Localización en interiores utilizando infraestructura de Internet de las Cosas

Proyecto de Grado

Francisco Crizul, Gerardo Gómez

Tutores:

Dr. Ing. Matías Richart

Dr. Ing. Eduardo Grampín

Montevideo – Uruguay

20 de abril de 2021

RESUMEN

Debido a la necesidad de contar con sistemas de localización en interiores (ILS) económicos, muchos investigadores se han centrado en los ILS basados en Wi-Fi, que utilizan la intensidad de la señal recibida (RSS) por la red de área local inalámbrica, que se recoge desde distintos puntos de los entornos interiores, llamados puntos de referencia.

La mayoría de las soluciones se basan en contar con una infraestructura de red disponible, como ser puntos de acceso Wi-Fi conectados a una red de datos cableada, con la necesidad de conexión a la red eléctrica. Esto genera que no se pueda contar con suficientes puntos de referencia, o que sea muy costoso su despliegue y mantenimiento.

Por otro lado, la Internet de las Cosas (IoT) es un concepto que refiere a lograr la interconexión de objetos físicos, estáticos y móviles, con el fin de capturar y transmitir datos y/o de que respondan a órdenes (comandos). Para lograr esto se cuenta con sensores, actuadores, elementos de cómputo, software y una infraestructura de comunicación para lograr la interconexión. Una arquitectura de IoT consiste en dispositivos finales que generan información en su rol sensor y reciben comandos en su rol actuador. Esta información se transmite hacia/desde plataformas de IoT "en la nube" a través de un componente específico de comunicación asincrónica (denominado broker), que hace disponibles los datos a módulos de almacenamiento y análisis que eventualmente toman acciones. Los mayores desafíos a nivel de transmisión de datos en este paradigma son las necesidades de largo alcance y bajo consumo energético, debido a que los dispositivos pueden estar ubicados en zonas de difícil acceso y sin la infraestructura necesaria.

En el presente proyecto se realiza un estudio del estado del arte de las tecnologías, mecanismos y soluciones existentes para localización en interiores. Se resuelve el problema de localización en interiores utilizando dispositivos de sensado y comunicaciones de IoT, donde los dispositivos cuentan con sistemas de comunicación de largo alcance. Se realiza la implementación y el despliegue de todas las partes de la arquitectura IoT (sensores, gateways, broker, etc.) necesarias para localizar elementos (personas, dispositivos o robots)

que se encuentran en un espacio determinado. Para esto se miden las señales emitidas por las interfaces Wi-Fi y Bluetooth de los dispositivos que se encuentren al alcance y, utilizando la tecnología LoRa, se envían a un servidor para su procesamiento. Se implementan dos mecanismos de localización existentes: Fingerprinting y Trilateración a partir de la RSS. También se implementa una aplicación para la visualización de los elementos localizados. Finalmente, se mide el desempeño del sistema y se evalúa la solución en un escenario de uso real.

Tabla de contenidos

Capítulo 1

Introducción

1.1. Motivación

La localización en interiores ha tenido un gran interés ya que da lugar a un gran número de servicios basados en la ubicación del usuario. Esto se ha potenciado dado que la mayoría de las personas cuenta con un dispositivo móvil, con una interfaz Wi-Fi y Bluetooth, haciendo posible la localización de personas sin un costo adicional. Algunos ejemplos de servicios que se puede brindar con localización en interiores son:

- Comercial: se pueden realizar estrategias comerciales en base al comportamiento de los usuarios. Por ejemplo, en un supermercado se puede identificar grupos de góndolas visitados por los clientes (las góndolas de los grupos identificados pueden ser ubicados más cerca), o a qué góndolas se dirige un cliente que compró determinado producto.
- Asistencia al usuario: en un aeropuerto, en base a los datos de su vuelo, se le podría avisar al usuario si no está cerca de su puerta de embarque mediante una notificación a su dispositivo móvil.
- Experiencia de usuario: un usuario en un supermercado o en un centro comercial podría saber los productos/tiendas que hay a su alrededor. También es posible llevar a cabo acciones automáticas al detectar la presencia del usuario. Por ejemplo, en una casa, abrir una puerta automáticamente si se trata del dueño de la misma.
- Servicios de salud: en caso de una emergencia en un hospital se podría asignar al médico libre más cercano al paciente.
- Seguridad: es posible detectar dispositivos desconocidos o no autorizados

a acceder a determinada área.

- Rastreo de recursos: puede ser útil para control de inventario o seguimiento de personal.

Para localización en exteriores existe el Sistema de Posicionamiento Global (GPS), ampliamente utilizado, siendo uno de los más exitosos para este fin [50]. Sin embargo, no es útil para interiores dado que la señal de los satélites no penetra obstáculos como las paredes de edificios. Es por esto que es necesario contar con sistemas que funcionen en interiores.

Existen diversos trabajos que utilizan puntos de acceso (AP) 802.11 como nodos de referencia para la localización, lo que hace necesario contar con una infraestructura de red de área local, además de que incrementa el costo de implantación si se quisiera aumentar el número de AP desplegados.

Con el auge de Internet de las Cosas (IoT) se han desarrollado dispositivos de bajo costo, con sensores incorporados, capaces de capturar datos del entorno (e.g., temperatura, humedad, presión atmosférica, movimiento, etc.), y de transmitirlos a largas distancias a un servidor central capaz de procesar esta información. Estos dispositivos pueden contar con interfaces Wi-Fi, Bluetooth, y otras tecnologías inalámbricas de largo alcance (e.g., LoRa), por lo que no requieren infraestructura de red muy grande, e incluso podrían prescindir de una red de área local. Además, por lo general, estos dispositivos funcionan a batería y están construidos para tener un alto rendimiento energético, por lo que tampoco sería necesario un despliegue de red eléctrica para su utilización. También suelen ser pequeños, pudiendo ser embebidos en cualquier objeto. Teniendo en cuenta todas estas ventajas, resulta de interés evaluar la factibilidad de utilizar esta tecnología para la localización en interiores.

1.2. Objetivos

1.2.1. Objetivos Generales

Se marca como objetivo general de este trabajo evaluar la viabilidad de utilizar equipamiento de IoT para la localización en interiores.

Actualmente, el grupo MINA, de la Facultad de Ingeniería, cuenta con el siguiente equipamiento IoT:

- Placas Sparkfun SPX-14893 [67], que cuentan con un módulo ESP32-WROOM-32 (con Wi-Fi y Bluetooth) [24] y un módem LoRa RFM95W.

- Un Gateway LoRa de The Things Network [57].
- Un Gateway LoRa Dragino LPS8 [23].

Se busca, entonces, utilizar estos componentes de hardware para construir una arquitectura de IoT para cumplir con el objetivo planteado.

1.2.2. Objetivos Específicos

Con el fin de cumplir con el objetivo antes mencionado, se marcaron los siguientes objetivos específicos:

1. Realizar un estudio del estado del arte de localización en interiores (tecnologías inalámbricas utilizadas, métodos y algoritmos existentes, y trabajos o sistemas realizados a nivel académico).

2. Implementar un sistema de localización en interiores, de punta a punta, utilizando tecnología y aplicando una arquitectura de IoT. Esto va desde la obtención de los datos en los sensores, hasta la visualización de los mismos luego de procesados. Para esto se deberá utilizar el hardware IoT provisto por la Facultad, lo que incluye placas con interfaces Wi-Fi, Bluetooth y LoRa, que actuarán de sensores, y Gateways LoRa, que serán responsables de enviar la información de los sensores al servidor central.

3. Realizar una evaluación del sistema desarrollado en cuanto a su desempeño y sus posibles aplicaciones, y la viabilidad de la utilización del hardware IoT disponible para localización en interiores.

1.3. Organización del Documento

En el capítulo 2 se describe el estudio realizado sobre el estado del arte de localización en interiores. En el capítulo 3 se presenta el diseño y la implementación del sistema propuesto. En el capítulo 4 se realiza la evaluación del sistema. Por último, en el capítulo 5 se presentan las conclusiones y el trabajo a futuro.

Capítulo 2

Estado del Arte

2.1. Introducción

En este capítulo se describirá el estado del arte de la localización en interiores. Esto abarcará tanto las técnicas como las tecnologías utilizadas, y algunos trabajos realizados para localización en interiores. Para los trabajos existentes, este proyecto se centra en los que utilizan las tecnologías Wi-Fi y Bluetooth, dado que son las tecnologías con las que se cuenta para realizar la localización. Por esta misma razón, se describirán las tramas 802.11 (tramas Wi-Fi) y los paquetes Bluetooth, de donde se tomará la información necesaria para identificar a los dispositivos a ser localizados. Por último, se hace una breve introducción a la tecnología y el hardware de IoT con que se cuenta para la realización de este trabajo.

2.2. Generalidades

En localización en interiores se utilizan varias tecnologías inalámbricas. Puede ser clasificada según el método para determinar la ubicación, haciendo uso de varios tipos de medidas de la señal (e.g., el tiempo, el ángulo o la intensidad con que ésta llega), y de la capa física o la infraestructura de los sensores de localización (i.e., la tecnología inalámbrica utilizada). Las medidas de la señal implican la transmisión y recepción de señales entre los componentes de hardware del sistema.

Existen varios problemas en la localización en interiores. Dos de ellos son inherentes a la propagación de las señales electromagnéticas, la falta de línea

de visión (NLoS, Non-Line of Sight) y el multi-trayecto (multipath). NLoS se da a causa a la presencia de obstáculos (personas, mobiliario, paredes, etc.). Al atravesar el medio una señal se va atenuando, es decir, su intensidad (RSS) va disminuyendo. Al encontrarse con estos obstáculos se tiene una atenuación adicional, haciendo que la RSS sea aún menor cuando llega al receptor [79]. Los multi-trayectos son causados por la reflexión de la onda al impactar con los obstáculos. Esto puede provocar que la misma señal llegue al receptor por diferentes caminos, con diferente ángulo y RSS, lo que puede afectar a los distintos métodos. Por esto es que se han realizado diversos trabajos, que intentan mitigar estos problemas.

Otro problema es la privacidad. La mayoría de los dispositivos móviles de usuario cuentan con una interfaz 802.11 (Wi-Fi) y un módulo Bluetooth. Estos componentes tienen una dirección de capa de enlace única por dispositivo: la dirección MAC en Wi-Fi y su análoga en Blueooth, la BD_ADDR. Con estas direcciones podría ser posible identificar a una persona, lo que permite rastrearla y con esto estudiar su comportamiento, a partir de sus recorridos. Para mitigar esto, los sistemas operativos Android e iOS realizan una anonimización, generando direcciones aleatorias [3, 37], lo que podría dar problemas si se quisiera rastrear un dispositivo en particular.

En localización existen los conceptos Localización Basada en Monitor (MBL) y Localización Basada en Dispositivo (DBL) [81], también llamados "Basados en Infraestructura" y "Basados en el Cliente" [44], respectivamente. En MBL el sistema calcula la ubicación del usuario en base a los datos recolectados de forma pasiva por los nodos ancla. Por otra parte, en DBL el dispositivo cliente calcula su ubicación en base a la información de sus nodos de referencia. MBL es utilizado principalmente en rastreo, y DBL en navegación.

2.3. Medidas de la Señal

Para los métodos de localización se realiza alguna medición de la señal, y éstos también pueden basarse en alguna de sus propiedades. Ejemplos de medidas pueden ser la intensidad, el ángulo o la fase con la que una señal llega al receptor, y un ejemplo de las propiedades puede ser la velocidad a la que se propagan las ondas (e.g., las ondas de radiofrecuencia se propagan a la velocidad de la luz).

A continuación se describen dos medidas bastante utilizadas en localización

en interiores.

2.3.1. RSSI (Received Signal Strength Indicator)

La Received Signal Strength (RSS) es la potencia real con la que una señal llega al receptor, usualmente medida en dBm (decibelios-miliWatts), y es una medida de la capa física [79]. El Received Signal Strength Indicator (RSSI) es el indicador de la RSS (y muchas veces confundido con ésta [81]). Es una medida relativa de la RSS, con unidades arbitrarias definidas por el fabricante del chip (e.g., Atheros usa valores de RSSI entre 0 y 60, Cisco usa valores entre 0 y 100), y es el valor entregado por la capa de enlace.

2.3.2. CSI (Channel State Information)

En muchos sistemas inalámbricos (e.g. IEEE 802.11) el ancho de banda de coherencia del canal es menor al ancho de banda de la señal, lo que hace que el canal sea selectivo en frecuencia (i.e., su respuesta varía según el canal de frecuencia) [81]. Además, en transceptores de múltiples antenas, la Respuesta de Frecuencia del Canal (CFR) para los pares de antenas podría variar considerablemente (dependiendo de la distancia entre las antenas y de la longitud de onda de la señal).

A diferencia de la RSS, que es la superposición de multi-trayectos con grandes variaciones de fase, la característica de capa física, Channel Response, es capaz de discriminar las características de multi-trayectos [79]. La Respuesta de Impulso del Canal (CIR) o su par de Fourier (i.e., la CFR), ambos valores de la capa física, normalmente son entregadas a las capas superiores como Información de Estado del Canal (CSI). La CIR tiene una granularidad mayor que la RSS, ya que puede capturar las respuestas, tanto de amplitud como de fase, del canal en diferentes frecuencias y entre pares de antenas transmisor-receptor separados. El CSI es una cantidad compleja y se puede escribir en forma polar como:

$$H(f_k) = \|H(f_k)\| \, e^{j \sin(\angle H_k)} \tag{2.1}$$

donde $H(f_k)$ es el CSI del subcarrier k, con frecuencia central f_k, $\|H(f_k)\|$ es la respuesta de amplitud del subcarrier k y $\angle H_k$ denota su fase (j es la unidad imaginaria, notación utilizada en el campo de la ingeniería eléctrica).

2.4. Métodos de Localización

Los métodos de localización son las técnicas utilizadas para llevar a cabo dicha tarea. Esto incluye la información de las señales de onda consideradas y los algoritmos utilizados, que hacen uso de esta información para realizar el cálculo de la ubicación del usuario. Hasta donde se pudo ver, existen tres tipos de técnicas de localización: *Triangulación*, *Análisis de Escena* y *Proximidad* [26, 50, 81]. En esta sección se hará una descripción de los distintos tipos de técnicas y sus algoritmos.

2.4.1. Triangulación

Hace uso de las propiedades geométricas de los triángulos para estimar la ubicación del dispositivo cliente. Este método tiene dos derivaciones: *Lateración* y *Angulación*.

2.4.1.1. Lateración

El método de Trilateración (o Multilateración, o N-lateración) consiste en calcular las coordenadas del usuario a partir de la distancia de éste a tres o más nodos ancla. Cada distancia determina una esfera. Si las distancias fueran exactas, la intersección de estas tres o más esferas sería un punto, que determinaría la ubicación del dispositivo cliente. Usualmente, las distancias calculadas no son exactas, por lo que la intersección puede ser una región o más de un punto, e incluso puede no existir intersección, por lo que es necesario aplicar algún algoritmo para estimar la ubicación del dispositivo cliente. Uno de los métodos más utilizados para trilateración es la Estimación por Mínimos Cuadrados [74]. En mínimos cuadrados se intenta minimizar la suma de los cuadrados del error. En este caso, se considera el error del cuadrado de la distancia de los nodos al usuario. La distancia del usuario al nodo i está dada por:

$$d_i = \sqrt{(x - x_i)^2 + (y - y_i)^2} \tag{2.2}$$

siendo (x, y) las coordenadas cartesianas de la ubicación del usuario y (x_i, y_i) las coordenadas cartesianas donde se encuentra el nodo i. El error del cuadrado de la distancia del usuario al nodo i es:

$$e_i = (x - x_i)^2 + (y - y_i)^2 - {d_i}^2 \tag{2.3}$$

A continuación, se describen los métodos encontrados que realizan lateración a partir de las distancias calculadas.

Basado en la Atenuación de la Señal

Este método, también llamado *Basado en RSS*, utiliza el RSSI, valor obtenido de la RSS. Se puede establecer una relación entre el RSSI y la distancia del emisor (Tx) al receptor (Rx), siendo que, a mayor distancia, su valor disminuye. Para esto, es necesario definir un modelo de propagación, teniendo en cuenta la atenuación de la señal en su trayecto. Un modelo sencillo, y de los más utilizados para interiores [74, 81], es el de la ecuación 2.4:

$$RSSI = -10n \cdot log_{10}\left(\frac{d}{d_0}\right) + RSSI_0 \tag{2.4}$$

donde $RSSI$ es el indicador de la intensidad de la señal en Rx, d la distancia de Rx a Tx y n el factor de atenuación, constante que depende del medio y de los obstáculos existentes entre Tx y Rx. El valor de n es 2 en espacio libre y varía entre 3 y 4 para interiores. $RSSI_0$ es el valor del RSSI en un punto de referencia, encontrado a una distancia d_0 de Rx. Suele tomarse $d_0 = 1$ m, por lo que $RSSI_0$ es el valor del RSSI a un metro de distancia. Una vez determinados los valores de los parámetros n y $RSSI_0$, es posible calcular la distancia al usuario utilizando la ecuación 2.5:

$$d = 10^{\left(\frac{RSSI_0 - RSSI}{10n}\right)} \tag{2.5}$$

El enfoque de Trilateración utilizando las distancias estimadas a partir del RSSI es simple y económico, pero carece de precisión, sobre todo en NLoS [79]. Es poco probable que se obtenga una alta precisión de localización sin la utilización de algoritmos más complejos.

Tiempo de Vuelo (ToF, Time of Flight)

ToF o Tiempo de Arribo (ToA, Time of Arrival) usa el tiempo de propagación de la señal para calcular la distancia entre el transmisor Tx y el receptor

Rx [19]. La distancia entre Tx y Rx se calcula multiplicando el tiempo de propagación por la velocidad de la luz $c = 3 \times 10^8$ m/s. En este método se puede usar Trilateración a partir de las distancias calculadas para estimar la ubicación del usuario. ToF requiere sincronización entre receptores y transmisores y que se transmitan timestamps junto con la señal. Con este método se tiene un error significativo debido al desvío de las señales producido por los obstáculos, lo que incrementa el tiempo en que la señal se propague de Tx a Rx. Sea t_1 el instante en que Tx_i envía un mensaje a Rx_j, que lo recibe en el instante t_2, donde $t_2 = t_1 + t_p$ (t_p es el tiempo en que la señal llega de Tx_i a Rx_j), la distancia entre Tx_i y Rx_j está dada por la ecuación 2.6:

$$D_{ij} = (t_2 - t_1) \times c \tag{2.6}$$

Requiere Line of Sight (LoS) para lograr una buena precisión.

Diferencia de Tiempo de Arribo (TDoA, Time Difference of Arrival)

TDoA utiliza la diferencia de los tiempos de propagación de las señales, desde el transmisor (dispositivo cliente), medidas en los receptores (nodos de referencia). Las medidas de TDoA ($T_{D(i,j)}$), desde el transmisor a los receptores i y j se traducen a valores de distancia $L_{D(i,j)} = c \times T_{D(i,j)}$, siendo c la velocidad de la luz. El transmisor estará ubicado en la hiperboloide dada por la ecuación 2.7:

$$\begin{aligned} L_{D(i,j)} = &\sqrt{(X_i - x)^2 + (Y_i - y)^2 + (Z_i - z)^2} \\ &- \sqrt{(X_j - x)^2 + (Y_j - y)^2 + (Z_j - z)^2} \end{aligned} \tag{2.7}$$

donde (X_i, Y_i, Z_i) y (X_j, Y_j, Z_j) son las coordenadas de los receptores (nodos de referencia) i y j, y (x, y, z) las coordenadas del emisor (usuario). Se necesita el TDoA de al menos tres nodos de referencia para calcular la ubicación del usuario, calculada como la intersección de las tres (o más) hiperboloides. El sistema de ecuaciones se puede resolver con regresión lineal [50] o aplicando la serie de Taylor. Se requiere una alta sincronización únicamente entre los receptores, a diferencia de ToF, pero su precisión también depende de la existencia

de LoS.

Tiempo de Retorno de Vuelo (RToF, Return Time of Flight)

RToF mide el tiempo de ida y de regreso de la propagación de la señal (i.e., transmisor-receptor-transmisor) para estimar la distancia de Tx a Rx [19]. La principal ventaja de RToF respecto a ToF es que se necesita una sincronización relativamente moderada entre Tx y Rx. Un problema significativo es que el retardo de la respuesta en el receptor depende la implementación de su hardware y del overhead del protocolo, pero éste puede ser ignorado si el tiempo de propagación transmisor-receptor es grande comparado con el tiempo de respuesta. Sin embargo, no puede ser ignorado en sistemas de corto alcance, como los de localización en interiores. Sea t_1 el instante en que Tx_i envía un mensaje a Rx_j, que lo recibe en el instante t_2, donde $t_2 = t_1 + t_p$. En el instante t_3, Rx_j envía un mensaje de regreso a Tx_i, y lo recibe en el instante t_4. Entonces, la distancia entre Tx_i y Rx_j se calcula de la siguiente manera:

$$D_{ij} = \frac{(t_2 - t_1) + (t_4 - t_3)}{2} \times c \tag{2.8}$$

Su precisión también depende la existencia de LoS.

