del mar, es decir, $1g = 9.8\frac{m}{s^2}$ [11].

Los acelerómetros MEMS más comunes se basan en la transducción capacitiva. La estructura de un acelerómetro MEMS de un solo eje está formada por una masa de silicio suspendida por cantilévers de silicio deformables (resortes). En presencia de una aceleración externa, la masa se ve afectada por una fuerza que deforma los resortes. El desplazamiento de la masa móvil desde su posición de reposo se traduce en un cambio capacitivo, utilizando electrodos específicos para esta medición [11]. La Figura 1-4 muestra un esquemático de un acelerómetro MEMS capacitivo.

Por otra parte, existen los acelerómetros piezoeléctricos, que funcionan basándose en el principio de la piezoelectricidad, que es la capacidad de ciertos materiales cristalinos para generar una carga eléctrica cuando se deforman [12]. Un acelerómetro piezoeléctrico se compone básicamente de uno o varios cristales piezoeléctricos unidos a una masa de prueba (o masa sísmica). En presencia de una aceleración externa, la masa sísmica provoca una deformación en los cristales, generando así una señal eléctrica.

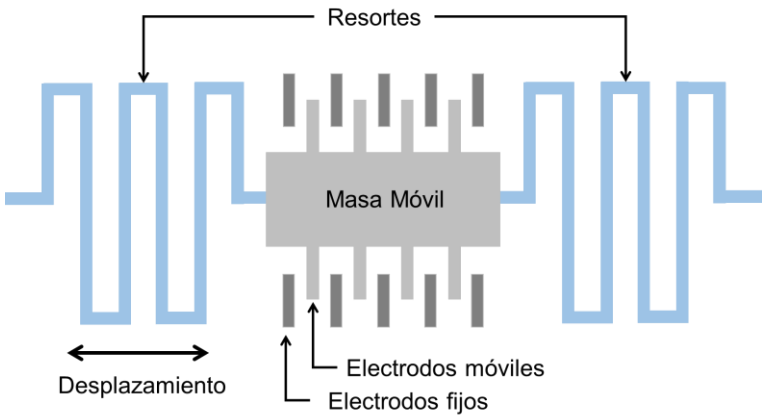


Figura 1-4. Esquemático de un acelerómetro MEMS capacitivo

1.2.2 Localización de pisadas en interiores

Anteriormente se mencionó que los métodos tradicionales (TOA, TDOA, RSS, AOA) fueron diseñados principalmente para sistemas RF o Wi-Fi, donde las señales a estudiar se propagan por el aire y se considera que la velocidad de propagación es constante.

Si bien estos métodos clásicos se han utilizado para localizar fuentes de vibraciones en sólidos [13], éstos no son una buena opción cuando los sólidos son dispersivos y amortiguados como el concreto [1], ya que en este medio las vibraciones se distorsionan a medida que viajan, dando como resultado una velocidad de propagación no constante [1], [9]. Por esta razón se han propuesto nuevos algoritmos basados en los tradicionales que superan el inconveniente de trabajar en medios dispersivos y amortiguados como el concreto. Estos nuevos métodos también tienen diferentes niveles de complejidad y requerimientos de infraestructura. A continuación, se hace un repaso al estado del arte de la localización de pisadas en interiores.

1.3 Estado del arte

1.3.1 Localización de pisadas

Al hacer una revisión de la literatura, nos encontramos que algunos de los métodos de localización de pisadas en interiores necesitan almacenar y procesar varias muestras de la señal. En [6], [7], [8] utilizan métodos basados en RSS, por lo que es necesario calcular la potencia promedio de la señal del acelerómetro dentro de una ventana de tiempo (ver Figura 1-5). En [1] se almacena la señal de aceleración y se estudian las vibraciones en una ventana de tiempo; posteriormente se hace la localización utilizando un método basado en las diferencias de tiempo de llegada (TDOA). En [10] se almacenan y procesan las señales completas provenientes de acelerómetros triaxiales (acelerómetros que miden aceleración en los tres ejes de coordenadas), para luego aplicar un método híbrido de localización que combina los métodos TDOA y AOA.

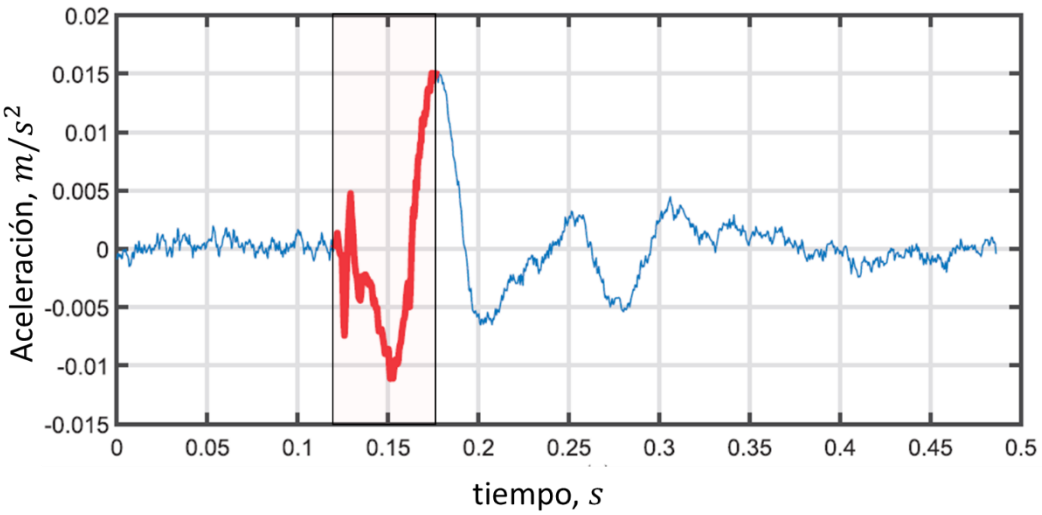


Figura 1-5. Señal de una pisada medida con un acelerómetro. Para hacer la localización se estudia la señal dentro de una ventana de tiempo. Imagen obtenida de [7]

Estos métodos reportan errores submétricos (menores a un metro) en la localización de las pisadas; sin embargo, el algoritmo SO-TDOA (Signo de

las Diferencias de los Tiempos de Llegada, del inglés “Sign Of the Time Differences Of Arrival”) propuesto en [9], puede hacer una estimación submétrica sin almacenar toda la información de la pisada, ya que sólo se requiere conocer el tiempo de llegada de la señal (TOA). Este TOA puede ser obtenido con un método de umbral (cuando la amplitud de la señal cruza un valor específico).

Como una muestra de la complejidad de este tipo de sistemas, los algoritmos de localización suelen probarse en habitaciones con pisos altamente instrumentados como los disponibles en Goodwin Hall [14]. Este es un edificio diseñado para una gran variedad de aplicaciones, incluyendo el monitoreo de la salud estructural, la dinámica del edificio y el movimiento humano. Este edificio está equipado con sensores piezoeléctricos de alta sensibilidad instalados bajo el piso y sujetados a las vigas estructurales del edificio como se muestra en la Figura 1-6.

Figura 1-6. Imagen de un acelerómetro sujetado a una viga del edificio Goodwin Hall. Imagen obtenida de [14]

Otra forma de obtener las mediciones de las vibraciones se presenta en [9] y [10], donde los sensores se colocan sobre el piso en ubicaciones distribuidas uniformemente a lo largo de la habitación. Por ejemplo, si se usan cuatro sensores, se coloca uno en cada esquina de la habitación como se muestra en la Figura 1-7. Sin embargo, en este trabajo se mostrará que, si se realiza un estudio de la ubicación de los sensores dentro de la habitación, se puede mejorar el desempeño del algoritmo de localización de pisadas.

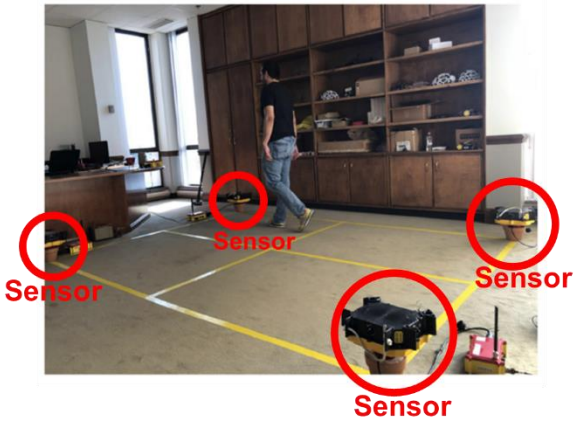


Figura 1-7. Pruebas en una habitación, donde los sensores se colocan en las esquinas del área de pruebas. Imagen obtenida de [10]

La complejidad de este estudio dependerá del algoritmo utilizado. En este trabajo se utiliza el algoritmo heurístico SO-TDOA, ya que éste sólo requiere los valores de los tiempos de llegada de la señal, disminuyéndose el almacenamiento de datos y simplificando el sistema. En el capítulo 2 se hablará con más detalle de este algoritmo.

1.3.2 Sensores

Los sensores utilizados en la literatura suelen tener un costo elevado, resultando económicamente inasequibles si se desea implementar una red de estos dispositivos. Por ejemplo, el modelo PCB352B [15] utilizado en las instalaciones de Goodwin Hall [6], [14] tiene un costo de $926 USD antes de impuestos (en el capítulo 3, se presenta una comparación de diferentes sensores). Por lo tanto, es deseable tener un bajo error de localización utilizando el menor número de sensores. Para esto, se requiere hacer un estudio en el que se obtenga el número de sensores necesarios para cada aplicación, y tomar una decisión en función de las limitaciones de espacio y presupuesto.

1.3.3 Resumen de las características de los sistemas de localización en la literatura

La Tabla 1-1 resume las características y resultados de diferentes sistemas de localización. En [1], [6], [7], [8], realizaron sus experimentos en Goodwin Hall [14], por lo que no había posibilidad de cambiar la ubicación de los sensores. Por su parte, en [9] y [10], los sensores se distribuyeron uniformemente en el piso de la habitación. Se observa que la mayoría de ellos reportan errores submétricos, sin embargo, ninguno estudia el número de sensores ni su ubicación dentro de la habitación.

Método	Sensor	No. de Sensores	Ubicación de los Sensores	Dimensiones de la habitación	No. De Pisadas	RMSE (m)	Referencia
TDOA	Acelerómetro Piezoeléctrico	12	No Optimizada*	25.5 m x 9.4 m	30	0.590	Poston, et al. [1]
RSS	Acelerómetro Piezoeléctrico	4	No Optimizada*	3 m x 1 m	13	0.253**	Alajlouni, et al. [6]
RSS	Acelerómetro Piezoeléctrico	6	No Optimizada*	13 m x 2 m	66	0.845	Alajlouni & Tarazaga [7]
RSS	Acelerómetro Piezoeléctrico	11	No Optimizada*	16 m x 2 m	162	1.020	Alajlouni & Tarazaga [8]
SO-TDOA	Acelerómetro Piezoeléctrico /Capacitivo	9	No Optimizada	3.6 m x 5.4 m	7	0.475	Bahroun, et al. [9]
ATDOA	Sismómetro	4	No Optimizada	4 m x 3 m	12	0.270**	Li, et al. [10]

Tabla 1-1. Resumen de las características y resultados de diferentes trabajos. *Realizado en Goodwin Hall. **Error promedio

1.4 Optimización

El número de sensores y su ubicación dentro de la habitación pueden ser optimizados para reducir el error de localización.

La optimización es el proceso en el que se busca el mínimo o máximo de una función. A esta función se le conoce como función objetivo. Todas las posibles soluciones al problema de optimización se encuentran dentro de un espacio acotado llamado espacio de búsqueda (ver Figura 1-8).

Un mínimo local de una función es un punto donde el valor de la función es menor o igual que los puntos cercanos dentro de su vecindario. Por otro lado, un mínimo global es un punto donde el valor de la función es menor o igual que todos los demás puntos en su dominio.

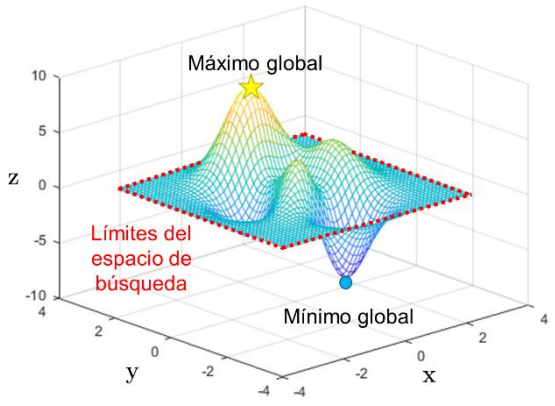


Figura 1-8. Función objetivo con un máximo y un mínimo globales

1.4.1 Clasificación de los métodos de optimización

Los métodos de optimización se pueden dividir en métodos derivativos y no derivativos [16]. Los métodos derivativos requieren conocimiento de la derivada de la función objetivo, mientras que los métodos no derivativos requieren información (evaluación) de la función objetivo y no de su derivada. Los métodos de optimización tradicionales son métodos basados en cálculo o

en búsquedas aleatorias [17]. La principal desventaja de estos métodos es que son optimizadores locales, es decir, pueden quedar atrapados en un mínimo local de la función objetivo.

Sin embargo, estos métodos pueden utilizarse como optimizadores globales mediante el uso de técnicas de inicialización “multi-start” [18]. Esto es, se realizan múltiples búsquedas locales desde diferentes puntos de inicio, lo que permite una exploración exhaustiva del espacio de búsqueda, evitando la optimación local. El mínimo local encontrado con el mejor valor de función objetivo es considerado como el óptimo global [18].

El proceso de búsqueda en los algoritmos de optimización finaliza cuando se cumple algún criterio de convergencia. Estos criterios se emplean para determinar cuándo un algoritmo de optimización ha alcanzado un resultado satisfactorio o se ha acercado lo suficiente al óptimo. Algunos criterios comunes de convergencia en algoritmos de optimización incluyen:

1. terminar el algoritmo cuando el cambio en el valor de la función objetivo f entre dos iteraciones consecutivas es pequeño

$$\left|\frac{f\big(\boldsymbol{x}(t+1)\big) - f\big(\boldsymbol{x}(t)\big)}{f\big(\boldsymbol{x}(t)\big)}\right| \leq \varepsilon$$

Donde $\boldsymbol{x}(t)$ es la solución en la iteración t y ε es un valor pequeño, por ejemplo 10^{-3}.

2. terminar el algoritmo cuando las derivadas parciales (componentes del gradiente) de la función f son pequeñas

$$\left|\frac{\partial f}{\partial x_j}\right| \leq \varepsilon, \qquad j = 1,2,\dots,n$$

Donde x_j es un elemento de la solución $\boldsymbol{x}(t)$ (es decir, $\boldsymbol{x}(t) = [x_1, x_2, \dots, x_n]$) y n es el número de elementos de la solución $\boldsymbol{x}(t)$.

3. terminar el algoritmo cuando el cambio en las soluciones entre dos iteraciones consecutivas es pequeño

$$|\boldsymbol{x}(t+1) - \boldsymbol{x}(t)| \leq \varepsilon$$

4. terminar el algoritmo cuando se alcanza cierto número de iteraciones

$$t = MaxIt$$

La elección de los criterios de convergencia depende del problema específico y del método de optimización utilizado [16].

1.4.2 Métodos modernos de optimización

Los algoritmos aproximados presentan una metodología alternativa a las técnicas de optimización tradicionales. Estos algoritmos se pueden dividir en heurísticos y metaheurísticos. 'Meta' proviene del griego y significa “nivel superior”, mientras que 'heurística' se refiere al arte de descubrir nuevas estrategias [17], [19].

Los métodos heurísticos ofrecen técnicas prácticas de resolución de problemas que producen soluciones satisfactorias, aunque no necesariamente óptimas [20]. Las metaheurísticas son heurísticas de nivel superior que buscan, generan o seleccionan heurísticas de nivel inferior (algoritmos de búsqueda parciales) capaces de proporcionar una solución suficientemente buena a un problema de optimización [17]. Si bien las metaheurísticas pueden proporcionar soluciones suficientemente buenas a un problema de optimización, no necesariamente son las óptimas [17], [21].

1.4.3 Optimización con Metaheurísticas

Los algoritmos metaheurísticos, también conocidos simplemente como metaheurísticas, poseen la capacidad de explorar extensivamente el espacio de búsqueda, lo que permite la optimización global sin quedar atrapados en mínimos locales. La capacidad de superar mínimos locales en la función

objetivo aumenta la probabilidad de encontrar el mínimo global y se le denomina robustez del método [21].

Las metaheurísticas se pueden dividir en tres clases: las que se inspiran en fenómenos físicos, las que se inspiran en los procesos evolutivos y las que se inspiran en el comportamiento de poblaciones o enjambres [22]. Como estas metaheurísticas se inspiran en la naturaleza, por lo general se les conoce como metaheurísticas bio-inspiradas. En este trabajo nos centramos únicamente en las metaheurísticas bio-inspiradas basadas en poblaciones o enjambres.

1.4.4 Inteligencia de Enjambre (Swarm Intelligence)

La inteligencia de enjambre es una inteligencia colectiva de grupos de individuos. Cada individuo tiene su propio comportamiento, sin embargo, al tenerse una interacción entre individuos, se presenta un comportamiento grupal, es decir, los individuos se ven influenciados por el resto del enjambre. Además, en los enjambres existe una memoria colectiva. La existencia de esta memoria hace que el comportamiento futuro del grupo esté influenciado por su comportamiento histórico [17].

Algunos ejemplos de enjambres son las parvadas de aves, las manadas de lobos, las colonias de abejas o de hormigas, entre otros. En los enjambres, los individuos cooperan para conseguir una meta en común, ya sea encontrar comida, migrar, reproducirse, etc.

Dentro de la inteligencia artificial existe un campo llamado “inteligencia computacional”, donde el comportamiento de los enjambres sirve como inspiración para crear algoritmos de optimización.

En estos algoritmos, una función se optimiza evaluando individuos, donde cada individuo representa una posible solución al problema de optimización [23].

Estos algoritmos son iterativos, comienzan con soluciones aleatorias y, con cada iteración, los individuos se mueven dentro del espacio de búsqueda tratando de alcanzar el mínimo de la función objetivo. Los individuos comparten y guardan información sobre el espacio de búsqueda, ayudándose mutuamente para evitar mínimos locales y utilizando la memoria para guardar la mejor solución obtenida hasta el momento. La dirección y magnitud de los desplazamientos de los individuos dependen del algoritmo utilizado.

Estos algoritmos tienen ciertos elementos de aleatoriedad por lo que se dice que son estocásticos. Con esto, se evita que los individuos queden atrapados en mínimos locales.

Cuando se proponen nuevos algoritmos de optimización, se suelen comparar con otros algoritmos para demostrar su efectividad optimizando algunas funciones específicas. En este trabajo, no buscamos desarrollar un nuevo algoritmo de optimización; sin embargo, se realiza una comparación de algunas metaheurísticas bio-inspiradas aplicadas al algoritmo SO-TDOA para seleccionar aquella que optimice mejor la ubicación de los sensores dentro de una habitación.

1.5 Conclusiones del capítulo

En este capítulo se presentaron diferentes métodos para localizar objetivos. Los tradicionales se basan en sistemas R.F., donde la velocidad de propagación se considera constante. Se pueden aplicar estos métodos en interiores, sin embargo, son invasivos y dependen de que el usuario decida llevar consigo un dispositivo para su localización.

Se vio que los métodos no invasivos incluyen sistemas que utilizan dispositivos inerciales para medir las vibraciones del piso producidas al caminar. Estos sistemas utilizan dispositivos costosos y, en la literatura, no se

presenta un estudio del número óptimo de sensores ni de su ubicación dentro de la habitación para minimizar el error de localización de pisadas.

De igual forma, se vio que, para hacer la localización, generalmente se analiza la señal completa producida por las pisadas al caminar. Sin embargo, se mencionó la existencia del algoritmo SO-TDOA que, por sus características, permite estimar la ubicación de las pisadas de una persona sin la necesidad de almacenar y procesar toda la información de la pisada.

Por lo anterior, en el siguiente capítulo se estudia a detalle este algoritmo. Posteriormente se propone una red de sensores con la cual se obtendrán las vibraciones en un piso de concreto producidas por las pisadas de una persona. Estas vibraciones serán utilizadas para determinar su ubicación dentro de una habitación. De igual forma, se hará un estudio del número de sensores a utilizar y se optimizará su ubicación dentro de la habitación utilizando metaheurísticas bio-inspiradas para minimizar el error de localización de las pisadas.

2 ASPECTOS TEÓRICOS

Para proponer una red de sensores y optimizar la cantidad y ubicación de estos, es necesario definir algunos conceptos y dar un breve repaso al algoritmo heurístico de localización de pisadas SO-TDOA propuesto en [9], así como a algunas metaheurísticas bio-inspiradas.

2.1 Algoritmo heurístico SO-TDOA [9]

Cuando se realiza una pisada, se generan ondas mecánicas longitudinales y transversales con diferentes componentes de frecuencia que viajan a través del piso. Las ondas viajeras transversales a la dirección de propagación (fuera del plano o flexurales) tienen componentes de baja frecuencia (entre 1-75 Hz [8]) y son las que dominan las vibraciones en el piso, ya que contienen la mayor parte de la energía de la onda generada en comparación con las ondas longitudinales [6], [9], [13].

Si ese paquete de ondas viajara sin distorsionarse, se podría medir la misma forma de onda en cualquier punto del medio sin importar qué tan lejanos o cercanos nos encontremos de la fuente de vibración.

Sin embargo, en un medio amortiguado y dispersivo como lo es el concreto, estas ondas se ven deformadas a medida que viajan. La deformación es debida a la atenuación y a la dispersión de la onda.

La dispersión es la no uniformidad de las velocidades de propagación de las diferentes componentes de frecuencia de una onda, es decir, no todas

las componentes de frecuencia de la onda viajan a la misma velocidad [6], [24], [25].

Además, en trabajos como en [9], se muestra que la velocidad de propagación de la onda también varía en función de la distancia que ésta recorre, es decir, la dispersión distorsiona la forma del paquete de ondas a medida que viaja y se aleja de la fuente de vibración.

Con esto en mente, en [9] propusieron el método de localización de pisadas SO-TDOA que considera la no uniformidad de la velocidad de propagación de la onda en el piso.

El algoritmo SO-TDOA se basa en el principio de que la señal llega primero a los sensores más cercanos a la fuente de vibración, incluso en un medio dispersivo y amortiguado como el concreto. Lo anterior se puede expresar con la siguiente ecuación

$$sign(t_{si} - t_{sj}) = sign(d_{si} - d_{sj}) \quad (2\text{-}1)$$

Donde $sign(\cdot)$ es el operador o función signo; (i,j) son un par de sensores; d_{si} es la distancia entre la ubicación del objetivo $\boldsymbol{p}_s$ y el sensor i; t_{si} es el tiempo de llegada del objetivo al sensor i.

El tiempo de llegada se define como el tiempo en el que la vibración, producida por un impacto sobre el piso, llega desde el objetivo hasta la ubicación de un sensor y se puede obtener mediante un criterio de umbral, es decir, cuando la amplitud de la señal cruza un valor específico (ver Figura 2-1).

El algoritmo SO-TDOA se divide en dos etapas: La etapa de segmentación y la etapa de medición y localización.

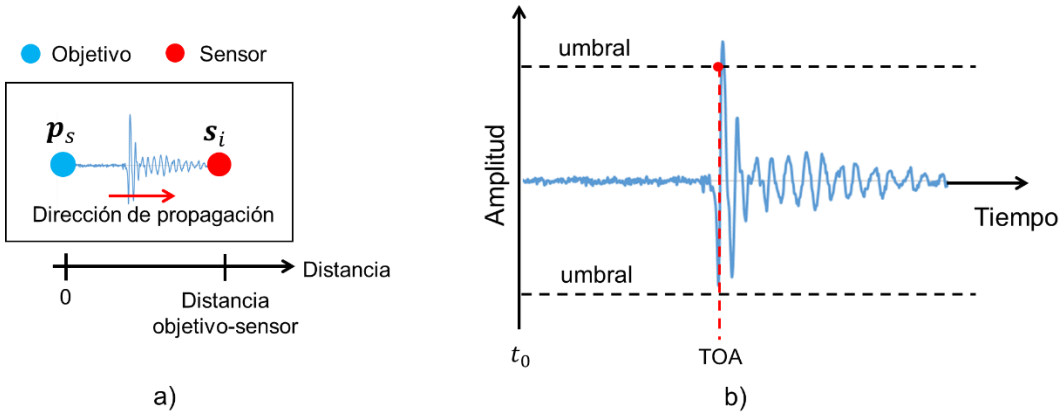


Figura 2-1. a) La vibración se propaga desde el objetivo ubicado en $\boldsymbol{p}_s$ hasta el sensor i ubicado en $\boldsymbol{s}_i$. b) El tiempo de llegada se obtiene usando un método de umbral y se mide a partir de un tiempo arbitrario t_0

2.1.1 Etapa de segmentación del piso

Primero, se selecciona la superficie de prueba y se colocan los sensores en ubicaciones conocidas. A continuación, se divide la superficie en Q número de regiones, donde cada región (R_k) está formada por bisectrices perpendiculares a la línea que une a cada par de sensores (ver Figura 2-2a). Cada región tendrá un vector característico $\boldsymbol{z}_k$ cuyos elementos toman valores +1 o -1, tal como se muestra en la ecuación (2-2). Donde cada elemento del vector $\boldsymbol{z}_k$ está dado por el signo de las diferencias de distancia entre un par de sensores (i, j) y una región R_k.

$$\boldsymbol{z}_k(l) = sign(d_{ki} - d_{kj}); \quad l = \frac{(j-2)(j-1)}{2} + i \qquad (2\text{-}2)$$

$$\forall\ (i,j) \in \{(1,2),(1,3),(2,3),(1,4),\ldots,(N-1,N)\}$$

Donde d_{ki} es la distancia entre la región R_k y el sensor i, d_{kj} es la distancia entre la región R_k y el sensor j y N es el número de sensores.

Finalmente, se calcula el centroide $\boldsymbol{p}_k^c$ de cada región R_k (ver Figura 2-2b). Esta etapa de segmentación se hace sólo una vez, después de fijar los sensores en el piso.

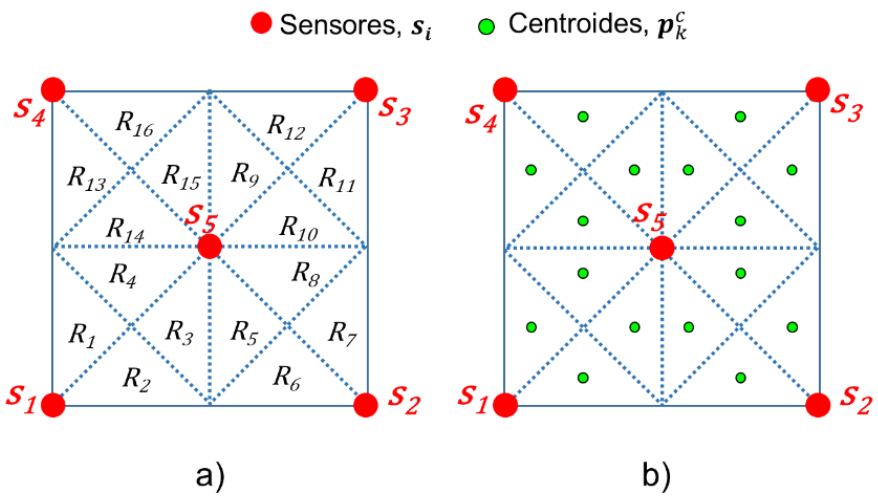


Figura 2-2. Segmentación de la habitación. a) Habitación dividida en regiones R_k, b) Centroides $\boldsymbol{p}_k^c$ de las regiones. Las líneas punteadas son las bisectrices que forman las regiones (estas líneas son imaginarias)

2.1.2 Etapa de medición y localización

Posterior a la selección de la superficie y colocación de los sensores, se realiza una pisada sobre el piso y se estima el tiempo de llegada de la vibración ($\hat{t}_{si}$) desde el objetivo hasta cada sensor i. Como anteriormente se mencionó, el tiempo de llegada se estima con un método de umbral, tomando como origen un tiempo arbitrario t_0 (ver Figura 2-1).

El umbral se define en función de la amplitud de la señal detectada por los nodos. Para ello, es necesario realizar caminatas alrededor de la habitación y medir la amplitud de las vibraciones generadas por las pisadas. Esto permitirá definir el umbral adecuado según las amplitudes registradas.

Después, se calcula el vector característico del objetivo ($\boldsymbol{z}_s$), el cual está dado por el signo de las diferencias de los tiempos de llegada medidos. Los elementos de $\boldsymbol{z}_s$ toman valores +1 o -1, tal como se describe en la ecuación (2-3).

$$\boldsymbol{z}_s(l) = sign\left(\hat{t}_{si} - \hat{t}_{sj}\right); \quad l = \frac{(j-2)(j-1)}{2} + i \tag{2-3}$$

A continuación, se calcula el conjunto de regiones M_r que minimizan la distancia de Hamming al vector característico $\boldsymbol{z}_s$.

$$M_r = arg \min_{k \in [1 \dots Q]} \sum_{a=1}^{N(N-1)/2} \left(\boldsymbol{z}_s(a) \oplus \boldsymbol{z}_k(a) \right); \quad M_r \subset [1 \dots Q] \tag{2-4}$$

donde $\oplus$ es el operador OR exclusivo y los elementos $\boldsymbol{z}_s(a)$ y $\boldsymbol{z}_k(a)$ representan los a-ésimos elementos de los vectores $\boldsymbol{z}_s$ y $\boldsymbol{z}_k$, respectivamente.