Fase de Arribo (PoA, Phase of Arrival)

Estima la distancia de Tx a Rx usando la diferencia de fase con que llega la señal de onda en diferentes antenas. Se asume que existe un retardo de propagación finito D_i entre Tx y Rx que puede expresarse como una fracción de la longitud de onda de la señal. Luego de calculadas las distancias se puede aplicar Trilateración para calcular la ubicación del usuario. La principal desventaja de este método es que requiere LoS para obtener una alta precisión.

2.4.1.2. Angulación

Ángulo de Arribo (AoA, Angle of Arrival)

La ubicación del dispositivo cliente se calcula como la intersección de rectas que pasan por los nodos ancla, cada una determinada por el ángulo en que llega la señal al nodo. Se requieren dos nodos ancla para localización 2D y tres para 3D, y no requiere sincronización entre los nodos. En esta técnica se usan arreglos de antenas [77] en el receptor para estimar el ángulo en que las señales llegan a éste, calculando la diferencia de tiempo en que la señal llega a cada elemento del arreglo de antenas. Brinda estimaciones precisas cuando el transmisor y el receptor están cerca, pero requiere hardware complejo y calibración cuidadosa en comparación con las técnicas basadas en RSSI. A medida que la distancia entre el transmisor y el receptor aumenta, la precisión disminuye, introduciendo un error alto en la estimación.

2.4.2. Análisis de Escena (Fingerprinting)

Las técnicas de localización basadas en Fingerprinting requieren un estudio previo del entorno (análisis de escena) para obtener huellas (fingerprints), usualmente con valores del RSSI, pero también del CSI. Con este método, el problema de localización se convierte en un problema de clasificación.

Existen las etapas offline y online. En la etapa offline se toman medidas de RSSI en determinados puntos del lugar, generando un mapa de datos (Radio Map), y se les asocia una etiqueta, que indica la clase a la que pertenecen. Las tuplas de medidas de RSSI etiquetadas $(etiqueta_i, RSSI_{j,1}, \ldots, RSSI_{j,n})$, que son fingerprints etiquetados, son utilizados para entrenar los algoritmos que serán utilizados en la etapa online. La etapa online hace referencia a cuando el sistema recibe fingerprints sin etiquetar $(RSSI_{j,1}, \ldots, RSSI_{j,n})$. Las medidas de esta etapa son comparadas con las de la etapa offline para estimar la ubicación del usuario.

A diferencia de Trilateración, este método realiza la estimación de la ubicación del usuario dentro de un conjunto discreto de ubicaciones, dado que se basa en una grilla discreta de las ubicaciones consideradas en la etapa offline. Estas técnicas pueden dar estimaciones precisas de la ubicación [81], pero son susceptibles a los cambios en el entorno.

En las técnicas de clasificación existen algunas métricas para medir el desempeño del modelo utilizado. Una de ellas es la *exactitud* (en inglés, *accuracy*), y es el porcentaje de aciertos sobre el total de estimaciones (o predicciones) [6]. También suele usarse la matriz de confusión. Las filas de la matriz de confusión

indican las clases reales y las columnas indican las clases determinadas por el algoritmo (las predicciones). Es decir que la celda $[i, j]$ de la matriz indica la cantidad de veces que, para una muestra de la clase i, se predijo que pertenece a la clase j. Por lo tanto, al disponer las filas y columnas en el mismo orden, se espera que los valores en su diagonal principal estén lo más cerca posible de los valores reales, y de cero en las demás celdas. Esta matriz permite visualizar qué tipos de aciertos y errores está teniendo el modelo.

A continuación se describen algunos de los algoritmos existentes para correlacionar las medidas online con las offline.

2.4.2.1. k-Nearest Neighbors (kNN)

Este método obtiene las k correspondencias más cercanas (en base a las medidas offline de RSSI) de las ubicaciones conocidas usando la raíz del error cuadrático medio (RMSE, Root Mean Square Error) [50]. Estas correspondencias son promediadas para obtener la estimación de la ubicación.

2.4.2.2. Support Vector Machine (SVM)

Este método es muy utilizado en Machine Learning (ML) y análisis estadístico y tiene una gran precisión. SVM también puede ser utilizado para localización usando medidas de RSSI online y offline [50]. A partir de tener los datos como puntos del espacio, el algoritmo encuentra una superficie (o un hiper-plano) que separe los datos de forma óptima para poder tomar una decisión de clasificación. El hiper-plano generado separa el espacio en regiones, de forma tal que los puntos son clasificados acorde a la región a la cual pertenecen.

2.4.2.3. Árboles de Decisión

Utiliza un modelo de decisiones similar a un árbol, en el que los nodos internos son pruebas sobre las entradas y las hojas son los posibles resultados de la clasificación [58]. En cada nodo, partiendo desde la raíz, se verifica una condición con respecto a la entrada, conduciendo a una decisión. El árbol es determinado durante el aprendizaje.

2.4.2.4. Bosques Aleatorios

Deriva de los árboles de decisión [22]. Se busca crear varios árboles de decisión distintos. Estos árboles se ensamblan de forma tal de obtener un mejor resultado global. Los Bosques Aleatorios agregan un factor de aleatoriedad al construir los árboles de decisión. En lugar de buscar la característica más importante a la hora de bifurcar un nodo, busca la mejor característica entre un subconjunto aleatorio de características.

2.4.2.5. Multi-layer Perceptron (MLP)

Un perceptrón es un algoritmo para clasificación binaria, ampliamente utilizado en Redes Neuronales [59]. Tiene una estructura simple, donde a partir de una combinación lineal de n entradas y una función de activación, se obtiene una salida binaria. La clave del algoritmo está en hallar los pesos óptimos de la combinación lineal para la mejor clasificación. Para esto se entrena el algoritmo con un conjunto de datos, actualizando los pesos de la combinación lineal a partir de la diferencia entre la salida del perceptrón y la muestra. El perceptrón sólo permite resolver problemas linealmente separables.

Un Perceptrón multi-capa parte del perceptrón para crear una red neuronal más compleja que permita resolver problemas de clasificación no linealmente separables. En esta red existe al menos una capa oculta entre la entrada y la salida, que consiste en varios perceptrones simples. La combinación de las salidas de esta capa pasa por una función de activación, lo que da el resultado final. El entrenamiento de este algoritmo consiste entonces en optimizar todos los pesos involucrados de las combinaciones para llegar a la mejor clasificación.

2.4.2.6. AdaBoost

Adaptive Boosting [31] es un meta-algoritmo, dado que utiliza uno o varios algoritmos para aprender. Al grupo de algoritmos se le conoce como ensamble y representa un clasificador con muy buena precisión. A los algoritmos que conforman el ensamble se les denomina clasificadores débiles, que son clasificadores con una precisión menor. Es adaptativo, porque durante el entrenamiento pondera de manera diferente a los datos. En cada iteración los datos testeados con clasificación incorrecta obtendrán un mayor peso para la siguiente iteración, y los datos clasificados correctamente obtendrán una importancia menor.

Esto permite que el conjunto de algoritmos se enfoque en los datos de prueba que no han sido clasificados correctamente.

2.4.2.7. Naive Bayes

Es un conjunto de algoritmos basados en aplicar el teorema de Bayes, con el supuesto "ingenuo" de independencia condicional entre cada par de características dado el valor de la variable de clase [64]. El teorema de Bayes establece la relación de la ecuación 2.9, dada la variable de clase y y el vector de características dependiente $x = (x_1, \ldots, x_n)$.

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y)P(x_1, \ldots, x_n \mid y)}{P(x_1, \ldots, x_n)} \tag{2.9}$$

El supuesto de independencia condicional establece lo siguiente:

$$P(x_i \mid y, x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) = P(x_i \mid y) \tag{2.10}$$

Tomando 2.10 para todo i, la relación de la ecuación 2.9 puede ser simplificada a:

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y) \prod_{i=1}^{n} P(x_i \mid y)}{P(x_1, \ldots, x_n)} \tag{2.11}$$

Dado que $P(x_1, \ldots, x_n)$ es constante dada una entrada, se puede usar la siguiente regla de clasificación:

$$\begin{gathered} P(y \mid x_1, \ldots, x_n) \propto P(y) \prod_{i=1}^{n} P(x_i \mid y) \\ \Downarrow \\ \hat{y} = \arg\max_{y} P(y) \prod_{i=1}^{n} P(x_i \mid y) \end{gathered} \tag{2.12}$$

Es decir que se intenta estimar el y que maximiza la expresión anterior.

Los distintos clasificadores Naive Bayes difieren principalmente en los supuestos que realizan en relación a la distribución de $P(x_i \mid y)$.

2.4.2.8. Linear and Quadratic Discriminant Analysis

Linear Discriminant Analysis (LDA) y Quadratic Discriminant Analysis (QDA) son dos clasificadores con una superficie de decisión lineal y una

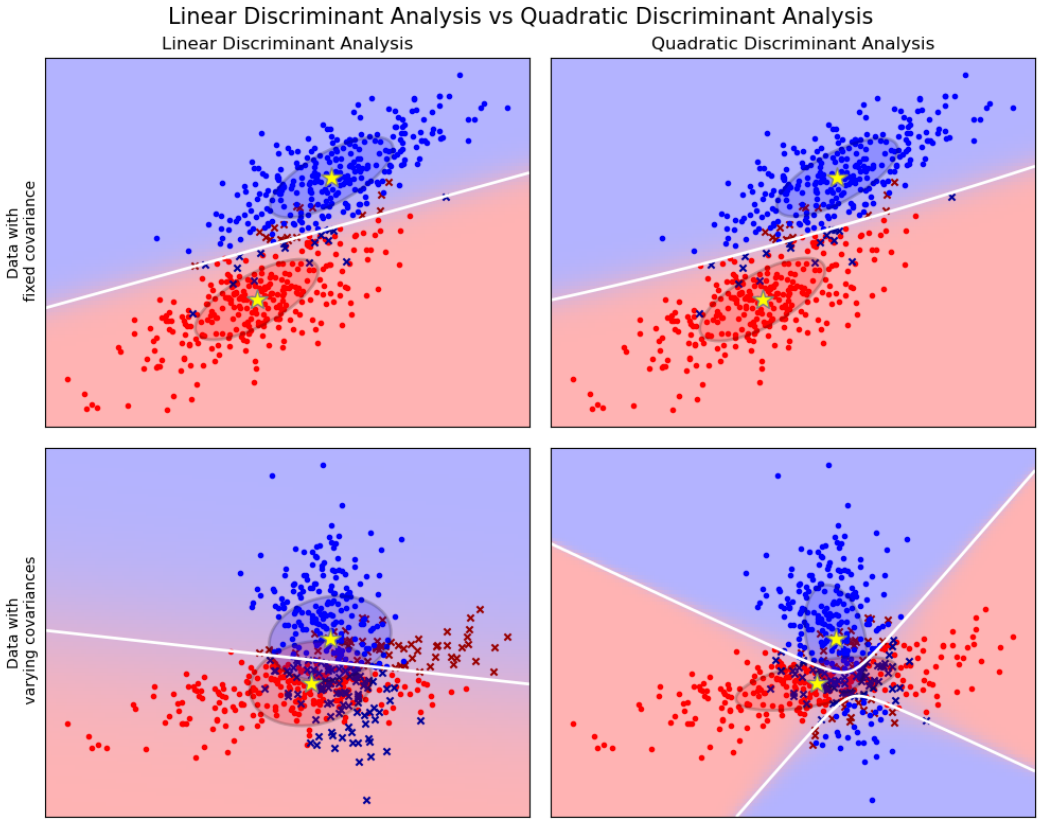


Figura 2.1: LDA vs. QDA (obtenida de [63])

cuadrática, respectivamente [63]. La figura 2.1 muestra las fronteras de decisión para LDA y QDA, y muestra que QDA es más flexible.

QDA es una generalización de LDA, y ambos pueden ser derivados de modelos probabilísticos simples, que modelen la distribución de la clase condicional de los datos, $P(X \mid y = k)$ para cada clase k. Las predicciones pueden ser obtenidas utilizando la regla de Bayes para cada muestra de entrenamiento $x \in \mathbb{R}^d$:

$$P(y = k \mid x) = \frac{P(x \mid y = k)P(y = k)}{P(x)} = \frac{P(x \mid y = k)P(y = k)}{\sum_l P(x \mid y = l) \cdot P(y = l)} \tag{2.13}$$

y se selecciona la clase k que maximiza la probabilidad. Más específicamente, para LDA y QDA, $P(x \mid y)$ es modelada como una distribución Gaussiana multivariante.

2.4.3. Proximidad

Los métodos de proximidad brindan información de ubicación relativa a un nodo ancla, o Nodo de Referencia (RN) [50]. Por lo general, se utiliza una grilla densa de RN, cuyas ubicaciones son conocidas. Cuando un único RN detecta a un dispositivo, se considera que se encuentra próximo a ese RN. Cuando más de un RN lo detecta, se considera que se encuentra próximo al RN que recibió la mayor RSS.

2.5. Tecnologías Utilizadas en Localización

A continuación, se describen las principales tecnologías inalámbricas utilizadas para localización en interiores, haciendo énfasis en Wi-Fi y Bluetooth, dado que son las tecnologías con las que se cuenta para realizar el presente trabajo.

2.5.1. Wi-Fi

El estándar IEEE 802.11, conocido como Wi-Fi, opera en las bandas de radio Industriales, Científicas y Médicas (ISM), no reguladas y de uso libre. Inicialmente, su alcance era de aproximadamente 100 m [50], lo que aumentó a cerca de 1 km [81] en IEEE 802.11ah, optimizada principalmente para servicios de Internet de las Cosas (IoT).

Actualmente, la mayoría de los dispositivos móviles (teléfonos, tablets, laptops, etc.) cuentan con una interfaz Wi-Fi, por lo que es ampliamente utilizado en localización en interiores. Dado que los Access Points (APs) Wi-Fi existentes pueden ser usados como nodos de referencia para la recolección de señales [81], es posible construir sistemas de localización básicos sin la necesidad de infraestructura adicional y con una precisión razonablemente buena. Sin embargo, las redes Wi-Fi existentes usualmente son instaladas para comunicación, optimizadas para maximizar la tasa de transferencia y la cobertura, y no para localización, por lo que se requiere de algoritmos eficientes para mejorar la precisión. Además, la interferencia en la banda ISM afecta esta precisión. Es posible usar las técnicas mencionadas anteriormente, y combinaciones de ellas (métodos híbridos) para localización.

Figura 2.2: Trama 802.11 (obtenida de [48])

2.5.1.1. Trama Wi-Fi

A continuación se describirá la trama 802.11, o trama Wi-Fi, ya que será utilizada en el presente trabajo, por contar con sensores con interfaces Wi-Fi.

La figura 2.2 muestra la trama 802.11 [48]. Los campos utilizados son *Control de Trama*, *Dirección 1*, *Dirección 2*, *Dirección 3* y *Dirección 4*:

- **Dirección 1**: contiene la dirección MAC del destinatario de la trama.
- **Dirección 2**: contiene la dirección MAC del emisor de la trama.
- **Dirección 3**: contiene el BSSID, que es la dirección MAC de la interfaz del router a la que está conectado el BSS en la que se encuentra el dispositivo destino.
- **Dirección 4**: se utiliza cuando los APs se envían tramas entre sí en modo ad-hoc, es decir, cuando las envían directamente sin pasar por un router.
- **Control de Trama**: De este campo se utilizan solamente los bits *DesdeAP* y *HaciaAP*, y según el valor de este par de bits, las direcciones 1 a 4 toman diferentes significados:
  - HaciaAP = DesdeAP = 0: se trata de una transmisión entre dispositivos de un mismo BSS
    - Dirección 1: dirección MAC destino
    - Dirección 2: dirección MAC origen
    - Dirección 3: BSSID
  - HaciaAP = 1, DesdeAP = 0: es una trama enviada a la infraestructura
    - Dirección 1: BSSID
    - Dirección 2: dirección MAC origen

    - Dirección 3: dirección MAC destino
- HaciaAP = 0, DesdeAP = 1: trama que se recibe desde la infraestructura
    - Dirección 1: dirección MAC destino
    - Dirección 2: BSSID
    - Dirección 3: dirección MAC origen
- HaciaAP = DesdeAP = 1: se trata de una trama enviada a un dispositivo en otro BSS
    - Dirección 1: dirección MAC del AP receptor
    - Dirección 2: dirección MAC del AP transmisor
    - Dirección 3: dirección MAC destino
    - Dirección 4: dirección MAC origen

2.5.2. Bluetooth

El estándar IEEE 802.15.1, Bluetooth [8], consiste en las especificaciones de la capa física y Control de Acceso al Medio (MAC) para conectar dispositivos inalámbricos disímiles. La versión Bluetooth Low Energy (BLE) puede alcanzar una tasa de transferencia de 24 Mbps y un alcance entre 70 y 100 metros con alta eficiencia energética [81]. La mayoría de las soluciones de localización basadas en BLE utilizan como dato de entrada el RSSI, ya que los sistemas basados en el RSS son menos complejos, aunque esto limita su precisión de localización.

Existen dos variantes del estándar, Basic Rate/Enhanced Data Rate (BR/EDR), también conocido como Bluetooth Classic, correspondiente a las versiones 1.x, 2.x, y 3.x, y BLE, que son las versiones 4.x y 5.x del estándar. Ambas variantes operan en la banda ISM a 2,4 GHz. Las versiones BR/EDR y BLE son incompatibles, no sólo porque el formato del paquete difiere en cada versión, sino que además las señales son moduladas y demoduladas de forma diferente.

En BR/EDR, el rango de frecuencias está dividido en 79 canales físicos, cada uno separado por 1 MHz del siguiente. La comunicación se realiza luego de haber establecido una conexión entre dos o más dispositivos, mediante una red que recibe el nombre de piconet. Una piconet consta de un maestro y hasta siete esclavos. La comunicación entre dispositivos es siempre entre el maestro y un esclavo, y nunca entre esclavos.

En BLE, se tienen 40 canales físicos, separados por 2 MHz. Los canales 37, 38 y 39 se usan exclusivamente para Advertisement, y se denominan *Primary Advertising Channels*. Los canales 0 al 36 son usados para transferencia de datos durante una conexión. Existen los mensajes de tipo broadcast y los dirigidos a un dispositivo específico. Los mensajes de tipo broadcast se denominan *Advertisement Packets*, y pueden ser leídos por cualquier dispositivo que se encuentre "escuchando" en los canales de Advertisement, a la espera de paquetes de este tipo. Para enviar paquetes de datos a un dispositivo específico es necesario haber establecido una conexión, mediante una piconet. En BLE una piconet consta de dos dispositivos, donde uno opera como maestro y el otro como esclavo.

Los sensores con los que se cuenta para realizar este proyecto, además de contar con una interfaz Wi-Fi, tienen una interfaz Bluetooth, que también será utilizada para la localización. A continuación se describirá las estructura de los paquetes Bluetooth, para cada versión del estándar, y se describirán los campos utilizados.

2.5.2.1. Paquete BR/EDR

A continuación, se describe brevemente el paquete BR/EDR, para tener una idea de su contenido, y los casos en los que puede ser utilizado para localización. La figura 2.3 muestra la estructura del paquete BR/EDR [9], que está dividido en tres componentes:

- **Access Code**: es utilizado para sincronización y para identificar los paquetes intercambiados en el canal físico. Cuando existe una conexión (piconet), el access code identifica a la piconet. Cuando no existe una conexión, se utilizan access codes específicos para el descubrimiento de otros dispositivos Bluetooth. En la figura 2.4 se muestra el formato del Campo Access Code del paquete.

- **Header**: el cabezal está estructurado como se muestra en la figura 2.5. El único campo que contiene una dirección en el cabezal es *LT_ADDR*, de 3 bits, pero se trata de una dirección lógica dentro de la piconet, y sirve para identificar al dispositivo esclavo origen o destino del paquete, o 000b para broadcast del maestro a los esclavos.

- **Payload**: En este campo se envía la carga útil. Los paquetes en los que

se puede encontrar una BD_ADDR es en los de tipo *Frequency Hop Synchronization* (FHS), utilizados en los mensajes *Inquiry Response* y *Page Master Response*, explicados más adelante. El formato del Payload de los paquetes FHS está dado por la figura 2.6. Los campos *LAP*, *NAP* y *UAP* corresponden a la parte menos significativa, la parte no significativa, y a la parte alta de la BD_ADDR (figura 2.7) del dispositivo que envía el paquete *Inquiry Response*.

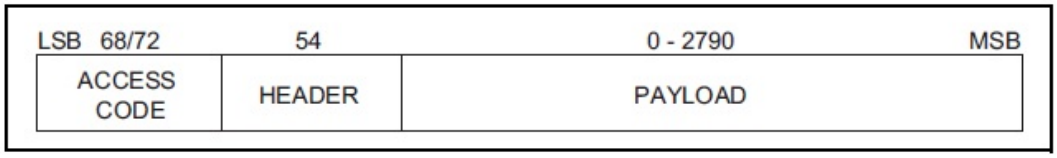


Figura 2.3: Paquete BR/EDR (obtenido de [9])

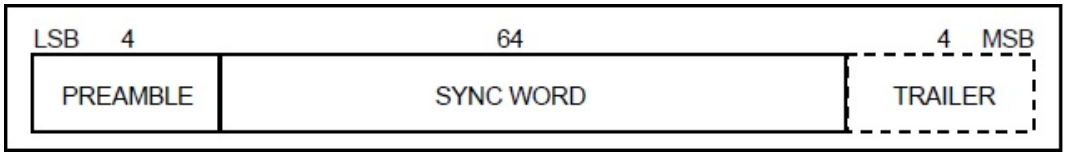


Figura 2.4: Access Code del Paquete BR/EDR (obtenido de [9])

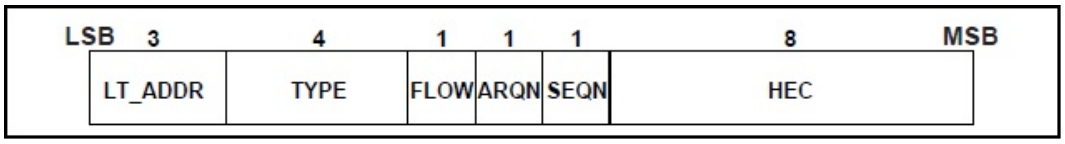


Figura 2.5: Cabezal del Paquete BR/EDR (obtenido de [9])

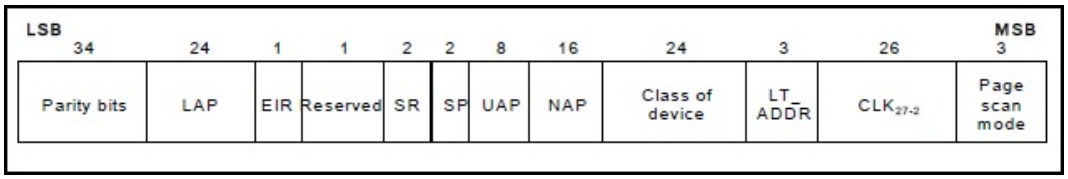


Figura 2.6: Payload de un Paquete BR/EDR de tipo FHS (Inquiry Response o Page Master Response) (obtenido de [9])

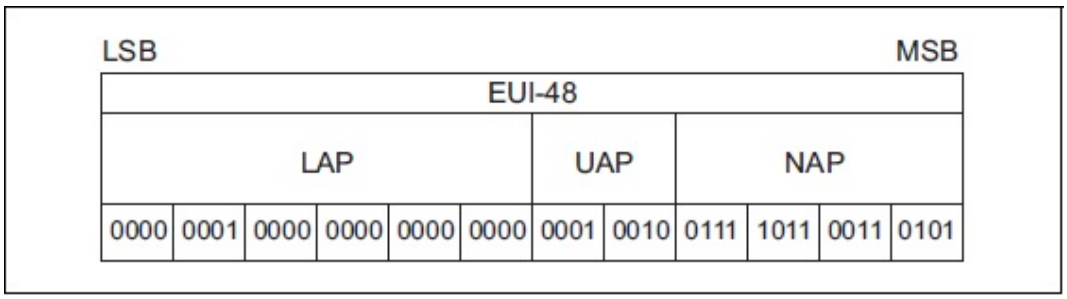


Figura 2.7: Formato de la BD_ADDR (obtenido de [8])

Para establecer una conexión a una piconet primero es necesario realizar un proceso de descubrimiento, para descubrir los dispositivos que aceptan una conexión. Los pasos para establecer una conexión son los siguientes:

1. El dispositivo que actuará de maestro envía periódicamente mensajes de tipo *ID Packet*, o mensajes *Inquiry*, con un *Access Code* específico para esto, con el objetivo de encontrar a los dispositivos que están dispuestos a ser descubiertos.
2. Los dispositivos esclavos responden al mensaje *Inquiry* con un mensaje *Inquiry Response*, en un paquete de tipo FHS, por lo que contiene su BD_ADDR.
3. El maestro, al recibir el *Inquiry Resaponse*, envía un mensaje *Page Master Response*, también en un paquete de tipo FHS, que contiene su BD_ADDR, además de otra información utilizada para sincronización.

2.5.2.2. Paquete BLE

A continuación se hace una breve descripción del paquete BLE, y se indica en qué tipos de mensajes es posible identificar un dispositivo por su BD_ADDR. La figura 2.8 muestra la estructura del paquete BLE [10]. Al igual que en BR/EDR, no todos los paquetes de BLE se tiene una BD_ADDR para identificar a los dispositivos emisores.

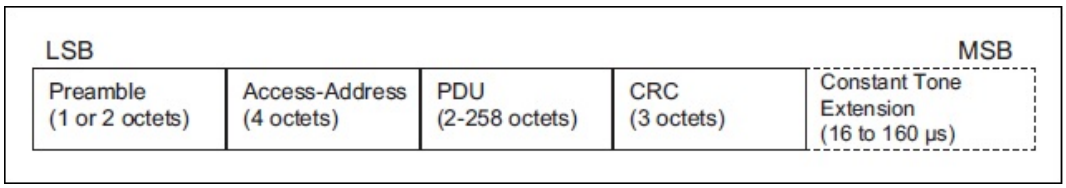


Figura 2.8: Paquete BLE (obtenido de [10])

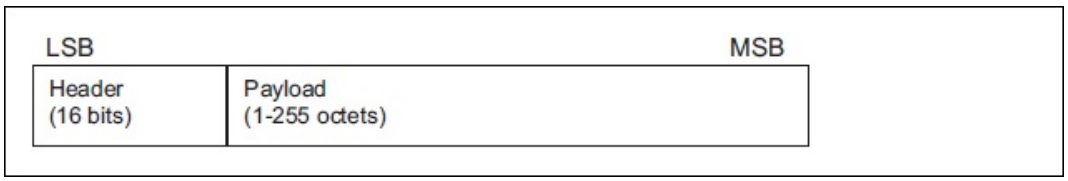


Figura 2.9: Advertising Physical Channel PDU de un Paquete BLE (obtenido de [10])

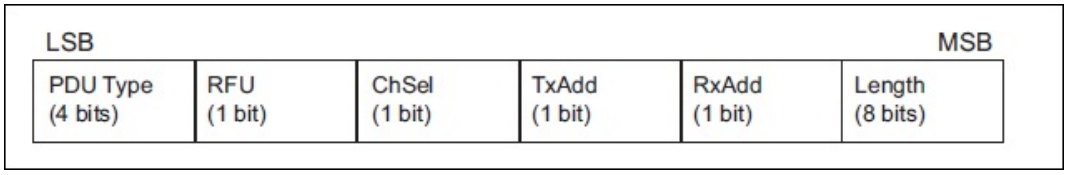


Figura 2.10: Cabezal del Advertising Physical Channel PDU de un Paquete BLE (obtenido de [10])

- **Access Address (AA)**: En caso de tratarse de un paquete de Advertisment, este campo tiene el valor fijo 0x8E89BED6, mientras que si es un paquete de datos, es decir que existe una conexión, identifica a la conexión, y es único a nivel de dispositivo.

- **PDU**: Se denomina *Advertising Physical Channel PDU* o *Data Physical Channel PDU*, según si se envía por un canal de advertisement o uno de datos, respectivamente.

  - Advertising Physical Channel PDU: Como muestra la figura 2.9, este PDU consta de un cabezal (figura 2.10) y un payload, cuyo formato depende del tipo de PDU, indicado en el campo *PDU Type* del cabezal del PDU.

  - Data Physical Channel PDU: El formato de este PDU depende de los datos que se están enviando, y en este caso no se envía un identificador de dispositivo, ya que la identificación de los paquetes en una red se realiza a través del *Access Address*.

A continuación se describen los tipos de PDU de los canales de Advertisement que son los que pueden contener un identificador de dispositivo (BD_ADDR), y que pueden ser utilizados en localización sin la necesidad de que exista una conexión. Como se dijo antes, el tipo de PDU está dado por el

campo *PDU Type*, presente en el cabezal del PDU, y determina el formato del payload. Existen los *Advertising PDUs* (*ADV_IND*, *ADV_DIRECT_IND*, *ADV_NONCONN_IND*, *ADV_SCAN_IND*), los *Scanning PDUs* (*SCAN_REQ*, *SCAN_RSP*) y los *Initiating PDUs* (*CONNECT_IND*):

- **ADV_IND**: *Advertising Indications (Connectable and Scannable Undirected Advertising)*, figura 2.11. Advertisement no dirigido, enviado por los dispositivos que quieren ser descubiertos para establecer una conexión con cualquier dispositivo que lo detecte. El campo *AdvA* contiene la BD_ADDR del dispositivo emisor (Advertiser). Un dispositivo es *Connectable* si acepta conexiones (mensajes de tipo CONNECT_IND), y *Scannable* si acepta mensajes de tipo SCAN_REQ.

- **ADV_DIRECT_IND**: *Connectable Directed Advertising*, figura 2.12. Similar a ADV_IND, pero dirigido a un dispositivo específico. El campo *AdvA* contiene la BD_ADDR del dispositivo emisor, quien espera por una conexión, y *TargetA* la BD_ADDR del dispositivo al que está dirigido el paquete. En este caso, el dispositivo AdvA acepta conexiones únicamente de TargetA.

- **ADV_NONCONN_IND**: *Non-connectable, Non-scannable, Undirected Advertising*, figura 2.11. El formato de este payload es igual al ADV_IND. Son mensajes Broadcast, enviados por un dispositivo que no espera por una conexión. En el campo *AdvData* se envían los datos del mensaje.

- **ADV_SCAN_IND**: *Scannable Undirected Advertising*, figura 2.11. Igual que ADV_NONCONN_IND, pero este tipo de mensaje indica que el dispositivo que lo envía acepta mensajes de tipo SCAN_REQ.

- **SCAN_REQ**: *Scan Request*, figura 2.13. El campo *ScanA* indica la BD_ADDR del dispositivo que envía el mensaje SCAN_REQ, y *AdvA* la BD_ADDR del dispositivo al que es dirigido. Sirve para pedir información adicional de un dispositivo del que se recibió un mensaje *Scannable*.

- **SCAN_RSP**: *Scan Response*, figura 2.14. Sirve para enviar información adicional del dispositivo advertiser. Es enviado como respuesta a un SCAN_REQ. El campo *AdvA* contiene la BD_ADDR del dispositivo que envía el mensaje, y *ScanRspData* puede contener información del dispositivo.

- **CONNECT_IND**: *Connection Indication*, figura 2.15. Es utilizado para iniciar una conexión. El campo *InitA* es la BD_ADDR del dispositivo que inicia la conexión, y *AdvA* la BD_ADDR del dispositivo destinatario del mensaje. En el campo *LLData* se envían datos necesarios para establecer la conexión (i.e., el *Access Address* de la conexión a crear y otros datos para sincronización).

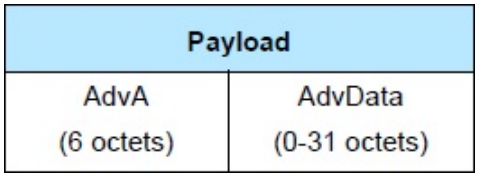


Figura 2.11: Payload de los PDU ADV_IND, ADV_NONCONN_IND y ADV_SCAN_IND de un Paquete BLE (obtenido de [10])

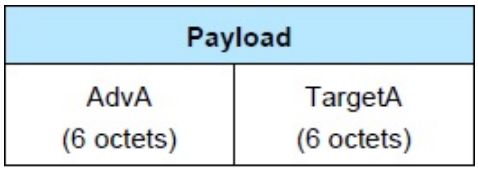


Figura 2.12: Payload del PDU ADV_DIRECT_IND de un Paquete BLE (obtenido de [10])

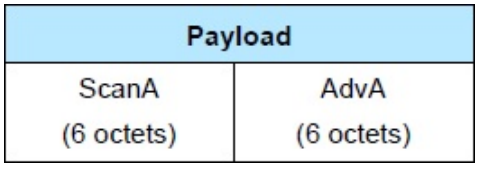


Figura 2.13: Payload del PDU SCAN_REQ de un Paquete BLE (obtenido de [10])

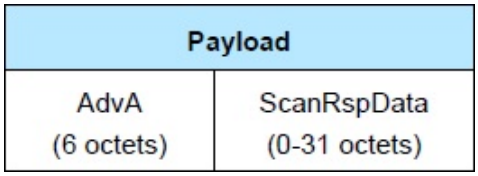


Figura 2.14: Payload del PDU SCAN_RSP de un Paquete BLE (obtenido de [10])

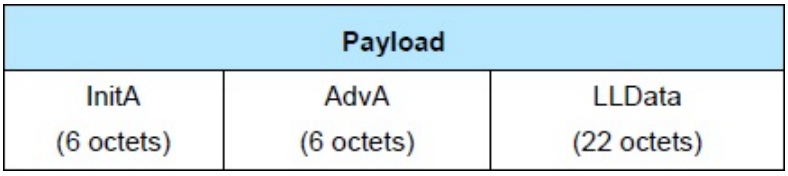


Figura 2.15: Payload del PDU CONNECT_IND de un Paquete BLE (obtenido de [10])

En todos los casos anteriores, las BD_ADDR enviadas pueden ser la pública o una aleatoria, lo que es indicado por los campos *RxAdd* y *TxAdd* del cabezal del PDU. *TxAdd* y *RxAdd* hacen referencia a la BD_ADDR del emisor y el destinatario, respectivamente, del mensaje, donde 0 indica que se trata de una BD_ADDR pública y 1 que se trata de una aleatoria.

Cabe mencionar que en la especificación de BLE se define el *Link Layer Device Filtering* ([8] Vol. 6, Parte B, Sección 4.3), que establece cómo filtrar los paquetes dirigidos. En caso de que el destinatario de un paquete dirigido no sea el dispositivo que lo recibe, el controlador de la capa de enlace lo descarta.

2.5.3. Zigbee

Zigbee está diseñado bajo el estándar IEEE 802.15.4, y se basa en la capa física y MAC para redes de área personal de bajo costo, baja tasa de transferencia, y eficiencia energética [81]. Zigbee define los niveles más altos en el stack de protocolos y básicamente se usa en Redes de Sensores Inalámbricos (WSN). La Capa de Red es responsable del enrutamiento con saltos múltiples (multi-hop) y de la organización de la red, mientras que la capa de aplicación se encarga de la comunicación distribuida y del desarrollo de la aplicación. La desventaja de esta tecnología es que no está disponible en la mayoría de los dispositivos de usuario.

2.5.4. RFID

Dispositivo de Identificación por Radio Frecuencia (RFID) está diseñado principalmente para la transmisión y almacenamiento de información enviando campos electromagnéticos desde transmisores a cualquier dispositivo de Radio Frequency (RF) compatible [81]. Un sistema RFID consiste en un lector que se puede comunicar con las etiquetas (tags) RFID. Los tags emiten datos que

los lectores pueden leer, mediante un protocolo predefinido. Hay dos tipos de sistemas RFID:

- RFID Activos: Operan en los rangos de frecuencia Frecuencia Ultra Alta (UHF) y de microondas. Cuentan con una fuente de energía y transmiten su ID periódicamente y pueden operar a cientos de metros del lector RFID. Los RFIDs activos pueden ser usados para localización y rastreo de objetos, dado que tienen un alcance aceptable, son de bajo costo, y pueden ser fácilmente embebidos en los objetos a rastrear. La desventaja es que la tecnología RFID activa no logra una buena precisión sub-metro y no está disponible en la mayoría de los dispositivos de los usuarios.
- RFID Pasivos: Operan sin batería, pero están limitados a distancias cortas (entre 1 y 2 metros). Son más pequeños, más livianos y menos costosos que los activos. Trabajan en las frecuencias bajas, altas, UHF y de microondas. Su alcance limitado los hace inadecuados para localización en interiores.

2.5.5. UWB

En Banda Ultra Ancha - Ultra Wideband (UWB) se envían pulsos ultracortos con períodos menores a 1 ns a una frecuencia entre 3.1 y 10.6 GHz, usando un ciclo de trabajo bastante bajo [50], lo que reduce el consumo de energía. UWB ha sido una tecnología bastante utilizada para localización en interiores, dado que no tiene interferencia con otras señales (dado su tipo y espectro de señal diferentes). Las señales de UWB pueden penetrar varios materiales, incluyendo paredes, pero los metales y los líquidos pueden interferir con ellas. Dada la corta duración de los pulsos de UWB, son menos sensibles a los efectos de trayectos múltiples, lo que permite una estimación precisa utilizando ToF, que ha mostrado una precisión de localización de hasta 10 cm [81]. Sin embargo, el lento avance en el desarrollo de esta tecnología ha limitado su uso en dispositivos de usuario.

2.5.6. Luz Visible

La Comunicación por Luz Visible (VLC) es una tecnología para transferencia de alta velocidad [47] que usa la luz visible entre 400 y 800 THz, y es modulada y emitida por Diodos Emisores de Luz (LEDs). Las técnicas de

localización basadas en luz visible utilizan sensores de luz para medir la ubicación y dirección de los LEDs. En este caso, AoA es considerada la técnica de localización más precisa [47]. La principal limitante es que se requiere LoS entre el LED y el sensor para una localización precisa.

2.5.7. Señal Acústica

Con esta tecnología se aprovechan los micrófonos presentes en los teléfonos inteligentes para capturar las señales acústicas emitidas por los RNs. Lo más común ha sido transmitir señales acústicas moduladas, conteniendo timestamps, y son capturadas por los sensores del micrófono, para luego estimar la ubicación utilizando ToF [51]. También se puede usar los cambios de fase y de frecuencias causados por el efecto Doppler, debido a los teléfonos en movimiento, para calcular la velocidad en que éstos se mueven. Si bien ha mostrado tener buena precisión en la localización, tiene como desventaja que solo las señales acústicas en las bandas audibles (menores a 20 KHz) brindan estimaciones precisas. Por esta razón la potencia de transmisión debería ser lo suficientemente baja para no causar contaminación sonora, y son necesarios algoritmos avanzados de procesamiento de señales para lograr la detección en el receptor. Otra desventaja es la necesidad de infraestructura adicional (i.e., fuentes acústicas/RNs) y la alta tasa de actualización, que impacta en la batería del dispositivo.

2.5.8. Ultrasonido

Los sistemas de localización que utilizan ultrasonido como tecnología utilizan el método ToF de las señales (mayores a 20 KHz) y la velocidad del sonido para calcular la distancia entre Tx y Rx. Brindan una precisión de localización a nivel de centímetros [81] y rastrean múltiples nodos móviles al mismo tiempo, con una alta eficiencia energética y sin filtraciones entre habitaciones. La transmisión de señales de ultrasonido suelen ser acompañadas de un pulso de RF para sincronización. A diferencia de las señales de RF, la velocidad del sonido varía significativamente ante cambios de temperatura o de la humedad del ambiente, por lo que se suelen instalar sensores de temperatura junto con los sistemas de ultrasonido para tener en cuenta estos factores [40]. Una desventaja es que una fuente permanente de sonido podría degradar el desempeño del sistema sensiblemente.

2.6. Tecnología y Hardware de IoT

En esta sección se hace una breve introducción a la tecnología y el hardware de IoT disponible para la realización del presente proyecto.

2.6.1. LoRa y LoRaWAN

2.6.1.1. LoRa

LoRa (Long Range) es una técnica de modulación de espectro ensanchado (spread spectrum modulation), derivada de la tecnología Chirp Spread Spectrum (CSS) [66], y es propiedad de Semtech. Es una implementación de capa física.

Fue diseñado para IoT y para operar en dispositivos a batería, requiriendo un muy bajo consumo de energía (duración de hasta diez años), para envío de pequeñas cantidades de información. Puede tener un alcance de hasta 5 Km en áreas urbanas y hasta 15 Km en áreas rurales (LoS).

Se definen canales de subida (uplink) con un ancho de banda de 125 KHz o 500 KHz, y de bajada (downlink) con un ancho de banda de 500 KHz. La codificación de los datos se denomina *Spreading Factor* (SF), e indica la cantidad de bits a utilizar para representar un símbolo. Trabaja con seis valores de SF (SF7 a SF12), y cuanto mayor es el SF, más lejos podrá ser transmitida la información, dado que será capaz que corregir mayor cantidad de errores en la transmisión. En los canales de uplink de 125 KHz se puede usar los SF 7 a 10, y en los canales de uplink de 500 KHz, es posible usar los seis valores de SF.

La tasa de transferencia (DR, Data Rate) depende del ancho de banda utilizado y del SF. Para la banda AU915-928 se definen los DR que se muestran en la tabla 2.1, donde también se indica el tamaño máximo de payload para cada uno.