Finalmente, se obtiene la estimación de la ubicación de la pisada $\widehat{\boldsymbol{p}}_s$, promediando los centroides de las regiones que minimizaron la distancia de Hamming al vector característico del objetivo.

$$\widehat{\boldsymbol{p}}_s = \frac{1}{|M_r|} \sum_{r \in M_r} \boldsymbol{p}_r^c \tag{2-5}$$

Donde $|M_r|$ es la cardinalidad del conjunto M_r.

Es decir, en ausencia de errores de medición, la estimación de la ubicación de una pisada $\widehat{\boldsymbol{p}}_s$ está dada por el centroide de la región $\boldsymbol{p}_k^c$ cuyo vector característico $\boldsymbol{z}_k$ minimiza la distancia de Hammimg al vector $\boldsymbol{z}_s$.

Por ejemplo, en la Figura 2-3a, la pisada $\boldsymbol{p}_s$ se encuentra dentro de la región R_1, entonces la ubicación estimada será el centroide $\boldsymbol{p}_1^c$ como se muestra en la Figura 2-3b. Es decir, en este algoritmo, el error de localización de las pisadas depende del número y la forma de las regiones, por lo tanto, el error de estimación está directamente relacionado con el número y la ubicación de los sensores. Por esta razón, se buscará optimizar estos valores para reducir el error de localización de pisadas. A continuación, se presentan con detalle algunos métodos de optimización bio-inspirados.

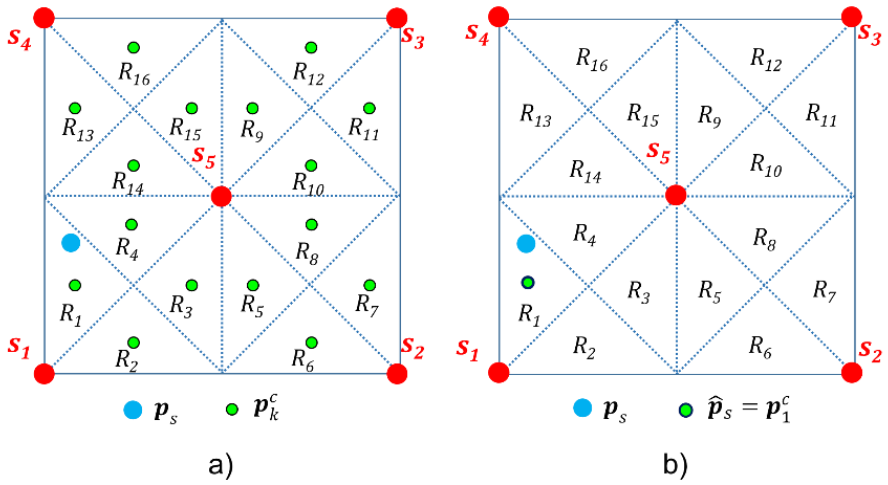


Figura 2-3. a) Una pisada $\boldsymbol{p}_s$ en una habitación con 16 regiones. b) La ubicación estimada de la pisada está en el centroide de la región donde ocurrió la pisada

2.2 Metaheurísticas bio-inspiradas basadas en poblaciones

En el capítulo 1 se presentó una breve introducción a la optimización utilizando algoritmos metaheurísticos bio-inspirados, también conocidos simplemente como metaheurísticas. A continuación, se describen en detalle estas técnicas, con el fin de emplearlas en el capítulo 3 para la optimización de la red de sensores.

Las metaheurísticas se pueden dividir en dos grandes clases: aquellas basadas en una única solución ("*single-solution-based metaheuristics*") y las basadas en poblaciones ("*population-based metaheuristics*"). Se considera que las primeras están más orientadas a la explotación, mientras que las segundas están más orientadas a la exploración [17].

"Exploración" se refiere a la habilidad del algoritmo para buscar nuevas soluciones alejadas de la solución actual, es decir, la exploración es una búsqueda global. "Explotación", por su parte, se refiere a la habilidad del algoritmo para buscar la mejor solución cercana a la solución actual, es decir, la explotación es una búsqueda local [26]. Un buen algoritmo de optimización debe balancear adecuadamente la exploración y la explotación [26], [27].

Como se mencionó en la sección 1.4.4, las metaheurísticas bio-inspiradas basadas en poblaciones son algoritmos estocásticos que se inspiran en la naturaleza. Con estos algoritmos se optimiza una función evaluando individuos, donde cada individuo representa una solución potencial [23].

Estos algoritmos son iterativos, se inician con soluciones aleatorias y en cada iteración los individuos se desplazan dentro del espacio de búsqueda para tratar de encontrar el mínimo de la función objetivo.

Los desplazamientos de los individuos dependerán del algoritmo utilizado. En cada iteración, se evalúa la posición de los individuos en la

función a optimizar. A esta evaluación se le conoce como *fitness* y sirve para saber si los individuos se están aproximando al mínimo de la función.

Por ejemplo, la Figura 2-4 ilustra cómo una población de individuos encuentra el mínimo de la función “Peaks” [28], la cual está definida como:

$$z = 3(1-x)^2 e^{-x^2-(y+1)^2} - 10\left(\frac{x}{5} - x^3 - y^5\right)e^{-x^2-y^2} - \frac{1}{3}e^{-(x+1)^2-y^2}$$

Cada individuo i es una posible solución (x_i, y_i) y tiene un *fitness* z_i.

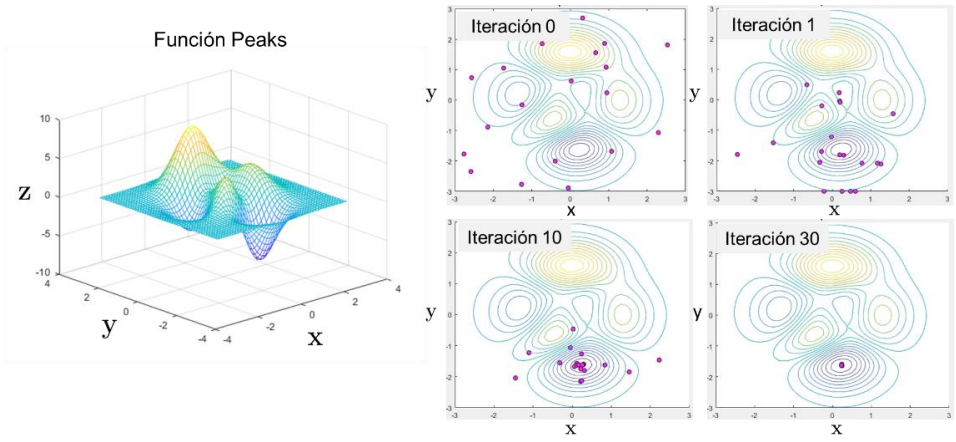


Figura 2-4. Individuos encontrando el mínimo de la función "Peaks". Izquierda: Gráfica 3D de la función. Derecha: Gráfica de contorno y ubicación de los individuos (puntos morados) en diferentes iteraciones

Debido a que existe una gran variedad de problemas (funciones objetivo) con diferentes características (condiciones y restricciones), no existe un algoritmo que optimice universalmente mejor que los demás [29]. Es por esto que se han desarrollado una gran variedad de metaheurísticas bio-inspiradas. Entre las más populares se encuentran el Particle Swarm Optimization (PSO) [30], Artificial Bee Colony (ABC) [31], Gray Wolf Optimizer (GWO) [22] y combinaciones de ellas para aprovechar las fortalezas de cada método. Por ejemplo, el algoritmo Hybrid PSO-GWO (HPSGWO), propuesto en [27], aprovecha la exploración (búsqueda global) del GWO y la explotación (búsqueda local) del PSO. A continuación, se realiza un repaso de estos algoritmos.

2.2.1 Optimización por Enjambre de Partículas (*Particle Swarm Optimization,* PSO) [30]

Este algoritmo está inspirado en el comportamiento social y la dinámica de movimiento de las parvadas de aves y bancos de peces. En este algoritmo a los individuos se les conoce como partículas y cada una de ellas representa una posible solución.

El PSO encuentra la mejor solución global ajustando el desplazamiento de cada partícula en función de dos aspectos: su mejor posición personal (aspecto cognitivo) y la mejor posición global de todas las partículas del enjambre (aspecto social).

Cada partícula i tiene su propia trayectoria, esto es, tiene una posición $\boldsymbol{x}_i$ y una velocidad $\boldsymbol{v}_i$, y se mueve en el espacio de búsqueda actualizando sucesivamente su trayectoria de acuerdo con la siguiente ecuación

$$\boldsymbol{v}_i(t+1) = \boldsymbol{v}_i(t) + cr_1[\boldsymbol{x}_i^*(t) - \boldsymbol{x}_i(t)] + cr_2[\boldsymbol{x}^g(t) - \boldsymbol{x}_i(t)]$$

$$x_i(t+1) = x_i(t) + v_i(t+1), i = 1, 2, \dots, M$$

Donde t es la iteración actual; M es el número de partículas del enjambre; c, conocido como constante de aceleración, tiene valor mayor que cero ($c > 0$); r_1 y r_2 son números aleatorios distribuidos de manera uniforme dentro de $[0, 1]$ (cualquier valor dentro del intervalo $[0, 1]$ tiene la misma probabilidad de ser seleccionado); $\boldsymbol{x}_i^*$ es la mejor solución de la partícula hasta el momento; $\boldsymbol{x}^g$ es la mejor solución global. El algoritmo termina su búsqueda cuando se cumple un criterio de parada, el cual puede ser un número máximo de iteraciones, $t = MaxIt$.

Existen diferentes variantes del algoritmo original del PSO, las cuales buscan mejorar su comportamiento. Por ejemplo, se puede agregar un parámetro llamado peso de inercia w_{damp} para modificar la velocidad de las partículas, y se utilizan dos constantes de aceleración, c_1 y c_2, que le dan una ponderación distinta a los movimientos individuales y sociales de las

partículas, los cuales reciben el nombre de parámetro cognitivo y social, respectivamente.

$$\boldsymbol{v}_i(t+1) = w_{damp}\boldsymbol{v}_i(t) + c_1 r_1[\boldsymbol{x}_i^*(t) - \boldsymbol{x}_i(t)] + c_2 r_2[\boldsymbol{x}^g(t) - \boldsymbol{x}_i(t)]$$

$$\boldsymbol{x}_i(t+1) = \boldsymbol{x}_i(t) + \boldsymbol{v}_i(t+1), i = 1, 2, \dots, M$$

Tanto w_{damp} como c_1 y c_2 han sido estudiados en [32], [33] para encontrar sus valores óptimos. Algunos valores típicos reportados son $c_1 = c_2 = 2$ y $w_{damp} = 0.7$.

Sin embargo, en [32] se encontró que variando w_{damp} se tienen mejores resultados. Por ejemplo, se puede variar en cada iteración como: $w_{damp} = \frac{MaxIt - t}{MaxIt}$. Donde $MaxIt$ es el número máximo de iteraciones.

Pseudocódigo del Algoritmo PSO [17]:

1. Iniciar $t = 1$
   Inicializar cada partícula de la población en posiciones $\boldsymbol{x}_i$ y velocidades $\boldsymbol{v}_i$ aleatorias, para $i = 1, 2, \dots, M$.
2. **Repetir hasta que** se cumpla $t = MaxIt$:
   a. Calcular el valor *fitness* para cada partícula i
      Si el *fitness* de cada partícula i es mejor que su mejor solución, entonces actualizar $\boldsymbol{x}_i^*(t)$
   b. Determinar la ubicación de la partícula con el mejor *fitness* y actualizar $\boldsymbol{x}^g(t)$ si es necesario
   c. **Para cada** partícula i:
      Calcular su velocidad $\boldsymbol{v}_i(t+1)$
      Fin del ciclo
   d. Actualizar la posición x_i de cada partícula i
   e. Aumentar $t = t + 1$

   Fin del ciclo principal

2.2.2 Algoritmo de Colonia de Abejas Artificiales (*Artificial Bee Colony*, ABC) [31]

Este algoritmo se inspira en la danza que realiza un enjambre de abejas durante su proceso de búsqueda de alimento. En la colonia artificial, las abejas se dividen en dos grandes clases: obreras (*employed bees*) y desempleadas

(*unemployed bees)*. Estas últimas se dividen a su vez en abejas vigías (*onlooker bees*) y exploradoras (*scouts*).

En el algoritmo ABC, las fuentes de alimento representan posibles soluciones y la cantidad de néctar en una fuente refleja la calidad de cada solución (fitness). Cada fuente está asociada a una abeja obrera, y un número igual de abejas vigías explora estas fuentes para evaluar su calidad. Tanto las abejas vigías como las obreras buscan mejores fuentes de alimento moviéndose aleatoriamente hacia otra fuente dentro de su vecindario[17].

Este algoritmo consta de tres fases: fase de obreras, fase de vigías y fase de exploradoras.

Primero, se envían las abejas obreras a fuentes de alimento aleatorias $\boldsymbol{x}_i$ y comienza la fase de obreras. En esta fase, las obreras buscan en su vecindario una nueva fuente de comida con mayor néctar, es decir, realizan una búsqueda local y se quedan en la fuente que tiene mejor *fitness* (se aplica una selección voraz o *greedy selection* entre ambas fuentes). Luego, las obreras pasan la información de su fuente de comida a las abejas vigías.

La fase de vigías comienza cuando todas las obreras han compartido la información de su fuente de alimento. En esta fase, las vigías seleccionan una fuente de alimento i con una probabilidad P_i dada por $P_i = f_i / \sum_{j=1}^{M} f_i$

Donde f_i es el *fitness* de la fuente de aliento i y M es el número total de fuentes de alimento. Este *fitness* se suele determinar como:

$$f_i = \begin{cases} 1/(1 + f(\boldsymbol{x}_i)), & si\ \ f(\boldsymbol{x}_i) \geq 0 \\ 1 + |f(\boldsymbol{x}_i)|, & si\ \ f(\boldsymbol{x}_i) < 0 \end{cases}$$

Donde $f(\boldsymbol{x}_i)$ es la función objetivo evaluada en la ubicación de la fuente de alimento $\boldsymbol{x}_i$.

Una vez seleccionada la fuente de alimento, las abejas vigía hacen una búsqueda local similar a la de las obreras, buscando una mejor fuente de alimento en su vecindario y eligiendo la que tenga mejor *fitness* (*greedy*

selection). Si el néctar disminuye o se agota después de varios intentos (límite de abandono de búsqueda local, *limit*), indicando un mínimo local, entonces, las abejas obreras de esas fuentes se convierten en exploradoras.

En la fase de exploradoras, las exploradoras buscan aleatoriamente una nueva fuente de comida $\boldsymbol{x}_i$. Después, vuelven a ser obreras.

Las tres fases se repiten hasta que se cumpla un criterio de parada, como un límite máximo de iteraciones $t = MaxIt$ o un cambio pequeño en el valor de la función entre iteraciones consecutivas $\left|\frac{f(\boldsymbol{x}(t+1))-f(\boldsymbol{x}(t))}{f(\boldsymbol{x}(t))}\right| \leq \varepsilon$ (por ejemplo $\varepsilon = 10^{-3}$). Después de cada iteración, se registra la mejor solución, lo que facilita el abandono de las fuentes que son inicialmente pobres o que se han empobrecido debido a la explotación.

Pseudocódigo del Algoritmo ABC [31]:

1. Inicializar los parámetros (número de abejas, número de intentos y número máximo de iteraciones).
2. Generar fuentes de alimento distribuidas aleatoriamente en el espacio de búsqueda y evaluar su néctar (*fitness*).
3. Enviar a las obreras a las fuentes de alimento.
4. **Repetir hasta que** se cumpla un criterio de parada:
    - **a.** **Para** cada abeja obrera:
        Encontrar una nueva fuente de alimento en su vecindario y evaluar su *fitness*.
        Quedarse en la fuente con mejor *fitness*.
        Fin de la fase de obreras
    - **b.** Calcular la probabilidad P_i para cada fuente de alimento.
    - **c.** **Para** cada abeja vigía:
        Para cada fuente de alimento i:
        if ($rand() < P_i$)
        Enviar a la abeja vigía a la fuente de alimento de la i-ésima obrera.
        Encontrar una nueva fuente de alimento en su vecindario y evaluar su *fitness*.
        Quedarse en la fuente con mejor *fitness*.
        Fin del if
        Fin del ciclo
        Fin de la fase de vigías
    - **d.** **Si** alguna obrera se volvió exploradora:
        Enviarla a alguna nueva fuente de alimento aleatoria.
        Fin de la fase de exploradoras

Fin del ciclo principal

2.2.3 Optimizador de Lobo Gris (*Gray Wolf Optimizer*, GWO) [22]

Este algoritmo se inspira en el método de caza de las manadas de lobos grises, donde cada lobo tiene un rol y una jerarquía dentro del grupo. Los alfa (α) (hembra y macho) son los líderes y todos se someten ante ellos, no necesariamente son los más fuertes, pero sí los más capaces para dirigir a la manada. Luego están los beta (β), cuyo rol es ayudar a dirigir y controlar a la manada, y en caso de muerte o envejecimiento de los alfa, estos toman su lugar. En el siguiente nivel jerárquico se encuentran los delta (δ), estos son los exploradores, centinelas, ancianos, cazadores y cuidadores de la manada. En el último nivel se encuentran los omegas (ω), los cuales son sometidos por todos los lobos de niveles jerárquicos superiores.

Las principales etapas de la caza de los lobos grises son las siguientes [22]:

1. Rastrear, seguir y acercarse a la presa
2. Perseguir, rodear y acosar a la presa hasta que deje de moverse
3. Atacar a la presa

En el algoritmo GWO cada lobo representa una posible solución al problema de optimización y la presa simboliza el mínimo (o máximo) que se desea encontrar en el espacio de búsqueda. En este algoritmo se supone que sólo existe un alfa (α), un beta (β) y un delta (δ), las demás posibles soluciones son consideradas como los lobos omegas (ω).

Para simular el proceso de la caza de los lobos, se hace la suposición de que los lobos alfa, beta y delta son los más cercanos al mínimo, así que, para determinar la jerarquía de cada lobo, se evalúan en la función a optimizar. Así, el alfa (α) es la mejor solución, el beta (β) es la segunda mejor solución y el delta (δ) es la tercera mejor solución. El resto de las soluciones son omegas (ω).

Entonces, se almacenan las tres mejores soluciones ($\boldsymbol{x}_\alpha$, $\boldsymbol{x}_\beta$, $\boldsymbol{x}_\delta$) y el resto de los lobos (soluciones $\boldsymbol{x}_i$) actualizan su posición de acuerdo con las mejores tres de la siguiente forma:

$$\boldsymbol{x}_i(t+1) = \frac{\boldsymbol{x}_1 + \boldsymbol{x}_2 + \boldsymbol{x}_3}{3}$$

Donde t es la iteración actual,

$$\boldsymbol{x}_1 = \boldsymbol{x}_\alpha - \boldsymbol{A}_1 \cdot \boldsymbol{D}_\alpha; \quad \boldsymbol{x}_2 = \boldsymbol{x}_\beta - \boldsymbol{A}_2 \cdot \boldsymbol{D}_\beta; \quad \boldsymbol{x}_3 = \boldsymbol{x}_\delta - \boldsymbol{A}_3 \cdot \boldsymbol{D}_\delta$$

$$\boldsymbol{D}_\alpha = |\boldsymbol{C}_1 \cdot \boldsymbol{x}_\alpha - \boldsymbol{x}_i|; \quad \boldsymbol{D}_\beta = \left|\boldsymbol{C}_2 \cdot \boldsymbol{x}_\beta - \boldsymbol{x}_i\right|; \quad \boldsymbol{D}_\delta = |\boldsymbol{C}_3 \cdot \boldsymbol{x}_\delta - \boldsymbol{x}_i|$$

Los vectores $\boldsymbol{A}$ y $\boldsymbol{C}$ son vectores de coeficientes que determinan el movimiento de cada lobo omega y se calculan de la siguiente forma:

$$\boldsymbol{A} = 2\boldsymbol{a} \cdot \boldsymbol{r}_1 - \boldsymbol{a}; \qquad \boldsymbol{C} = 2 \cdot \boldsymbol{r}_2$$

Donde las componentes de $\boldsymbol{a}$ van decreciendo en cada iteración de manera lineal desde 2 hasta 0. Los vectores $\boldsymbol{r}_1$ y $\boldsymbol{r}_2$ son vectores aleatorios en $[0,1]$.

Gráficamente las ecuaciones de $\boldsymbol{x}_1$, $\boldsymbol{x}_2$ y $\boldsymbol{x}_3$ representan al lobo moviéndose aleatoriamente alrededor de la presa. Por ejemplo, supongamos que el problema de optimización es de dos dimensiones y que el lobo alfa se encuentra en la posición ($x,\ y$), entonces $\boldsymbol{x}_1$ llevaría al lobo omega a una de las nuevas posiciones que se ilustran en la Figura 2-5a, por ejemplo ($x^*,\ y^*$). Los vectores $\boldsymbol{r}_1$ y $\boldsymbol{r}_2$ hacen que la nueva posición sea aleatoria. Como la nueva posición $\boldsymbol{x}_i$ de los lobos omega (o cualquier otro lobo) se actualiza no solo con $\boldsymbol{x}_1$, sino que también con $\boldsymbol{x}_2$ y $\boldsymbol{x}_3$, entonces se tiene un movimiento aleatorio, pero influenciado por la posición de los lobos alfa, beta y delta, por lo que los lobos se moverán a una posición aleatoria alrededor de la presa como se muestra en la Figura 2-5b.

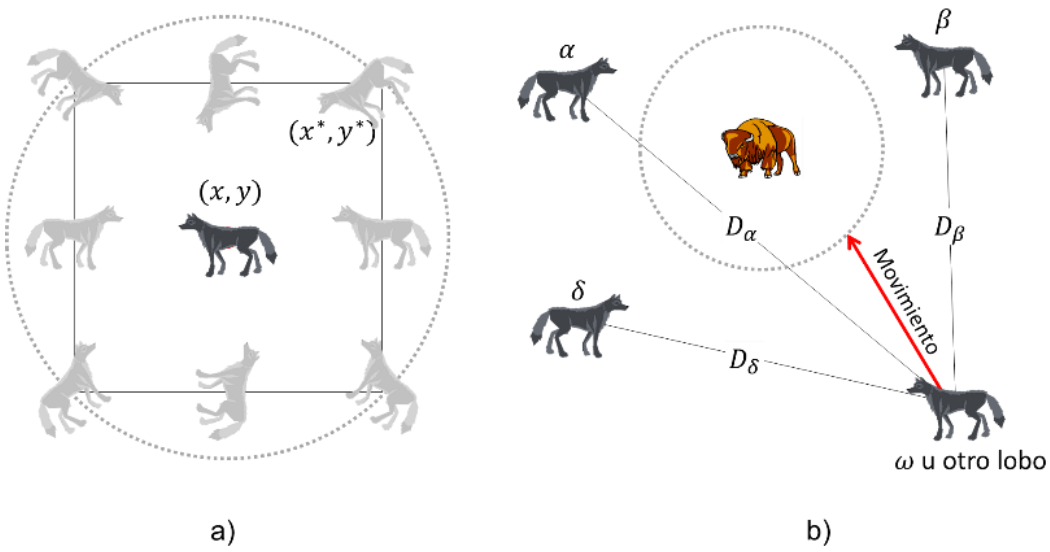


Figura 2-5. a) El lobo se puede mover a alguna de las posiciones sombreadas alrededor del alfa. b) Los lobos se mueven a una posición aleatoria alrededor de la presa

Este algoritmo equilibra la exploración y la explotación. Cuando la magnitud del vector $|\boldsymbol{A}| > 1$, el lobo se aleja de la presa (exploración), pero cuando $|\boldsymbol{A}| < 1$, se acerca a la presa (explotación). Esta dinámica ayuda al algoritmo a evitar mínimos locales, permitiéndole hacer búsquedas globales. Sin embargo, aunque $|\boldsymbol{A}|$ toma valores aleatorios en cada iteración, tiende a disminuir debido al vector $\boldsymbol{a}$, el cual decrece de manera lineal con las iteraciones. Por su parte, el vector $\boldsymbol{C}$ siempre toma valores aleatorios, favoreciendo la exploración y evitando mínimos locales incluso en las últimas iteraciones.

Finalmente, tras mover a todos los lobos, se determina cuáles son los nuevos alfa, beta y delta y el proceso se repite hasta que se cumpla una condición de parada, la cual puede ser un número máximo de iteraciones o un cambio pequeño en el valor de la función entre iteraciones consecutivas.

Pseudocódigo del Algoritmo GWO [22]:

1. Inicializar la población de lobos $\boldsymbol{x}_i$ $(i = 1, 2, \dots, M)$ en ubicaciones aleatorias.
2. Inicializar $\boldsymbol{a}$, $\boldsymbol{A}$ y $\boldsymbol{C}$.
3. Calcular el *fitness* de cada lobo y determinar su jerarquía.

   ($\boldsymbol{x}_\alpha$ es la mejor solución, $\boldsymbol{x}_\beta$ es la segunda mejor solución, $\boldsymbol{x}_\delta$ es la tercera mejor solución).

4. **Repetir mientras** $(t < MaxIt)$
   - **a.** **Para** cada lobo i:
     Actualizar su posición $\boldsymbol{x}_i$.
     Fin del ciclo
   - **b.** Actualizar $\boldsymbol{a}$, $\boldsymbol{A}$ y $\boldsymbol{C}$.
   - **c.** Calcular el *fitness* de cada lobo y actualizar las jerarquías.

   Fin del ciclo principal

5. Regresar el óptimo $\boldsymbol{x}_\alpha$

2.2.4 Algoritmo híbrido PSO-GWO (HPSGWO) [27]

Como anteriormente se mencionó, en ocasiones se hacen hibridaciones de algunas metaheurísticas para combinar las ventajas de ambas. El algoritmo HPSGWO combina los métodos PSO y GWO. Esta combinación es beneficiosa porque cuando el PSO se acerca a una buena solución, las partículas tienden hacia ella, incluso si es un mínimo local, lo que significa que la capacidad de explotación del PSO es alta en relación con su capacidad de exploración.

El método HPSGWO sustituye algunas partículas del PSO por una partícula mejorada con el GWO para reducir la posibilidad de que el algoritmo PSO quede atrapado en un mínimo local. La sustitución de partículas ocurre con una baja probabilidad ($prob$), lo que permite que el bucle principal del PSO se ejecute mientras algunas partículas entran en un bucle anidado del GWO.

Los ciclos anidados hacen que el algoritmo HPSGWO se ejecute en un tiempo mayor que el PSO y el GWO originales, por lo que el número de iteraciones ($MaxItW$), la probabilidad ($prob$) y población de lobos (N_w) del ciclo anidado del GWO deben ser pequeños [27]. Por ejemplo $N_w = 10$, $MaxItW = 10$, $prob = 0.01$.

Pseudocódigo del Algoritmo HPSGWO [27] :

1. Definir parámetros:
   $MaxIt$: Número máximo de iteraciones
   N_p: Número máximo de partículas PSO
   $prob$: Pequeña probabilidad definida por el usuario
   N_w: Número de lobos (población pequeña)
   $MaxItW$: Número pequeño de iteraciones del GWO

2. Inicializar partículas $\boldsymbol{x}_i$ $(i = 1, 2, \dots, M)$ en ubicaciones aleatorias.
3. **Repetir mientras** $(t < MaxIt)$
   a. **Para** cada partícula i:
   Ejecutar PSO
   Actualizar la velocidad y posición de la partícula i
   if $rand(0,1) < prob$ **then** *(número pequeño de probabilidad)*
   Ejecutar GWO
   Inicializar $\boldsymbol{a}$, $\boldsymbol{A}$ y $\boldsymbol{C}$.
   Repetir mientras $k < MaxItW$ (*número pequeño de iteraciones)*
   Para cada lobo *(población pequeña)*:
   Actualizar su posición del alfa, beta y delta.
   Actualizar $\boldsymbol{a}$, $\boldsymbol{A}$ y $\boldsymbol{C}$.
   Fin del ciclo
   Fin del ciclo
   Posición de la partícula i = promedio de los tres mejores lobos
   Fin del if
   Fin del ciclo
   Fin del ciclo principal

2.3 Conclusiones del capítulo

En este capítulo se presentó el algoritmo SO-TDOA para la localización de pisadas, el cual será empleado en nuestro sistema. Además, se presentaron diferentes algoritmos de optimización metaheurísticos bio-inspirados que serán empleados para optimizar el número y la ubicación de los nodos de la red en el siguiente capitulo.

Como se mostró, el algoritmo SO-TDOA permite obtener la localización de una persona a partir de la medición de los tiempos de llegada de la señal. Por lo tanto, si se diseña una red de sensores capaz de obtener los TOAs, no será necesario almacenar y procesar completas las señales de cada pisada, lo cual simplifica el sistema y reduce el almacenamiento de datos. De igual

forma, se vio que, en este algoritmo, el error de localización depende del número y la ubicación de los sensores. Por lo tanto, en los capítulos posteriores se buscará optimizar estos valores con la ayuda de algoritmos metaheurísticos bio-inspirados. Estos algoritmos de optimización son de fácil implementación ya que se utilizan como funciones, para las cuales sólo se requiere ajustar unos pocos parámetros como el número de individuos, el número de iteraciones, entre otros, dependiendo del algoritmo empleado.

3 ANÁLISIS Y DISEÑO

En el capítulo anterior se mostró que para realizar la localización de pisadas utilizando el algoritmo SO-TDOA, únicamente se necesitan conocer los tiempos de llegada de la señal, desde el objetivo hasta los sensores. Esto permite el uso de una red de sensores para detectar el cruce de umbral sin necesidad de almacenar y analizar toda la información de la pisada en una computadora, lo cual simplifica el sistema.

En este capítulo se presenta el diseño de una red inalámbrica de sensores inteligentes para obtener las vibraciones producidas por los impactos de los pies de una persona sobre un piso de concreto y con esto determinar su ubicación dentro de una habitación. En este trabajo, definimos un sensor inteligente como un dispositivo que mide las vibraciones del piso, realiza la detección del cruce de umbral y forma parte de una red de sensores.