2.6.1.2. LoRaWAN

Por otro lado, LoRaWAN es un protocolo abierto Low Power Wide Area Network (LPWAN), basado en la tecnología LoRa, para comunicación bi-direccional con cifrado de datos. Opera en las bandas de frecuencias ISM. En Uruguay se utiliza el plan de frecuencias definido para Australia (915-928 MHz) [20, 52]. Los componentes principales de una red LoRaWAN son:

Data Rate	Configuración	Máximo Payload (Bytes)
0	SF12 / 125 kHz	59
1	SF11 / 125 kHz	59
2	SF10 / 125 kHz	59
3	SF9 / 125 kHz	123
4	SF8 / 125 kHz	250
5	SF7 / 125 kHz	250
6	SF8 / 500 kHz	250
7	RFU	No definido
8	SF12 / 500 kHz	61
9	SF11 / 500 kHz	137
10	SF10 / 500 kHz	250
11	SF9 / 500 kHz	250
12	SF8 / 500 kHz	250
13	SF7 / 500 kHz	250
14	RFU	No definido
15	Definido en LoRaWAN	No definido

Tabla 2.1: Data Rate para AU915-928, sin límite de tiempo (obtenido de [52])

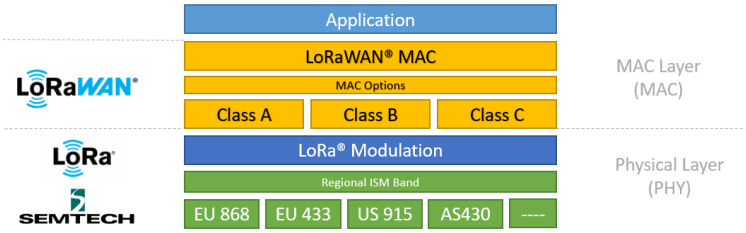


Figura 2.16: Stack de LoRaWAN (obtenido de [66])

- Dispositivos Finales (*LoRa-based End Devices*): Son los sensores o actuadores, que cuentan con un módulo LoRa, y generalmente son dispositivos a batería.

- Gateways (*LoRaWAN Gateways*): Reciben los mensajes de todos los dispositivos finales que están al alcance, y los redirige al servidor LoRaWAN (LNS, LoRaWAN Network Server), mediante una conexión IP. No existe una asociación entre los dispositivos finales y los gateways, varios gateways pueden reenviar los mensajes de un mismo dispositivo final.

- Servidor de Red (*Network Server*): El LNS administra la red, es decir, gestiona los parámetros de la red y establece las conexiones seguras AES de 128 bits, de extremo a extremo, para la transmisión de los datos. El LNS también asegura la autenticidad de los dispositivos finales y la integridad de los mensajes, y no tiene acceso a los datos de aplicación.

- Servidores de Aplicación (*Application Servers*): Son los responsables de interpretar y procesar la información enviada por los dispositivos finales (uplinks). También son quienes generan datos para los dispositivos finales (downlinks).

Existen tres clases de dispositivos LoRaWAN (nodos, o dispositivos finales), A, B y C [53]:

- **Clase A**: Esta clase debe ser soportada por todos los dispositivos LoRaWAN. La comunicación siempre es iniciada por el dispositivo final, y es asincrónica. El envío de cada mensaje (uplink) consta de tres ventanas, Tx, Rx1 y Rx2 [65]. La ventana Tx es en la que se realiza el uplink, y las ventanas Rx1 y Rx2 son intervalos de tiempo en los que el nodo clase A espera por mensajes de la red (downlinks). No se requiere que estos dispositivos esperen por downlinks en otro momento que no sea Rx1 o Rx2. Esto hace que la clase A sea el modo de operación con menor consumo de energía, dado que mientras no hay un uplink, el dispositivo puede estar en modo sleep.

- **Clase B**: Tiene las ventanas Tx y Rx de la clase A, y se agrega una sincronización con la red, utilizando balizas (beacons) periódicas y ventanas de donwlink a horas programadas. Esto permite recibir downlinks

de la red con una latencia determinista, pero con un consumo adicional de energía.

- **Clase C**: Tiene las ventanas Tx y Rx de la clase A y, además, el dispositivo puede recibir downlinks en cualquier momento, siempre que no esté realizando un envío. Con esto, el servidor de la red puede iniciar una comunicación en cualquier momento. Esto requiere un consumo aún mayor que en la clase B, por lo que es apropiado para dispositivos conectados continuamente a una fuente de energía.

2.6.2. ESP32 y ESP-IDF

ESP32 es una serie de microcontroladores programables con tecnología de 40 nm, con conectividad Wi-Fi y Bluetooth integrada, y un bajo consumo de energía [24]. Para el desarrollo sobre estos módulos, el fabricante brinda el ESP-IDF (Espressif IoT Development Framework) [25], un framework para del desarrollo de IoT.

Las placas disponibles para este proyecto contienen un módulo ESP32-WROOM-32 y un transceptor LoRa RFM95W [67], que permite crear una puerta de enlace LoRa de un solo canal.

2.6.3. LuaRTOS

LuaRTOS es un sistema operativo de tiempo real diseñado para ejecutar en sistemas embebidos, disponible para las plataformas ESP32, ESP8266 y PIC32MZ [75]. Su arquitectura consta de tres capas:

1. **Top Layer**: La capa superior es un intérprete de Lua 5.3.4, al que se le agrega una API, compuesta por módulos, para acceso al hardware y a otros servicios (Lua Threads, LoRaWAN, MQTT, etc.).

2. **Middle Layer**: Le sigue una capa con un micro-kernel de tiempo real, desarrollado utilizando FreeRTOS [1].

3. **Bottom Layer**: La capa inferior es la abstracción al hardware, y es la que interactúa directamente con el hardware, haciendo uso del framework ESP-IDF.

Permite acceder a la API Lua mediante la programación de scripts escritos en este lenguaje. Los scripts Lua pueden ser editados directamente desde el editor de LuaRTOS o pueden ser creados externamente y copiados al file system de la placa. Estos scripts luego son compilados y ejecutados por LuaRTOS.

Entre los módulos que brinda están:

- *net.wf*: para acceder a la interfaz Wi-Fi.
- *bt*: para acceder al módulo de Blutooth en la versión Low Energy.
- *lora*: para realizar envíos mediante LoRa/LoRaWAN, y para trabajar como un gateway LoRaWAN.
- *thread*: para manejo y sincronización de hilos.

Brinda una consola, accesible a través del puerto serial, que permite ejecutar comandos, scripts Lua y editar archivos.

Una vez instalado el sistema operativo en el microcontrolador, es el firmware que será ejecutado por éste al iniciar. LuaRTOS permite que, de forma automática, luego del inicio se ejecute código de usuario mediante la creación de los scripts opcionales *system.lua* y *autorun.lua*, y son ejecutados en ese orden. La idea del script *system.lua* es que en éste se realicen las configuraciones del sistema necesarias (i.e., modificar configuraciones por omisión, colocar la configuración de conexión a redes, etc.). Por otro lado, el script *autorun.lua* está pensado para configurar y ejecutar la aplicación de usuario.

2.7. Trabajos Existentes

2.7.1. Introducción

En esta sección se presentan algunos trabajos académicos de sistemas de localización en interiores que hacen uso de Wi-Fi y Bluetooth. Se hace una separación en sistemas MBL Wi-Fi, MBL Bluetooth, DBL Wi-Fi y DBL Bluetooth.

Según se pudo ver, en los trabajos encontrados definen el error como la distancia de la ubicación real a la ubicación calculada, independientemente del método utilizado. En la mayoría definen su precisión como la mediana del error, y en algunos casos mencionan el error promedio. Dada la Función de

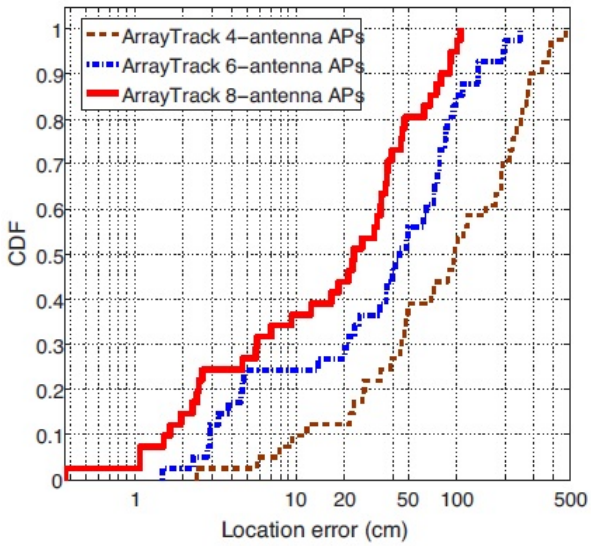


Figura 2.17: ArrayTrack - CDF (obtenido de [77])

Distribución Acumulada (CDF) del error $f(x) = P(error \leq x)$, la mediana es el valor e_m tal que $f(e_m) = 0,5$, es decir que el 50 % de las estimaciones de la ubicación tienen un error menor o igual a e_m[1]. Por lo tanto, cuando se habla de precisión, se está haciendo referencia a este valor. Para los trabajos presentados se mostrará la CDF, cuando esté disponible, y se indicará su precisión.

2.7.2. MBL con Wi-Fi

2.7.2.1. ArrayTrack

ArrayTrack [77] utiliza AoA medido en los APs Wi-Fi. Tiene una precisión de 23 cm cuando utiliza APs con ocho antenas. Para un cálculo preciso de AoA usa una versión modificada del algoritmo MUSIC (Multiple Signal Classification) [62]. Tiene una gran precisión en tiempo real y es escalable, pero una de sus limitaciones es que requiere hardware más específico, dado que utiliza un gran número de antenas. Cada AP consta de dos placas WARP (Wireless open-Access Research Platform), de Rice, basadas en FPGA (Field Programmable Gate Array). Muestran que a medida que agregan APs, la precisión aumenta. En la figura 2.17 se muestra la CDF de ArrayTrack.

[1]La mediana también se define como el valor que se encuentra en el centro de una secuencia ordenada de datos.

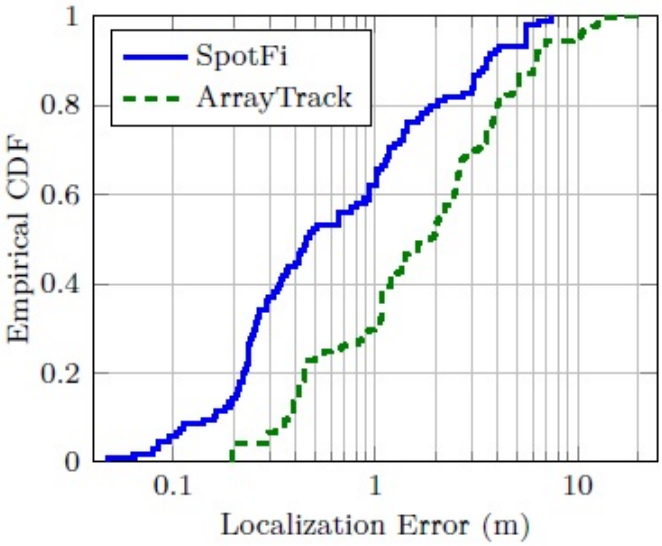


Figura 2.18: SpotFi - CDF (obtenido de [43])

2.7.2.2. SpotFi

SpotFi [43] hace uso del CSI y el RSSI, y utiliza AoA y ToF. Tiene una precisión de 40 cm. Usa interfaces Wi-Fi estándar (NICs Wi-Fi Intel 5300), sin necesidad de hardware costoso ni de Fingerprinting. Utiliza APs con múltiples antenas y la técnica AoA. Combina las estimaciones de los algoritmos AoA y ToF haciendo uso del CSI. No es apropiado para MBL en tiempo real dado que no puede calcular la ubicación con un número limitado de señales. La figura 2.18, muestra la CDF del error (la compara con ArrayTrack cuando utiliza cuatro antenas).

2.7.2.3. Chronos

Chronos [70] es un sistema con un único AP con varias antenas (NIC Wi-Fi Intel 5300) que utiliza ToF como método de localización. Tanto el receptor como el emisor usan saltos de frecuencia para usar varios canales y combinar el procesamiento de varias señales. Esto hace que no sea escalable por el alto consumo de energía que requiere. Tiene una precisión de localización de 65 cm en LoS y de 98 cm en Non-Line of Sight (NLoS). La figura 2.19 muestra la CDF para Chronos.

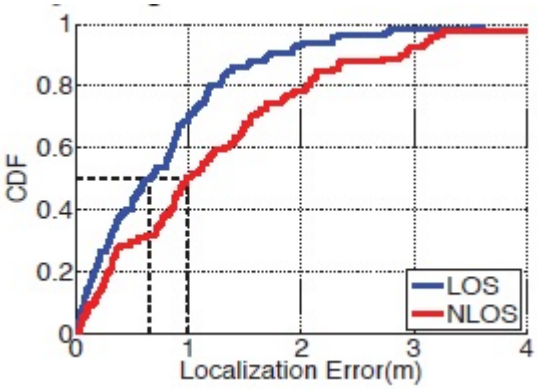


Figura 2.19: Chronos - CDF (obtenido de [70])

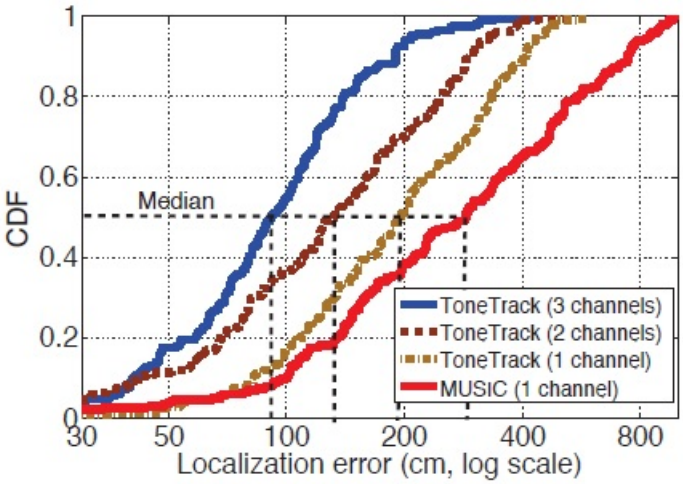


Figura 2.20: ToneTrack - CDF (obtenido de [78])

2.7.2.4. ToneTrack

ToneTrack [78] usa ToF para obtener estimaciones en tiempo real con una precisión de 0,9 m. Combina los datos de ToF obtenidos en los saltos de frecuencia (canales) del dispositivo del usuario. Afirman que al combinar la información de los diferentes canales, se obtiene una mayor precisión. Los resultados numéricos muestran que obtiene medidas bastante precisas y en tiempo real. Al igual que Chronos, se basa en el principio de combinar la información de diferentes canales. No menciona si se puede utilizar en hardware de uso comercial. En la figura 2.20 se muestran las CDFs del error para ToneTrack, según la cantidad de canales utilizados.

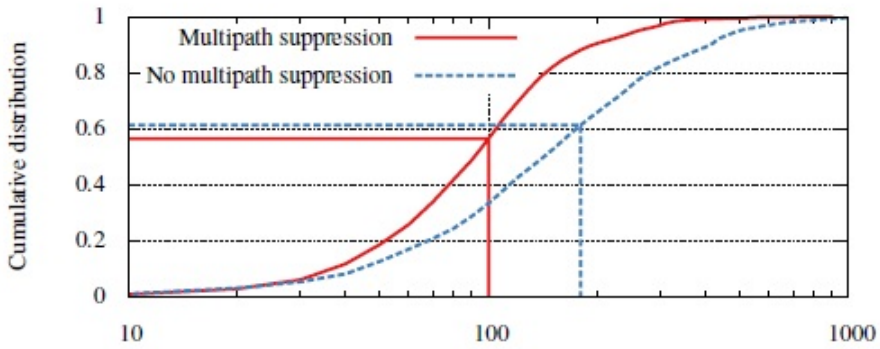


Figura 2.21: Phaser - CDF (obtenido de [34])

2.7.2.5. Phaser

Phaser [34] es una extensión de ArrayTrack (2.7.2.1), que utiliza hardware Wi-Fi disponible comercialmente para obtener el AoA, a partir de los valores obtenidos del CSI. Utiliza dos interfaces Intel 5300 802.11 de tres antenas, haciendo que ambos compartan una antena, obteniendo un total de cinco antenas. Alcanza una precisión aproximada de 2 m en condiciones normales y de 1 m eliminando el efecto de trayectos múltiples, como muestra la figura 2.21 con la CDF.

2.7.2.6. DeepFi

DeepFi [73] es un sistema de Fingerprinting basado en Deep Learning, utilizando el CSI. En la etapa offline entrenan una red neuronal (deep network con cuatro capas ocultas), y en la etapa online utilizan un método probabilístico para estimar la ubicación. Utilizan un único AP, utilizando una NIC Intel 5300. Obtiene una precisión de 0,95 m. Aproximadamente el 60 % de los puntos de prueba tiene un error por debajo de 1,5 m. La figura 2.22 muestra la CDF para DeepFi, y la compara con la de otros trabajos.

2.7.2.7. Redpin

Redpin [11] localiza dispositivos a nivel de habitación, con una precisión de 2,32 m. Utiliza Fingerprinting del RSSI de señales de Wi-Fi y Bluetooth de los APs Wi-Fi visibles y de los dispositivos Bluetooth no portables. Para la etapa online utiliza k-Nearest Neighbors (kNN) y Support Vector Machine

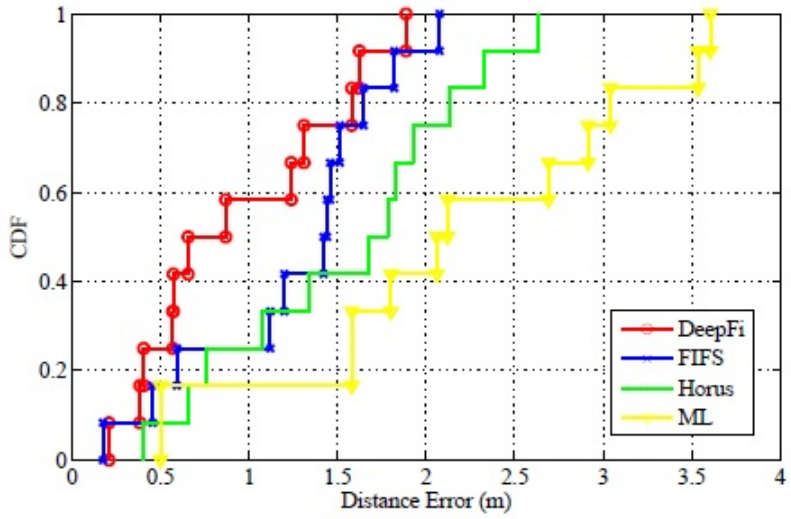


Figura 2.22: DeepFi - CDF (obtenido de [73])

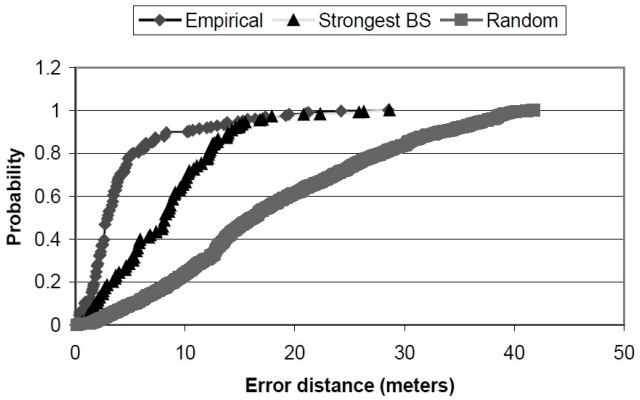


Figura 2.23: RADAR - CDF (obtenido de [5])

(SVM). No es útil para localización en interiores. Hace una comparación con LoCo (2.7.4.4), y muestra que LoCo es más preciso y tiene menor latencia que Redpin.

2.7.2.8. RADAR

RADAR [5] toma valores offline del RSSI para generar un Radio Map, que luego son correlacionados al valor online, lo que da la ubicación del usuario. Tiene una precisión de 2,94 m, la figura 2.23 muestra la CDF. Guvenc y col. [38] aplican el filtro de Kalman para mejorar la localización, lo que hace que se obtenga una precisión de 2,5 m.

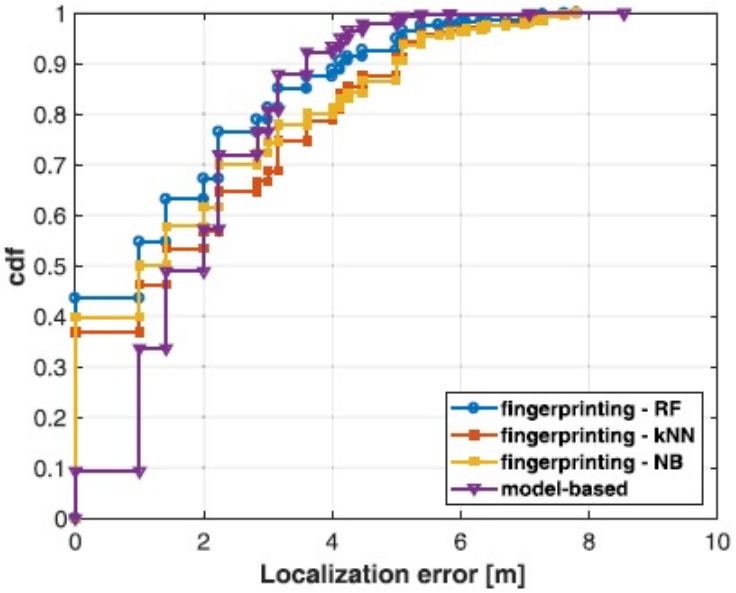


Figura 2.24: Redondi y Cesana [60] - CDF

2.7.2.9. Building up Knowledge through Passive Wi-Fi Probes

En [60] utilizan placas Rasperry Pi 3 como nodos ancla, donde de forma pasiva recolectan las tramas *Probe Request* de los dispositivos Wi-Fi al alcance. Utilizan el RSSI y dos técnicas de localización, una es Trilateración, y la segunda un método con Fingerprinting. Para el método de Trilateración utilizan el modelo de propagación mencionado en 2.4.1.1. Para Fingerprinting utilizan los algoritmos kNN, Naive Bayes (NB) y Random Forest (RF). Comparan los resultados de utilizar cuatro y seis nodos ancla, comprobando que con la segunda opción se obtiene un mejor resultado. Las figuras 2.24 y 2.25 muestran la CDF y el error promedio, respectivamente, para los métodos utilizados. La precisión es algo mayor a 1 m para RF y NB, y próxima a 2 m para kNN y el modelo de propagación. Llegan a la conclusión de que, sin importar el algoritmo utilizado, el error depende de la cantidad de nodos ancla que se instalen, y éste disminuye al aumentar la cantidad de nodos.

2.7.3. MBL con Bluetooth

2.7.3.1. Bluetooth Location Network (BLN)

Bluetooth Location Network (BLN) [35] utiliza RNs Bluetooth. El dispositivo se comunica con los RNs, que luego envía la información de ubicación a

Localization Method	Mean Error [m]
Fingerprint (RF)	2.06
Model-based	2.3
Fingerprint (RF)	1.5
Model-based	1.9

Figura 2.25: Redondi y Cesana [60] - Error promedio

un nodo maestro. Tiene un nivel de precisión de una habitación, lo que lo hace más apropiado para proximidad. Tiene un tiempo de respuesta de 11 segundos, por lo que no es apropiado para tiempo real.

2.7.3.2. Bluetooth Indoor Positioning System (BIPS)

Bluetooth Indoor Positioning System (BIPS) [13] tiene un rango menor a 10 m y es eficiente en cuanto al consumo de energía. Un dispositivo con Bluetooth se comunica con RNs bluetooth que luego usan un "*BIPS-server*" para obtener un estimado de la localización. Los RNs están interconectados mediante una red, por lo que pueden intercambiar información. Los RNs actúan como nodos maestros que detectan a los nodos esclavos (dispositivos de usuarios) en su vecindad y transfieren datos entre los usuarios y el RN. Puede localizar usuarios estáticos o que se muevan lentamente. Los clientes, para ser localizados, deben estar conectados a la piconet como esclavos, lo que agrega tiempos de descubrimiento y conexión a la piconet antes de poder realizar la localización. Debido a la latencia que presenta, no es apropiado para tiempo real.

2.7.3.3. Bluepass

Bluepass [21] usa el RSSI del dispositivo del usuario en el RN para estimar la distancia. Consiste en un servidor central, un servidor local, un dispositivo Bluetooth de detección y una aplicación en el dispositivo del usuario. El usuario debe tener la aplicación instalada y estar identificado a través de ésta (inicio de sesión). El servidor local brinda un mapa del lugar y el central relaciona los diferentes mapas. Se obtiene un error cuadrático medio (MSE) de 2,33 m.

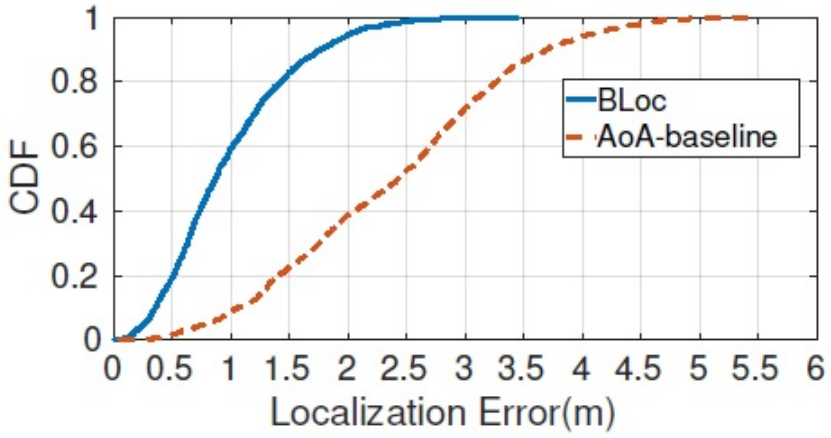


Figura 2.26: BLoc - CDF (obtenido de [4])

2.7.3.4. Ibeacon Based Proximity and Indoor Localization System

Zafari [83] usa iBeacons [41]. Los valores de RSSI son obtenidos de los iBeacons en el dispositivo del usuario, y son enviados a un servidor que ejecuta diferentes algoritmos. En el servidor se usa Particle Filter (PF) y métodos en cascada usando Kalman Filter-Particle Filter (KF-PF) y Particle Filter-Extended Kalman Filter (PF-EKF) para mejorar la precisión de localización. En promedio, PF, KF-PF y PF-EKF alcanzan una precisión de 1,441 m, 1,03 m y 0,95 m respectivamente. Es bastante preciso y consume poca energía, pero tiene un retardo significativo, y al requerir un despliegue de iBeacons, conlleva un costo adicional.

2.7.3.5. BLoc

En [4] se propone uno de los primeros trabajos usando el CSI para BLE. Utiliza AoA y tiene una precisión de entre 0,86 m y 0,92 m, y el 90 % de las estimaciones de localización tienen un error menor o igual a 1,75 m. En la figura 2.26 se muestra la CDF del error.

2.7.3.6. Rethinking Ranging of Unmodified BLE Peripherals in Smart City Infrastructure

En [42] utilizan la técnica ToF para localización. Se propone un algoritmo de *multipath profiling*, que consiste en explotar el RSSI en los diferentes canales, y utilizar el retardo de propagación de los diferentes trayectos de la señal para

rastrear cualquier etiqueta BLE en un ambiente de interiores. Afirman que casi no existen interfaces BLE que permitan obtener el CSI. Para medir la precisión, calcula el promedio del error absoluto en tres escenarios:

1) Corredor (LoS): Corredor en el que hay línea directa de visión. Se logra un error promedio de 1,83 m.
2) Corredor (NLoS): En el mismo corredor que antes, pero con obstáculos, se obtiene un error promedio de 4 m.
3) Laboratorio (LoS): Habitación, más chica que el corredor anterior. Se obtiene un error promedio de 1,5 m.

Para indicar el error global del sistema, hace un promedio de los tres errores anteriores, que es 2,44 m.

2.7.4. DBL con Wi-Fi

2.7.4.1. Zero-Configuration, Robust Indoor Localization: Theory and Experimentation

Lim y col. [49] presentan un sistema basado en el RSSI (también funcionaría en el modo MBL) que no requiere una etapa offline de Fingerprinting. Los APs obtienen los RSSI de los demás APs para obtener un mapa online de RSSI. Luego, el cliente (o la infraestructura, según el modo) mide el RSSI que es mapeado para estimar la ubicación del usuario. Tiene una precisión de 1,76 m. Requiere infraestructura adicional (monitores adicionales para mejorar el desempeño), lo que conlleva a un costo extra. Requiere un número de muestreo que puede incurrir en un retardo en la localización. La figura 2.27 muestra la CDF del error de este sistema.

2.7.4.2. Horus

Horus [80] utiliza el RSSI con una etapa de entrenamiento y Fingerprinting y una etapa online con un método probabilístico. Tiene una precisión de 39 cm, y el 90 % de las estimaciones de la localización tiene un error menor o igual a 86 cm. Es sensible a cambios en el entorno, debido al Fingerprinting y la etapa de entrenamiento offline. La figura 2.28 muestra la CDF de Horus, y se compara contra RADAR y otro sistema.

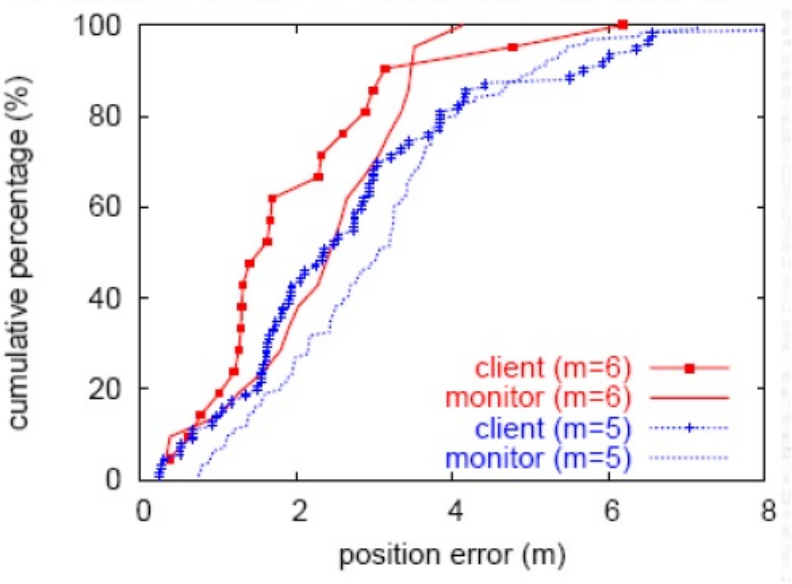


Figura 2.27: Lim y col. [49] - CDF

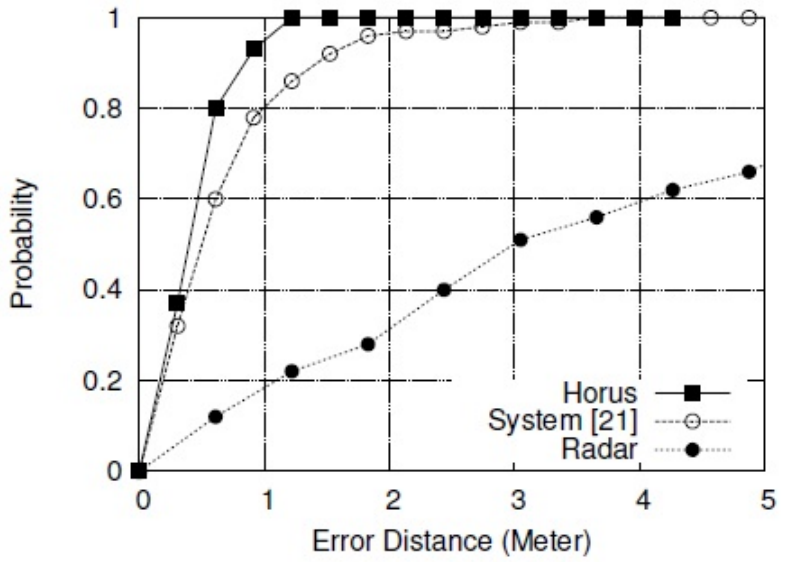


Figura 2.28: Horus - CDF (obtenido de [80])

2.7.4.3. Ubicarse

Ubicarse [46] usa una formulación de Synthetic Aperture Radar (SAR) en el dispositivo de usuario para obtener su ubicación de forma precisa. Funciona en dispositivos de usuario con al menos dos antenas. Utiliza AoA. Para simular el arreglo de antenas, el usuario debe rotar el dispositivo para emular SAR, dado que el principio básico es capturar snapshots del canal mientras el usuario rota el dispositivo en un trayecto, con lo que se obtienen AoA precisos. Tiene una precisión de 39 cm en localización 3D. Además, usando algoritmos de *Stereo Vision* y la cámara del dispositivo, brinda funcionalidad precisa de geoetiquetado.

2.7.4.4. LoCo

LoCo [7] usa APs Wi-Fi y un framework propietario para obtener un clasificador a nivel de habitación. Útil para proximidad, pero no para localización en interiores.

2.7.4.5. FILA

En [76] presentan un trabajo que utiliza el CSI y proponen dos métodos para localización. Crean un modelo de propagación, y usan un método probabilístico basado en Fingerprinting, ambos a partir del CSI. Afirman que se trata del primer trabajo que utiliza la información de capa física, el CSI, para ambos métodos. El modelo de propagación propuesto es el de la ecuación 2.14:

$$d = \frac{1}{4\pi}\left[\left(\frac{c}{f_0 \times |CSI_{eff}|}\right)^2 \times \sigma\right]^{\frac{1}{n}} \tag{2.14}$$

donde CSI_{eff} es el promedio del CSI de los subcarriers en el dominio de frecuencias, c es la velocidad de la onda, σ el factor del entorno (*environment factor*) y n el exponente de pérdida de trayecto (*path loss fading exponent*). Para calcular σ y n utilizan un algoritmo basado en Supervised Learning. Las figuras 2.29 y 2.30 muestran la CDF del error para el modelo propagación y para el método probabilístico, respectivamente. La precisión para el modelo de propagación es menor a 0,5 m, mientras que para el método probabilístico es menor a 0,6 m.

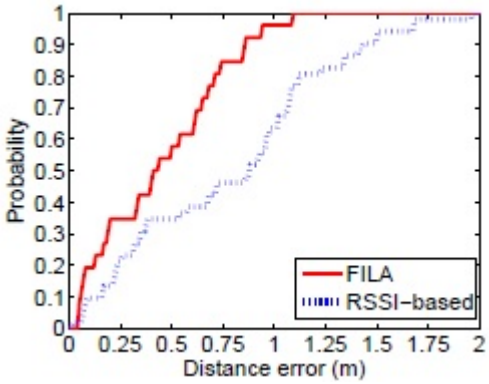


Figura 2.29: FILA - Modelo de Propagación - CDF (obtenido de [76])

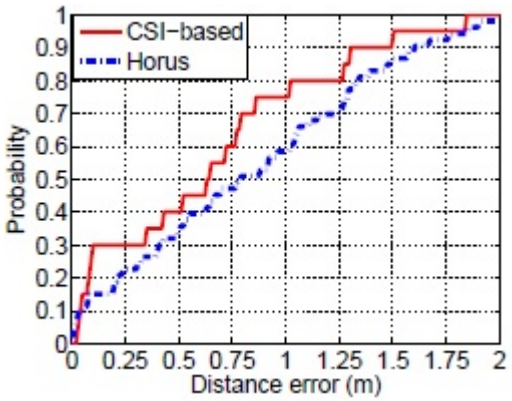


Figura 2.30: FILA - Método Probabilístico - CDF (obtenido de [76])

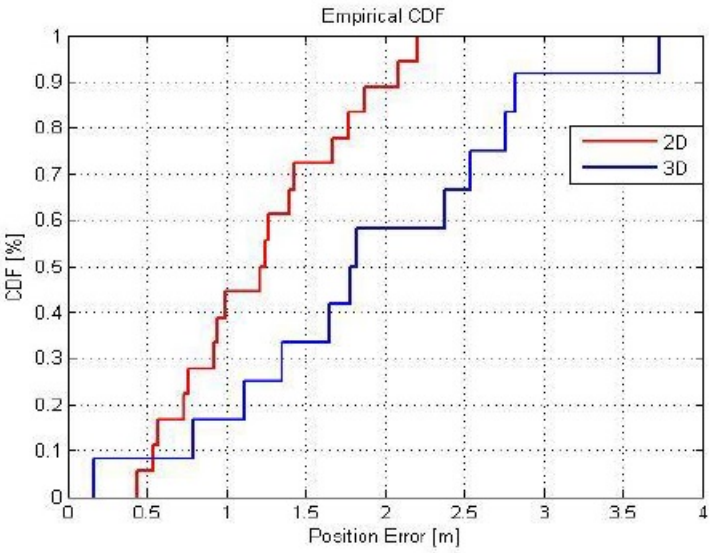


Figura 2.31: Farid y col. [27] - CDF

2.7.4.6. A WLAN Fingerprinting Based Indoor Localization Technique via Artificial Neural Network

Farid y col. [27] usan el RSSI y Fingerprinting. Utilizan una Red Neuronal Artificial (ANN) de tipo Perceptrón Multi-Capa (MLP) con dos capas ocultas. Realiza localización 2D y 3D. Para la localización 3D, coloca los APs a diferentes alturas. Tiene una precisión de 1,22 m para 2D y 1,91 m para 3D. En la figura 2.31 se muestra la CDF para este sistema.

2.7.5. DBL con Bluetooth

2.7.5.1. Enhancing iBeacon Based Micro-Location with Particle Filtering

Zafari y Papapanagiotou [82] proponen un sistema basado en iBeacons utilizando el RSSI. Se utilizan beacons como RNs, que transmiten iBeacons periódicamente. El dispositivo de usuario procesa los iBeacons y usa Particle Filtering (PF) para localizar al usuario, con una precisión de 0,97 m. No es apropiado para tiempo real debido a la limitación inherente al CoreLocation Framework de iOS, que no permite obtener el valor del RSSI a períodos menores a 1 s. Un inconveniente es que PF en el dispositivo consume mucha batería, reduciendo la carga considerablemente.

2.7.5.2. Improving Indoor Localization Using Bluetooth Low Energy Beacons

Kriz, Maly y Kozel [45] combinan localización basada en iBeacons BLE con Wi-Fi para mejorar la precisión global. Utiliza Fingerprinting con RSSI, recolectado de varios RNs. En la etapa online usa kNN como método de localización. El uso de iBeacons en conjunto con Wi-Fi mejora la precisión en un 23 %, y logra una precisión de 0,77 m. No es apropiado para tiempo real por la latencia que presenta en la obtención de los valores de RSSI de los distintos RNs.

2.7.5.3. RSSI-Based Indoor Localization With the Internet of Things

Sadowski y Spachos [61] comparan el desempeño de Wi-Fi, BLE, Zigbee y LoRaWan usando el RSSI y Trilateración, y evalúan el consumo de energía al utilizar dispositivos para IoT. Sus pruebas muestran que Wi-Fi se desempeña sensiblemente mejor que las otras tecnologías evaluadas.

2.7.6. Otros trabajos

2.7.6.1. Aplicacion de IoT con diversas tecnologías inalámbricas

[20] se trata de un proyecto de grado del grupo MINA del año 2019. En éste se desarrolló un sistema que detecta los dispositivos Wi-Fi al alcance de los nodos ancla, que son placas Sparkfun SPX-14893 [67]. Luego, la información es enviada a un servidor utilizando LoRaWAN. Las placas utilizadas cuentan con un módulo ESP32, y en éste se instaló el sistema operativo LuaRTOS [75], al que se le agregó una función para poder utilizar la interfaz Wi-Fi en modo monitor y así poder examinar todas las tramas 802.11 que le llegan. El proyecto de grado citado puede ser visto como un sistema de proximidad, o de localización con precisión a nivel de habitación (si bien, esta precisión no fue medida en ese proyecto), dado que al momento de contar los dispositivos detectados, considera que cada uno se encuentra en la habitación donde está el nodo que lo detectó con la mayor intensidad de señal (mayor valor de RSSI).

El presente proyecto utiliza parte del desarrollo allí realizado, lo relacionado a la detección de los dispositivos Wi-Fi, dado que también se utilizarán placas Sparkfun SPX-14893 como nodos. También hace uso de la experiencia obtenida

al trabajar con LoRa y LoRaWAN para el envío de la información de los dispositivos detectados.

2.7.6.2. Localización Indoor Basada en Wi-Fi

[12] también es un proyecto de grado de la Facultad de Ingeniería, para el Instituto de Ingeniería Eléctrica del año 2018, de localización en interiores para el Museo Nacional de Artes Visuales (MNAV). Se trata de un sistema para asistir a los visitantes del museo, y dar información de las obras cercanas, según la ubicación del usuario. Para la localización utiliza el framework FIND3 [28], que utiliza distintos algoritmos de Fingerprinting. Las huellas (fingerprints) son clasificadas en alguna de las zonas definidas en base a la ubicación de las obras en consideración. Como se explica más adelante, FIND3 tiene el modo activo y el modo pasivo. En el proyecto citado se hizo uso del modo activo.

En el presente proyecto, que también utiliza el framework FIND3, se hace uso de la experiencia obtenida en el proyecto citado, en lo que tiene que ver con optimizaciones realizadas a FIND3, sobre todo en la eliminación de algunos de los algoritmos que el framework utiliza, dado que algunos presentaban errores y otros aumentaban el tiempo del entrenamiento de los algoritmos, además de que su remoción no generó una menor precisión, sino lo contrario.

2.8. Frameworks para Desarrollo

A continuación se listan los frameworks y bibliotecas encontrados para desarrollo de localización en interiores.

2.8.1. FIND3

Framework for Internal Navigation and Discovery versión 3 (FIND3) [28] es un framework para localización en interiores. Es un software libre de código abierto, publicado bajo licencia MIT. Se basa en Fingerprinting y Machine Learning, utilizando los valores de RSSI. Es un sistema MBL, ya que la localización se calcula siempre en el servidor. Permite dos modos:

- **Activo**: Requiere una aplicación en el dispositivo cliente (dispositivo a ser localizado), ya que es éste quien debe detectar las señales de los nodos

ancla, junto con el RSSI, para generar un fingerprint y enviarlo al servidor principal. Un fingerprint es una colección de pares (*dirección MAC nodo ancla, RSSI*).
Para este modo el framework provee una aplicación desarrollada en Android para los dispositivos cliente, en la que los nodos ancla son todos los APs que se encuentren al alcance (propios o de terceros). Esta aplicación genera mensajes de tipo *Probe Request*, a los que los APs responden con un *Probe Response*, de los que toma la dirección MAC del AP y el RSSI. Esta aplicación cliente también está disponible para PCs con el sistema operativo Linux (requiere una NIC en modo monitor) y para placas ESP32. No se encuentra disponible para dispositivos móviles iOS dado que este sistema operativo no permite a las aplicaciones realizar un Wi-Fi scan (enviar mensajes Probe Request).