Por otra parte, en este capítulo se presenta una comparación de las metaheurísticas previamente mencionadas, aplicadas al algoritmo SO-TDOA, para seleccionar la que mejor optimiza el número y la ubicación de los nodos de la red de sensores.

3.1 Propuesta de la red de sensores inteligentes

En este trabajo, presentamos una propuesta de red inalámbrica de sensores inteligentes para la localización de pasos, donde todos los nodos (sensores inteligentes) se comunican bidireccionalmente con un nodo central,

también llamado Coordinador (unidad de procesamiento), utilizando una topología de punto a multipunto, como se ilustra en la Figura 3-1a.

La Figura 3-1b muestra un diagrama a bloques de la red de sensores propuesta. Cada nodo consta de un acelerómetro capacitivo para medir las vibraciones y detectar los cruces de umbral; un microcontrolador para configurar y leer el sensor; y un módulo transmisor/receptor (TX/RX) inalámbrico para comunicarse con el Coordinador. El Coordinador consta de un módulo inalámbrico TX/RX y de una computadora donde se realiza el procesamiento de los datos (el equipo utilizado, así como sus especificaciones, se mencionan más adelante en esta sección).

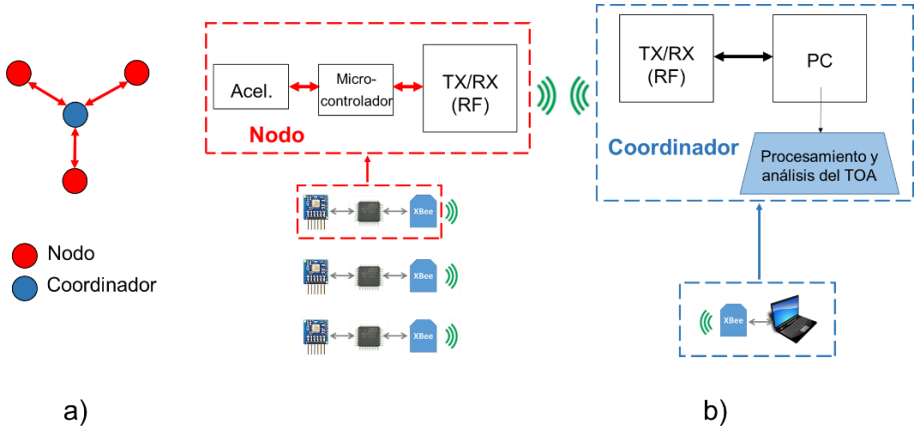


Figura 3-1. a) Topología punto a multipunto, donde el nodo central se comunica bidireccionalmente con los otros nodos de la red. b) Diagrama a bloques de la red de sensores propuesta.

La Figura 3-2 muestra los diagramas de flujo del coordinador y los nodos. El diagrama del coordinador se implementa en una computadora usando Python 3 y Matlab, mientras que el diagrama de los nodos se implementa en un microcontrolador usando lenguaje C. La primera tarea del coordinador es formar la red. A continuación, envía una bandera a todos los nodos para indicarles que comiencen la medición del tiempo de llegada (TOA). Una vez recibida la bandera, los nodos comienzan a medir el tiempo usando un contador digital y revisan continuamente si la amplitud de la vibración cruza

el umbral. Cuando un sensor detecta que la señal cruza el umbral, éste envía el tiempo de llegada al coordinador (ver Figura 3-3).

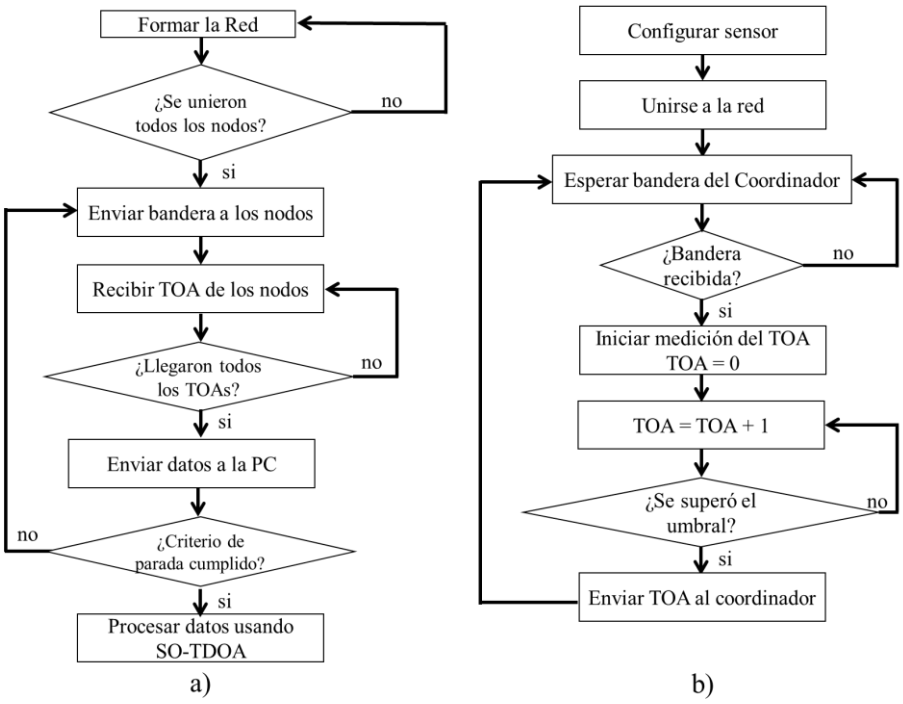


Figura 3-2. Diagramas de flujo de la red de sensores. a) Coordinador. b) Nodos

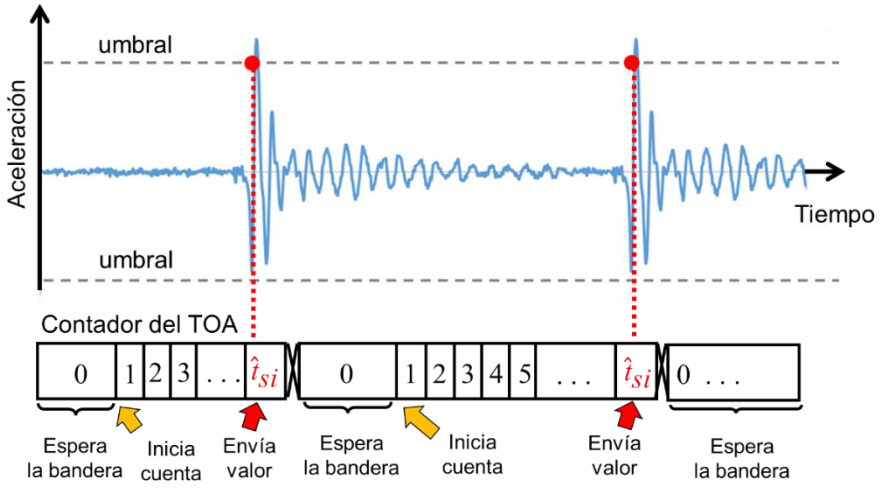


Figura 3-3. Estimación del TOA. El sensor i comienza a medir el tiempo después de recibir la bandera proveniente del coordinador. El TOA se obtiene cuando la señal cruza el umbral

Cuando el coordinador recibe los TOA de todos los nodos, los envía a una computadora para almacenarlos y posteriormente estimar la ubicación de la pisada usando la segunda etapa del algoritmo SO-TDOA. Una vez hecho lo anterior, se vuelve a enviar la bandera a los nodos y el ciclo se repite continuamente. Esta técnica permite a los nodos enviar únicamente un dato por pisada, por lo que los módulos inalámbricos pueden trabajar a una baja velocidad de transmisión/recepción de datos, lo que se traduce en un menor consumo de energía.

Es importante mencionar que esta primera propuesta del sistema no funciona en tiempo real. El sistema almacena todas las mediciones en la computadora y tras un cierto número de pisadas, estima la trayectoria formada con los impactos de los pies.

A continuación, se explica de manera detallada cada bloque del sistema mostrado en la Figura 3-1.

3.2 Acelerómetro

La red requiere sensores capaces de detectar las vibraciones en el piso producidas por los impactos de los pies, por lo que se requiere una alta sensibilidad y un bajo nivel de ruido para que la señal no se pierda con el ruido. Además, los sensores deben tener un ancho de banda capaz de captar vibraciones de baja frecuencia, como las generadas por pisadas en un piso de concreto, que están por debajo de los 500 Hz [8]. De ser posible, también deben contar con detección de cruce de umbral, así como con ajuste de offset de la señal.

La Tabla 3-1 muestra las características de los sensores piezoeléctricos PCB352B [8] utilizados en Goodwin Hall [6]. El costo de estos sensores rebasa el presupuesto asignado para este proyecto. Por esta razón, se realizó la búsqueda de sensores con características similares. La Tabla 3-1 muestra la comparación de estos acelerómetros. Se considera que el EVAL-ADXL355-PMDZ (tarjeta de evaluación que contiene el sensor ADXL355) es una buena opción para reemplazar a los utilizados en la literatura, tal como se describe a continuación.

Sensor	PCB352B	PCB356B18	ADXL327BCPZ	EVAL-ADXL355-PMDZ
Ejes de medición	X, Y, Z	X, Y, Z	X, Y, Z	X, Y, Z
Tipo de sensor	Analógico Piezoeléctrico	Analógico Piezoeléctrico	Analógico Capacitivo	Digital Capacitivo
Sensibilidad	1000 mV/g	1000 mV/g	420 mV/g	812 mV/g @ ±2g
Rango de medición	±5g	±5g	±2.5g	Seleccionable ±2g, ±4g, ±8g
Ancho de banda	10kHz	3kHz	5.5kHz	1kHz
Densidad de ruido (equivalente en gs)	$15\ \mu g/\sqrt{Hz}$ ($0.6mg$ @1kHz)	$0.4\ \mu g/\sqrt{Hz}$ ($16\mu g$ @1kHz)	$250\ \mu g/\sqrt{Hz}$ ($10mg$ @1kHz)	$25\ \mu g/\sqrt{Hz}$ ($1mg$ @1kHz)
Precio antes de impuestos (USD)	$ 926	$ 1865	$ 10.48	$ 47.13

Tabla 3-1. Comparación de acelerómetros comerciales

Si bien el ancho de banda del ADXL355 es menor que los utilizados en la literatura, se sabe que las vibraciones dominantes producidas por las pisadas se encuentran por debajo de 500Hz [8], por lo que el ancho de banda del ADXL355 (1 kHz) cumple con los requisitos para nuestra aplicación.

También, este sensor cuenta con diferentes circuitos digitales integrados que se ajustan a nuestras necesidades. Cuenta, entre otros, con un circuito de detección de cruce de umbral, un filtro pasa bajas para reducir el ruido de la señal y un circuito para modificar el offset de la misma [34].

3.2.1 Montaje del acelerómetro

En la literatura, los sensores suelen montarse de dos formas: fijándolos a las vigas de la estructura del edificio [14] o colocándolos sobre el piso de tal forma que se puedan medir las vibraciones perpendiculares al plano del piso [9], [10].

La primera opción consiste en preparar una superficie plana donde se montará el acelerómetro y luego hacer una perforación. Posteriormente, se acopla un perno al acelerómetro y se introduce en la perforación del piso (ver Figura 3-4). Esta técnica es recomendada cuando la instalación será permanente y no existan inconvenientes en perforar la superficie de prueba.

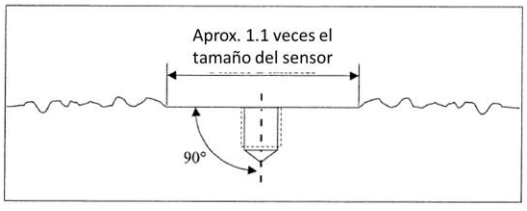


Figura 3-4. Preparación de la superficie para montar el acelerómetro (imagen obtenida de [15])

La segunda opción de montaje consiste en colocar el sensor sobre el piso y fijarlo mediante pegamento instantáneo (por ejemplo, Loctite 454 [15]) o con una estructura que evite que el sensor se mueva durante las pruebas. Estas técnicas son utilizadas para instalaciones temporales o cuando la superficie de prueba no puede ser perforada (ver Figura 3-5).

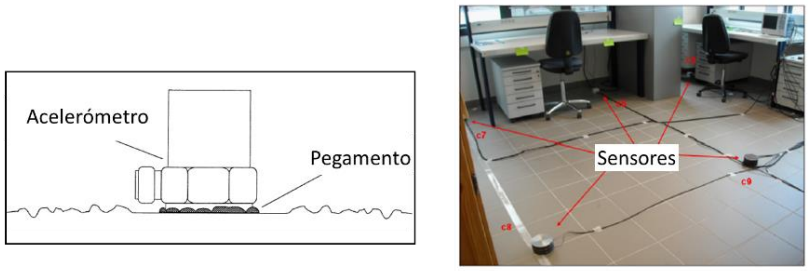


Figura 3-5. Ejemplos de montaje temporal de los sensores. a) Con pegamento instantáneo (imagen obtenida de [15]). b) Con cinta doble cara (imagen obtenida de [9])

En este proyecto, la tarjeta EVAL-ADXL355-PMDZ (de aquí en adelante llamada acelerómetro ADXL355, para simplificar) se coloca sobre el piso de tal forma que el eje z se encuentra perpendicular al plano del piso (ver Figura 3-6) y se utiliza una base fabricada con resina epóxica para evitar que se mueva durante las pruebas (ver Figura 3-7). Esta base fue fabricada para que el sensor se ajuste de manera precisa en ella y sea fácil de montar y desmontar.

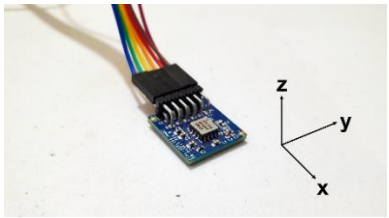


Figura 3-6. Acelerómetro ADXL355 colocado sobre el piso

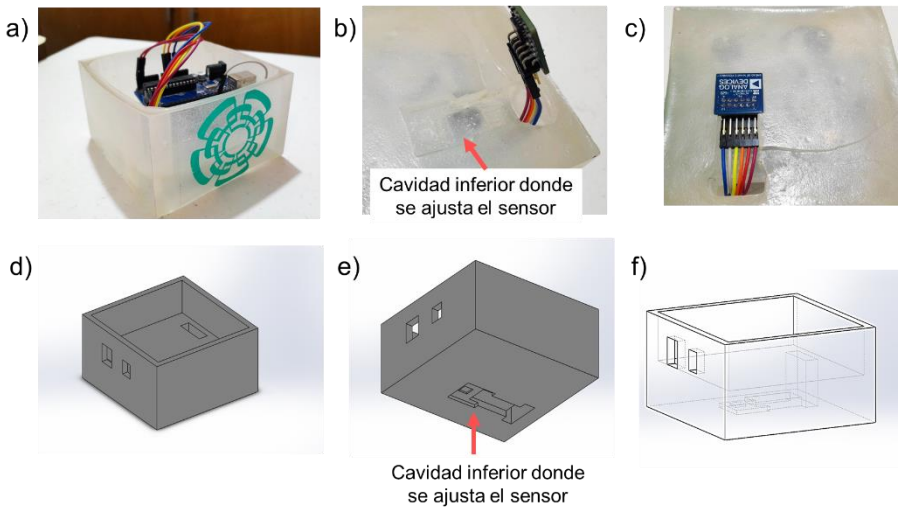


Figura 3-7. Base de resina epóxica. a) Se muestra la base montada sobre una mesa. b) Cavidad en la parte inferior con la forma del sensor. c) Vista inferior con el sensor montado d, e, f) isométricos de la base del acelerómetro, se muestran las cavidades donde se ajusta el sensor

3.2.2 Registros del acelerómetro

Los circuitos digitales del ADXL355 se configuran modificando los registros del dispositivo mediante comunicación I^2C o SPI. Cada registro consta de 8 bits y tiene una dirección única, por lo que es necesario especificar dicha dirección para configurarlos. Esta configuración se puede hacer con la ayuda de un microcontrolador, del cual se hablará en la siguiente sección.

A continuación, se mencionan las principales tareas de los registros utilizados para la detección del cruce de umbral (RESET, STATUS, POWER_CTL, RANGE, OFFSET_Z_H, OFFSET_Z_L, ACT_EN, ACT_THRESH_H, ACT_THRESH_L, INT_MAP). Para conocer los detalles completos de estos registros se puede consultar la hoja de datos del acelerómetro [34]. Por otra parte, en el Anexo B se presenta un repaso de algunos conceptos de sistemas digitales como auxiliares para esta sección.

3.2.2.1 Reset

Cuando se trabaja con circuitos digitales, es recomendable reiniciarlos antes de realizar cualquier tarea para conocer su estado inicial. El registro RESET cumple con esa función.

3.2.2.2 Lectura de la aceleración

Las mediciones de aceleración en el eje z se almacenan en 20 bits separados en tres registros (ZDATA1, ZDATA2, ZDATA3). Los datos están justificados a la izquierda y en complemento a dos. Esto es, los 8 bits más significativos (ZDATA [19:12]) se almacenan en el registro ZDATA3, los siguientes 8 (ZDATA [11:4]) en ZDATA2 y los 4 menos significativos (ZDATA [3:0]) en ZDATA1. Entonces, para conocer la aceleración medida por el acelerómetro, se deben leer estos registros.

En este trabajo, los registros ZDATA sólo son utilizados para hacer pruebas preliminares, ya que, como anteriormente se mencionó, sólo requerimos conocer el momento en el que la señal supera un umbral.

3.2.2.3 Velocidad de transferencia de datos

Para configurar al acelerómetro se utiliza comunicación I^2C. La velocidad de transferencia de datos se configura con el bit I2C_HS del registro RANGE.

Si I2C_HS = 0 se tiene el modo rápido (fast mode = 400 kHz), si I2C_HS = 1 se tiene el modo de alta velocidad (high speed mode = 3.4 MHz). La velocidad se selecciona de acuerdo con las especificaciones del microcontrolador con el que se está trabajando. En este proyecto, se utiliza el ATmega328P disponible en las placas de Arduino Nano, cuya velocidad máxima es de 400 kHz, por lo que debe seleccionarse el modo rápido.

3.2.2.4 Resolución de las mediciones

Por otra parte, el acelerómetro puede ser programado para escalas de ±2.048g, ±4.096g y ±8.192g (g = 9.81 m/s^2). Esto se hace configurando los bits Range del registro RANGE.

Al seleccionar una escala, se tendrá una resolución diferente en la medición de la aceleración. Esto es, se pueden medir cambios de aceleración de acuerdo con el bit menos significativo de ZDATA. La resolución de la medición está dada por:

$$LSB = \frac{Rango}{2^{n-1}} \tag{3-1}$$

Donde $n = 20$ es el número de bits del acelerómetro y $Rango$ es la escala seleccionada (2.048g, 4.096g o 8.192g).

La Tabla 3-2 muestra la resolución para cada escala. En este proyecto se selecciona la mayor resolución, por lo que se elige la escala de ±2.048 g.

Rango	2.048g	4.096g	8.192g
Resolución (LSB)	3.906 x 10^{-6} g	7.812 x 10^{-6} g	15.625 x 10^{-6} g

Tabla 3-2. Resolución del acelerómetro de acuerdo con la escala seleccionada

La medición en gs del acelerómetro se puede obtener como:

$$acel_g = acel \times LSB \tag{3-2}$$

Donde $acel$ es el valor en decimal de la aceleración medida por el acelerómetro. Por ejemplo, si el acelerómetro entrega una lectura de 256,000 (en binario, 0011 1110 1000 0000 0000_2), y se tiene una escala de ±2.048g, entonces, usando las ecuaciones 3-1 y 3-2, se tendrá una aceleración de 1g.

$$acel_g = 256{,}000 \times 3.906x10^{-6} = 1g$$

3.2.2.5 Offset de la señal

Al colocar el acelerómetro sobre el piso (como se especificó en la sección anterior), idealmente éste entrega una lectura en el eje z de +1g, es decir, los 20 bits del sensor entregan un valor de 256,000. En la práctica, este valor puede estar desplazado ligeramente (tiene un offset), además la señal tiene ruido, por lo que es necesario conocer el valor promedio de la señal para quitar el offset (ver Figura 3-8).

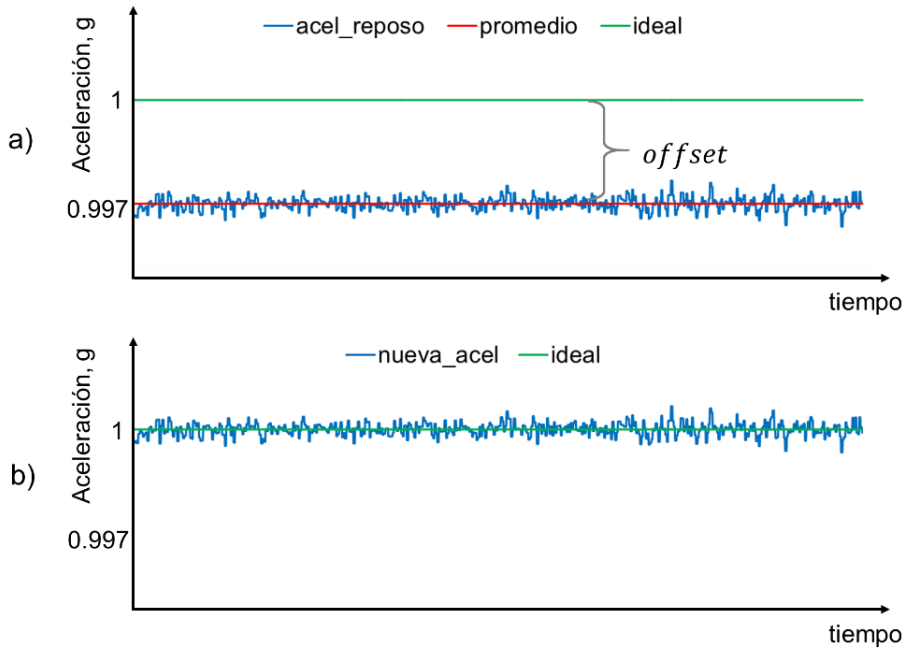


Figura 3-8 Offset. a) En reposo la señal es ruidosa y tiene un offset. b) Al quitar el offset, la señal en reposo se encuentra en +1g

Entonces, se desplaza la señal para que se tenga una lectura de +1g en reposo. Los registros OFFSET_Z_H y OFFSET_Z_L cumplen con esa función. El offset aplicado debe estar en complemento a 2 y el valor de los 16 bits de OFFSET_Z corresponde al valor de los 16 bits más significativos de ZDATA.

Por ejemplo, si el sensor 1, en reposo, entrega una aceleración promedio $acel_{reposo}$ =255,254 (0011 1110 0101 0001 0110_2), equivalente a

+0.9971g, entonces el offset necesario está dado por la diferencia de +1g y $acel_{reposo}$, esto es:

$$offset = 1g - acel_{reposo} \quad (3\text{-}3)$$

sustituyendo

$$offset = 256{,}000 - 255{,}254 = 746$$

Este número en binario es $offset = 0011\ 1110\ 0100\ 0111\ 1100_2$. Este valor debe ser escrito en complemento a 2 en los registros OFFSET_Z_H y OFFSET_Z_L, es decir

$$offset_{C2} = C_2(offset) \quad (3\text{-}4)$$

Donde $C_2(\cdot)$ es la operación complemento a 2 (ver Anexo B).

Entonces

$$offset_{C2} = 1111\ 1111\ 1101\ 0001\ 0110_2$$

Este número en hexadecimal es $offset_{C2} = 0xFFD16$. De este número, los 16 bits más significativos son los que se utilizan para los registros OFFSET_Z_H y OFFSET_Z_L. Entonces estos registros quedan como: OFFSET_Z_H = 0xFF, OFFSET_Z_L = 0xD1.

En la parte experimental, el offset será obtenido para cada sensor, ya que cada sensor de la red requerirá un valor específico para entregar +1g en reposo.

3.2.2.6 Umbral

El umbral se define con los registros ACT_THRESH_H y ACT_THRESH_L de la siguiente forma:

Los 16 bits de ACT_THRESH corresponden con el valor de los bits ZDATA[18:3]. Si, por ejemplo, el umbral es ± 0.005 g a partir de una señal

montada en +1 g (ver Figura 3-9), entonces el umbral a programar es 1.005 g. Utilizando la ecuación (3-2) se tiene que

$$umbral = \frac{umbral_g}{LSB} \tag{3-5}$$

sustituyendo

$$umbral = \frac{1.005g}{2.048g} \times 2^{19} = 257{,}280$$

Este número en binario es $umbral = 0011\ 1110\ 1101\ 0000\ 0000_2$. Tomando los bits 3 al 18 se tiene que ACT_THRESH = 0111 1101 1010 0000_2 o en hexadecimal ACT_THRESH = 0x7DA0, es decir, los registros para un umbral de ± 0.005 g a partir de una señal montada en +1 g son: ACT_THRESH_H = 0x7D y ACT_THRESH_L =0xA0

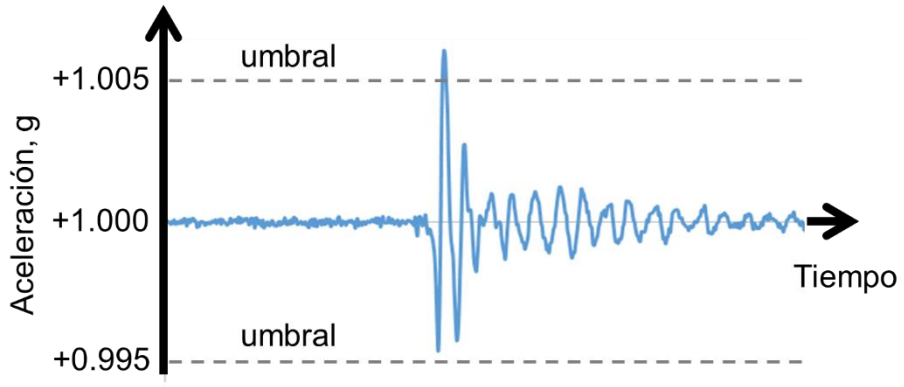


Figura 3-9. Definición de los umbrales a partir de una pisada (imagen ilustrativa, no se muestra una pisada real)

3.2.2.7 Interrupciones

El ADXL355 cuenta con interrupciones, las cuales activan alguno de los bits del registro STATUS cuando se presenta el evento que activó la interrupción.

Un evento que puede activar una interrupción es el cruce de umbral. Si el sensor detecta que la aceleración en cualquiera de los tres ejes (X, Y, Z) superó un umbral definido, entonces el bit Activity del registro STATUS se

activa. Estas interrupciones pueden ser asignadas a un pin externo del acelerómetro (INT1 o INT2). De tal forma que, cuando se presenta el cruce de umbral, el pin INTx entrega un 1 o un 0, según haya sido configurado. Para borrar el registro STATUS es necesario leerlo, lo cual hará que INTx regrese a su estado inicial.

Los registros y sus bits relacionados con el cruce de umbral son los siguientes: El Bit INT_POL determina cómo será la polaridad de la interrupción (INT_POL = 0 activa en bajo; INT_POL = 1 activa en alto). El bit ACT_Z del registro ACT_EN habilita el eje z para la detección de actividad. Los bits ACT_EN1 y ACT_EN2 del registro INT_MAP asignan las interrupciones por cruce de umbral en los pines INT1 e INT2 respectivamente.

Por ejemplo, supóngase que se tiene el sensor sobre el piso, la señal está montada en +1g en el eje z y se define un umbral de ±0.005g. Además, se asigna la interrupción por cruce de umbral en el pin externo INT1 y la polaridad de la interrupción es activo en alto. Finalmente, supóngase que se configura al sensor para que la interrupción sólo se pueda activar con las mediciones del eje z.

En un principio, el pin INT1 permanecerá en estado bajo (ver Figura 3-10a). Cuando se presenta una vibración producida por una pisada y llega al acelerómetro, la aceleración cruza el umbral. Esto provoca que el bit Activity del registro STATUS se active (pasa a estado alto), lo cual a su vez hace que el pin INT1 cambie a estado alto (la Figura 3-10b muestra que el pin INT1 toma el mismo valor que el bit Activity). Tanto el pin INT1 como el bit Activity, permanecen en estado alto. Para que cambien de estado es necesario leer el registro STATUS del acelerómetro con la ayuda de un microcontrolador mediante comunicación I2C (con los pines SCL y SDA como se muestra en la Figura 3-10b). Por lo que es importante leerlo antes de que se presente una nueva pisada para poder detectar de nuevo el cruce de umbral.