- **Pasivo**: Es necesaria infraestructura que actúe de nodos ancla. En este modo, los nodos ancla son quienes deben detectar a los clientes y enviar la información al servidor principal.
  Para el modo pasivo, el framework brinda un software para configurar como sensores PCs con sistemas operativos Linux y OS X (requiere una NIC en modo monitor), y placas Raspberry Pi.

Permite el uso de cualquier interfaz (no solo Wi-Fi y Bluetooth), dado que la interfaz se identifica por el nombre que se envíe al servidor. Además, podría usarse cualquier valor, y no solo el RSSI[1], haciendo de este framework una herramienta bastante flexible para el uso de Fingerprinting y con un bajo costo.

Permite crear lo que denomina familias, donde cada familia se puede corresponder a un edificio. Para cada familia, se definen ubicaciones. Cada ubicación será una clase, para la clasificación en Fingerprinting.

2.8.1.1. Localización

A partir de los datos recolectados de los sensores realiza una clasificación usando varios algoritmos de Machine Learning (ML). Para el entrenamiento de los algoritmos, separa los datos de forma aleatoria, tomando un 70 %

[1]Si bien no se probó enviar otro valor que no fuera el RSSI, por cómo está implementado FIND3, debería ser posible enviar cualquier valor que se tome como variable para Fingerprinting.

para entrenar los algoritmos y el 30 % restante para su evaluación. Con esta evaluación calcula la exactitud por zona y la global (porcentaje de aciertos sobre predicciones). Los datos de aprendizaje consisten en identificadores únicos (típicamente direcciones MAC o BD_ADDR) y valores de la señal (Bluetooth, Wi-Fi, o cualquier otra señal) y una etiqueta que identifica a la ubicación donde las señales fueron capturadas. Una vez entrenados los diferentes algoritmos de ML utilizados, el sistema realiza la clasificación haciendo un ensamble de estos algoritmos, obteniendo una probabilidad para cada una de las zonas definidas. En la implementación actual se usan diez clasificadores:

1. k-Nearest Neighbours (kNN)
2. Support Vector Machine (SVM) lineal
3. Support Vector Machine (SVM) no lineal
4. Árboles de Decisión
5. Bosques Aleatorios
6. Multi-layer Perceptron (MLP)
7. AdaBoost
8. Gaussian Naive Bayes
9. Quadratic Discriminant Analysis
10. Extended Naive Bayes

Los algoritmos del 1 al 9, descritos en 2.4.2, están implementados en la biblioteca scikit-learn de Python, mientras que el 10 es una implementación del desarrollador de FIND3.

Según dice en el sitio del framework, alcanza una precisión menor a 3 m^2. No hay información sobre la precisión mas que esa mención, por lo que se podría interpretar que es una precisión de 0,98 m (radio de una circunferencia con esa área), o que las zonas de tamaño mínimo pueden ser cuadrados de aproximadamente 1,75 m de lado.

2.8.1.2. Arquitectura

El framework consta de los siguientes componentes:

- **Main Server**: brinda una API REST para el almacenamiento de huellas y para obtener la ubicación de un dispositivo (entre otros). También brinda una interfaz gráfica básica que permite ver en tiempo real la actividad de los dispositivos clientes en los últimos quince minutos. Se comunica con el *AI Server* para enviar las huellas y calcular la ubicación

del cliente.
Este componente está implementado en el lenguaje Go.

- **AI Server**: es el servidor de Machine Learning. A través de una API REST recibe los fingerprints y ejecuta los algoritmos para realizar la clasificación, calculando las probabilidades de cada ubicación en función de un fingerprint. Está implementado en Python.

- **Base de datos**: se trata de una base de datos SQLite. Se almacenan los datos de familias, ubicaciones, dispositivos y datos de calibración de los algoritmos.

2.8.2. Anyplace

Anyplace [2] es un software libre de código abierto, publicado bajo la licencia MIT. Está en las categorías DBL y MBL, dado que permite el cálculo de la localización en el dispositivo del usuario y en el servidor. Requiere una aplicación para ejecutar en un teléfono móvil con sistema operativo Android, iOS o Windows Phone.

Como framework de localización, brinda las mismas prestaciones que FIND3, pero no tiene un modo pasivo, por lo que requiere de una aplicación en el cliente para recolectar la información de los AP que tiene al alcance.

2.8.2.1. Localización

Está basado en Fingerprinting del RSSI. La aplicación en el dispositivo del usuario toma el RSSI de los AP Wi-Fi al alcance y los envía al servidor. Además, hace uso del giroscopio, el acelerómetro y de la brújula digital del dispositivo para mejorar la navegación. Los algoritmos utilizados para localización son kNN y Mínimo Error Cuadrático Medio (MMSE). Tiene una precisión de 1,96 m.

2.8.2.2. Arquitectura

Consiste en cinco componentes principales:

- **Servidor**: desarrollado en Scala. Sigue una arquitectura de Big Data y brinda una API Web basada en JSON para los mapas, la navegación y la localización. Utiliza Couchbase como base de datos de backend. Brinda

direcciones de navegación al usuario basado en los Radio Maps almacenados. Incluye un módulo de privacidad que permite que la aplicación en el cliente calcule su ubicación en el dispositivo descargando datos mínimos del Radio Map. Almacena datos de edificios, planos de las plantas, y puntos de interés en formato JSON. Además, tiene varios módulos para Crowdsourcing[1] para el Fingerprinting.

- **Architect**: aplicación web desarrollada con AngularJS. Permite el diseño de los mapas, haciendo uso de la Google API para los mapas, direcciones, mapas de calor, etc. Permite insertar el plano de un edificio sobre Google Maps, trabajar con varias plantas, ingresar puntos de interés dentro del edificio y conectarlos para indicar caminos posibles para dar indicaciones de navegación, enviar invitaciones para Crowdsourcing para recolectar Radio Maps Wi-Fi y definir un edificio como público o privado.

- **Viewer**: aplicación web que permite la visualización de los modelos de edificios en Anyplace. Permite ver el mapa de los edificios sin la necesidad de descargar una aplicación.

- **Logger y Navigator**: es la aplicación cliente, desarrollada para Android y consta de dos componentes: Logger y Navigator. Navigator permite ver la ubicación del usuario y navegar entre puntos de interés. A partir de la geolocalización obtenida con la Google Geolocation API, descarga el plano, los puntos de interés y el Radio Map del edificio en que se encuentra, y calcula la localización. Logger recolecta los valores de RSSI de los AP Wi-Fi cercanos y los envía al servidor. Es usada por usuarios voluntarios para contribuir con datos de RSSI y para Crowdsourcing.

- **Data store**: almacena los modelos de los mapas de los edificios y la información de los Radio Maps.

La figura 2.32 muestra la arquitectura del sistema.

2.8.3. Redpin

Redpin [11] se encuentra en la categoría MBL, y es el trabajo descrito en 2.7.2.7. Fue creado para alcanzar una precisión tal que identificara, al menos,

[1]Crowdsourcing: Tarea que puede ser llevada a cabo por un gran número de colaboradores.

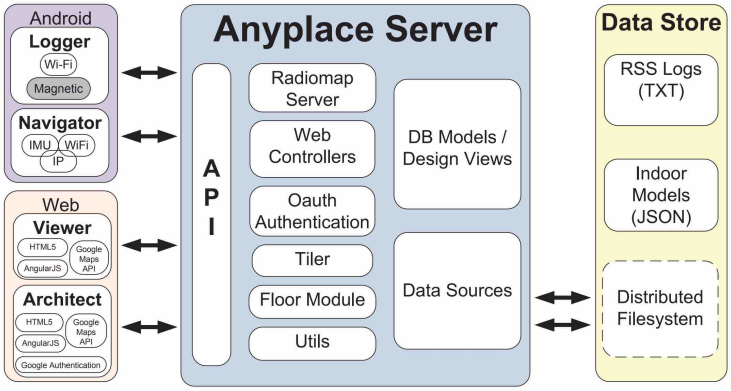


Figura 2.32: Arquitectura de Anyplace (obtenido de [2])

la habitación en la que se encuentra el cliente. Es de código abierto y está publicado bajo la licencia GNU LGPLv3. El proyecto no se mantiene desde el 2010, y la aplicación cliente para Android está implementada usando la API 8 (versión 2.2.x de Android), lo que hace que no funcione en dispositivos más nuevos.

2.8.3.1. Localización

Cada ubicación se corresponde a una etiqueta, y el proceso de localización se basa en encontrar la etiqueta que mejor se corresponde a la huella online. Usa dos algoritmos de ubicación: Si una huella consiste en pocas medidas usa kNN, mientras que para huellas con muchas medidas usa SVM. Como se mencionó antes, tiene una precisión de 2,32 m.

2.8.3.2. Arquitectura

Las figuras 2.33 y 2.34 muestran la arquitectura general del sistema y la arquitectura de la aplicación en el dispositivo móvil, respectivamente. Sus componentes son los siguientes:

- **Sniffer**: recolecta información de los dispositivos móviles al alcance para crear las huellas. Se ejecuta en el dispositivo cliente, y lee las señales de los AP disponibles. La aplicación cliente está implementada para Android y iPhone, y para celulares más antiguos se utilizó Java Micro Edition y Symbian Series 60.

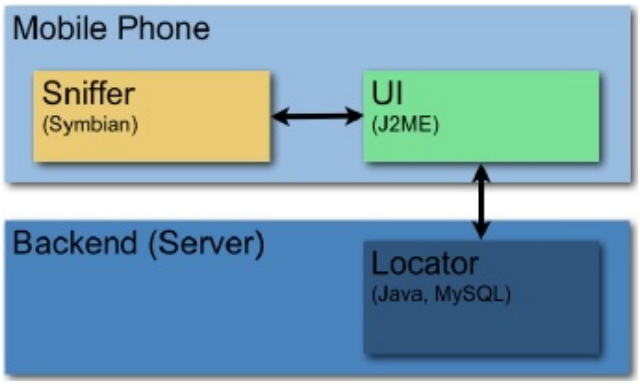


Figura 2.33: Arquitectura de Redpin (obtenido de [11])

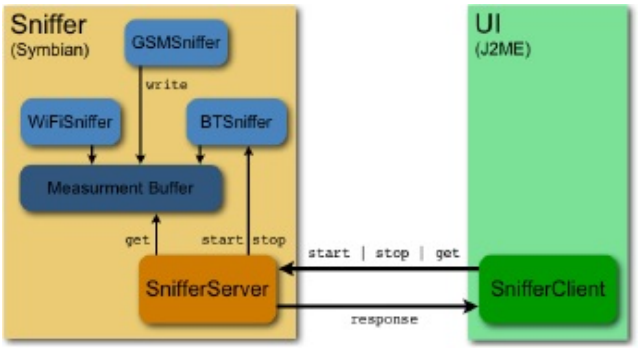


Figura 2.34: Arquitectura de la aplicación móvil de Redpin (obtenido de [11])

- **Locator**: almacena las huellas y contiene los algoritmos para localizar los dispositivos móviles. Puede ser ejecutado en un servidor central o por separado en cada dispositivo móvil. El servidor está implementado en Java SE 6.0 y utiliza SQLite (por omisión) o MySQL (configurable) como base de datos.

- **Servidor**: brinda los servicios de almacenamiento de las huellas, almacenamiento y obtención de los mapas, y servicios para localizar un dispositivo a partir de una huella. Los servicios son accesibles vía TCP/IP, utilizando un protocolo de aplicación definido por el desarrollador, que consta de mensajes en formato JSON.

2.8.4. Trilateration (biblioteca)

Se trata de una biblioteca desarrollada en Java [33] que resuelve la trilateración utilizando un optimizador de mínimos cuadrados no lineal, utilizando el algoritmo de Levenberg-Marquardt [39], implementado en la biblioteca Commons Math de Apache [30].

Capítulo 3

Diseño e Implementación de la Solución

3.1. Introducción

Luego de una primera etapa, donde se analizaron distintos métodos de localización existentes referentes al tema, dado el hardware disponible se optó por utilizar la técnica Fingerprinting (sección 2.4.2), utilizando el framework FIND3 en su modo pasivo (sección 2.8.1), y un modelo de propagación de la señal a partir del RSSI (sección 2.4.1.1) con Trilateración, utilizando la biblioteca Trilateration (sección 2.8.4). Son dos métodos que pueden ser implementados a partir de las señales Wi-Fi y Bluetooth capturadas por los módulos ESP32 LoRa 1-CH Gateway [67]. Otros métodos que se evaluaron fueron los basados en el tiempo de propagación, como por ejemplo TDoA (sección 2.4.1.1), pero esto requiere de una sincronización entre los nodos que no era posible conseguir con el hardware disponible. Tampoco fue posible usar una técnica de Angulación (2.4.1.2) ni obtener el CSI, ya que para esto se requiere hardware con múltiples antenas.

También se analizó si optar por una solución basada en infraestructura (MBL) o en el cliente (DBL) (ver sección 2.2), optando por una solución basada en infraestructura, ya que se evaluaron las siguientes ventajas y desventajas de esta solución:

Ventajas:

- Multiplataforma, no requiere implementación en los dispositivos,

pudiendo localizar cualquier dispositivo sin importar el sistema operativo o su arquitectura.

- Consumo energético, se puede evitar el consumo excesivo en los dispositivos.

Desventajas:

- Privacidad, se podría identificar al usuario a partir de la dirección MAC del dispositivo.
- Dada la anonimización que realizan los sistemas operativos iOS y Android, podría no ser posible reconocer dispositivos que sí son conocidos.

En el caso de querer localizar dispositivos conocidos (e.g., robots u otros objetos con hardware IoT embebido), las desventajas antes mencionadas no se cumplirían. La segunda desventaja tampoco se cumpliría si los dispositivos no implementan ningún tipo de anonimización.

Además, se analizaron diferentes Gateways LoRa, optando por el uso del Gateway Dragino, por haber presentado un mejor desempeño, como se detalla más adelante en este capítulo.

Para simplificar, a partir de esta sección los dispositivos a ser localizados serán nombrados como "clientes" y los nodos ancla como "nodos". En el caso de las direcciones MAC en Wi-Fi y las BD_ADDR en Bluetooth, ambas serán nombradas como "direcciones MAC", diferenciando en caso de ser necesario.

3.2. Requerimientos

1. Los clientes serán cualquier dispositivo que emita señales Wi-Fi o Bluetooth.

2. El hardware para los nodos son módulos ESP32 LoRa 1-CH Gateway.

3. Se debe realizar la localización de los clientes en interiores.

4. La comunicación de los nodos con el sistema central debe ser realizada utulizando LoRaWAN.

3.3. Pruebas preliminares

En una etapa preliminar al diseño e implementación se analizó la intensidad de la señal de los módulos ESP32 LoRa 1-CH Gateway. Al realizar mediciones del RSSI a 1 m de distancia se detectó que la intensidad de la señal varía según la orientación del cliente respecto al nodo. Estando a 1 m y girando el cliente 90° se detectó una diferencia de 10 dBm en los valores de RSSI. Esto sucede tanto para la interfaz Wi-Fi como para Bluetooth, dado que utilizan la misma antena. Para verificar si era un problema que se pudiera evitar, se utilizó como cliente un notebook con una interfaz Wi-Fi con una antena omnidireccional y se notó que la variación al rotar esta antena es bastante menor. De todas formas, el problema persiste, ya que los nodos también son módulos ESP32 LoRa 1-CH Gateway, y al cambiar la ubicación del cliente, el ángulo entre el cliente y el nodo cambia, generando valores de RSSI diferentes para la misma distancia. Se continuó usando el modulo ESP32 LoRa 1-CH Gateway como cliente ya que no era factible el uso de un notebook, en cuanto a la movilidad y el tiempo de duración de la batería dadas las pruebas a realizar.

Otro problema que se vio con el RSSI fue que, a medida que aumenta la distancia, y el valor del RSSI disminuye, se comienza a perder la relación entre la intensidad de la señal y la distancia. Esto es, ante valores de RSSI muy chicos, pequeñas variaciones en su valor hacen que las distancias calculadas aumenten considerablemente.

3.4. Arquitectura del Sistema

La Facultad puso a disposición un PC en donde se instalaron máquinas virtuales (VM) con Ubuntu Server 18.04.4 LTS [68]. Algunos de los componentes de la arquitectura se instalaron en una VM para su posterior evaluación. Se utilizó el software VirtualBox 6.1.16 [71] para crear y alojar las VM necesarias, configurando una red NAT [72] entre ellas. En la figura 3.1 se puede visualizar la arquitectura global y las interacciones entre sus componentes. En las siguientes secciones se detalla cada uno de los componentes.

Al configurar una red NAT, las VM se pueden conectar entre ellas, pero desde equipos externos de la red no se puede acceder a las VM. Para que el Gateway LoRa tenga acceso al Server LoRa, en la máquina host se especifican puertos que luego son redirigidos a la VM en cuestión.

En el apéndice E se detallan los pasos para la instalación de los principales componentes de la arquitectura.

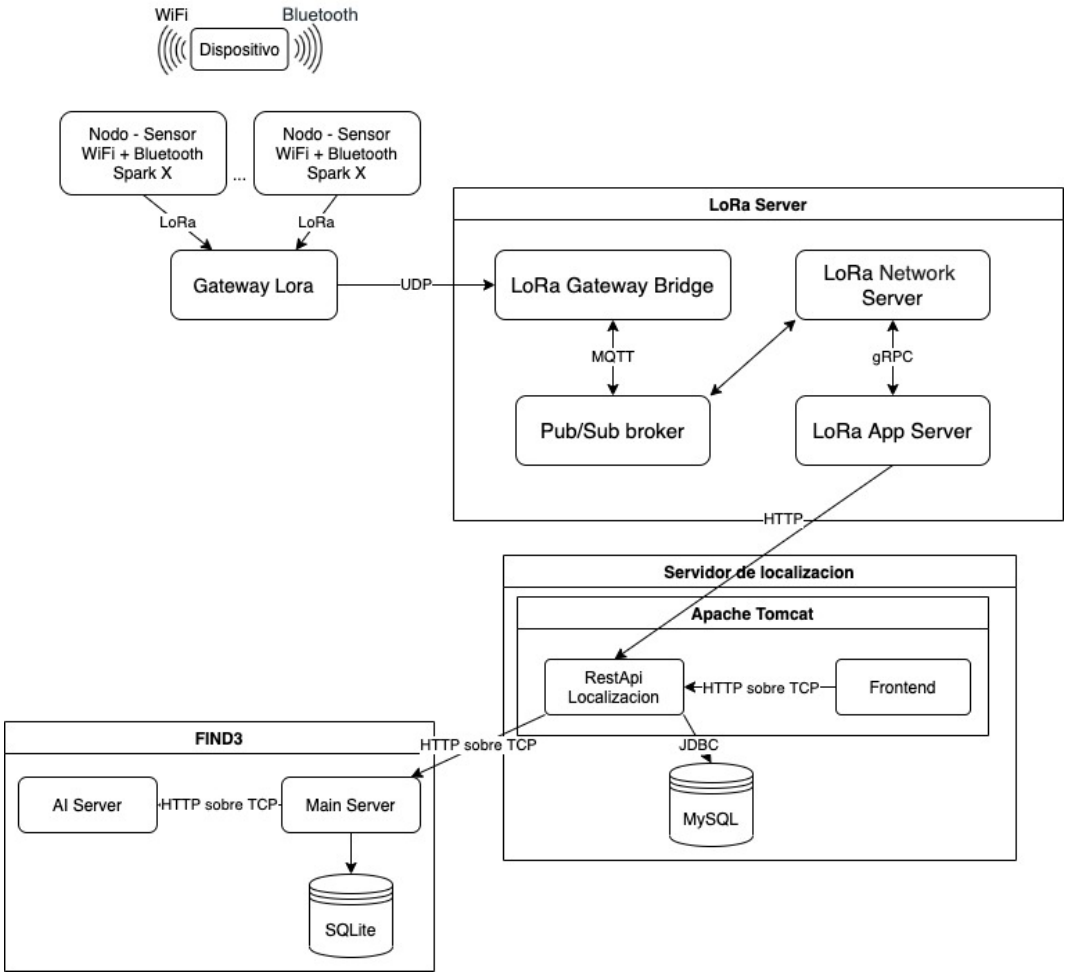


Figura 3.1: Diagrama de los componentes de arquitectura

3.4.1. Dispositivos

Los dispositivos son los clientes que serán localizados. Pueden ser cualquier dispositivo que cuente con una interfaz Wi-Fi y/o Bluetooth (BLE o BR/EDR). No requieren de una aplicación específica para esto, ya que son localizados captando sus señales Wi-Fi o Bluetooth, sin la necesidad de que haya sido establecida una conexión. Para este trabajo, pensando en el caso de uso de robots con un módulo ESP32 embebido con el sistema operativo LuaRTOS (ver 2.6.3), se implementó un script Lua que emite datos Wi-Fi y BLE, con el fin de que sean detectados por los nodos. Los datos Wi-Fi emitidos son tramas

Probe Request generadas con una función de la API Lua de LuaRTOS, agregada en este trabajo para poder realizar un "scan Wi-Fi" en un único canal, con el fin de ahorrar batería, ya que la función original lo hace para los 14 canales Wi-Fi. En el caso de BLE, se emiten paquetes de Advertisement con una función que ya se encontraba en la API.