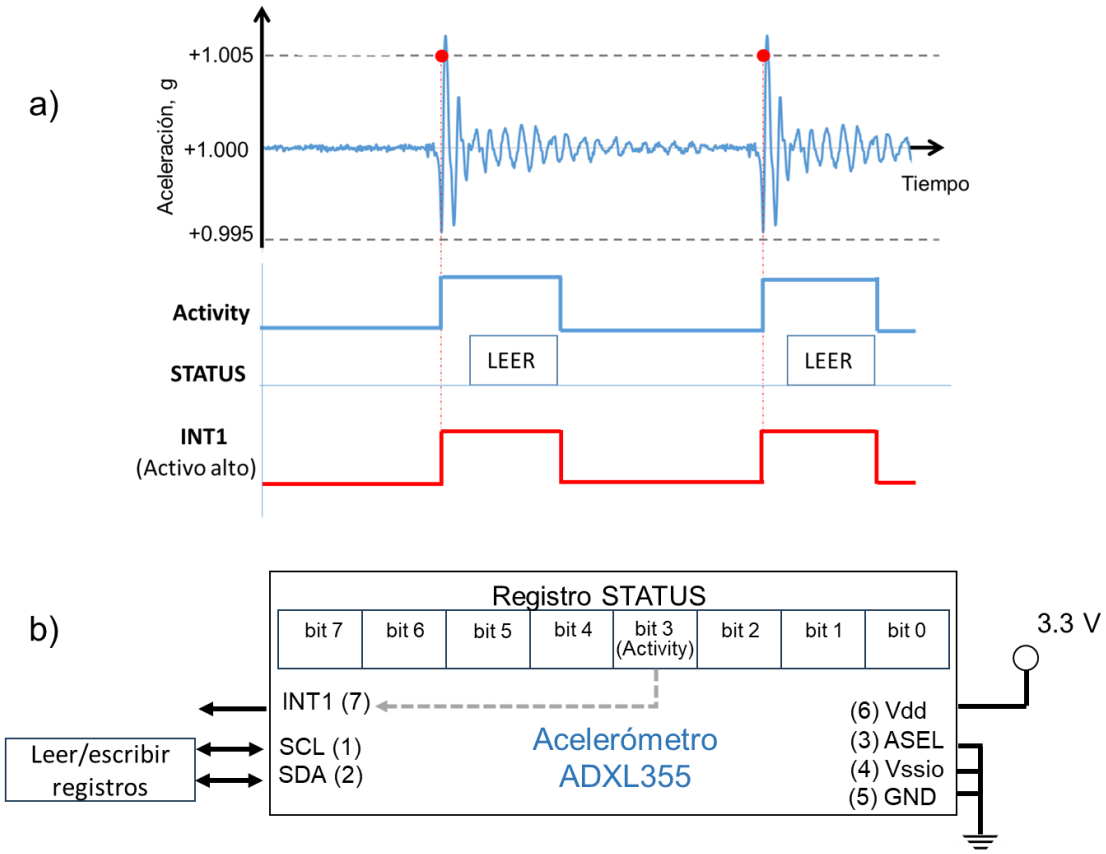


Figura 3-10. Detección de cruce de umbral. a) El pin INT1 cambia de estado cuando se presenta el cruce de umbral y permanece en estado alto hasta que se lee el registro STATUS. b) el registro STATUS del acelerómetro contiene al bit Activity; el estado del pin INT1 depende del estado del bit Activity

Si se conecta el pin INT1 del sensor a un pin de interrupción del microcontrolador, se pueden ejecutar funciones para realizar tareas específicas. En la siguiente sección se hablará de las tareas a realizar con esta interrupción.

3.3 Microcontrolador

Para configurar los registros vistos en la sección anterior, se utiliza el microcontrolador ATmega328P disponible en las placas de Arduino Nano. Este, además de llevar a cabo la comunicación I^2C, cuenta con interrupciones externas que se pueden asociar a un pin del microcontrolador (intPin). Si intPin se conecta al pin INT1 del acelerómetro como se muestra en la Figura 3-11, entonces es posible utilizar el cruce de umbral para calcular el tiempo de llegada de la señal (TOA).

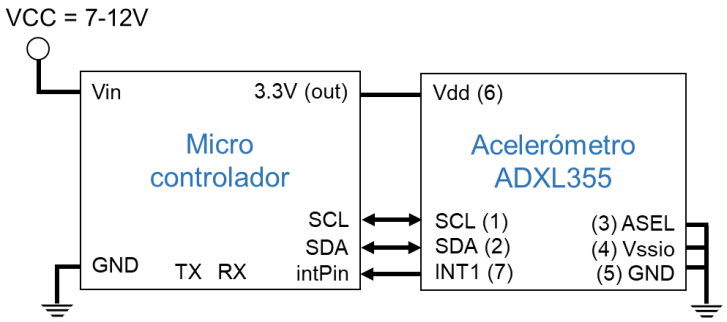


Figura 3-11. Diagrama a bloques de las conexiones eléctricas del acelerómetro y el microcontrolador

Esto es, cuando el nodo i recibe la bandera de parte del coordinador, en el microcontrolador se ejecuta un contador del TOA (cada cuenta equivale a 3.4 µs). En el momento en el que el acelerómetro detecta el cruce de umbral, INT1 cambia a estado alto. Al detectarse el flanco de subida de la señal en intPin, se ejecuta una interrupción y se detiene el contador del TOA. Luego, se envía el contador hacia el coordinador con un identificador (Ni, donde $i = 1, 2, \dots, N$) del nodo que detectó el cruce de umbral. Este dato se envía utilizando comunicación serial, es decir, por el pin TX (más adelante, en la sección de los módulos inalámbricos, se verá que para enviar o recibir información vía RF se requiere comunicación serial). A continuación, se lee el registro STATUS del acelerómetro para cambiar INT1 a estado bajo. Luego, se resetea el contador del TOA. Finalmente, el nodo permanece en espera de

la bandera de parte del Coordinador (leyendo el pin RX del puerto serie) y se repite el ciclo. Todo lo anterior se resume en la Figura 3-12.

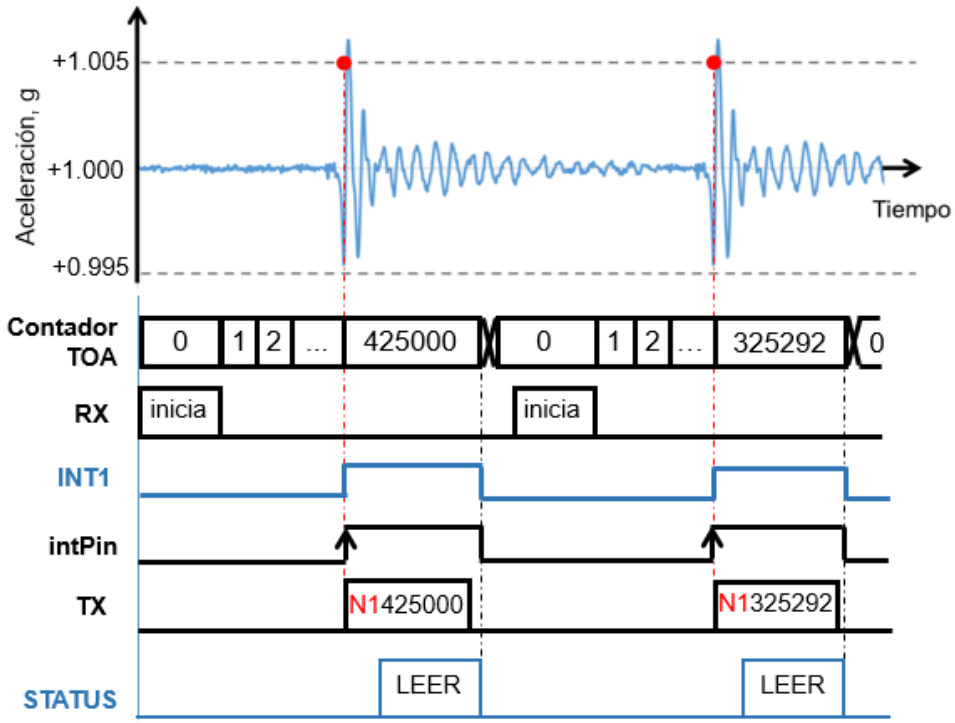


Figura 3-12. Señales involucradas en el cálculo del TOA del nodo 1 (las señales negras son del microcontrolador, las azules del acelerómetro)

El contador se envía con el identificador (Ni) del nodo, para que en el coordinador se ordene la información sin importar cuál nodo detecte primero el cruce de umbral.

Se utilizan 21% de los recursos del microcontrolador (el código utiliza 6,492 bytes de los 32,256 bytes disponibles en la memoria flash, mientras que las variables globales utilizan 440 bytes de los 2,048 bytes disponibles en la SRAM).

En la Figura 3-11 se observa que el microcontrolador cuenta con una fuente de 3.3V para alimentar al acelerómetro, por lo que sólo se requiere una fuente de 7-12V para alimentar a ambos circuitos.

3.4 Módulos TX/RX inalámbricos

Para realizar la red inalámbrica se utilizan módulos *XBee* Serie 3 (o simplemente *XBee3*) [35]. Algunas de sus características se muestran en la Tabla 3-3.

Velocidad de datos	RF 250Kbps, Serial hasta 1Mbps
Alcance en interiores/urbanos	Hasta 60m
Alcance en exteriores/línea de vista	Hasta 1200m
Interfaz serial de datos	UART, SPI, I2C
Banda de frecuencias	ISM 2.4GHz
Voltaje de alimentación	2.1 a 3.6V
Corriente de transmisión	40mA
Corriente de recepción	17mA
Corriente de apagado	2µA

Tabla 3-3. Características de los módulos inalámbricos XBee3 [35]

3.4.1 Tipos de dispositivos

En una red de módulos *XBee3* existen tres tipos de dispositivos: Coordinador, Router y End Device (ver Figura 3-13).

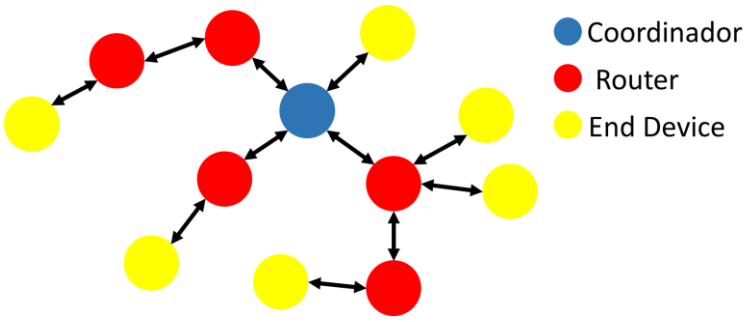


Figura 3-13. Tipos de dispositivos en una red de módulos XBee

A grandes rasgos, los roles de cada dispositivo son los siguientes: El Coordinador forma la red y permite que otros dispositivos (Routers y End Devices) se unan a ella. El Router se puede unir a una red existente, permitir que otros dispositivos se unan a ella y actuar como mensajero entre dispositivos (cuando estos están demasiado alejados para comunicarse entre

ellos). Los End Devices por su parte, se pueden unir a redes existentes, pero no pueden permitir que otros dispositivos se unan a ella y tampoco pueden actuar como mensajeros. Sin embargo, los End Devices pueden entrar en modo de suspensión (*sleep mode*), lo cual permite reducir el consumo de energía.

En este proyecto la red está conformada por un Coordinador y varios Routers conectados directamente a él. No se utilizan end devices ya que no es necesario el modo de suspensión. La cantidad de Routers o nodos de la red será determinada en la sección 3.6.3.

3.4.2 Modos de transmisión y recepción de datos

Cada uno de los módulos *XBee3* tiene dos direcciones: una dirección única de 64 bits (la cual es asignada de fábrica y no puede ser modificada) y una dirección asignada de 16 bits (la cual se le asigna cuando se une a la red). Para que un dispositivo se comunique con otro, es necesario que se encuentren dentro de la misma red y que al transmisor se le indique la dirección del receptor (ya sea la dirección de 16 o 64 bits).

Cuando se comunican dos módulos de manera bidireccional se conoce como modo punto a punto (ver Figura 3-14a, donde MY es la dirección del dispositivo y DL es la dirección hacia donde se transmite). Por otra parte, existe el modo *broadcast*, donde el transmisor (que puede ser cualquiera de los nodos de la red, incluyendo al coordinador) envía la información a todos los nodos de la red. La dirección para hacer transmisión *broadcast* es 0xFFFF (ver Figura 3-14b).

En este proyecto, el modo *broadcast* es utilizado para que el coordinador envíe la bandera a todos los nodos de la red. Por su parte, los nodos de la red sólo requieren enviar información al coordinador, por lo que se configuran para transmitir datos a la dirección del coordinador. La dirección reservada para el coordinador es 0x0000 (ver Figura 3-14b).

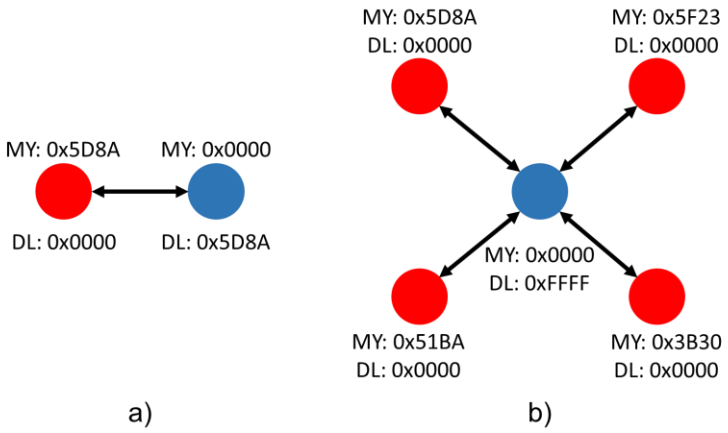


Figura 3-14. a) Los módulos se comunican punto a punto. b) El coordinador transmite en modo *Broadcast*

3.4.3 Conexiones eléctricas de los módulos *XBee3*

Las conexiones eléctricas para operar a los módulos XBee3 se muestran en la Figura 3-15. Se observa que se requiere una fuente de alimentación de 3.3 V y un par de alambres para hacer transmisión y recepción de datos de manera serial. Todos los datos que ingresan mediante cable por la terminal DIN, son enviados vía RF a la dirección de destino DL. Por otra parte, si se reciben datos vía RF, estos salen mediante cable por la terminal DOUT del *XBee3*.

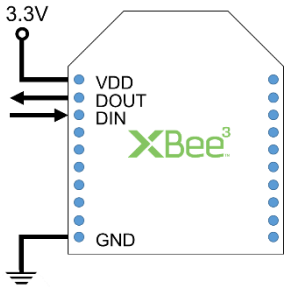


Figura 3-15. Conexiones eléctricas del *XBee3*

En los nodos de la red de este proyecto, los pines DOUT y DIN se conectan mediante cable a los pines RX y TX del microcontrolador como se muestra en la Figura 3-16. De tal forma que, en los nodos, se puede enviar el

TOA vía RF hacia el coordinador escribiéndolo en el pin DIN del *XBee3* y se puede recibir la bandera vía RF de parte del coordinador leyendo el pin DOUT del *XBee3*.

Por otro lado, en el coordinador se requiere un convertidor serial a USB, con esto la comunicación alámbrica se puede llevar a cabo con una computadora en lugar de con un microcontrolador (ver Figura 3-16).

El convertidor USB-Serie utilizado en el coordinador es el SparkFun XBee Explorer Dongle que se muestra en la Figura 3-17.

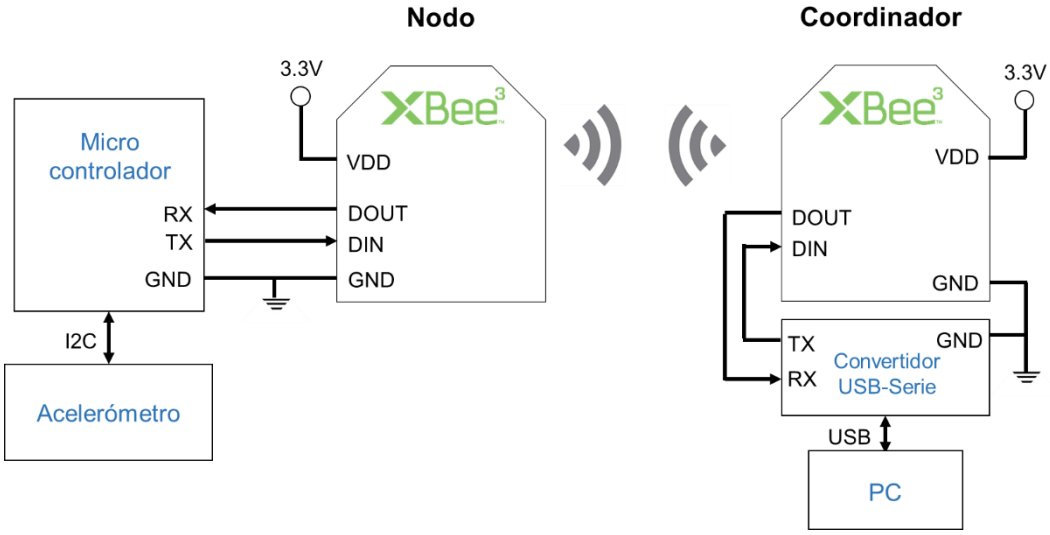


Figura 3-16. Conexiones eléctricas de los módulos *XBee3*

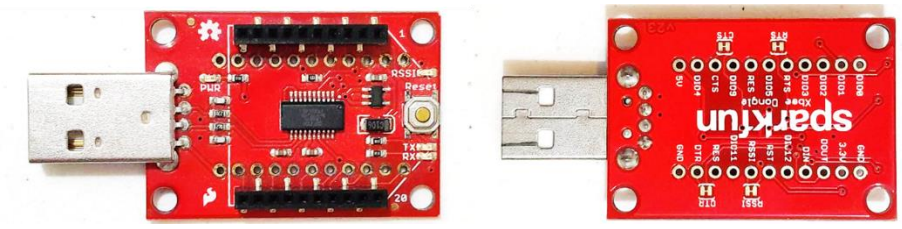


Figura 3-17. SparkFun XBee Explorer Dongle usado para conectar el coordinador con la computadora. Vista superior e inferior

3.5 Conexiones eléctricas de la Red Inalámbrica de Sensores Inteligentes

Las conexiones del acelerómetro, el microcontrolador y los módulos *XBee3* vistos anteriormente, forman la red inalámbrica de sensores inteligentes. La Figura 3-18 muestra un diagrama a bloques de las conexiones eléctricas de la red.

Con esta red, se envía la bandera de inicio desde el Coordinador (PC) hacia los nodos. Y en los nodos, se hace la detección del cruce de umbral, así como la medición y envío del TOA hacia el coordinador.

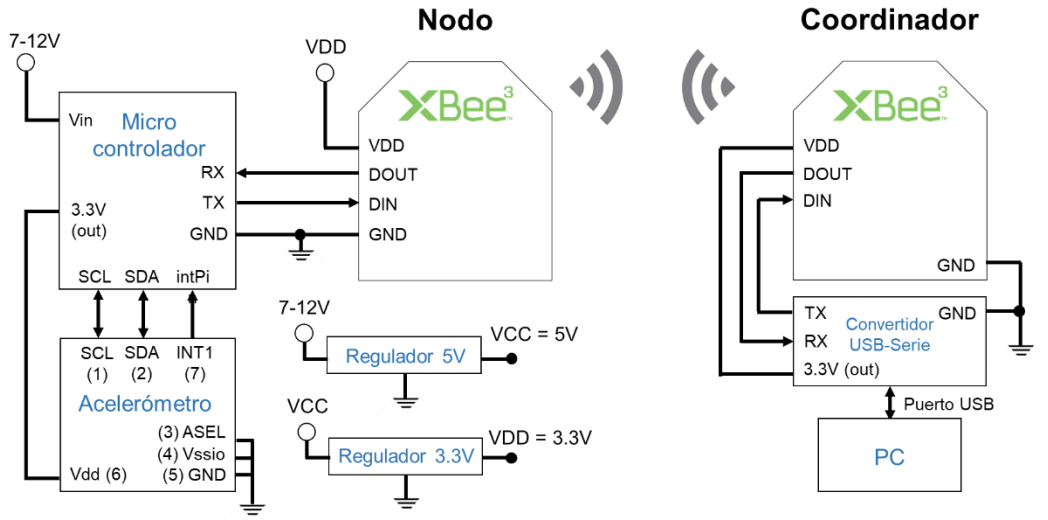


Figura 3-18. Diagrama a bloques de las conexiones eléctricas del coordinador y un nodo

La Figura 3-19 muestra el diseño topológico de la PCB para los nodos de la red. La Tabla 3-4 resume los componentes utilizados, así como su precio. Los capacitores que aparecen en la tabla sirven para darle estabilidad a los reguladores de voltaje (para más detalles se pueden consultar sus hojas de especificaciones). El LED y el resistor se utilizan para indicar cuando el nodo se ha conectado a la red (para más detalles consultar la hoja de datos de los módulos *XBee3*).

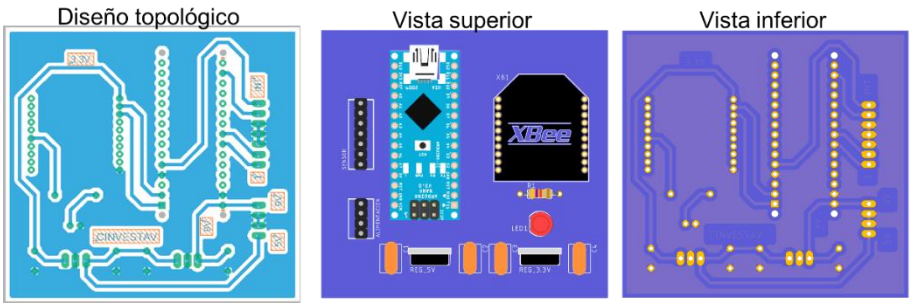


Figura 3-19. PCB para los nodos de la red

Componente	Matrícula o Características	Cantidad	Precio Unitario (USD)
Acelerómetro	ADXL355	1	$ 47.13
Módulo RF	XBee Serie 3	1	$ 26.98
Microcontrolador	Arduino Nano	1	$ 6.00
Regulador 3.3V	LF33CV	1	$ 2.26
Regulador 5V	L7805CV	1	$ 0.90
Headers	2.54 mm (tira de 40 pines)	1	$ 0.57
Headers	2 mm (tira de 40 pines)	1	$ 0.17
Capacitor electrolítico	2.2 uF/50 V	1	$ 0.11
Capacitor cerámico	0.1 uF/50 V	2	$ 0.06
Capacitor cerámico	0.33 uF/50 V	1	$ 0.06
LED rojo	5mm	1	$0.06
Resistor	1 kOhm/0.25 W	1	$ 0.03
		TOTAL	$84.33

Tabla 3-4. Lista de materiales para la PCB de un nodo

3.5.1 Consumo de energía

El consumo de energía de los dispositivos utilizados se resume en la Tabla 3-5, es decir, cada nodo inalámbrico de la red tiene un consumo máximo de 359 mA.

Dispositivo	Voltaje (V)	Corriente (mA)
Acelerómetro [34]	3.3	100
Microcontrolador [36]	7-12	40 (por cada pin), 19 (alimentación)
XBee [35]	2.1 – 3.6	40 (Transmisión), 17 (recepción)

Tabla 3-5. Consumo de energía de los dispositivos utilizados en la red

3.6 Optimización de la red de sensores

En la sección 2.2 se mencionó que, en el algoritmo heurístico SO-TDOA, el error de localización de las pisadas depende del número y la forma de las regiones, es decir, el error de estimación está directamente relacionado con el número y la ubicación de los sensores.

El problema de seleccionar el número de sensores radica en que estará limitado tanto por el presupuesto disponible como por el error de localización máximo permitido para toda la habitación. En cuanto a la ubicación de los sensores, podría suponerse que el error de localización será menor si se distribuyen de manera uniforme a lo largo de la habitación, como se ilustra en la Figura 3-20. Más adelante mostraremos que esto no es lo más conveniente, además, surgiría la pregunta ¿cómo se distribuyen de manera uniforme 6, 7 u 8 sensores dentro de una habitación rectangular?

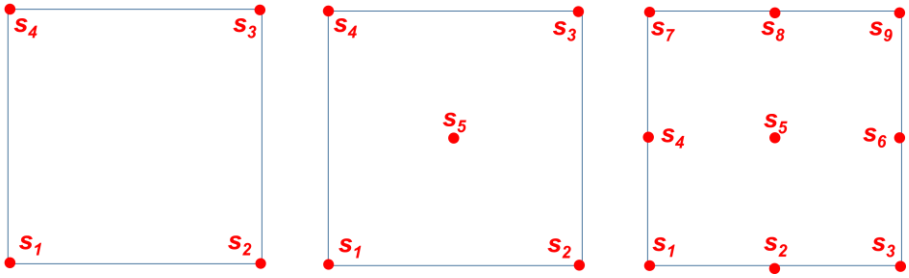


Figura 3-20. Sensores distribuidos uniformemente dentro de una habitación cuadrada

A continuación, se presenta una propuesta para optimizar el número y la ubicación de los sensores con el objetivo de minimizar el error de localización.

3.6.1 Definición del Problema de Optimización

Supóngase que se tienen 5 sensores $\boldsymbol{s}_i$ distribuidos uniformemente a lo largo de la habitación como se muestra en la Figura 3-21a. Si se cambia la ubicación de los sensores (ver Figura 3-21b), es posible obtener un mayor número de centroides $\boldsymbol{p}_k^c$ y con esto disminuir el error de localización para toda la habitación.

Además, en la Figura 3-21b se observa que, al cambiar la ubicación de los sensores, se modifican completamente todas las regiones, por lo que encontrar el número y la ubicación de los sensores que minimicen el error de localización es una tarea prohibitiva si se realiza manualmente. Por lo tanto, en este trabajo, la ubicación de los sensores se optimiza computacionalmente utilizando metaheurísticas bio-inspiradas.

A continuación, se define la función objetivo (función a minimizar) que se utilizará para la optimización.

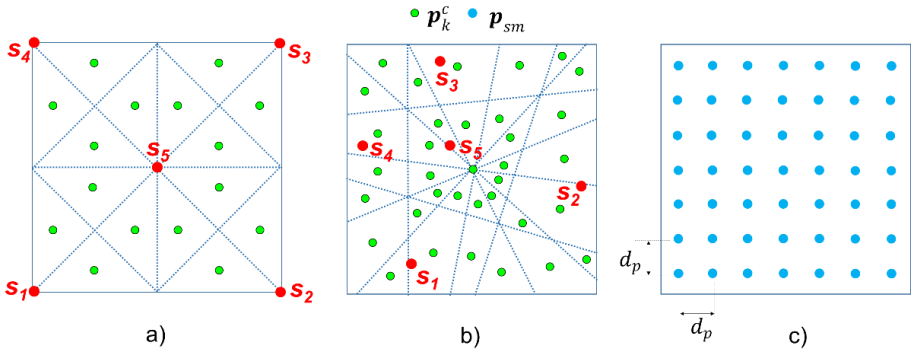


Figura 3-21. Centroides en una habitación para sensores distribuidos: a) uniformemente b) de manera no uniforme. c) Pisadas distribuidas uniformemente en la habitación

La optimización se realiza con pisadas distribuidas uniformemente en la habitación (ver Figura 3-21c), por lo tanto, la optimización consiste en encontrar la ubicación de los sensores que minimicen el error cuadrático medio (MSE) de la estimación de la localización de estas pisadas.

Entonces la función objetivo está dada por el MSE y se calcula como:

$$MSE = \frac{1}{N_p}\sum_{m=1}^{Np}(error_m)^2 \quad (3\text{-}6)$$

Donde:

N_p es el número de pisadas y $error_m$ es el error de localización de la pisada m y se calcula como:

$$error_m = |\boldsymbol{p}_{sm} - \widehat{\boldsymbol{p}}_{sm}|$$

$$error_m = \sqrt{\left(p_{sm_x} - \hat{p}_{sm_x}\right)^2 + \left(p_{sm_y} - \hat{p}_{sm_y}\right)^2} \quad (3\text{-}7)$$

Donde:

$\boldsymbol{p}_{sm} = \left(p_{sm_x}, p_{sm_y}\right)$ es la ubicación real de la pisada m,

$\widehat{\boldsymbol{p}}_{sm} = \left(\hat{p}_{sm_x}, \hat{p}_{sm_y}\right)$ es la estimación de la pisada m usando el algoritmo SO-TDOA.

Como se discutió anteriormente, en el algoritmo SO-TDOA, la habitación de prueba se divide en regiones. El número y la forma de estas regiones dependen tanto de la cantidad como de la ubicación de los sensores. En consecuencia, el error de localización de pisadas se ve sustancialmente afectado por estos parámetros.

Sustituyendo las ecuaciones (2-2) - (2-5) y (3-7) en la ecuación (3-6), la función objetivo queda como:

$$MSE = \frac{1}{N_p}\sum_{m=1}^{Np}\left|\boldsymbol{p}_{sm} - \frac{1}{\left|arg\min_{k\in[1\ldots Q]}\sum_{a=1}^{N(N-1)/2}\left(\boldsymbol{z}_{sm}(a)\oplus\boldsymbol{z}_k(a)\right)\right|}\sum_{r\in M_{rm}}\boldsymbol{p}_{rm}^{c}\right|^2$$

Donde M_{rm} es el conjunto de regiones r que minimizan la distancia de Hamming entre el vector característico medido $\boldsymbol{z}_{sm}$ y el vector característico $\boldsymbol{z}_k$

$\oplus$ es el operador OR exclusiva,

$\boldsymbol{z}_{sm}(a)$ representa el a-ésimo elemento del vector $\boldsymbol{z}_s$ de la pisada m,

$\boldsymbol{p}_{rm}^{c}$ es el centroide de la región r que minimiza la distancia de Hamming.

De tal forma que la función objetivo depende tanto del número de sensores N como de su ubicación $\boldsymbol{s}_i = (s_{ix}, s_{iy})$ debido al vector característico $\boldsymbol{z}_k$ (Ecuación (2-2)), el cual se puede reescribir como:

$$\boldsymbol{z}_k = sign\left(\sqrt{\left(s_{ix} - p_{kx}^{c}\right)^2 + \left(s_{iy} - p_{ky}^{c}\right)^2} - \sqrt{\left(s_{jx} - p_{kx}^{c}\right)^2 + \left(s_{jy} - p_{ky}^{c}\right)^2}\right); \quad l = \frac{(j-2)(j-1)}{2} + i$$

Si el vector $\boldsymbol{S} = [s_{1x}\ s_{1y}\ s_{2x}\ s_{2y}\ \ldots\ s_{Nx}\ s_{Ny}]$ contiene todas las coordenadas $\boldsymbol{s}_i = (s_{ix}, s_{iy})$ de los sensores, donde $i = 1, 2, \ldots, N$ y N es el número de sensores, entonces, la optimización consiste en encontrar el vector $\boldsymbol{S}$ que minimice el MSE dentro de un espacio de búsqueda $2N$-dimensional. Los límites (*boundaries*) de la función objetivo están dadas por los límites espaciales de la habitación. Entonces, el problema de optimización se puede formular de la siguiente manera:

Minimizar $MSE(\boldsymbol{S})$ sujeto a $s_{ix} \in (0, U_x); \quad s_{iy} \in (0, U_y)$

Donde U_x, U_y son los valores máximos para las dimensiones horizontal y vertical de la habitación, respectivamente.

3.6.2 Características del problema de optimización

Como anteriormente se mencionó, el espacio de búsqueda es un espacio limitado que contiene todas las posibles soluciones del problema de optimización. En este trabajo, el espacio de búsqueda está acotado por las dimensiones de la habitación.

Para hacer las pruebas se selecciona una habitación de 2.5 m x 3.3 m. Esta habitación tiene un piso de concreto de 11 cm de espesor cubierto con azulejos cerámicos y está ubicada en el segundo piso de una casa residencial.

Los límites de la función objetivo son 0.2 m ≤ s_{ix} ≤ 2.3 m y 0.2 m ≤ s_{iy}≤ 3.1 m como se muestra en la Figura 3-22a. Las pisadas para la optimización se distribuyen uniformemente cada 0.25 m como se muestra en la Figura 3-22b.

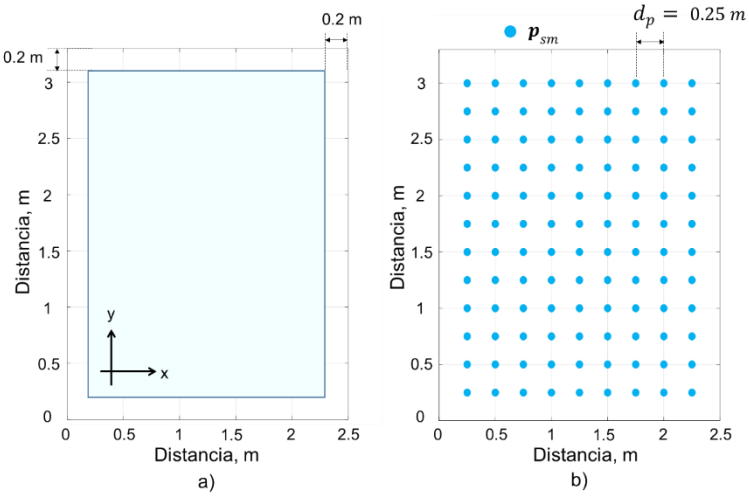


Figura 3-22 a) Habitación y límites para la optimización. b) Pisadas distribuidas uniformemente a lo largo de la habitación

3.6.3 Estudio del número de sensores

Para determinar el número de sensores, se optimiza el MSE para $N = 3,4,\ldots,10$. La optimización se puede realizar con cualquiera de las metaheurísticas antes mencionadas (PSO, ABC, GWO, HPSGWO). En este caso se selecciona el HPSGWO, sin embargo, es importante mencionar que, una vez seleccionado el número de sensores, su ubicación se optimizará con todas las metaheurísticas para seleccionar la mejor de ellas.

El tamaño de la población es $N_p = 50$, este valor se seleccionó teniendo en cuenta los valores típicos utilizados en la literatura [27]. Además, como se presentará más adelante, con este valor logramos que el algoritmo converja

en pocas iteraciones (menos de 200). Se ejecutó el algoritmo metaheurístico durante 200 iteraciones (condición de parada $MaxIt = 200$), repitiendo el proceso 10 veces y seleccionando puntos iniciales aleatorios para cada ejecución, con el fin de aumentar la posibilidad de encontrar una solución global. Los parámetros del HPSGWO son: $c_1 = c_2 = 1.49445$, $w = \frac{MaxIt - t}{MaxIt}$ [32], número de lobos $N_w = 10$, número de iteraciones de los lobos $MaxItW = 10$, probabilidad $prob = 0.01$ [27].

En la Figura 3-23 se muestra la evolución del MSE de las 10 corridas para el caso de 3 sensores en la habitación. Se observa que el error disminuye a medida que pasan las iteraciones. Además, en cada corrida se encuentra un mínimo diferente, por lo que se calcula el promedio de las 10 corridas para comparar el MSE con respecto al resto de número de sensores.

La Figura 3-24 muestra la evolución de la convergencia promedio de las 10 corridas para los diferentes números de sensores ($N = 3, 4, \dots 10$). Se observa que, como era esperado, si se tiene un mayor número de sensores, el MSE disminuye.

Si se obtiene la raíz cuadrada del MSE_prom de la última iteración (los óptimos encontrados), se obtiene la Figura 3-25. Finalmente, haciendo un ajuste de la curva, se obtiene que el RMSE_prom se ajusta al modelo exponencial:

$$RMSE\_prom = 1.082e^{-0.2896N}$$

Donde $N = 3, 4, \dots 10$, es el número de sensores

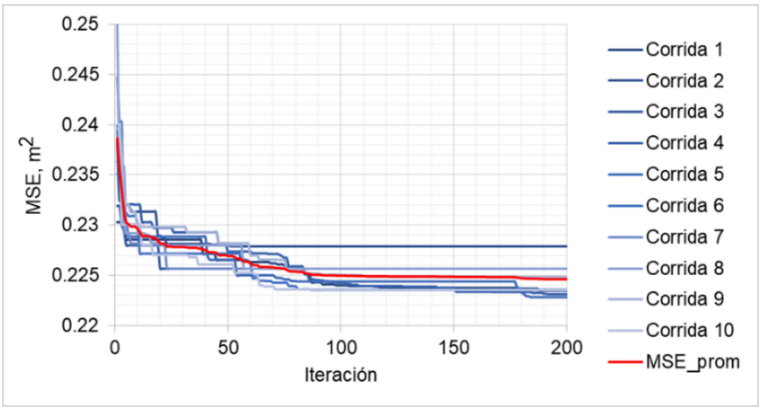


Figura 3-23. Evolución del algoritmo HPSGWO para tres sensores

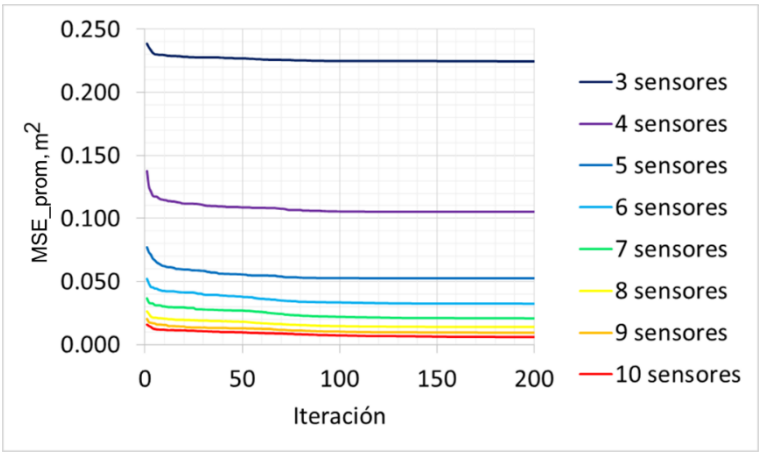


Figura 3-24. Evolución promedio del MSE para diferentes números de sensores

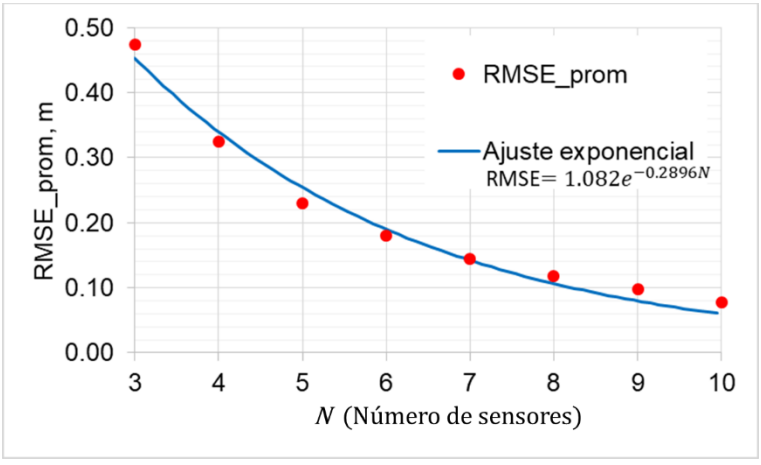


Figura 3-25. RMSE_prom en función del número de sesores N

La Figura 3-26 muestra la ubicación óptima de los sensores para los diferentes casos. Se observa que en ninguno de ellos los sensores quedan distribuidos uniformemente en la habitación.

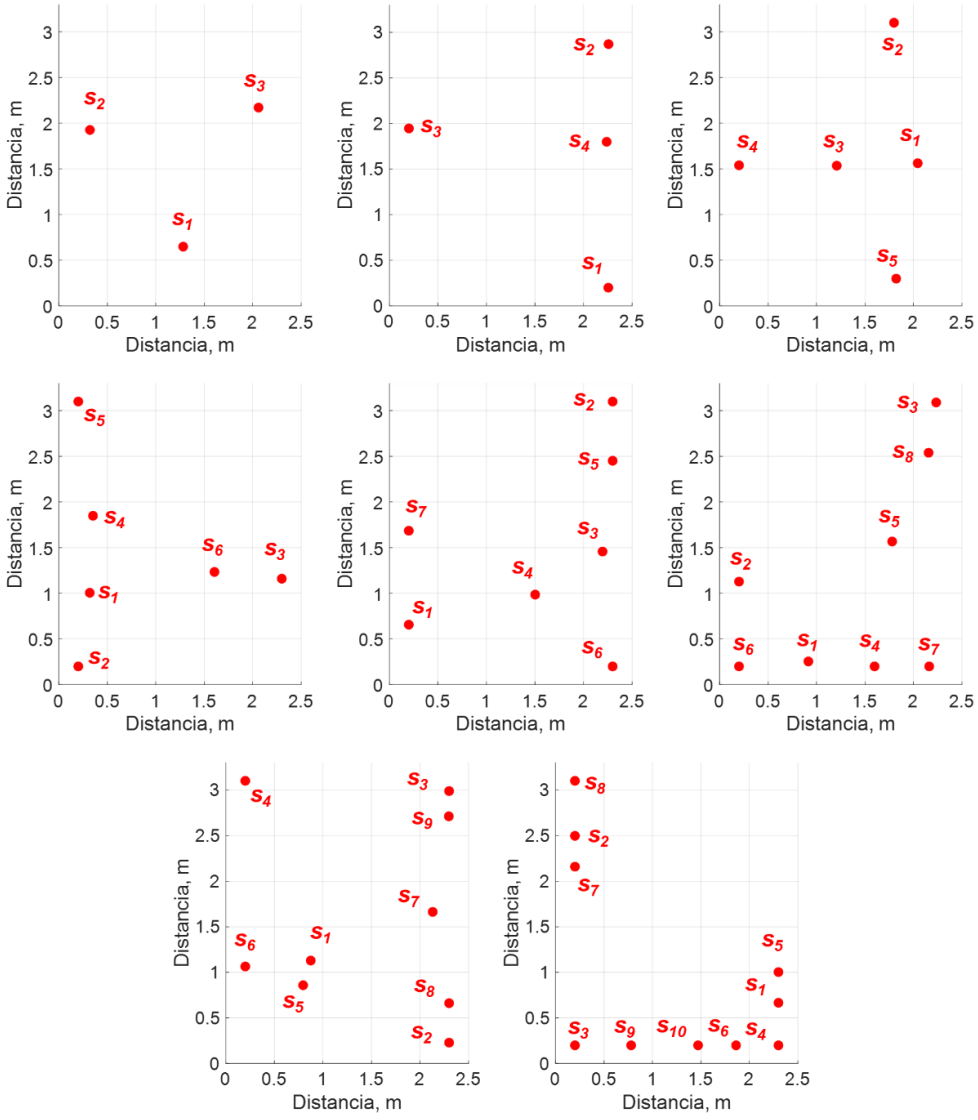


Figura 3-26. Ubicación optimizada de 3, 4, …, 10 sensores

Como anteriormente se mencionó, el número de sensores se selecciona en función del error de localización máximo permitido para toda la habitación, teniendo en cuenta el presupuesto disponible para el proyecto. En este trabajo se seleccionan 5 sensores, ya que con estos se puede tener un error de

localización de aproximadamente 25 cm para toda la habitación (ver Figura 3-25).

A continuación, se presenta la optimización de la ubicación de estos 5 sensores empleando las diferentes metaheurísticas antes mencionadas.

3.6.4 Optimización de la ubicación de los sensores

Como se tienen 5 sensores, el vector ***S*** es de 10 dimensiones, esto es, el espacio de búsqueda es 10-dimensional. Para esta optimización, el tamaño de la población es $N_p = 50$, la condición de parada es $MaxIt = 200$ y cada algoritmo se ejecuta 20 veces. Los parámetros del PSO son: $c_1 = c_2 = 1.49445$, $w = \frac{MaxIt - t}{MaxIt}$ [1]. Los parámetros del GWO para el HPSGWO son: número de lobos $N_w = 10$, número de iteraciones $MaxItW = 10$, probabilidad $prob = 0.01$. El límite de abandono de búsqueda local del ABC es *limit* = 10 [37].

Los algoritmos se implementan en MATLAB R2022a en una PC con un procesador Intel Core i7-7500U.

La Tabla 3-6 muestra el MSE mínimo y máximo, así como la desviación estándar (DE) para cada algoritmo. Se observa que el HPSGWO entrega el menor MSE de todas las corridas (0.047 m^2). Por otra parte, la Figura 3-27 muestra la evolución promedio del MSE de las 20 corridas de todos los algoritmos. Se observa que, aunque todos optimizan el problema adecuadamente, el algoritmo GWO converge a un valor promedio menor (0.051 m^2), sin embargo, el HPSGWO converge con menos iteraciones.

Algoritmo de Optimización	MSE Mín. (m^2)	MSE Máx. (m^2)	DE (m^2)
No Optimizado	0.091	0.091	0
ABC	0.048	0.055	0.0017
GWO	0.048	0.059	0.0033
PSO	0.050	0.063	0.0028
HPSGWO	0.047	0.058	0.0027

Tabla 3-6. Comparación de los diferentes algoritmos de optimización

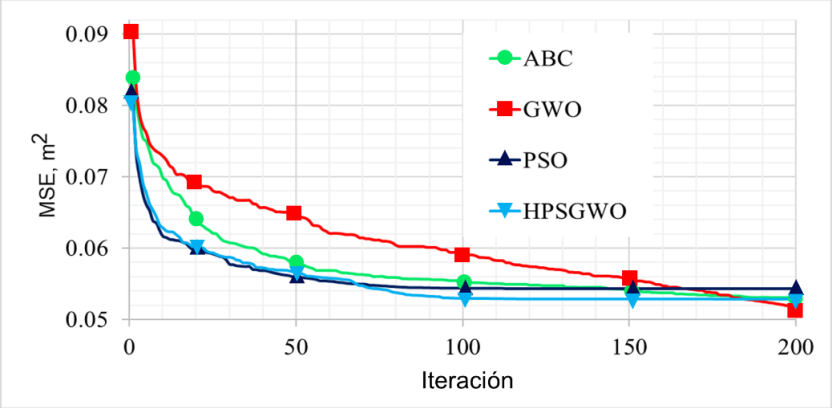


Figura 3-27. Evolución promedio de las diferentes metaheurísticas bio-inspiradas

La Figura 3-28 muestra la ubicación de los sensores después de la optimización empleando los algoritmos ABC, GWO y PSO.

Debido a la naturaleza inherente de las metaheurísticas, no podemos garantizar que se haya encontrado el mínimo global. Sin embargo, la solución obtenida se considera satisfactoria ya que se redujo el RMSE para toda la habitación en un 27.9% en comparación con las ubicaciones no optimizadas mostradas en la Figura 3-29a (RMSE no optimizado = 0.301 m; RMSE optimizado = 0.217 m). La Figura 3-29b muestra las ubicaciones optimizadas de los sensores, indicando que las nuevas posiciones aumentan el número de centroides $\boldsymbol{p}_k^c$.

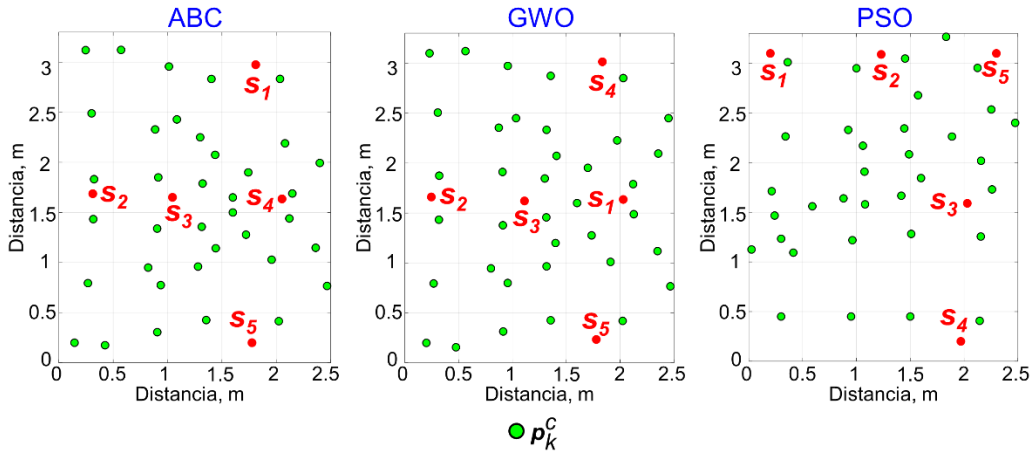


Figura 3-28. Sensores optimizados y centroides $\boldsymbol{p}_k^c$ de la habitación

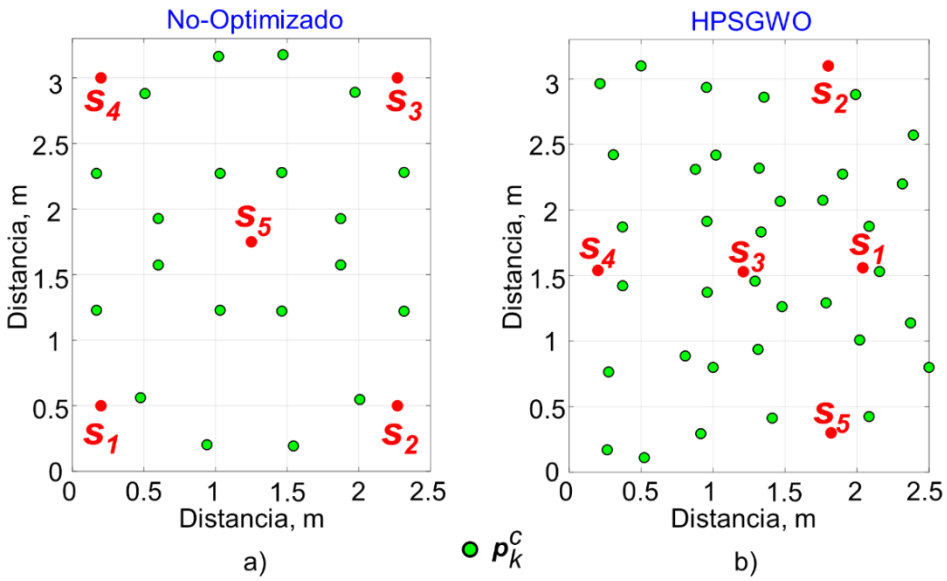


Figura 3-29. Sensores y centroides de la habitación. a) No optimizado. b) Optimizado con HPSGWO

En el siguiente capítulo se presentan pruebas experimentales de la red propuesta y optimizada con HPSGWO.

3.7 Conclusiones del capítulo

En este capítulo se identificó un acelerómetro de bajo costo y se propuso una red inalámbrica de sensores inteligentes para capturar las vibraciones producidas por el impacto de los pies de una persona sobre un piso de concreto.

A diferencia de la literatura, nuestra red está diseñada para permitir que los nodos transmitan un solo dato por pisada, lo que facilita bajas tasas de transmisión/recepción de datos. Además, la red lleva a cabo la detección de cruce de umbral directamente en los nodos, por lo que no es necesario realizar este cálculo en la computadora, lo que simplifica el sistema. Además, está

diseñada para determinar el tiempo de llegada (TOA) de las señales sin necesidad de almacenar una gran cantidad de datos para cada pisada.

En la literatura, la ubicación precisa de los sensores no es un enfoque principal. Por lo tanto, en este trabajo, se optimizaron el número de sensores y su ubicación en la habitación utilizando metaheurísticas bioinspiradas para minimizar el error de localización de las pisadas, demostrando que la aplicación estratégica del algoritmo SO-TDOA puede disminuir sustancialmente los errores de localización.

El estudio del número de sensores de la red mostró que el error de localización para toda la habitación tiene una disminución similar a una función exponencial a medida que se incrementa el número de sensores. Por otro lado, la optimización de la ubicación de los sensores redujo el error de localización en más de un 25% en comparación con una red no optimizada.

4 RESULTADOS EXPERIMENTALES

En este capítulo se presentan las pruebas experimentales de la red de sensores propuesta y optimizada en el capítulo anterior.

4.1 Definición del umbral

Para definir el umbral, se realizan un conjunto de pruebas. La primera consiste en colocar un sensor en el centro de la habitación y caminar alrededor de él sobre un círculo de 1 m de radio. Las tres componentes de aceleración (x, y, z) se envían a una computadora y se muestran en la Figura 4-1. Se observa que, como se mencionó en el capítulo 2, la componente perpendicular al plano del piso (z) es la que domina las vibraciones en el mismo, por lo que únicamente se trabaja con esta componente.

Posteriormente, se coloca el sensor en un extremo de la habitación y se realizan impactos de pisadas siguiendo las dos trayectorias que se muestran en la Figura 4-2. En esta misma figura se muestran las vibraciones de ambas trayectorias. Se observa que, aunque la amplitud de las vibraciones varía, los picos tienen una amplitud mayor a 0.005 g.

Por lo tanto, se define el umbral con un valor de ±0.005g tomando como referencia +1g, es decir, el TOA se obtendrá cuando la señal supere +1.005g o caiga por debajo de +0.995g.

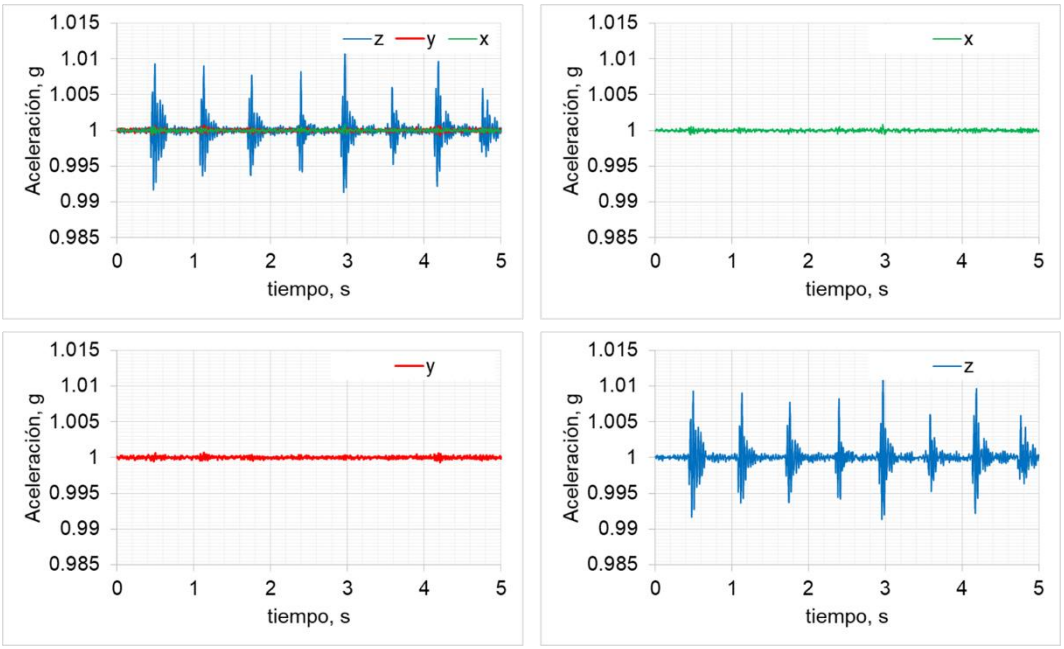


Figura 4-1. Componentes x, y, z de las vibraciones en el piso producidas por pisadas al caminar alrededor de un círculo de 1 m de radio

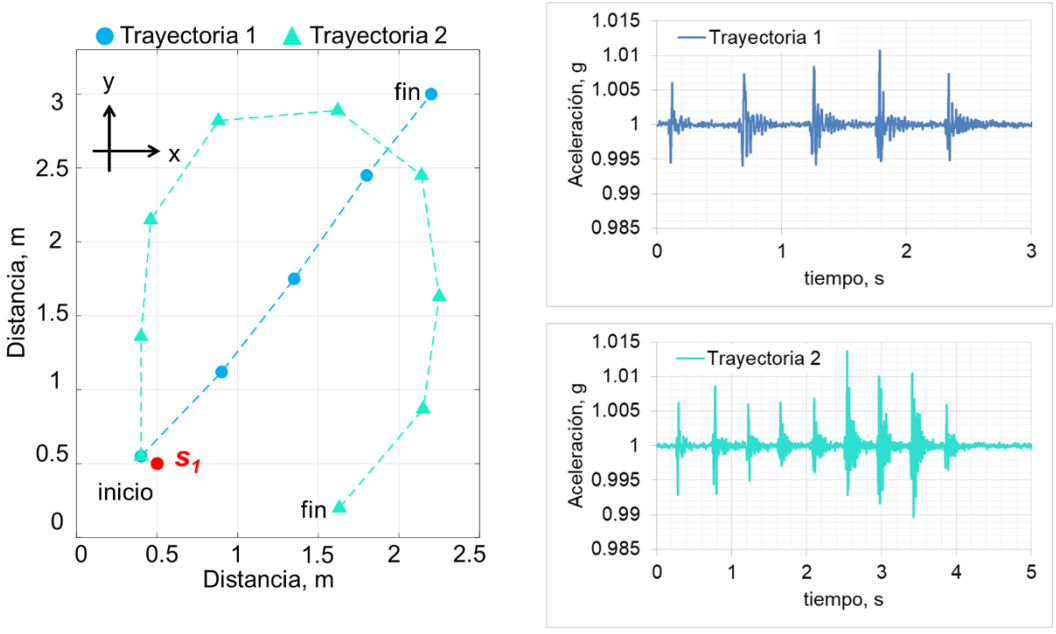


Figura 4-2. Trayectorias para definición de umbral y vibraciones obtenidas con el sensor

4.2 Ajuste de la Red de Sensores

Antes de hacer mediciones de las vibraciones, es necesario asegurarse de que los nodos de la red estén sincronizados. Para esto, el coordinador envía una bandera a todos los nodos para que inicien un contador. Transcurrido un tiempo $Tref = 20\ seg$, el coordinador vuelve a enviar otra bandera para que los nodos detengan su contador y le envíen su valor. Lo anterior se repite 10 veces.

Una vez recibidos, se promedian los contadores de cada nodo y se toma como referencia al nodo con menor valor. La Tabla 4-1 muestra que todos los nodos tienen una diferencia (ti4) respecto al nodo N4.

	N1	N2	N3	N4	N5
Tref (s)	20.018	20.030	20.010	20	20.026
ti4 (ms)	18.676	30.822	10.981	0	26.316

Tabla 4-1. Diferencias en los contadores

Se repite lo anterior para diferentes valores de $Tref$ (1, 2, 3, …, 10, 15 segundos). Los resultados se muestran en la Tabla 4-2. Si se grafican los datos de esta tabla (ver Figura 4-3), y se hace un ajuste de las curvas, se obtiene una expresión para ajustar los tiempos en cada nodo.

$Tref$ (s)	t14 (ms)	t24 (ms)	t34 (ms)	t44 (ms)	t54 (ms)
1	0.934	1.556	0.572	0	1.321
2	1.879	3.100	1.117	0	2.639
3	2.798	4.632	1.684	0	3.944
4	3.758	6.181	2.221	0	5.274
5	4.668	7.731	2.770	0	6.570
6	5.609	9.247	3.326	0	7.882
7	6.535	10.796	3.862	0	9.195
8	7.462	12.340	4.393	0	10.495
9	8.415	13.880	4.959	0	11.820
10	9.364	15.413	5.506	0	13.140
15	14.042	23.161	8.252	0	19.810
20	18.676	30.822	10.981	0	26.316

Tabla 4-2. Diferencias en los contadores con respecto al nodo 4 para distintos $Tref$

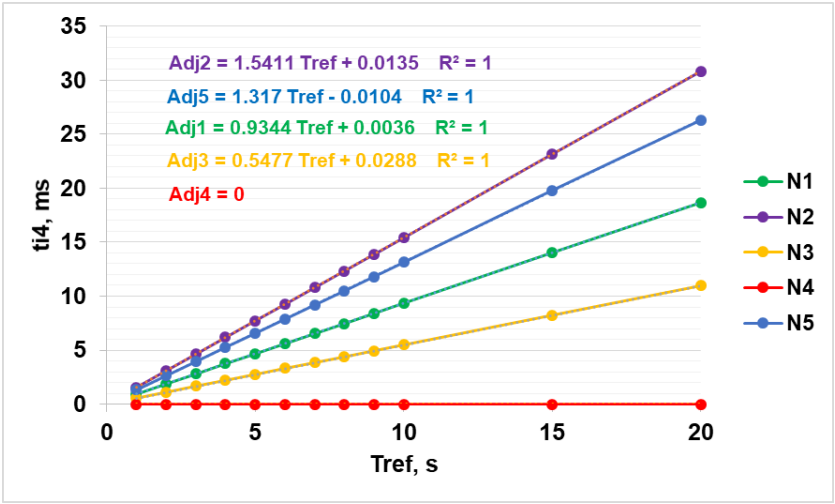


Figura 4-3. Diferencias en los contadores tomando como referencia a N4

El ajuste queda como:

$$Adj1 = (0.9344\,Tref + 0.0036) \times 10^{-3}$$

$$Adj2 = (1.5411\,Tref + 0.0135) \times 10^{-3}$$

$$Adj3 = (0.5477\,Tref + 0.0288) \times 10^{-3}$$

$$Adj4 = 0$$

$$Adj5 = (1.317\,Tref + 0.0104) \times 10^{-3}$$

Donde $Tref$ está dado en segundos, $Adji$ es el ajuste para el nodo i.