3.4.2. Nodos - Sensores

Los nodos, o sensores, son módulos ESP32 LoRa 1-CH Gateway [67], que cuentan con interfaces Wi-Fi, Bluetooth y LoRa. Son los que se encargan de recolectar la información del entorno, utilizando las interfaces Wi-Fi y Bluetooth, obteniendo así las direcciones MAC y el RSSI de los mensajes que envían los clientes que se encuentran al alcance. Si bien las direcciones MAC son únicas y pueden ser usadas para la identificación de individuos, lo que podría traer un problema de privacidad, en la solución planteada no se desarrolla ningún método para anonimizar los datos recolectados, como podría ser la generación de algún tipo de hash como lo hace el sistema Cisco Meraki [18]. Estos módulos serán distribuidos de forma tal que se abarque toda la zona donde se quiere realizar la localización, intentando colocarlos lo más equiespaciados posible, dependiendo del lugar donde se vaya a probar el sistema. Para Wi-Fi soportan únicamente la banda de frecuencias 2,4 GHz, no siendo posible detectar clientes que transmitan en la banda de frecuencias de 5 GHz.

En los nodos se instaló el sistema operativo de tiempo real LuaRTOS para ESP32 (sección 2.6.3), que debió ser modificado para agregar funcionalidades a la API Lua. Se agregaron funciones al módulo *bt* (Bluetooth) para utilizar la versión BR/EDR, ya que en su versión original sólo permite utilizar BLE. Para esto, se agregó la posibilidad de configurar el módulo *bt* en modo dual (BLE y BR/EDR a la vez), o elegir uno de los modos (BLE o BR/EDR). Al elegir el modo deseado, se libera la memoria del modo no utilizado, y se agregó una función para liberar toda la memoria del módulo *bt* en caso de que no se vaya a utilizar. Esta variante de LuaRTOS también cuenta con las modificaciones realizadas en el proyecto de grado mencionado en 2.7.6.1, para utilizar la interfaz Wi-Fi en modo monitor. A las modificaciones realizadas en ese proyecto, se le agregó la posibilidad de obtener las tramas Wi-Fi de un único canal, ya que en ese trabajo, al ejecutar la función, se recorren los 14 canales Wi-Fi. Poder obtener las tramas de un único canal permite configurar

un cliente para que emita tramas sólo en ese canal, disminuyendo el tiempo de escaneo. En el apéndice A se describen detalladamente las modificaciones realizadas a LuaRTOS para la implementación de los nodos.

Se crearon los scripts Lua necesarios para la programación de los nodos. De forma concurrente, para cada interfaz, los nodos recaban la información del entorno durante 3 segundos y envían, bajo el protocolo LoRaWAN, mensajes que contienen la interfaz (Wi-Fi o Bluetooth), la dirección MAC del nodo y del cliente, y el RSSI. Luego de enviar esta información, el hilo correspondiente a la interfaz entra en modo de suspensión durante 2 segundos. En los scripts Lua es posible configurar los tiempos de escaneo y de suspensión, además de las interfaces a usar (Wi-Fi y/o Bluetooth). A su vez, para Bluetooth, se puede configurar si se quiere usar BLE y BR/EDR o sólo una de ellas, y se implementaron dos scripts, uno para que BLE y BR/EDR se usen de forma concurrente y otro para que se usen de forma secuencial.

En cuanto al envío LoRa, se modificó LuaRTOS para utilizar el plan de frecuencias definido para Uruguay. Como los nodos actúan en el rol de sensor, y no esperan recibir información, están programados como dispositivos finales de clase A, según el protocolo LoRaWAN (2.6.1).

En los módulos disponibles, el chip utilizado para Wi-Fi y Bluetooth es el mismo, por lo que el framework (ESP-IDF) debe hacer una alternancia cuando se está usando ambas tecnologías. Hay que tener esto en cuenta a la hora de programar los scripts, sobre todo en los tiempos de scan que se utilicen para BLE.

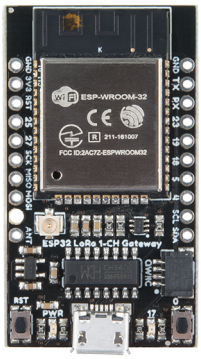


Figura 3.2: ESP32 LoRa 1-CH Gateway (obtenida de [67])

3.4.3. Gateway LoRa

El gateway LoRa se encarga de reenviar los datos recibidos de los nodos a un servidor LoRaWAN. Se evaluaron los siguientes tres gateways:

- ESP32 LoRa 1-CH Gateway

  Se instaló y se probó un firmware para que el módulo ESP32 actúe como gateway. Dado que estas placas son de un único canal, se apreció mucha pérdida de información, debido a las colisiones a causa de que todos los nodos emitían en el mismo canal. A medida que se agregaban nodos las pérdidas aumentaban, impidiendo una recolección de datos para permitir una correcta localización, al grado de que, probando con cuatro nodos, al cabo de 60 segundos no se tenía información de todos los nodos.
  Para este caso fue necesario modificar el código de LuaRTOS para que los nodos emitieran los mensajes en una única frecuencia.

- The Things Gateway [57]

  Se configuró y probó el gateway The Things Gateway, de 8 canales, creado por The Things Network. Se pudo ver que tiene menor pérdida de información que el gateway ESP32. Tiene como limitante que sólo permite utilizar el servidor de The Things Network a través del puerto UDP 1700. Esto es una desventaja dado que, a priori, no es posible acceder a ese puerto fuera de la red de Facultad. Otra limitante es una restricción impuesta por The Things Network llamada *Fair Access Policy*, que permite un airtime máximo de 30 segundos para uplinks y hasta diez mensajes de downlink [56]. Esta restricción es por nodo cada 24 horas. Al momento de realizadas estas pruebas esta restricción no se pudo constatar, sin embargo, en el sitio de The Things Network indican que en un futuro sí realizarán este control.
  Para este gateway se programaron los nodos para emitir los mensajes en los 8 canales del gateway.

- Dragino FLPS8 Indoor LoRaWAN Gateway [23]

  Es un gateway de 8 canales. Presenta menos pérdida de datos que los anteriores, y permite el uso de cualquier servidor LoRa que cumpla con el protocolo UDP Forwarding.
  Para este gateway también se programaron los nodos para emitir mensajes en los 8 canales del gateway.

Las pérdidas de mensajes se deben principalmente a que se programó el envío LoRa indicando que el servidor LoRa no envíe un mensaje de confirmación. Esto hace que los paquetes perdidos por colisiones en el canal no sean reenviados. Las colisiones se pueden dar en el canal LoRa, o en el reenvío del gateway al Lora Bridge, ya que utiliza UDP para esto. Las pruebas para determinar la pérdida de mensajes se realizaron usando cuatro nodos que envían mensajes al mismo tiempo, y se monitorizó uno de ellos comparando la cantidad de mensajes emitidos contra la cantidad recibida en el servidor LoRa. En cada nodo se envía un mensaje cada cuatro segundos, que es el tiempo aproximado desde que comienza la transmisión de un mensaje LoRa hasta que termina la segunda ventana Rx. Las pruebas realizadas fueron las siguientes:

1) 1 gateway ESP32 de un canal.
   4 nodos configurados en el mismo canal:
   44 % de los mensajes recibidos.

2) 2 gateways ESP32 de un canal, configurados en distintos canales.
   2 nodos por gateway:
   55,88 % de los mensajes recibidos.

3) 4 gateways ESP32 de un canal, cada uno en un canal diferente.
   1 nodo por gateway:
   74,51 % de los mensajes recibidos.

4) 1 gateway de The Things Network de 8 canales.
   4 nodos programados para enviar en los 8 canales de este gateway:
   87 % de los mensajes recibidos.

5) 1 Gateway Dragino de ocho canales. Se configuró el gateway en el plan de frecuencias de AU915-928.
   4 nodos programados para enviar en los 8 canales de este gateway:
   93,63 % de los mensajes recibidos.

Se optó por el uso del Gateway Dragino, ya que es el gateway que tiene menos pérdida de datos, y no tiene restricciones en cuanto al servidor LoRa a utilizar. Se encarga de recibir los mensajes enviados por los nodos mediante LoRaWAN [55] y los reenvía al servidor LoRa. El gateway se comunica mediante Wi-Fi a una red en donde se encuentra el servidor LoRa, enviando los mensajes recibidos mediante el protocolo UDP al puerto 1700.

Figura 3.3: Dragino FLPS8 Indoor LoRaWAN Gateway (obtenida de [23])

3.4.4. Servidor LoRa

Para el servidor LoRa se utilizó la solución *ChirpStack, open-source LoRaWAN® Network Server stack* [17], que provee tres componentes que permiten la comunicación desde el gateway a un sistema final. Se decidió usar esta solución por contar con la experiencia previa del proyecto de grado mencionado en 2.7.6.1, además de tratarse de una solución de código abierto en constante mantenimiento. El stack de ChirpStack se configuró en una VM dedicada, en la cual se instalaron los subcomponentes LoRa Bridge, LoRa Network Server y LoRa Application Server en el mismo servidor. A continuación se describe cada componente y su interacción.

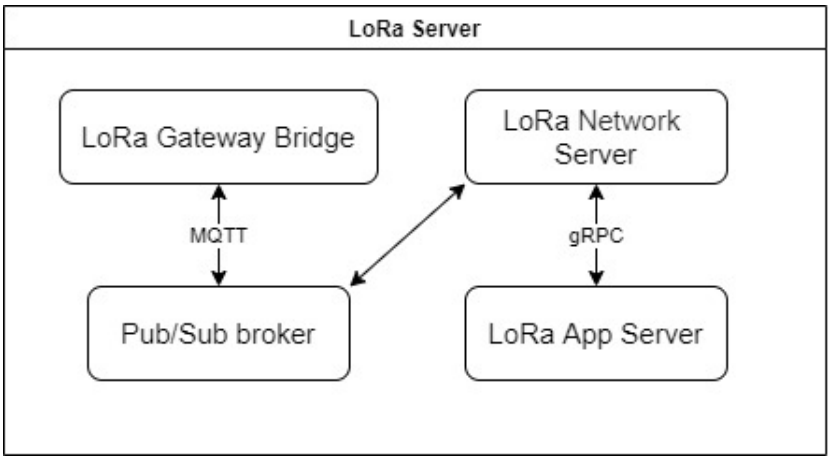


Figura 3.4: Arquitectura de LoRa Server

- LoRa Bridge

  El componente LoRa Bridge se encarga de transformar los paquetes recibidos en el puerto UDP 1700, que son enviados por los gateways utilizando el protocolo Semtech UDP Protocol [69], a un formato común [54] (JSON o Protobuf), que es interpretado por los demás componentes de ChirpStack. LoRa Bridge coloca estos mensajes transformados en un agente MQTT.

- LoRa Network Server

  Este componente se encarga de obtener los datos del agente MQTT, eliminar las tramas duplicadas de diferentes gateways y enviar los datos al servidor de aplicación.

- LoRa Application Server

  Este componente proporciona una interfaz web y una API para la gestión del sistema. Se configuró una integración HTTP para reenviar al servidor de localización los uplinks recibidos.

3.4.5. Servidor de localización

Este componente consta de un servidor Apache Tomcat 7 y un motor de base de datos MySQL, que se instalaron en una VM dedicada. En el Apache Tomcat ejecutan dos aplicaciones, que se detallan a continuación.

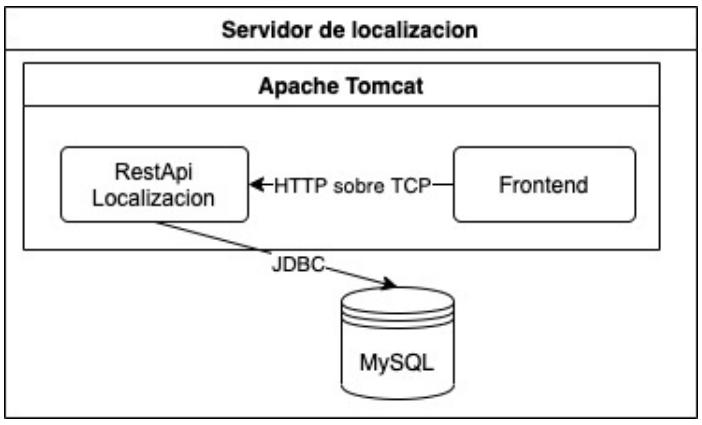


Figura 3.5: Servidor de localización

3.4.5.1. REST API de localización

Este componente es una aplicación web desarrollada con Java 1.8 que ejecuta en Apache Tomcat 7. Utiliza una base de datos MySQL, en la que se almacena:

- Configuración de los algoritmos.
- Datos del plano.
- Tipo de algoritmo a usar, Fingerprinting con FIND3 o Trilateración.
- Zonas definidas para el método de Fingerprinting. Para cada zona se determina un punto en el plano y se configuran las coordinadas cartesianas de éste.
- Información recabada por los nodos (información de los clientes junto con el RSSI).
- Ubicaciones reales de los clientes. Son las ubicaciones donde realmente se encontraba cada cliente en el momento de la recolección de datos, y sirven para comparar la ubicación calculada con la real (esta información debe ser registrada de forma manual).
- Ubicaciones de los clientes calculadas por el algoritmo configurado.
- Información de los nodos. Esto es, la dirección MAC, BD_ADDR, coordenadas de su ubicación, y los parámetros del modelo de propagación para cada interfaz (RSSI a un metro y n para Wi-Fi y para Bluetooth).

La API expone los siguientes servicios REST:

- Recibir mensajes de los nodos

  Este servicio es invocado por LoRa Application Server al recibir un mensaje de uplink (configurado en el servidor LoRa). A este servicio le llegan dos tipos de mensaje, la información recolectada por los nodos, y los inicios de sistema de los nodos. Los mensajes con la información de las medidas tomadas por los nodos contienen la dirección MAC del nodo, la dirección MAC del cliente y el RSSI de la señal que contenía la información emitida por el cliente. Los mensajes de inicio (o reinicio) sirven para

saber que el nodo está activo, ya que, como se menciona en 3.5 (Dificultades Encontradas), por problemas de memoria los nodos se reinician cada cierto tiempo.

- Obtener información del plano

  Retorna la información del plano, incluyendo la URL a la imagen del plano, las medidas del lugar y la lista de nodos activos. Este servicio es usado por el componente Frontend.

- Obtener información de los clientes localizados

  Este servicio retorna la ubicación de los clientes localizados en la última ventana de tiempo desde la fecha que se consulta. Permite filtrar por dirección MAC o interfaz del cliente. Este servicio es usado por el Frontend.

- Activar/desactivar el modo aprendizaje de FIND3

  Este servicio activa o desactiva el modo de aprendizaje en FIND3. Realiza un POST a FIND3 comunicando la acción.

- Aprendizaje offline

  El propósito de este servicio es poder procesar la información almacenada en la base de datos en un momento posterior a su obtención, con el fin de entrenar los algoritmos de Fingerprinting (FIND3) y de evaluar los métodos de localización implementados tantas veces como sea necesario, sin la necesidad de hacer una recolección de datos cada vez. Es muy útil para probar y evaluar el desempeño del sistema. Es necesario haber recolectado información suficiente por parte de los nodos, y haber almacenado la información de las ubicaciones reales de los clientes (esto lo realiza la persona que realiza la recolección de datos) con los que se va a realizar el entrenamiento o la evaluación de los algoritmos de localización.

  Recibe un rango de fechas, que indica el período de las ubicaciones reales precargadas que se debe considerar, y un parámetro opcional que indica si se quiere evaluar el sistema (en caso contrario, se quiere entrenar los algoritmos de Fingerprinting). En el caso de entrenamiento, configura a FIND3 en modo aprendizaje para los clientes de las ubicaciones reales y le

envía la información recolectada por los nodos. En el modo de evaluación, calcula la ubicación a partir de los datos recolectados por los nodos (con el método de localización configurado) y la compara con la ubicación real, con el fin de evaluar el sistema.

La API se encarga de aplicar cualquiera de los dos métodos preestablecidos (Fingerprinting o Trilateración), de forma transparente para los demás componentes de la arquitectura. De esta forma es posible cambiar el método de localización o agregar otro sin necesidad de cambiar los demás componentes de la arquitectura. Mediante los servicios REST se puede obtener información del plano y las ubicaciones de los clientes en un momento dado, permitiendo la interoperabilidad con cualquier sistema que realice consultas vía servicios REST.

El cálculo de ubicaciones se realiza cada 30 segundos, con los datos recolectados por los nodos en los últimos 30 segundos. Este tiempo es configurable, y se eligió este valor porque, a partir de pruebas realizadas, se vio que era el tiempo en que se tenía información suficiente (de todos o casi todos los nodos) para realizar la localización de los clientes detectados con alguno de los métodos implementados. Esto se debe a dos factores, uno es la demora en el envío de los mensajes LoRa desde los nodos (esto se explica en 3.5), y otro es la pérdida de mensajes. Puede suceder que en la ventana de tiempo configurada (en este caso, 30 segundos) un nodo recolecte más de un valor de RSSI para un mismo cliente. En ese caso, se toma el promedio de los valores de RSSI obtenidos.

Algoritmo de trilateración

Uno de los métodos utilizados es el de trilateración a partir del RSSI (sección 2.4.1.1). Para realizar este cálculo se utilizó la biblioteca Trilateration, mencionada en 2.8.4, que dado el conjunto de coordenadas cartesianas de los nodos y las distancias de éstos al cliente (calculadas a partir del RSSI utilizando el modelo de propagación), calcula la ubicación del cliente (coordenadas cartesianas) utilizando Mínimos Cuadrados.

Algoritmos de Fingerprinting

Además se utilizó el método de Fingerprinting con varios algoritmos de Machine Learning, utilizando el framework FIND3, que se detalló en la sección 2.8.1.

Al realizar la localización con FIND3 se utilizan algoritmos de clasificación, por lo que éste determina una zona dentro del conjunto de zonas previamente definidas. En este caso se establece como ubicación del cliente el centro de la zona determinada por FIND3 (coordenadas configuradas para esa zona). Cuando se utiliza este método, estas coordenadas son las que retorna el servicio de localización como ubicación del cliente.

En el apéndice B se detalla la configuración de esta API (parámetros necesarios, configurables en la base de datos) y el formato de entrada y salida de los servicios REST brindados por ésta.

3.4.5.2. Frontend web

Este componente es una web estática realizada con HTML y JavaScript. Implementa una interfaz de usuario que permite visualizar el plano (ver figura 3.6), los nodos y los clientes localizados. Muestra detalle de la dirección MAC y el tipo de interfaz. La información es actualizada cada 10 segundos, comunicándose mediante la API REST del servidor de localización.

Además, brinda una pantalla que permite activar y desactivar el modo aprendizaje para Fingerprinting para las zonas que fueron definidas para este proyecto.

3.4.6. FIND3

Este componente tiene toda la arquitectura de FIND3 (descrito en la sección 2.8.1), en una VM dedicada.

Para este proyecto se utiliza el modo pasivo del framework, ya que son los nodos quienes estarán reportando datos de los clientes que se tiene alcance, sin una participación activa de estos últimos. La razón principal por la que se utilizó este framework es que permite el modo pasivo, y que se contaba con la experiencia de un proyecto de grado anterior.

Cada nodo envía, de forma independiente, la información de las señales que captura de los clientes, por lo que FIND3 recibe esta información por separado. Para esto, en el modo pasivo de FIND3 se define una ventana de tiempo durante la cual se almacenarán los datos que le llegan antes de generar una huella (fingerprint). Durante este tiempo, FIND3 almacena para cada cliente i el conjunto de valores $F_i = \{(N_1, RSSI_1)_i, (N_2, RSSI_2)_i, \ldots, (N_n, RSSI_n)_i\}$

Figura 3.6: Plano: Cliente con interfaces Wi-Fi y Bluetooth y Ubicación Real

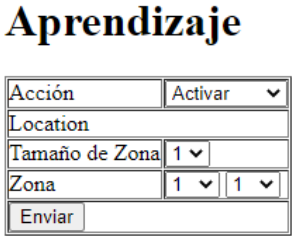


Figura 3.7: Frontend para aprendizaje

correspondiente a los n nodos que capturaron señales de i. Luego de transcurrido ese tiempo, F_i se considera una huella. Esta ventana de tiempo es utilizada tanto durante la etapa offline (entrenamiento) como la online (clasificación),

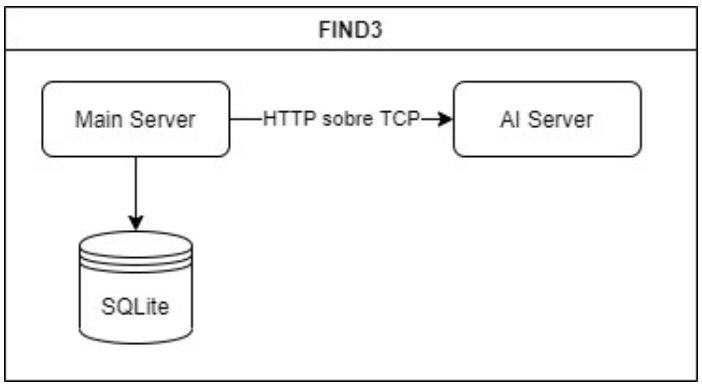


Figura 3.8: Servidor de FIND3

es configurable y se utilizó el mismo valor que se usó para localización (30 segundos), mencionado en 3.4.5.1.