Entones, cuando se reciban los tiempos de llegada $\hat{t}_{si}$, estos estarán ajustados como:

$$t_{s1} = \hat{t}_{s1} - Adj1$$

$$t_{s2} = \hat{t}_{s2} - Adj2$$

$$t_{s3} = \hat{t}_{s3} - Adj3$$

$$t_{s4} = \hat{t}_{s4} - Adj4$$

$$t_{s5} = \hat{t}_{s5} - Adj5$$

4.3 Trayectorias en la habitación

A continuación, se colocan los 5 nodos de la red en las ubicaciones no optimizadas ($\boldsymbol{S_{no\_opt}} = [0.20\ 0.50\ 2.27\ 0.50\ 2.27\ 3.00\ 0.20\ 3.00\ 1.25\ 1.75]$), con el eje z perpendicular al plano del piso como se muestra en la Figura 4-4.

Luego, se obtiene el offset para que cada uno de los sensores entregue, en reposo, una señal montada sobre +1g. La Figura 4-5 muestra las señales de los 5 sensores antes y después de ajustar el offset.

Figura 4-4. Fotografía de la habitación de pruebas. Los sensores se colocan en las ubicaciones no optimizadas

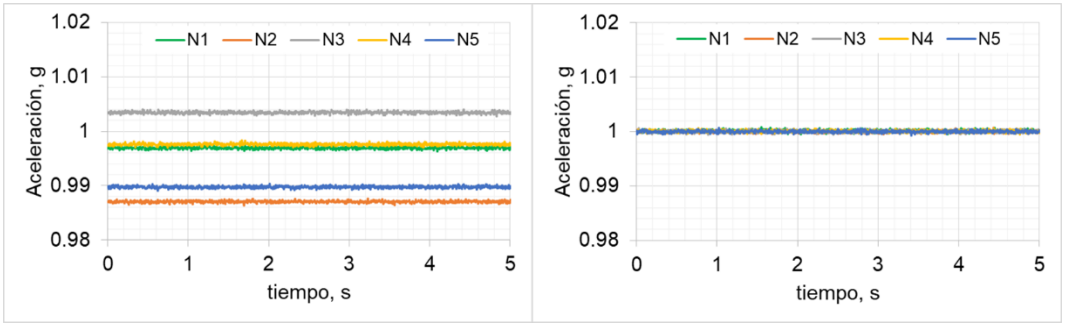


Figura 4-5. Señal de los acelerómetros en reposo antes (izquierda) y después (derecha) de ajustar el offset

Una vez conectada la red, se realizan impactos con el pie en diferentes ubicaciones de la habitación, siguiendo las tres trayectorias mostradas en la Figura 4-6. Cada ubicación es impactada 10 veces.

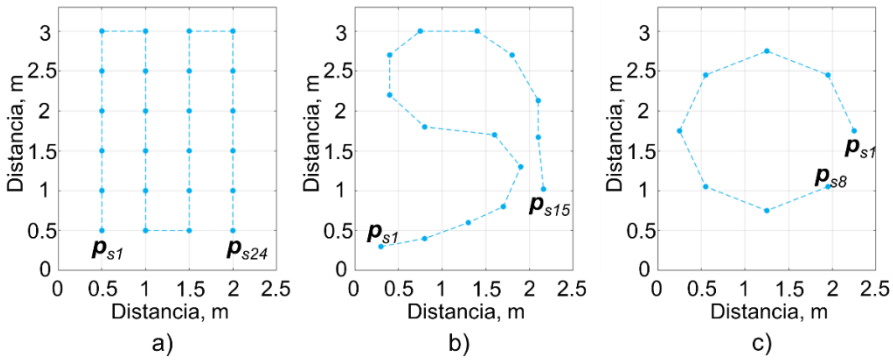


Figura 4-6. Se realizan diez impactos en cada punto de las trayectorias propuestas. a) Trayectoria 1 b) Trayectoria 2 c) Trayectoria 3

La Figura 4-7 muestra las estimaciones de las tres trayectorias con y sin el ajuste de los TOA visto en la sección anterior. La Figura 4-8 muestra el RMSE de cada ubicación de las tres trayectorias. Se observa que sin el ajuste, se tienen errores de estimación más altos.

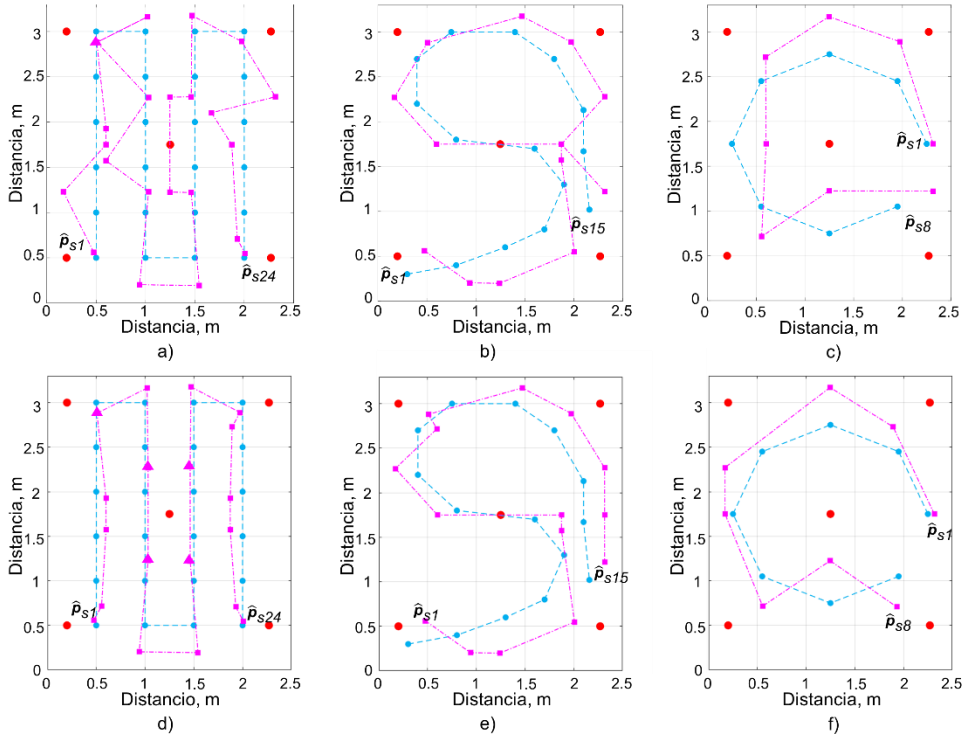


Figura 4-7. Trayectorias estimadas. a), b), c) sin ajuste de los TOA. d), e), f) con ajuste de los TOA. Los círculos azules representan las trayectorias reales; los cuadrados rosas denotan las trayectorias estimadas, y los triángulos rosas indican ubicaciones donde coinciden dos estimaciones de dos diferentes pisadas

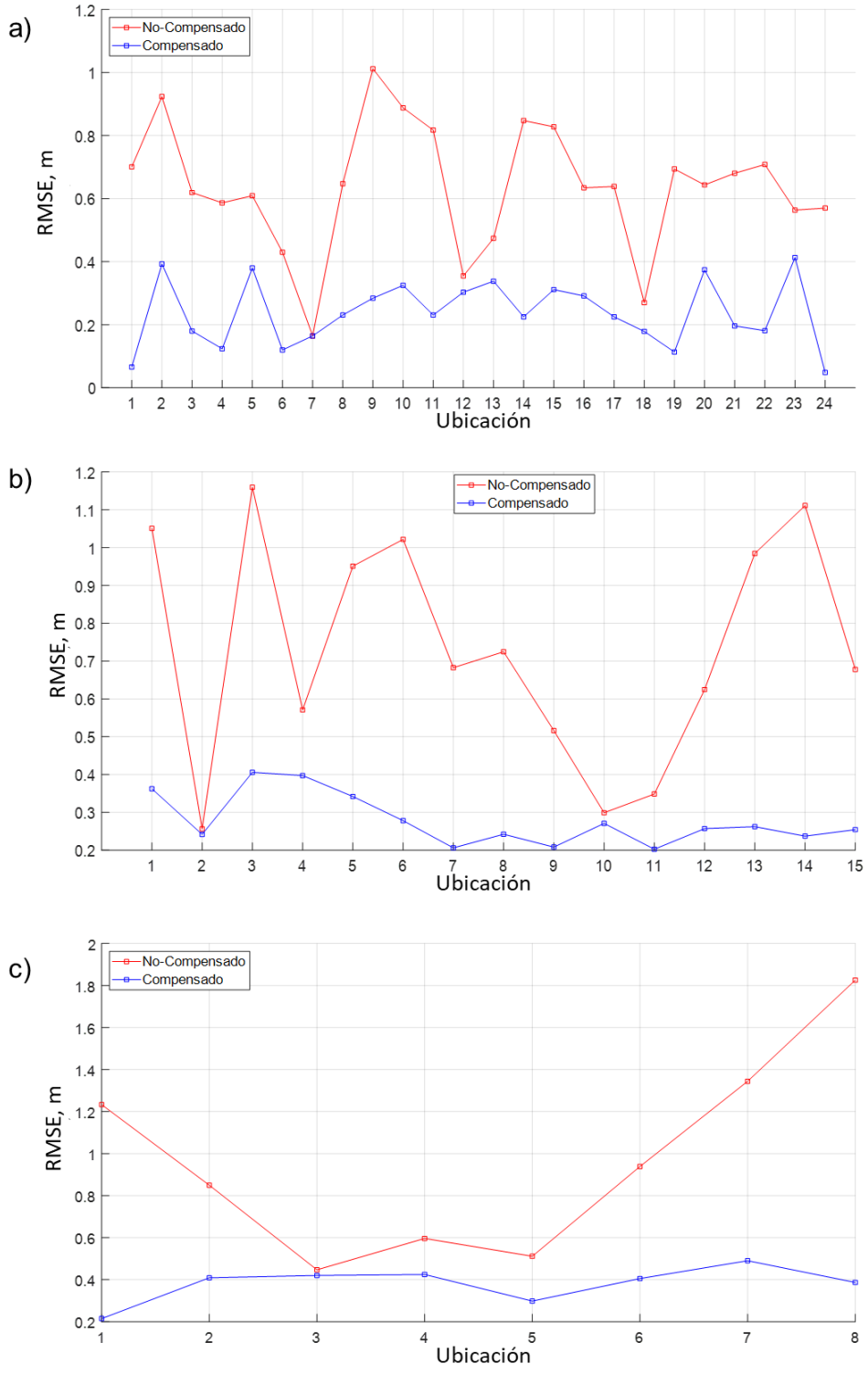


Figura 4-8. Comparación del RMSE para diferentes ubicaciones antes y después de ajustar los TOA. a) Trayectoria 1, b) Trayectoria 2, c) Trayectoria 3

4.3.1 Resultados de la red optimizada

A continuación, se repiten las pruebas anteriores para los sensores colocados en las ubicaciones optimizadas con el algoritmo HPSGWO ($\boldsymbol{S_{opt}} = [2.04\ 1.56\ 1.80\ 3.10\ 1.20\ 1.53\ 0.20\ 1.54\ 1.82\ 0.30]$). La Figura 4-9 muestra las trayectorias obtenidas para ambos casos (optimizado y no optimizado). La Figura 4-10 muestra el RMSE de cada ubicación de las tres trayectorias. Se observa que la optimización reduce el error de estimación en el 74.46% de las ubicaciones propuestas (reduce el error en 35 de 47 ubicaciones). Por otra parte, la Tabla 4-3 muestra el RMSE promedio de cada trayectoria. Se observa que, después de la optimización, el RMSE se reduce entre 18.24% y 46.78% para las tres trayectorias propuestas.

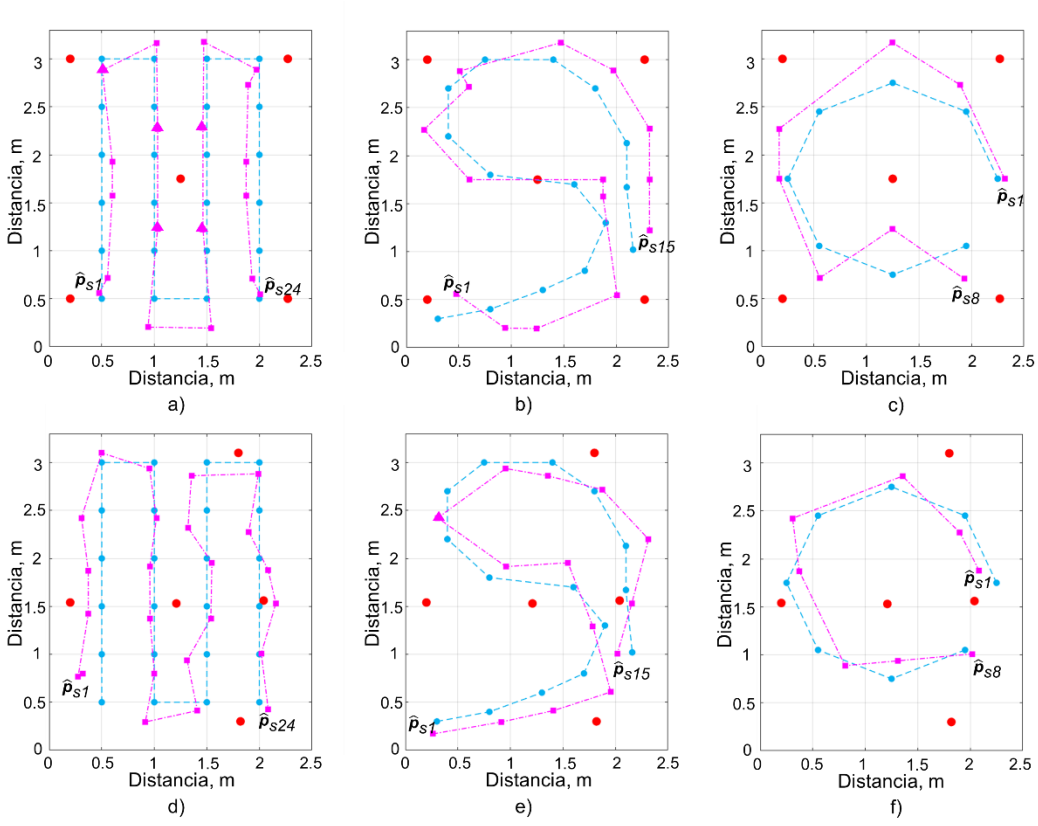


Figura 4-9. Trayectorias estimadas. a), b), c) red sin optimizar. d), e), f) red optimizada. Los círculos azules representan las trayectorias reales; los cuadrados rosas denotan las trayectorias estimadas, y los triángulos rosas indican ubicaciones donde coinciden dos estimaciones de dos diferentes pisadas

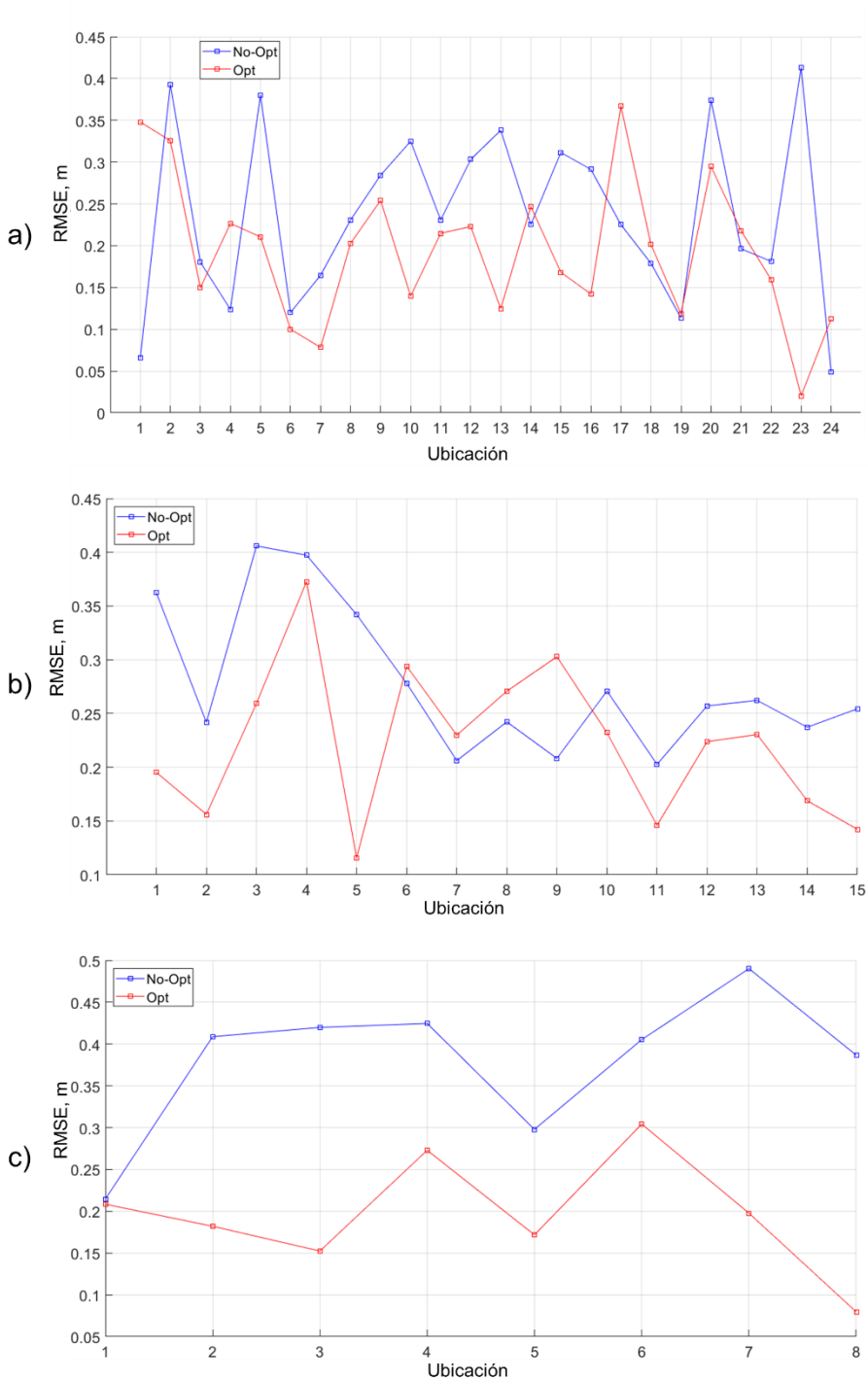


Figura 4-10. Comparación del RMSE para diferentes ubicaciones antes y después de optimizar la red de sensores. a) Trayectoria 1, b) Trayectoria 2, c) Trayectoria 3

	Trayectoria 1 RMSE (m)	**Trayectoria 2 RMSE (m)**	**Trayectoria 3 RMSE (m)**
No-Optimizada	0.259	0.285	0.389
Optimizada (HPSGWO)	0.211	0.233	0.207
Reducción del Error	*18.53%*	*18.24%*	*46.78%*

Tabla 4-3. Comparación del RMSE antes y después de la optimización para las tres trayectorias propuestas

4.4 Conclusiones del capítulo

Se probó de manera experimental el sistema inalámbrico para la adquisición de señales producidas por los impactos de los pies de una persona sobre un piso de concreto, utilizando acelerómetros colocados en un entorno cotidiano.

De acuerdo con los resultados obtenidos en este capítulo, se puede concluir que la optimización de la red de sensores ayuda a reducir el error de localización de pisadas en interiores. La optimización se hace para elegir el número de sensores, así como su ubicación dentro de la habitación.

De igual forma, se observó que al ser una red inalámbrica se presenta una asincronía en los TOA de los nodos, la cual provocaba grandes errores de localización. Esta asincronía se corrigió de manera experimental, sin embargo, debe estudiarse con más detalle sus causas para futuras aplicaciones de los módulos *XBee3*.

5 DISCUSIÓN

Los resultados obtenidos en el capítulo anterior muestran que, cuando se optimiza la red de sensores, efectivamente se tiene una disminución del error de localización. Aunque puede parecer que no existe mucha diferencia entre optimizar y no optimizar la red, dado que se tiene una disminución del error de entre 5 y 18 cm para las tres trayectorias propuestas, en la Tabla 5-1 se muestra la comparación del error para una red no optimizada de 6 sensores y una red optimizada de 5 sensores para las mismas tres trayectorias (la Figura 5-1 muestra la comparación de las trayectorias estimadas para ambos casos). Se observa que la optimizada entrega un error de localización menor de entre 18.73% y 37.34% (entre 5.3 y 12.3 cm) para las tres trayectorias propuestas, aun cuando se tiene un sensor menos.

La comparación anterior muestra que la optimización, además de reducir el error de localización, puede ayudar a reducir el costo del sistema, que, para este caso en particular, sería una reducción del 16.66% ($84.33 USD (ver Tabla 3-4)).

	Trayectoria 1	**Trayectoria 2**	**Trayectoria 3**
6 sensores No-Optimizada	0.2691 m	0.2867 m	0.3304 m
5 sensores Optimizada	0.211 m	0.233 m	0.207 m
Reducción del Error	*21.59%*	*18.73%*	*37.34%*

Tabla 5-1. Comparación del RMSE para una red no optimizada de 6 sensores y una red optimizada de 5 sensores

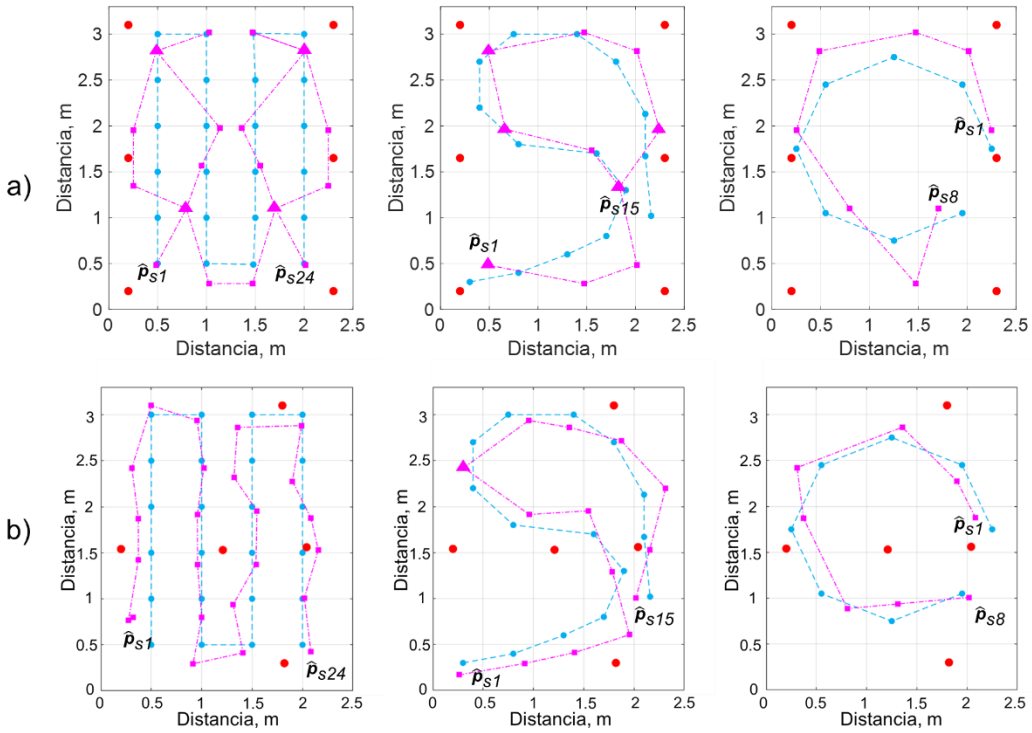


Figura 5-1. Trayectorias estimadas para: a) 6 sensores sin optimizar. b) 5 sensores optimizados. Los círculos azules representan las trayectorias reales; los cuadrados rosas denotan las trayectorias estimadas, y los triángulos rosas indican ubicaciones donde coinciden dos estimaciones de dos diferentes pisadas

La Tabla 5-2 resume las características y resultados de diferentes trabajos, incluyendo el presente estudio. Se observa que casi todos reportan errores submétricos, sin embargo, ninguno de ellos estudia la ubicación y número de sensores.

En este trabajo, al igual que en [9] y [10], los sensores fueron colocados sobre el piso de la habitación, lo cual puede ser un inconveniente ya que los sensores se convierten en obstáculos al momento de caminar. Sin embargo, se podría realizar la optimización de la ubicación de los sensores agregando restricciones a los algoritmos de optimización, por ejemplo, buscar la ubicación óptima, pero cuidando que los sensores queden debajo de los muebles de la habitación.

Lo anterior requeriría un estudio del comportamiento de la red cuando se tienen muebles dentro de la habitación, ya que, tanto en este trabajo como en la literatura reportada, se realizaron pruebas en habitaciones vacías.

Método	Sensor	No. de Sensores	Ubicación de los Sensores	Dimensiones de la habitación	No. De Pisadas	RMSE (m)	Referencia
TDOA	Acelerómetro Piezoeléctrico	12	No Optimizada*	25.5 m x 9.4 m	30	0.590	Poston, et al. [1]
RSS	Acelerómetro Piezoeléctrico	4	No Optimizada*	3 m x 1 m	13	0.253**	Alajlouni, et al. [6]
RSS	Acelerómetro Piezoeléctrico	6	No Optimizada*	13 m x 2 m	66	0.845	Alajlouni & Tarazaga [7]
RSS	Acelerómetro Piezoeléctrico	11	No Optimizada*	16 m x 2 m	162	1.020	Alajlouni & Tarazaga [8]
SO-TDOA	Acelerómetro Piezoeléctrico /Capacitivo	9	No Optimizada	3.6 m x 5.4 m	7	0.475	Bahroun, et al. [9]
ATDOA	Sismómetro	4	No Optimizada	4 m x 3 m	12	0.270**	Li, et al. [10]
SO-TDOA	Acelerómetro Capacitivo	5	No Optimizada	2.5 m x 3.3 m	24	0.259	Trabajo presente
SO-TDOA	Acelerómetro Capacitivo	5	Optimizada	2.5 m x 3.3 m	24	0.211	Trabajo presente

Tabla 5-2. Resumen de las características y resultados de diferentes trabajos. *Realizado en Goodwin Hall. **Error promedio

6 CONCLUSIONES GENERALES

En este trabajo se realizó una búsqueda de los diferentes métodos de localización de personas en interiores. Existen métodos invasivos y no invasivos. Uno de los objetivos de este proyecto era obtener la localización de pisadas con un método no invasivo, por lo que elegimos la localización basada en acelerómetros. Se encontró que, para vibraciones en el piso, no es recomendable utilizar los métodos clásicos de localización, ya que estos fueron pensados para aplicaciones donde el medio de propagación es el aire, es decir, para medios no dispersivos. Por lo que se seleccionó el algoritmo SO-TDOA para hacer la localización, ya que este algoritmo sólo requiere de los tiempos de llegada de la señal a los sensores.

Se propuso una red inalámbrica de sensores que, a diferencia de la literatura mencionada a lo largo de este trabajo, no requiere enviar y procesar una gran cantidad de información para localizar a una persona dentro de una habitación.

En la literatura revisada, no se suele dar mayor importancia al número y ubicación de los sensores de la red. En este trabajo se realizó la optimización de la red para minimizar el error de localización en toda la habitación, por lo que se optimizó el número de sensores y su ubicación dentro de la habitación. Para la optimización se emplearon algunas de las metaheurísticas bio-inspiradas más populares, tal es el caso de la Colonia Artificial de Abejas (ABC), Optimización por Enjambre de Partículas (PSO), Optimización de Lobo Gris (GWO) y el algoritmo híbrido PSO-GWO (HPSGWO). Todas ellas redujeron el error de localización en más de un 25% comparado con una red no optimizada.

Con la optimización de la ubicación de los sensores dentro de la habitación, se demostró que ésta puede ayudar a mejorar el desempeño del algoritmo de localización SO-TDOA. Además, el estudio del número de sensores mostró que el error de localización para toda la habitación tiene una disminución parecida a una exponencial a medida que se incrementa el número de sensores. Si bien es deseable tener el menor error usando el menor número de sensores, se debe seleccionar el número de sensores de acuerdo con la aplicación que se le dará al sistema.

En las pruebas experimentales del sistema propuesto, se observó una asincronía en los TOA de los nodos, la cual fue corregida de manera experimental, sin embargo, no se realizó un estudio a profundidad, por lo que quedará como trabajo a futuro.

Las pruebas experimentales, también mostraron que, luego de la optimización, el RMSE de localización disminuyó entre un 18.25% y un 46.78% para tres trayectorias propuestas.

Por otra parte, se comparó el RMSE para una red no optimizada de 6 sensores con una red optimizada de 5 sensores y se observó que la optimizada entrega un error de localización menor, entre un 18.73% y un 37.34% para las tres trayectorias propuestas, a pesar de tener un sensor menos.

Por lo tanto, se recomienda optimizar tanto el número de sensores a emplear como su ubicación antes de implementar algún algoritmo de localización. Esto permitirá tener un mejor seguimiento de la trayectoria seguida por una persona dentro de una habitación, además de reducir los costos del sistema.

Finalmente, los resultados indican que el sistema propuesto puede servir como base para el análisis de caminatas reales. Sin embargo, se necesitan pruebas adicionales para su aplicación debido a factores como la variabilidad en la marcha de diferentes personas, interferencias ambientales como muebles en la habitación, y la complejidad de procesar pisadas más rápidas y

más frecuentes en comparación con los impactos aislados utilizados en las pruebas del sistema actual.

7 TRABAJO A FUTURO

- Realizar experimentos en habitaciones con obstáculos (muebles) y con caminatas reales.

- Mejorar el sistema para localizar a una persona en tiempo real, incluyendo una interfaz amigable para el usuario.

- Realizar un estudio para localizar a más de una persona dentro de la habitación.

- Proponer un sistema de bajo costo en el que los sensores se encuentren montados bajo el piso de la habitación para evitar que sean un obstáculo para los ocupantes.

- Realizar un estudio de la asincronía de los módulos *XBee3* para redes inalámbricas.

8 REFERENCIAS

[1] J. D. Poston, R. M. Buehrer, and P. A. Tarazaga, “Indoor footstep localization from structural dynamics instrumentation,” *Mech Syst Signal Process*, vol. 88, pp. 224–239, 2017, doi: https://doi.org/10.1016/j.ymssp.2016.11.023.

[2] H. Lee, J. W. Park, and A. Helal, “Estimation of Indoor Physical Activity Level Based on Footstep Vibration Signal Measured by MEMS Accelerometer for Personal Health Care Under Smart Home Environments,” *Mobile Entity Localization and Tracking in GPS-less Environnments*, vol. 5801, pp. 148–162, 2009.

[3] S. A. Zekavat (Reza) and R. M. Buehrer, *Handbook of Position Location Theory, Practice, and Advances*, 1st ed. John Wiley & Sons, 2019.

[4] S. Drira and I. F. C. Smith, “A framework for occupancy detection and tracking using floor-vibration signals,” *Mech Syst Signal Process*, vol. 168, 2022, doi: https://doi.org/10.1016/j.ymssp.2021.108472.

[5] C. Liu, L. Xiey, C. Wangy, J. Wu, and S. Luy, “Track Your Foot Step: Anchor-Free Indoor Localization Based on Sensing Users’ Foot Steps,” in *2016 IEEE 13th International Conference on Mobile Ad Hoc and Sensor Systems*, 2016, pp. 201–209.

[6] S. Alajlouni, M. Albakri, and P. Tarazaga, “Impact localization in dispersive waveguides based on energy-attenuation of waves with the traveled distance,” *Mech Syst Signal Process*, vol. 105, pp. 361–376, 2018, doi: https://doi.org/10.1016/j.ymssp.2017.12.007.

[7] S. Alajlouni and P. Tarazaga, “A new fast and calibration-free method for footstep impact localization in an instrumented floor,” *Journal of Vibration and Control*, vol. 25, no. 10, pp. 1–10, 2019, doi: https://doi.org/10.1177/1077546319829943.

[8] S. Alajlouni and P. Tarazaga, “A passive energy-based method for footstep impact localization, using an underfloor accelerometer sensor network with Kalman filtering,” *Journal of Vibration and Control*, vol. 26, no. 11–12, pp. 941–951, 2020, doi: https://doi.org/10.1177/1077546319890520.

[9] R. Bahroun, O. Michel, F. Frassati, M. Carmona, and J. L. Lacoume, “New algorithm for footstep localization using seismic sensors in an indoor environment,” *J Sound Vib*, vol. 333, no. 3, pp. 1046–1066, 2014, doi: https://doi.org/10.1016/j.jsv.2013.10.004.

[10] F. Li, J. Clemente, M. Valero, Z. Tse, S. Li, and W. Z. Song, “Smart Home Monitoring System via Footstep-Induced Vibrations,” *IEEE Syst J*, vol. 14, no. 3, pp. 3383–3389, 2020, doi: https://doi.org/10.1109/JSYST.2019.2937960.

[11] B. Vigna, P. Ferrari, F. F. Villa, E. Lasalandra, and S. Zerbini, *Silicon Sensors and Actuators. The Feynman Roadmap*, 1st ed. Springer Cham, 2022. doi: https://doi.org/10.1007/978-3-030-80135-9.

[12] P. Regtien and E. Dertien, *Sensors for Mechatronics*, 2nd ed. Elsevier, 2018.

[13] S. M. Ziola and M. R. Gorman, “Source location in thin plates using cross-correlation,” *Journal of Acoustical Society of America*, no. 90, pp. 2551–2556, 1991.

[14] J. M. Hamilton, “Design and Implementation of Vibration Data Acquisition in Goodwin Hall for Structural Health Monitoring, Human Motion, and Energy Harvesting Research,” Blacksburg, 2015.

[15] PCB Piezotronics, “Model 352B. Installation and Operating Manual,” NY, 2002.

[16] S. S. Rao, *Engineering optimization: Theory and Practice*, 5th ed. John Wiley & Sons, 2020.

[17] K.-L. Du and M. Swamy, *Search and Optimization by Metaheuristics Techniques and Algorithms Inspired by Nature*, 1st ed. Birkhäuser Cham, 2016. doi: https://doi.org/10.1007/978-3-319-41192-7.

[18] L. Liberti and N. Maculan, Eds., *Global Optimization. From Theory to Implementation*, 1st ed. Springer New York, NY, 2006. doi: https://doi.org/10.1007/0-387-30528-9.

[19] E.-G. Talbi, *Metaheuristics: from design to implementation*. John Wiley & Sons, 2009.

[20] F. Rothlauf, *Design of Modern Heuristics. Principles and Application*. Springer Berlin, Heidelberg, 2011. doi: https://doi.org/10.1007/978-3-540-72962-4.

[21] M. Cavazzuti, *Optimization methods. From Theory to Design Scientific and Technological Aspects in Mechanics*, 1st ed. Springer Berlin Heidelberg, 2013. doi: https://doi.org/10.1007/978-3-642-31187-1.

[22] S. Mirjalili, S. M. Mirjalili, and A. Lewis, “Grey Wolf Optimizer,” *Advances in Engineering Software*, vol. 69, pp. 46–61, 2014, doi: 10.1016/j.advengsoft.2013.12.007.

[23] I. M. El-Hasnony, S. I. Barakat, and R. R. Mostafa, “Optimized ANFIS Model Using Hybrid Metaheuristic Algorithms for Parkinson’s Disease Prediction in IoT Environment,” *IEEE Access*, vol. 8, pp. 119252–119270, 2020, doi: https://doi.org/10.1109/ACCESS.2020.3005614.

[24] D. Halliday and R. Resnick, *Physics. Parts 1 and 2,*. John Wiley and Sons, 1978.

[25] F. Ciampa and M. Meo, “Acoustic emission source localization and velocity determination of the fundamental mode A0 using wavelet analysis and a Newton-based optimization technique,” *Smart Mater Struct*, vol. 19, no. 4, 2010, doi: https://doi.org/10.1088/0964-1726/19/4/045027.

[26] W. Zhao, L. Wang, and Z. Zhang, *New optimization algorithms and their applications*, 1st ed. Elsevier, 2021.

[27] F. A. Şenel, F. Gökçe, A. S. Yüksel, and T. Yiğit, “A novel hybrid PSO–GWO algorithm for optimization problems,” *Eng Comput*, vol. 35, no. 4, pp. 1359–1373, 2019, doi: https://doi.org/10.1007/s00366-018-0668-5.

[28] MathWorks, “peaks - Function for creating a sample surface.” Accessed: Jul. 22, 2024. [Online]. Available: https://www.mathworks.com/help/matlab/ref/peaks.html

[29] D. H. Wolpert and W. G. Macready, “No Free Lunch Theorems for Optimization,” *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, 1997, doi: https://doi.org/10.1109/4235.585893.

[30] J. Kennedy and R. Eberhart, “Particle swarm optimization,” in *Proceedings of ICNN’95 - International Conference on Neural Networks*, IEEE, 1995, pp. 1942–1948. doi: https://doi.org/10.1109/ICNN.1995.488968.

[31] D. Karaboga and B. Basturk, “A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm,” *Journal of Global Optimization*, vol. 39, no. 3, pp. 459–471, 2007, doi: https://doi.org/10.1007/s10898-007-9149-x.

[32] D. Cekus and D. Skrobek, “The influence of inertia weight on the Particle Swarm Optimization algorithm,” *Journal of Applied Mathematics and Computational Mechanics*, vol. 17, no. 4, pp. 5–11, 2018, doi: http://dx.doi.org/10.17512/jamcm.2018.4.01.

[33] R. C. Eberhart and Y. Shi, “Comparing Inertia Weights and Constriction Factors in Particle Swarm Optimization,” in *Proceedings of the 2000 Congress on Evolutionary Computation. CEC00 (Cat. No.00TH8512)*, 2000, pp. 84–88. doi: https://doi.org/10.1109/CEC.2000.870279.

[34] Analog Devices, “Low Noise, Low Drift, Low Power, 3-Axis MEMS Accelerometers. ADXL354/ADXL355 datasheet,” 2020. Accessed: Jan. 01, 2023. [Online]. Available: https://www.analog.com/media/en/technical-documentation/data-sheets/adxl354_adxl355.pdf

[35] DIGI, “DIGI XBEE3 ZIGBEE 3.0, Easy-to-add connectivity in a compact, low-power, low-profile footprint.” Accessed: Jan. 01, 2023. [Online]. Available: https://www.digi.com/products/embedded-systems/digi-xbee/rf-modules/2-4-ghz-rf-modules/xbee3-zigbee-3#specifications

[36] Arduino, “Arduino Store.” Accessed: Jan. 01, 2023. [Online]. Available: https://store.arduino.cc/products/arduino-nano

[37] B. Akay and D. Karaboga, “Parameter Tuning for the Artificial Bee Colony Algorithm,” in *International Conference on Computational Collective Intelligence. ICCCI 2009: Computational Collective Intelligence. Semantic Web, Social Networks and Multiagent Systems*, 2009, pp. 608–619. doi: https://doi.org/10.1007/978-3-642-04441-0_53.

9 ANEXOS

9.1 Anexo A. Métodos clásicos de localización de objetivos

A continuación, se presenta un breve repaso de los métodos tradicionales para localizar objetivos, los cuales pueden estudiarse con mayor detalle en [3].

9.1.1 Localización basada en tiempos de llegada (TOA)

El TOA se define como el tiempo en el que la excitación del medio llega desde el objetivo hasta un sensor y se puede obtener mediante un criterio de umbral, es decir, cuando la amplitud de la señal cruza un valor específico (ver Figura 9-1).

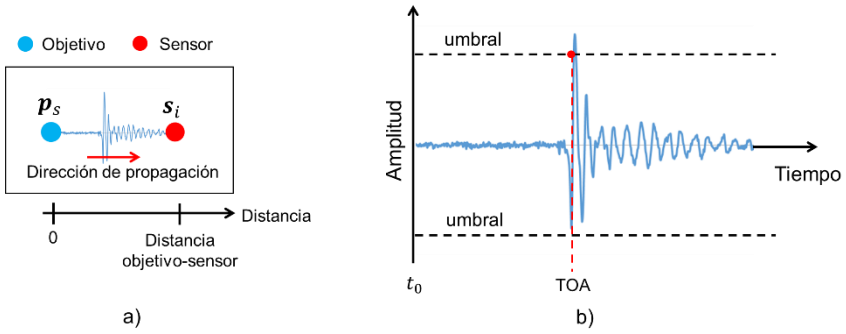


Figura 9-1. a) La excitación se propaga desde el objetivo ubicado en p_s hasta el sensor i ubicado en s_i. b) El tiempo de llegada se obtiene con un criterio de umbral y se mide a partir de un tiempo arbitrario t_0.

En el método de localización basado en tiempos de llegada (TOA), se utiliza la técnica de trilateración, la cual consiste en lo siguiente:

Si se produce una excitación en la ubicación desconocida $\boldsymbol{p}_s = (p_{sx}, p_{sy})$ y se tienen tres sensores en las ubicaciones conocidas $\boldsymbol{s}_i = (s_{ix}, s_{iy})$, donde $i = 1, 2, 3$, entonces la distancia entre el objetivo y el sensor i está dada por

$$d_{si} = t_{si}c = \sqrt{(s_{ix} - p_{sx})^2 + (s_{iy} - p_{sy})^2} \qquad (8.1)$$

Donde t_{si} es el TOA del objetivo al sensor i y c es la velocidad de propagación del medio.

Si se conoce la velocidad de propagación y el TOA, entonces (8.1) es un sistema de tres ecuaciones con dos incógnitas. De tal forma que al resolverlo se puede obtener la ubicación $\boldsymbol{p}_s$ del objetivo.

De manera geométrica, esta solución es el punto de intersección de tres círculos con radio d_{si} centrados en las coordenadas (s_{ix}, s_{iy}), tal como se muestra en la Figura 9-2.

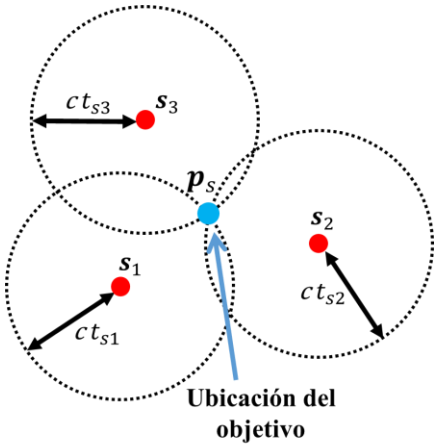


Figura 9-2. Trilateración. La ubicación del objetivo está en la intersección de las tres circunferencias formadas con las distancias objetivo-sensor

9.1.2 Localización basada en las diferencias de los tiempos de llegada (TDOA)

Este método se basa en las diferencias de tiempo de llegada de la señal a cada par de sensores, tomando como referencia a alguno de ellos.

Si se tienen tres sensores en las ubicaciones conocidas $\boldsymbol{s}_i = (s_{ix}, s_{iy})$, donde $i = 1, 2, 3$ y se toma como referencia al sensor 1, el TDOA entre el par de sensores $(1, j)$ está dado por

$$t_{1j} = t_{sj} - t_{s1} = \frac{d_{sj} - d_{s1}}{c} \qquad para\ j = 2, 3$$

Entonces de (1) tenemos que

$$t_{1j} = \frac{\sqrt{(s_{jx} - p_{sx})^2 + (s_{jy} - p_{sy})^2} - \sqrt{(s_{1x} - p_{sx})^2 + (s_{1y} - p_{sy})^2}}{c}$$

Despejando

$$\sqrt{(s_{jx} - p_{sx})^2 + (s_{jy} - p_{sy})^2} - \sqrt{(s_{1x} - p_{sx})^2 + (s_{1y} - p_{sy})^2} = t_{1j}c \qquad (8.2)$$

Si conocemos la ubicación de los sensores, los TOA y la velocidad de propagación, entonces (8.2) es un sistema de dos ecuaciones no lineales con dos incógnitas. De tal forma que al resolverlo se puede obtener la ubicación $\boldsymbol{p}_s$ del objetivo.

Geométricamente, esta solución es el punto de cruce de un par de hipérbolas formadas con las diferencias de tiempo de llegada t_{12} y t_{13}, cuyos focos son la ubicación de los pares de sensores $(1, 2)$ y $(1, 3)$ como se muestra en la Figura 9-3.

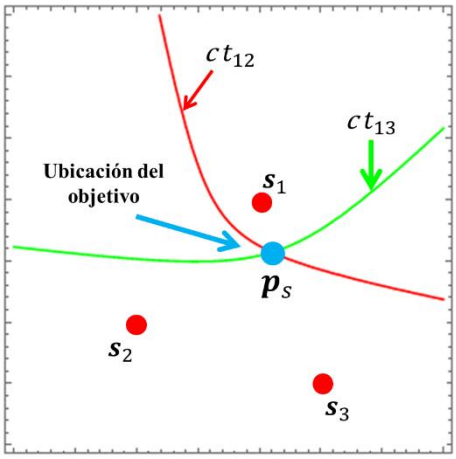


Figura 9-3. Localización basada en TDOA. El objetivo se localiza en la intersección de dos hipérbolas formadas con los TDOA tomando como referencia al sensor más cercano al objetivo.

9.1.3 Localización basada en la fuerza de la señal recibida (RSS)

En este método de localización de objetivos, al igual que en el método TOA, se utiliza trilateración para estimar la ubicación del objetivo. La diferencia está en que, en lugar de medir el TOA, la localización se hace a partir de la amplitud de la señal que reciben los sensores.

Geométricamente, se tienen círculos cuyos radios (distancia objetivo-sensor) están dados por la RSS. Y la ubicación del objetivo es el punto de intersección de los círculos.

9.1.4 Localización basada en el ángulo de llegada (AOA)

En la localización basada en el ángulo de llegada, los sensores deben contar con la tecnología para obtener el ángulo de llegada de la señal. Por ejemplo, para el caso de señales RF, los sensores deben contar con un arreglo

de antenas que, con la ayuda de algoritmos de procesamiento de señales, puedan estimar el ángulo de llegada de la señal.

Supóngase que se tienen dos nodos en las ubicaciones conocidas $\boldsymbol{s}_1 = (s_{1x}, s_{1y})$ y $\boldsymbol{s}_2 = (s_{2x}, s_{2y})$. Si el objetivo (ubicado en la posición desconocida $\boldsymbol{p}_s = (p_{sx}, p_{sy})$) genera una excitación, ésta llegará a los sensores en las direcciones φ_1 y φ_2 como se muestra en la Figura 9-4. Si se toma el eje x como referencia, entones los ángulos están dados por:

$$\tan(\varphi_1) = \frac{p_{sy} - s_{1y}}{p_{sx} - s_{1x}} \quad ; \quad \tan(\varphi_2) = \frac{p_{sy} - s_{2y}}{p_{sx} - s_{2x}} \qquad (8.3)$$

Si los sensores son capaces de estimar el ángulo de llegada de la señal, entonces (8.3) es un sistema de dos ecuaciones con dos incógnitas. De tal forma que, al resolverlo, se obtiene la ubicación $\boldsymbol{p}_s$ del objetivo.

Geométricamente, el objetivo se localiza en el punto de intersección de las dos líneas cuyas direcciones son φ_1 y φ_2.

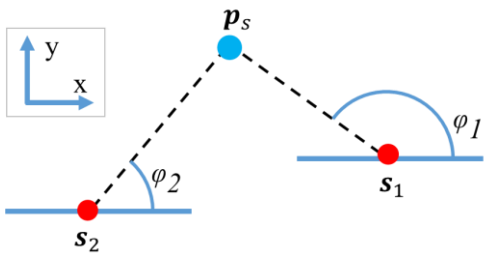


Figura 9-4. Localización basada en AOA. El objetivo se localiza en la intersección de las dos líneas cuyas direcciones son φ_1 y φ_2

9.2 Anexo B. Algunos conceptos de sistemas digitales

Este apartado tiene la intención de dar un breve repaso a algunos conceptos de sistemas digitales para complementar los temas vistos en el capítulo 3.

9.2.1 Señal digital

Una señal digital es aquella que tiene un nivel de tensión discreto en cualquier instante de tiempo. Por ejemplo, idealmente, una señal digital de 3.3V tiene 0 o 3.3 V en cualquier instante de tiempo como se muestra en la Figura 9-5. La tensión de 0V se define como estado bajo o estado 0, mientras que 3.3V se define como estado alto o estado 1. A la transición de la señal al pasar de estado bajo a estado alto se le conoce como flanco de subida, mientras que a la transición de estado alto a estado bajo se le conoce como flanco de bajada.

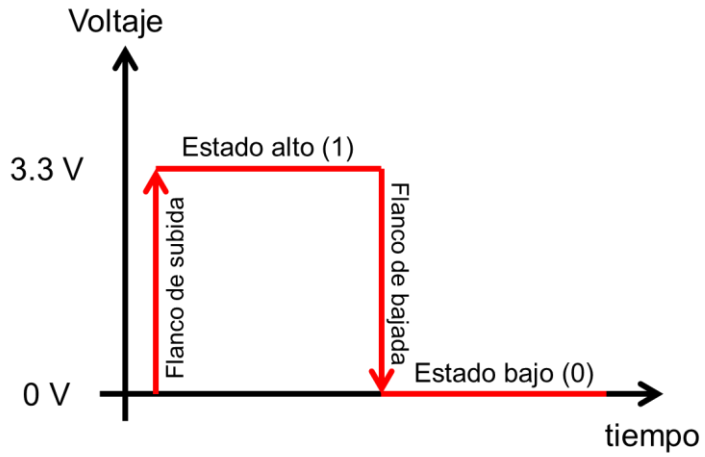


Figura 9-5. Niveles binarios de una señal digital

La señal digital de la Figura 9-5 tiene 2 bits: 10. Normalmente, las señales de datos están formadas por 4 o más bits. A una señal de 4 bits se le

conoce como nibble, mientras que a una de 8 bits se le conoce como byte. La Figura 9-6 muestra una señal de 8 bits (10110100).

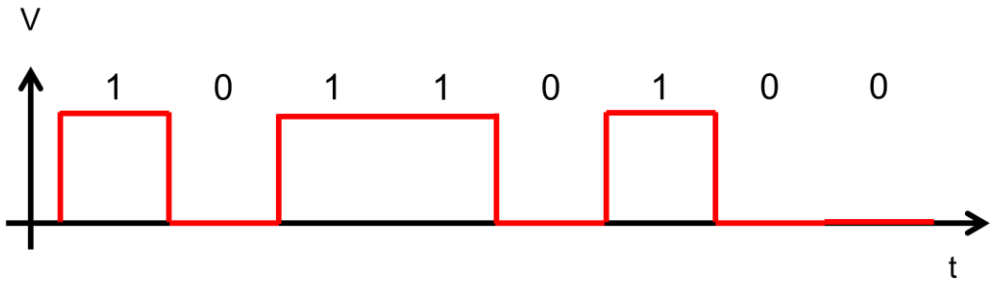


Figura 9-6. Señal digital de 8 bits

9.2.2 Registros

Los registros son circuitos digitales que sirven para almacenar datos. Si se tiene un registro de 8 bits llamado DATA que almacena el dato 00101101, sus bits se verían como en la Figura 9-7, es decir, todos los bits están presentes al mismo tiempo. De tal forma que se pueden cambiar los valores de los bits de manera independiente. Al bit DATA[0] se le conoce como bit menos significativo, mientras que al bit DATA[7] como bit más significativo.

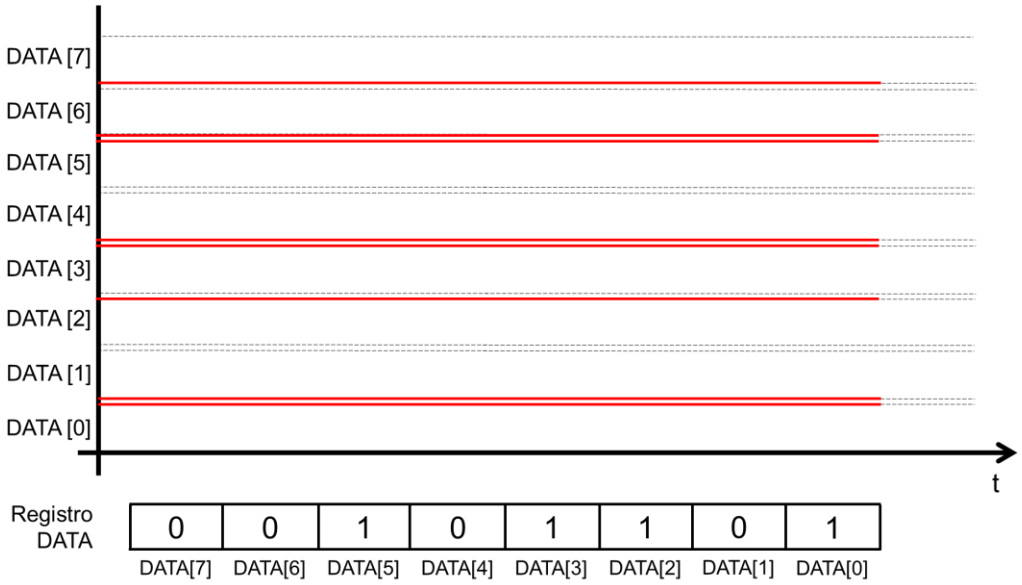


Figura 9-7. Registro DATA y sus bits

Si se quiere almacenar el dato 1001 en un registro de 8 bits, hay dos opciones: justificado a la izquierda y justificado a la derecha, tal como se muestra en la Figura 9-8.

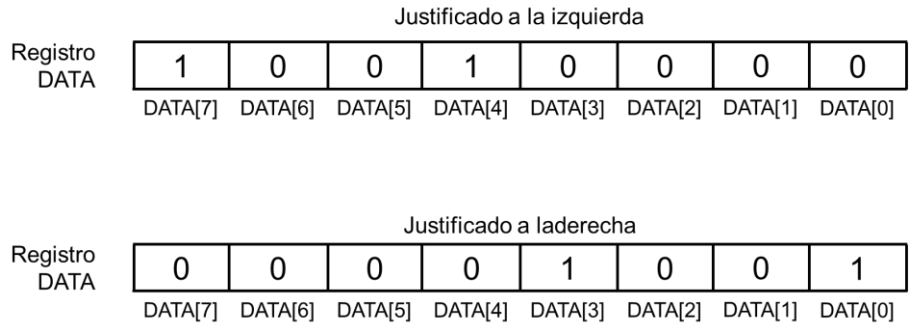


Figura 9-8. El mismo dato almacenado justificado a la izquierda y a la derecha

9.2.3 Sistemas de numeración decimal, binario y hexadecimal

Los humanos contamos en el sistema decimal, el cual tiene dígitos del 0 al 9, sin embargo, en las computadoras se utiliza el sistema binario que consta de dos dígitos, 0 y 1.

Además de los binarios, se suele utilizar el sistema hexadecimal, ya que con este se pueden representar grandes cantidades de información utilizando pocos símbolos. El sistema hexadecimal consta de 16 símbolos, del 0 al 9 y de la A a la F. La equivalencia de los números en binario, decimal y hexadecimal se muestran en la Tabla 9-1.

Para convertir de binario a hexadecimal, se reescribe el número en grupos de 4 bits de derecha a izquierda, es decir, del bit menos significativo al bit más significativo.

Por ejemplo, el número 6546 en decimal, convertido a binario es 1100110010010, si separamos en grupos de 4 bits, tendríamos:

1 1001 1001 0010

En este caso al último grupo de la izquierda le faltan tres bits para completar 4, entonces se completa con 0s.

0001 1001 1001 0010

Finalmente, se realiza la conversión a hexadecimal de acuerdo a la Tabla 9-1. Por lo que 6546 en decimal equivale a 1992 en hexadecimal.

Para evitar confusión entre los números del sistema decimal y del hexadecimal, estos últimos suelen escribirse con el prefijo “0x”, entonces 6546 es equivalente en hexadecimal a 0x1992. Por su parte, los binarios pueden distinguirse por el prefijo “0b” o bien por el subíndice 2 después del número.

$6546 = 1100110010010_2 = 0x1992$

Numero en hexadecimal	Equivalente en decimal	Equivalente en binario de 4 dígitos
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111

Tabla 9-1. Números Hexadecimales y sus equivalencias

9.2.4 Complemento a dos

El complemento a dos es el tipo de codificación utilizado por la mayoría de los microprocesadores, con el cual se pueden representar números enteros con signo (positivos, negativos y cero). Cuando el bit más significativo (MSB) tiene valor 0, se trata de un número positivo, mientras que si el MSB es 1, se

trata de un número negativo. Para obtener un número negativo en complemento a dos, se realizan las siguientes operaciones:

1) Se realiza la operación NOT a cada bit
2) Se suma un 1 al resultado

Por ejemplo, el +5 en 4 bits es 0101, mientras que su negativo en complemento a dos es:

$$C_2(5) = NOT(0101) + 1$$

$$C_2(5) = 1010 + 1$$

$$C_2(5) = 1011$$

El intervalo de valores en decimal que pueden ser representados en complemento a dos es $-[2^{n-1}]$ a $+[2^{n-1} - 1]$, donde n es el número de bits del sistema.

Por ejemplo, para las lecturas del acelerómetro de 20 bits, el intervalo de valores que se pueden representar son de -524288 a +524287.

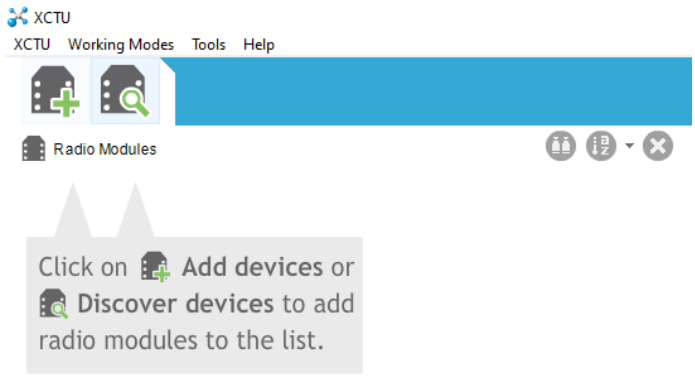
Utilizando la ecuación 3-2 se tienen un intervalo de -2.048 a +2.047 g.

9.3 Anexo C. Configuración de los módulos XBee3

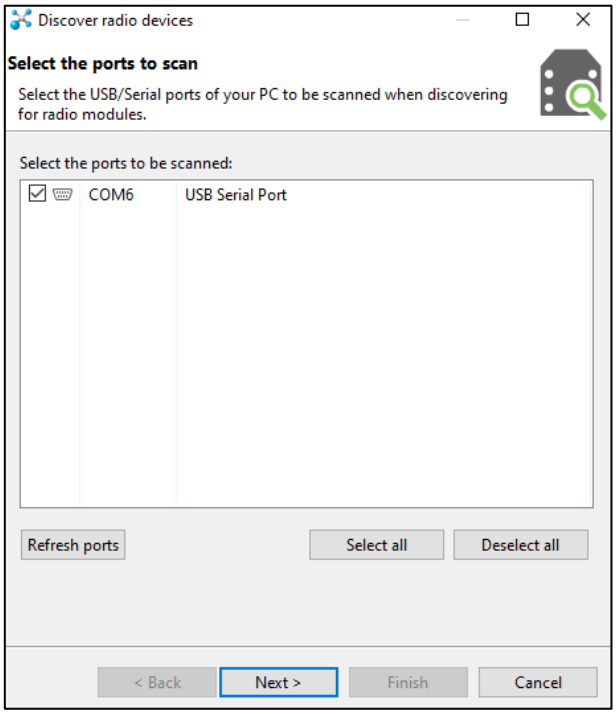
Para configurar los módulos XBee3 se utiliza el software XCTU en conjunto con la SparkFun XBee Explorer Dongle.

A continuación se presentan los parámetros a configurar en cada dispositivo de la red de sensores.

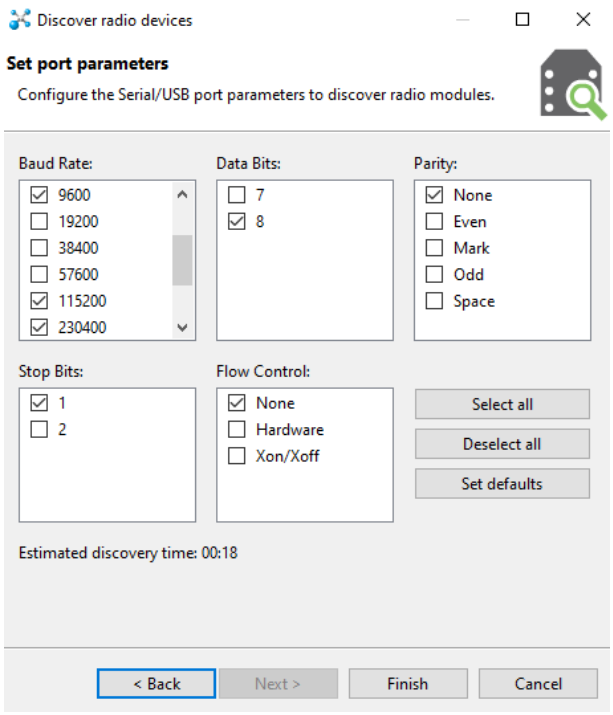
Se busca un nuevo dispositivo



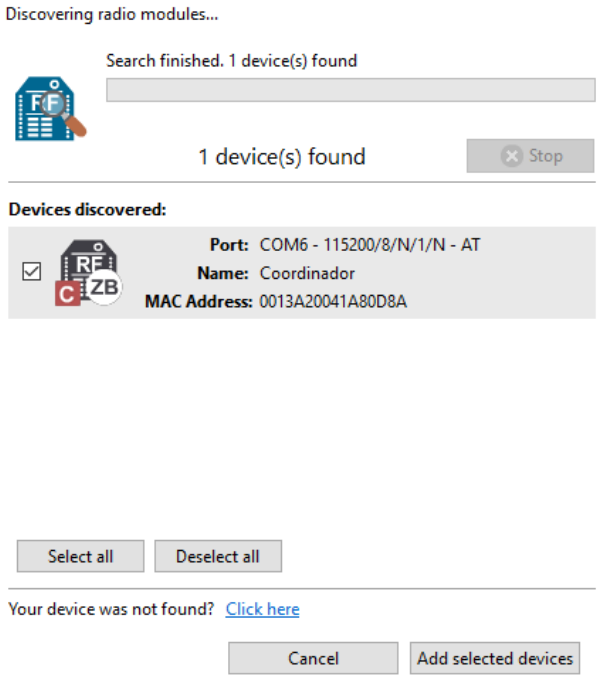
Y se selecciona el puerto del módulo XBee3.



Luego se seleccionan las características del dispositivo que se desea encontrar.

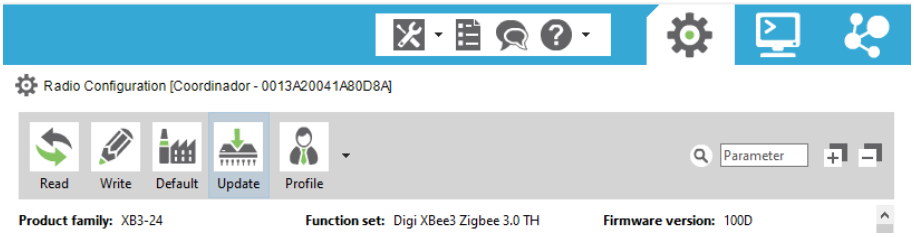


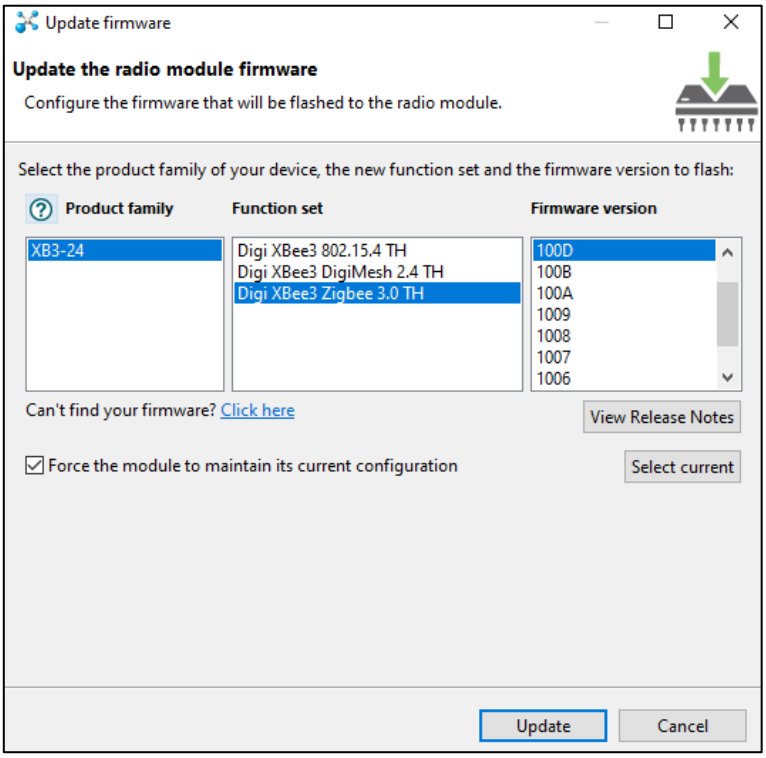
A continuación, se agrega el dispositivo encontrado



y se realizan las siguientes configuraciones:

Primero se actualiza el Firmware, seleccionando el tipo de red Zigbee 3.0 y la versión 100D (todos los dispositivos de la red deberán tener la misma configuración).

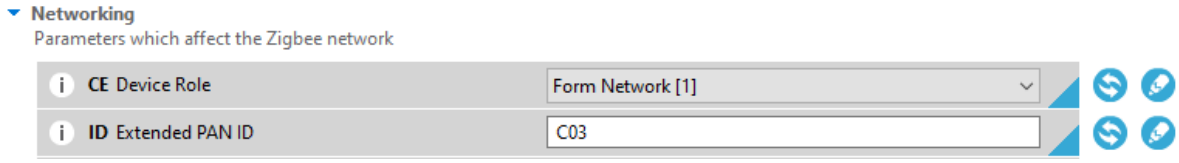




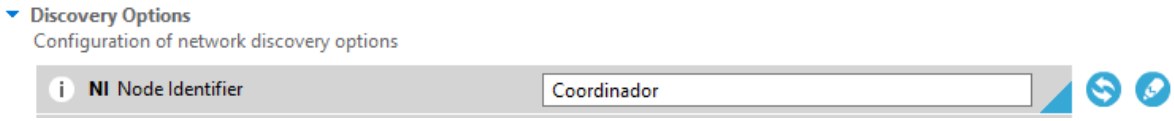
Luego, dependiendo de rol del dispositivo, se configuran los siguientes parámetros:

9.3.1 Coordinador

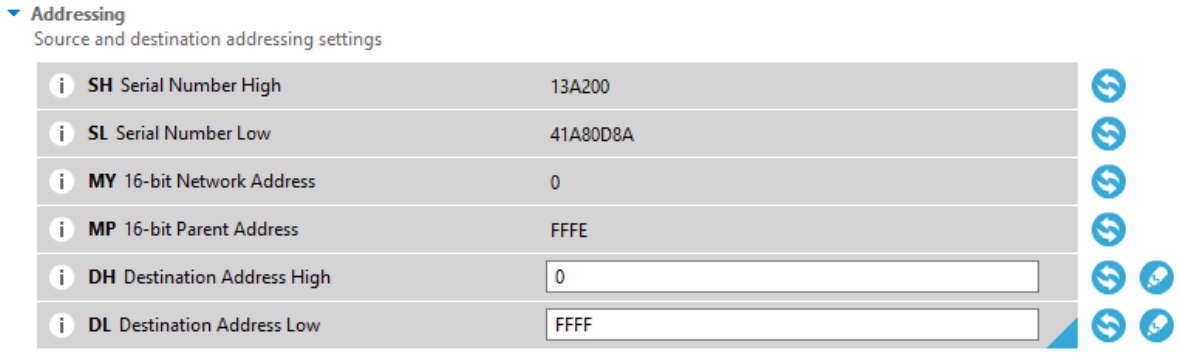
- Se configura el rol del dispositivo como coordinador o formador de la red.
- Se define el identificador de la red (Extended PAN ID) para que sólo se puedan unir los dispositivos con ese ID.



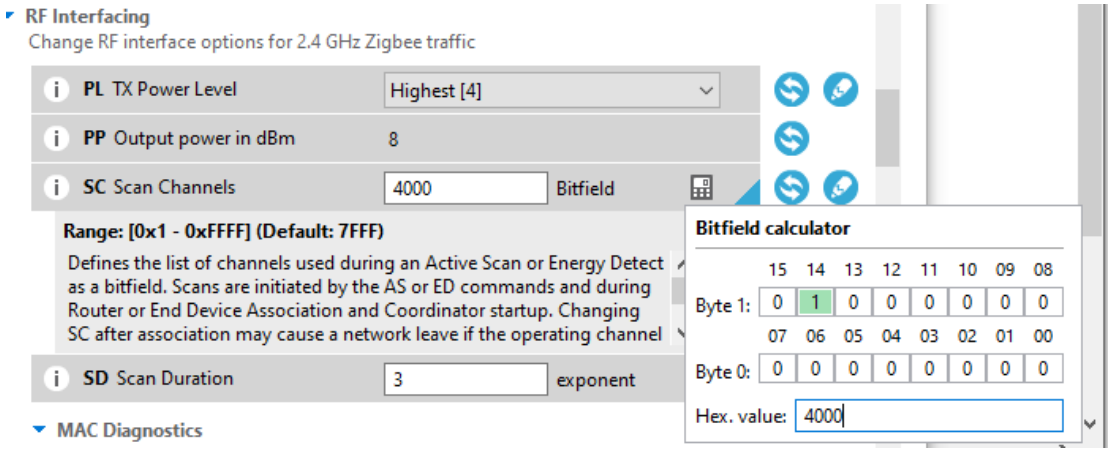
- Se configura el identificador del nodo (puede ser cualquier valor, sólo sirve para saber “en lenguaje humano” de qué nodo se trata



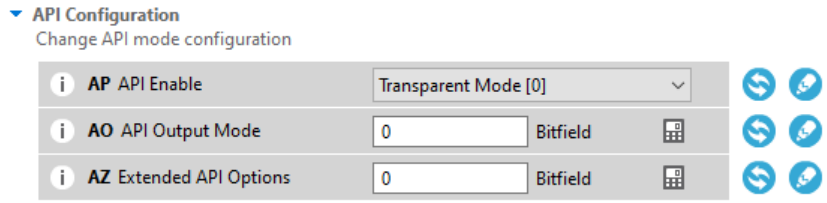
- Se configura como dirección de destino 0x0000 0000 0000 FFFF para que haga una transmisión Broadcast



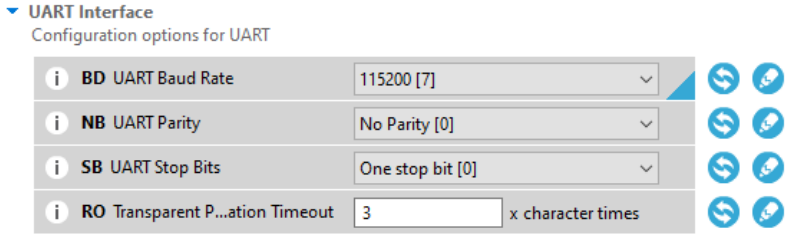
- Definimos algún canal en el cual buscar o formar la red (sólo se deja un canal para que todos los dispositivos se encuentren. Originalmente busca en todos los 26 canales (7FFF) y se asigna cualquiera de ellos a la red). En este caso se eligió el canal 14



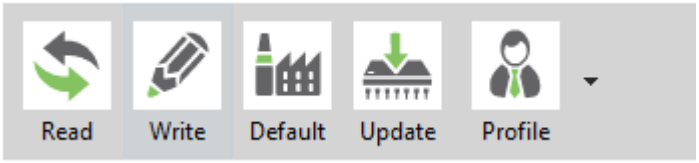
- Definimos el modo transparente



- Finalmente se configuran las características de la comunicación serial, a una velocidad de 115200 bauds, no paridad y un bit de parada

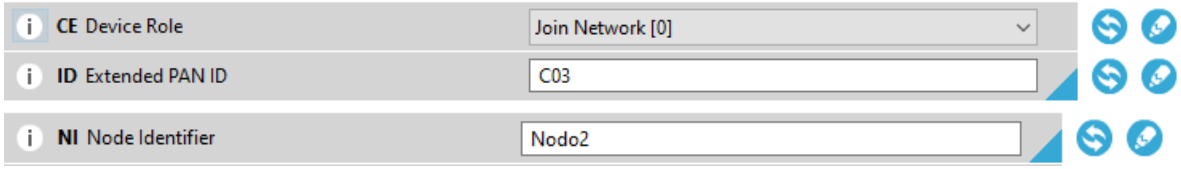


Una vez seleccionados los parámetros anteriorres, se escriben en el dispositivo

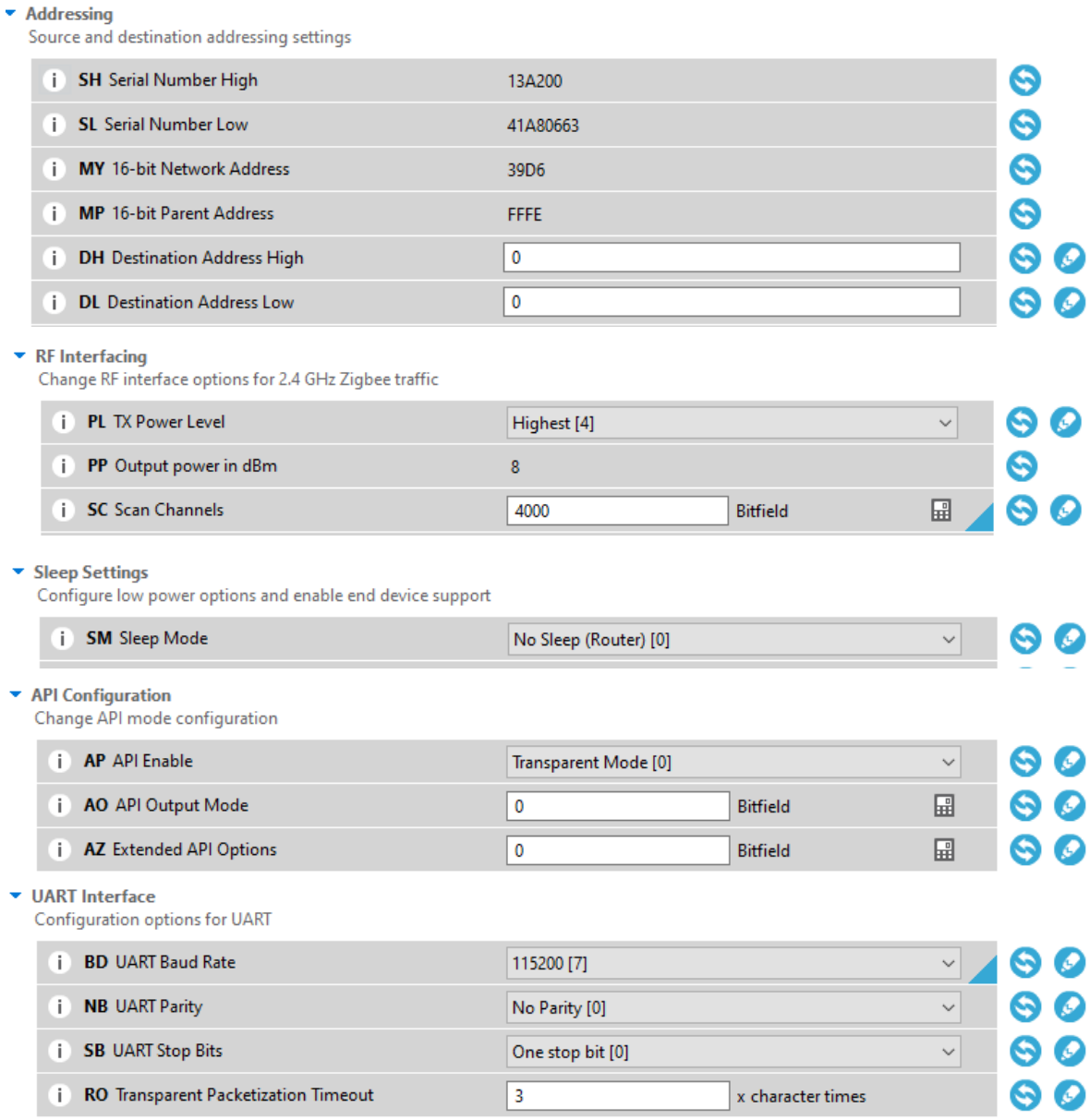


9.3.2 Nodos

- Se configura similar al coordinador, sólo cambian un par de parámetros. En este caso los nodos son tipo Router, por lo que se selecciona Join Network y se escribe el identificador del Nodo correspondiente.



- Se configura como dirección de destino la del coordinador (0)



Con esta configuración, el coordinador envía datos a todos los nodos al mismo tiempo y todos los nodos pueden enviar datos unicamente al coordinador.

9.4 Anexo D. Códigos para la red de sensores

A continuación, se presentan los códigos para la red de sensores. En caso de algún error, favor de comunicarse con el autor.

9.4.1 Programa en Arduino para los Nodos

```
//En este programa se configuran los registros del sensor ADXL355,
//Este programa espera a que se le envíe una bandera ('a') a través del puerto
serie, una vez recibida la bandera

#include <Wire.h>
#include <math.h>

//ACELEROMETRO
#define ADXL355 0x1D // dirección del acelerómetro (fabricante) con ASEL=0 (MISO)
Pin 4 del chip

//Direcciones de los registros a utilizar
#define XDATA3  0x08
#define XDATA2  0x09
#define XDATA1  0x0A
#define YDATA3  0x0B
#define YDATA2  0x0C
#define YDATA1  0x0D
#define ZDATA3  0x0E
#define ZDATA2  0x0F
#define ZDATA1  0x10

#define STATUS    0x04
#define ACT_EN    0x24
#define FILTER    0x28
#define RANGE     0x2C
#define POWER_CTL 0x2D
#define RESET     0x2F

#define ACT_THRESH_H     0x25
#define ACT_THRESH_L     0x26
#define ACT_COUNT     0x27
#define INT_MAP     0x2A
#define OFFSET_Z_H     0x22
#define OFFSET_Z_L     0x23
```

```
//Variables

long z1,z2,z3;//Lecturas de aceleraciones
long Ac_z, Ac_zg;//Aceleración en cada eje
String Ac_z_string;
int a=0;

char dato_leido = 0; // for incoming serial data
int bandera = 0;//Bandera que nos indica cuándo se recibe la orden de empezar a
enviar datos a través del puerto serie
unsigned long muestra = 0;

// Para interrupción
const int intPin = 2; //Pin digital 2
//Variables globales que se pueden modificar en la función de interrupción.
volatile int Inter=0; //Bandera que servirá para detener al contador
volatile unsigned long cnt=0;//número de ciclos que pasan antes de presentarse una
pisada. Cuenta cada Ts. Al momento no sé cuánto vale Ts
volatile long cnt1=0;// contador que se incrementa cada 4,294,967,290 cuentas de
cnt. Se pone en caso de desbordarse el cnt1
volatile int bandera2=0;

void Transmitir();
void LeerSTATUS();
void ParарContador();

void setup()
{

  //Comunicaciones
  Serial.begin(115200); //115200, SERIAL_8E1

  //Configuración acelerómetro
  Wire.begin();//comunicación I2C

  Wire.beginTransmission(ADXL355);
  Wire.write(RESET); //registro inicial a escribir
  Wire.write(0x52);//Reset a todos los registros
  Wire.endTransmission();

  //Hay que hacer una nueva transmisión aquí, ya que la dirección se autoincrementa
de acuerdo a la hoja de datos
  Wire.beginTransmission(ADXL355);
  Wire.write(FILTER);//Nueva dirección a escribir
  Wire.write(0x04); // ODR 250HZ, LowPassFilter 62.5Hz
```

```
  Wire.endTransmission();

  //Hay que hacer una nueva transmisión aquí, ya que la dirección se autoincrementa
de acuerdo a la hoja de datos
  Wire.beginTransmission(ADXL355);
  Wire.write(POWER_CTL);//Nueva dirección a escribir
  Wire.write(0x06); // Power Control -DRDY deshabilitado; Procesamiento de
temperatura deshabilitado;Measure mode
  Wire.endTransmission();

 //Hay que hacer una nueva transmisión aquí, ya que la dirección se autoincrementa
de acuerdo a la hoja de datos
  Wire.beginTransmission(ADXL355);
  Wire.write(RANGE); //registro inicial a escribir
  Wire.write(0x41);//Range -I2C speed=Fast mode; +-2g; INTx activo en alto (0x41
Activo en alto; 0x01 Activo en bajo)
  Wire.endTransmission();

  //Hay que hacer una nueva transmisión aquí, ya que la dirección se autoincrementa
de acuerdo a la hoja de datos
  Wire.beginTransmission(ADXL355);
  Wire.write(INT_MAP); //registro inicial a escribir
  Wire.write(0x08);//Interrupción de actividad habilitada en INT1
  Wire.endTransmission();

  //Hay que hacer una nueva transmisión aquí, ya que la dirección se autoincrementa
de acuerdo a la hoja de datos
  Wire.beginTransmission(ADXL355);
  Wire.write(ACT_EN); //registro inicial a escribir
  Wire.write(0x04);//Sólo el eje Z es una componente de la detección de Actividad
  Wire.endTransmission();

  //Umbral de aceleración.
  //ACT_THRESH(15:0) Coincide con los bits (18:3) de
ZDATA   0XXX  XXXX  XXXX  XXXX  X000
  //Hay que hacer una nueva transmisión aquí, ya que la dirección se autoincrementa
de acuerdo a la hoja de datos
  Wire.beginTransmission(ADXL355);
  Wire.write(ACT_THRESH_H); //registro inicial a escribir
  Wire.write(0x7D);//Umbral bits más significativos
  Wire.endTransmission();

  //Hay que hacer una nueva transmisión aquí, ya que la dirección se autoincrementa
de acuerdo a la hoja de datos
  Wire.beginTransmission(ADXL355);
```

```
  Wire.write(ACT_THRESH_L); //registro inicial a escribir
  Wire.write(0xA0);//Umbral bits menos significativos
  Wire.endTransmission();

  //Hay que hacer una nueva transmisión aquí, ya que la dirección se autoincrementa
de acuerdo a la hoja de datos
  Wire.beginTransmission(ADXL355);
  Wire.write(ACT_COUNT); //registro inicial a escribir
  Wire.write(0x01);//Número de eventos consecutivos requeridos por arriba del
umbral para la detección de Actividad
  Wire.endTransmission();

  //OFFFSET (poniendo la señal en +1g)
  //OFFSET_Z(15:0) coincide con los bits (19:4) del ZDATA
  //Hay que hacer una nueva transmisión aquí, ya que la dirección se autoincrementa
de acuerdo a la hoja de datos
  Wire.beginTransmission(ADXL355);
  Wire.write(OFFSET_Z_H); //registro inicial a escribir
  Wire.write(0xFF);//Offset bits más significativos.
  Wire.endTransmission();
  //Hay que hacer una nueva transmisión aquí, ya que la dirección se autoincrementa
de acuerdo a la hoja de datos
  Wire.beginTransmission(ADXL355);
  Wire.write(OFFSET_Z_L); //registro inicial a escribir
  Wire.write(0xCE);//Offset bits menos significativos.
  Wire.endTransmission();

  //Lectura del registro de estado
  Wire.beginTransmission(ADXL355);
  Wire.write(STATUS);//Registro inicial a leer, la dirección se autoincrementa de
acuerdo a la hoja de datos
  Wire.endTransmission();

  Wire.requestFrom(ADXL355,1);//1 BYTES A LEER
  a=Wire.read();

  //Interrupción
  pinMode(intPin,INPUT);
  attachInterrupt(digitalPinToInterrupt(intPin), PararContador, RISING);
//Interrupción con flancos de subida

}
```

```
void loop ()
{
  if (bandera == 0)// Si no se ha recibido la orden de empezar a enviar datos
  {
    Serial.flush();
    //Serial.println("Esperando Bandera");
    if (Serial.available() > 0)
    {
      dato_leido = Serial.read();
      if (dato_leido == 'a')//este dato se recibe desde Labview e indica que se
puede empezar a enviar los datos
      {
        bandera=1;
        //Lectura del registro de estado
        LeerSTATUS();
        //muestra=muestra+1;
      }
      else
      {
       //no hace nada
      }
    }
  }
  else//Ya se recibió la orden de enviar datos
  {

    if (bandera2==0)
    {
      if(cnt<=4294967290)//4294967290)//  5901686=aprox20seg ->1seg=295084
      {
        cnt=cnt+1;
      }
      else
      {
        cnt=0;
        cnt1=cnt1+1;
      }
    }
    else
    {
      Transmitir();
      cnt=0;
      cnt1=0;
      //Reseteando bandera para que el coordinador de instrucción de inicio de
medición
```

```
      bandera=0;
      bandera2=0;

    }
  }
}

void LeerSTATUS()
{
  Wire.beginTransmission(ADXL355);
  Wire.write(STATUS);//Registro inicial a leer, la dirección se autoincrementa de
acuerdo a la hoja de datos
  Wire.endTransmission();
  Wire.requestFrom(ADXL355,1);//1 BYTES A LEER
  a=Wire.read();
  //Serial.println("Leí el registro de estado: "+ String(a));
}

void ParарContador()
{
  bandera2=1;
}

void Transmitir()
{
  Serial.println("N1"+String(cnt));
}
```

9.4.2 Programa en Python del Coordinador

```
# -*- coding: utf-8 -*-
"""
Created on Wed Sep 29 14:14:28 2021

@author: luis_
"""

import serial, time,csv

file_name = "HP_19.csv"
num_paq = 61 #Numero de pisadas que recibirá
num_datos = 5 #Numero de nodos
bandera = 0
entrada="null"

i = 0

file = open(file_name,"w",newline='') #newline='' es para que no deje espacios
entre columnas

try:
    ser = serial.Serial("COM6", 115200)
#,bytesize=serial.EIGHTBITS,parity=serial.PARITY_EVEN,stopbits=serial.STOPBITS_ONE)


    print("Iniciando el sistema, espera")
    time.sleep(1)
    while bandera != 1:

        if entrada =='a':
            bandera = 1
            msj ="a".encode()
            #ser.write(msj)
            print("Bandera recibida")

        else:
            entrada=input("Presiona la letra 'a' para iniciar : ")
            time.sleep(1)
    bandera = 0

    print("Creando archivo")

    thewriter = csv.writer(file,delimiter=",")
```

```
    #thewriter.writerow(["N1", "N2", "N3", "N4", "N5"])

    time.sleep(0.1)
    print("Comienza ahora")

    lista = [0, 0, 0, 0, 0] # Se almacenan los datos recibidos vía RF
    z=[0, 0, 0, 0, 0] #Se acomodan los datos recibidos

    j=0
    while j<num_paq:#Repite hasta tener el número de pisadas esperadas

        ser.write(msj)

        #Lee los datos provenientes de los nodos
        i=0
        while i <num_datos: #Lee el puerto serie hasta que le lleguen los datos de
todos los nodos
            recibido = str(ser.readline())
            datos = recibido[2:][:-5] #Sólo se queda con los datos relevantes de la
cadena proveniente del coordinador
            lista[i] = datos# Hace una lista de los datos
            i=i+1
        print(lista)

        #Ordena los datos N1, N2, N3, N4, N5
        i=0
        while i<num_datos:
            dato=lista[i]
            #print(dato)
            if dato[1] == "1": # estamos buscando el número del identificador del
nodo N1,N2 o N3
                z[0]=dato[2:]# Sólo asignale los datos sin el identificador "N1"
                           # desde el índice 2 de la cadena hasta donde llegue
            elif dato[1] == "2":
                z[1]= dato[2:]
            elif dato[1] == "3":
                z[2] = dato[2:]
            elif dato[1] == "4":
                z[3] = dato[2:]
            elif dato[1] == "5":
                z[4] = dato[2:]

            i=i+1
        print(z)
        print(" ")
```

```
        #Una vez ordenados los datos, escribelos en una fila del archivo csv
        if j>=1: #Omite el primer paquete de datos que es basura
            thewriter.writerow(z)

        time.sleep(0.5) #tiempo para evitar las interrupciones continuas en los
nodos

        j=j+1

finally:
    file.close() # cierra archivo csv
    ser.close()  # cierra puerto serie
```