Al framework FIND3 se le hicieron algunas de las modificaciones realizadas en el proyecto de grado [12], mencionado en 2.7.6.2. Estas modificaciones fueron:

- Datos en el entrenamiento de los algoritmos: La versión original del framework limita la cantidad de fingerprints a un máximo de 1000[1] a la hora de entrenar los algoritmos. Estos datos son seleccionados de forma aleatoria en el conjunto total de datos. Esto hace que mucha de la información recolectada en la etapa de entrenamiento no se use, reduciendo la precisión del sistema.

- Se eliminó el algoritmo SVM no lineal, de $O(N^3)$, que aumentaba el tiempo de entrenamiento del sistema.

- Se eliminaron los algoritmos QDA, Gaussian Naive Bayes y Extended Naive Bayes. Los dos primeros no aportaban valor al resultado de clasificación, y el tercero, implementado por el desarrollador del framework, presentaba errores en su implementación.

En el proyecto citado afirman haber verificado que la eliminación de los algoritmos dio resultados mejores o iguales que con esos algoritmos. En el presente proyecto se realizó una prueba mínima y se pudo verificar lo que allí afirman.

[1]Según el desarrollador del framework, este límite es para evitar un uso excesivo de la memoria de su servidor, dado que en éste ejecuta una instancia de FIND3 de acceso público. Esto está mencionado en la sección de issues del proyecto en el sitio de GitHub (https://github.com/schollz/find3/issues/62#issuecomment-382916510).

Además, para el presente proyecto se modificó FIND3 para permitir tomar el 100 % de los fingerprints para entrenar los algoritmos, ya que en su versión original toma un 70 % para el entrenamiento y un 30 % para calcular la exactitud. Se decidió hacer esto para poder elegir los datos de entrenamiento y luego, en otra instancia, los datos para la evaluación del sistema. Esto es configurable, por lo que también se permite utilizar la separación original de los datos.

También se hicieron modificaciones por una lentitud que se presentaba al momento de recolectar datos para entrenamiento. Esto sucedía al configurar FIND3 en modo de aprendizaje. A medida que aumentaba la cantidad de datos se presentaba una sobrecarga en el servidor, lo que enlentecía bastante el proceso y generaba errores en la generación de los fingerprints. Los cambios realizados al framework se detallan en el apéndice D.

3.5. Dificultades Encontradas

Luego del análisis, se presentaron algunas dificultades en la implementación de los nodos. Dado que son los encargados de recolectar la información necesaria para la localización, se cree que solucionando algunos de estos problemas, la localización podría mejorar.

Algo a tener en cuenta en el caso de Bluetooth es que ESP-IDF (2.6.2) sigue el *Link Layer Device Filtering* de la especificación de Bluetooth (mencionada en 2.5.2.2), que establece que los paquetes de Advertisement dirigidos a otro dispositivo son descartados. Con esto, a diferencia de Wi-Fi que tiene el modo monitor, no es posible capturar todos los paquetes BLE, sino los no dirigidos o los dirigidos al dispositivo. Para el caso de BR/EDR no se encontró una política similar en la especificación[1], sin embargo, tampoco se encontró en la API de ESP-IDF una función que permita utilizar la interfaz Bluetooth en modo monitor, por lo que, para BR/EDR, los paquetes detectados son los que surgen del descubrimiento BR/EDR (mencionado en 2.5.2.1).

Para el envío mediante LoRa, LuaRTOS usa la biblioteca LMiC (LoRa MAC in C), originalmente desarrollada por IBM. Como ya se dijo en este capítulo, los nodos están programados como dispositivos de clase A, según el protocolo LoRaWAN, ya que no se espera recibir mensajes de la red. Como

[1] En BR/EDR no existe el concepto de Advertising, ni existen mensajes *Broadcast* sin una conexión.

se mencionó en 2.6.1, el envío de cada mensaje consta de tres ventanas, Tx, Rx1 y Rx2, siendo Tx de transmisión y Rx1 y Rx2 de recepción. Se intentó modificar la biblioteca LMiC para eliminar las ventanas Rx1 y Rx2, ya que agregan un tiempo importante en cada envío, pero esto no se pudo realizar por falta de tiempo, luego de algunos intentos. Vale la pena mencionar que la transmisión de un mensaje LoRaWAN (ventana Tx), desde las placas disponibles, demora unos pocos milisegundos, pero al considerar además el tiempo de las ventanas Rx1 y Rx2, cada envío LoRa demora, en promedio, 4 segundos. Dada esta demora en cada envío, se intentó enviar mensajes más grandes, con la información de más de una captura (información de dirección MAC de cliente y RSSI) en cada uno. Esto no fue posible dado que, a raíz de pruebas realizadas, se vio que el tamaño máximo de payload que se puede enviar es 51 bytes, si importar el *Data Rate* utilizado (se probó en todos los *Data Rates* que permite la API). La información correspondiente a una captura implica un payload de 32 bytes, por lo que no es posible enviar la información de más de una captura en un único mensaje (un ejemplo de payload es la cadena de caracteres "wf|112233445566|223344556677|-99"). Por lo tanto, finalmente se decidió enviar las capturas de a una. Esto genera una latencia considerable en la localización.

A raíz de diversas pruebas realizadas durante el desarrollo, se constató que luego de ejecutar las funciones del framework ESP-IDF que inician el *scan* en BLE y BR/EDR, y la que habilita el modo promiscuo en Wi-Fi, la memoria disponible disminuye[1], lo que provoca que luego de varias ejecuciones de estas funciones, la placa se reinicie y que, en algunas ocasiones, queden "bloqueadas", siendo necesario un reinicio por hardware. La disminución de la memoria disponible se comprobó utilizando la función *xPortGetFreeHeapSize()* de FreeRTOS. Por este motivo, en los scripts Lua, antes de iniciar el software del nodo, se configuró un reinicio de la placa a los 5 minutos de haber iniciado. En ese momento, también se envía un mensaje al servidor para registrar el inicio, lo que sirve para verificar que un nodo está funcionando en el caso de no estar recibiendo datos de clientes.

[1]Las pruebas se realizaron ejecutando las distintas funciones *scan* con el menor código posible, sin definir variables estáticas ni reservando memoria.

Capítulo 4

Evaluación de la Solución

4.1. Resultados Esperados

En esta sección se espera determinar si es factible la localización en interiores con el hardware proporcionado y los métodos de localización utilizados. De los métodos de localización interesa obtener el error y la precisión de cada uno, y posibles combinaciones o cantidades de nodos para que los métodos tengan un mejor resultado.

4.2. Escenario para las pruebas

Las pruebas se realizaron en el hall de planta baja del Instituto de Computación de la Facultad de Ingeniería (InCo). La figura 4.1 muestra el plano del lugar, sus medidas, los nodos con sus coordenadas cartesianas (unidades en metros) y el punto de origen (esquina inferior izquierda). Se tienen las siguientes consideraciones:

1 El plano se divide en cuadrantes de 60 cm de lado, aprovechando que el piso está cubierto de baldosas de ese tamaño (los cuadrados de la figura 4.1 no son los que se mencionan, sino que son de 2,4 m de lado).

2 Se utilizan 9 módulos ESP32 como nodos, numerados del 1 al 9, distribuidos como se muestra en la figura 4.1. Los nodos se encuentran escuchando las tramas Wi-Fi únicamente del canal 7, y los paquetes BLE en los tres canales de advertisement (37, 38 y 39).

3 Se utiliza como cliente un módulo ESP32 que emite datos Wi-Fi cada 500 ms y datos Bluetooth cada 160 ms. Estos tiempos se eligieron de forma empírica al realizar el despliegue del sistema dado que, luego de pruebas realizadas, se vio que el cliente siempre era detectado por los nodos con ambas interfaces. Los datos Wi-Fi son tramas *Probe Request* y se envían únicamente al canal 7. Los datos Bluetooth son advertisements BLE que contienen el nombre del dispositivo (generado como la concatenación de la BD_ADDR y el texto *"-LE"*), y se envían por los tres canales de advertisement.

4 Tanto los nodos como el cliente están aproximadamente a 30 cm del piso.

5 El cliente siempre está en el centro de algún cuadrante de la consideración 1, y posicionado verticalmente, con la antena hacia arriba, como muestra la figura 3.2.

6 Para cada punto de la consideración 5, durante la toma de datos, el cliente siempre iniciará orientado hacia la misma dirección, y será rotado de a 90° (0°, 90°, 180° y 270°) en intervalos de tiempo iguales. El eje de rotación considerado es el vertical.

7 Al momento de realizar la recolección de datos, se intentará que el entorno quede invariante.

8 Se recolectará la intensidad de la señal para las interfaces Wi-Fi y Bluetooth.

9 Como se comentó en la sección 3.3, el RSSI de la señal recibida en el nodo cambia considerablemente según la orientación del cliente, incluso sin cambiar la ubicación de los dispositivos.

En el apéndice G se muestran imágenes con el despliegue de los nodos y del cliente.

Para las pruebas se decidió usar un único canal para Wi-Fi (consideración 2 y consideración 3) para acelerar el proceso de recolección de datos Wi-Fi, ya que hacerlo en los 14 canales Wi-Fi insumía demasiado tiempo.

Figura 4.1: Plano del hall de la planta baja del InCo

4.3. Realización de las Pruebas

A continuación se detallan los pasos seguidos para realizar la evaluación del sistema y de los métodos implementados.

Materiales utilizados: Además de los componentes comentados en secciones anteriores, también se utilizaron transformadores con salida de 5 V con una potencia de 1 A, cuatro Redmi 18 W Power Bank 20000 mAh (uno para el gateway y el resto para los nodos), y un PNY T2200 Powerpack (para el cliente).

Recolección de los datos: La toma de datos insumió un total de 46 hs aproximadamente, y se realizó en dos instancias. En cada instancia se cubrió por completo la zona de localización, que es el hall de la planta baja del InCo. La primera instancia insumió unas 20 hs. y contiene el 38 % de los datos,

mientras que la segunda insumió unas 26 hs. y contiene el 62 % de los datos. La forma en que se recolectaron los datos fue la misma en ambas instancias, con la diferencia de que en la segunda se mantuvo al cliente durante más tiempo en cada cuadrante.

Utilización de los Datos: Los datos se usaron de forma distinta según el método:

- Trilateración: El 100 % de los datos recolectados fue utilizado para evaluar el método.
- Fingerprinting: Se decidió tomar los datos de la segunda instancia de recolección (62 %) para el entrenamiento de los algoritmos y los de la primera (38 %) para la evaluación de los mismos. En un principio se consideró seleccionar los datos de forma aleatoria, pero dada la variación de la intensidad de la señal según la orientación del cliente, se decidió hacer la separación de esta manera para tener un mayor control sobre esto.

Orientación del cliente: Tomando en cuenta la consideración 9 en 4.2, se probó el sistema sin considerar la orientación del cliente (rotándolo hacia todas las direcciones en cada ubicación) y con éste orientado siempre hacia una misma dirección. Con esto se quiso verificar si, dado el problema presente en el RSSI según el ángulo, hay alguna mejoría cuando el cliente no es rotado en las distintas ubicaciones. Dada la consideración 6 en 4.2, es posible tomar los datos para una misma orientación del cliente, para ambas instancias de recolección.

Combinaciones de nodos: Se probaron las siguientes combinaciones de nodos:

- todos los nodos
- sólo los nodos de los vértices
- sólo los nodos de las aristas

Cuando se mencione vértices y aristas, se estará haciendo referencia a los vértices y las aristas del rectángulo formado por los nodos, sin considerar el nodo del centro. Los nodos 1, 3, 7 y 9 corresponden a los vértices, y los nodos 2, 4, 6 y 8 a las aristas (ver figura 4.1). Con esto se quiso verificar qué

tanto influye el número y la distribución de los nodos, intentando con cada combinación abarcar lo máximo posible de la zona, manteniendo el despliegue de los nodos de la figura 4.1.

Métodos de localización: Se probaron los dos métodos de localización implementados, Trilateración y Fingerprinting. Primero se midió el desempeño de cada uno por separado, y luego se compararon los resultados de cada uno.

Cálculo del error: Para ambos métodos, el error e se calculó como la distancia de la ubicación real del cliente, X_R, a la ubicación calculada por el sistema, X_S:

$$e = d\left(X_R, X_S\right) \tag{4.1}$$

Medida de desempeño: Para cada prueba realizada se generó la CDF para el error, en la que $f(x) = P(e \leq x)$. Al igual que en la mayoría de los trabajos mencionados en la sección 2.7, en cada prueba se tomó como precisión el error x tal que $P(e \leq x) = 50\,\%$, es decir, su mediana. Además, como referencia, se indican los valores de $P(e \leq 1,0)$, $P(e \leq 1,5)$, $P(e \leq 2,0)$ y el error promedio. Para el método Fingerprinting se indica además la *exactitud* y se muestra la matriz de confusión, conceptos mencionados en 2.4.2.

Interfaces utilizadas: El desempeño se midió tomando los datos de ambas interfaces (Wi-Fi y Bluetooth), sin discriminarlos, y también se hizo una comparación de los resultados para cada interfaz, para determinar si la localización en la solución propuesta funciona mejor con una u otra tecnología.

Zonas para Fingerprinting: Para el método de clasificación se definieron tres divisiones distintas del plano, generando una grilla en cada caso. Cada división se compone por zonas cuadradas iguales, compuestas por las siguientes cantidades de cuadrantes (definidos según la consideración 1):

a) 1 cuadrante (zonas de 1×1)

b) 4 cuadrantes (zonas de 2×2)

c) 9 cuadrantes (zonas de 3×3)

La figura 4.13 (sección 4.4.2) muestra la disposición de los cuadrantes para cada tamaño de zona. El sistema es evaluado por separado para cada una de estas divisiones para determinar cómo impacta el tamaño de las zonas definidas en la precisión. Como se mencionó en 2.8.1, las zonas de tamaño mínimo en FIND3 serían cuadrados de 1,75 m de lado. Por este motivo, se eligió como zona de

mayor tamaño las de 1,8 m de lado (3×3), probando además tamaños de zonas de menor tamaño, de 0,6 m y 1,2 m de lado (1×1 y 2×2, respectivamente).

Consumo de energía de los nodos: En el apéndice F se mencionan las pruebas realizadas para evaluar el consumo de energía por parte de los nodos.

4.4. Resultados obtenidos

4.4.1. Trilateración

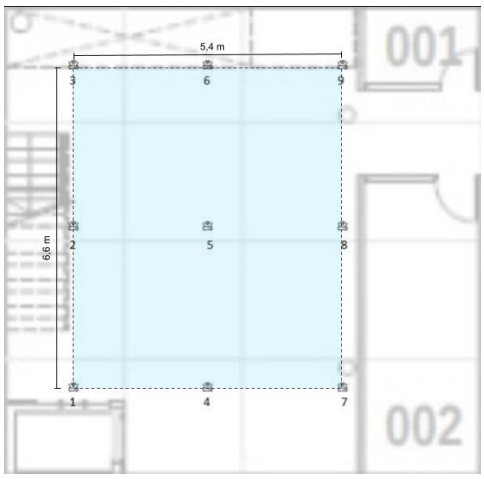
Con el fin de determinar si para este método es importante que el cliente esté siempre "entre nodos", para cada combinación de nodos (todos, sólo vértices, sólo aristas) se comparó el error diferenciando las ubicaciones del cliente durante la toma de datos en base a:

1. todas las ubicaciones en las que estuvo cliente (*todas las ubicaciones*)
2. sólo las ubicaciones dentro del cuadrilátero determinado por los nodos (*entre nodos*)
3. sólo las ubicaciones fuera de dicho cuadrilátero (*fuera de nodos*)

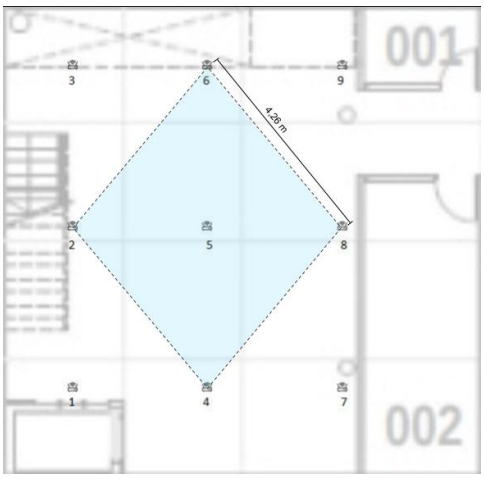
El cuadrilátero determinado por los nodos se trata de un rectángulo cuando se toman todos los nodos o sólo los de los vértices, y un rombo cuando se toman los nodos de las aristas (ver figura 4.2). A partir de ahora se dirá "*todas las ubicaciones*", "*entre nodos*" y "*fuera de nodos*" para hacer referencia a estas consideraciones respecto a las ubicaciones del cliente.

En las figuras 4.3, 4.4 y 4.5 se muestra la comparación de los errores considerando todas las ubicaciones, las ubicaciones entre nodos y las ubicaciones fuera de nodos, para el caso en que el cliente puede estar orientado hacia cualquier dirección. La figura 4.3 es para todos los nodos, la figura 4.4 los nodos de los vértices y la figura 4.5 para los nodos de las aristas. Allí se puede ver que el error siempre es menor si el cliente se encuentra entre nodos y mayor cuando se encuentra fuera de nodos.

En las figuras 4.6, 4.7 y 4.8 se muestra la comparación del error al considerar al cliente orientado siempre hacia una misma dirección. Igual que para todas las orientaciones, se observa que el error es menor cuando se encuentra entre nodos y mayor fuera de nodos, sin importar la combinación de nodos.



Nodos vértices que forman un rectángulo de 6,6 m × 5,4 m



Nodos aristas que forman un rombo de 4,26 m de lado

Figura 4.2: Determinación de cuadriláteros

De las pruebas realizadas considerando las ubicaciones dentro y fuera de nodos, se puede concluir que lo ideal sería que el cliente siempre se encuentre entre al menos tres nodos para obtener un error menor en la localización. Por lo tanto, y bajo el supuesto de que los nodos pueden ser colocados de forma tal que un cliente se encuentre siempre entre nodos, para el resto de las comparaciones se tomarán las ubicaciones entre nodos.

La figura 4.9 muestra la comparación del error para las distintas combinaciones de nodos, habiendo orientado al cliente hacia todas las direcciones (línea roja de las figuras 4.3, 4.4 y 4.5). La figura 4.10 muestra la misma comparación pero para el caso en que se orientó al cliente siempre hacia una misma dirección (línea roja de las figuras 4.6, 4.7 y 4.8). En ambas figuras se observa que el error es menor cuando se utilizaron los nodos en las aristas, algo mayor al utilizar todos los nodos, y mayor al utilizar sólo los de los vértices. Esto puede deberse a que los nodos en los vértices se encuentran más distanciados que los de las aristas, por lo que en los vértices se tienen distancias mayores al cliente, y se vio que a medida que aumenta la distancia (y así, el valor del RSSI disminuye) se comienza a perder la relación entre la intensidad de la señal y la distancia (ante valores de RSSI muy chicos, pequeñas variaciones hacen que las distancias calculadas aumenten considerablemente).

Si bien el resultado siempre es mejor para la combinación de nodos en las aristas, parecería más adecuado tomar la combinación de todos los nodos, ya que se ajusta más a la realidad. Ésta es la segunda mejor (luego de sólo

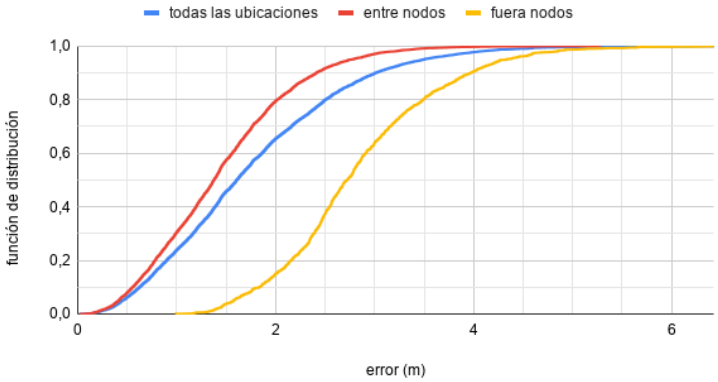


Figura 4.3: Trilateración - CDF del error con el cliente orientado hacia todas las direcciones usando todos los nodos

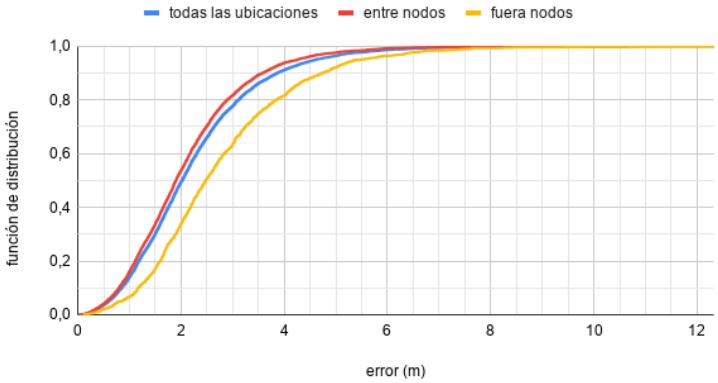


Figura 4.4: Trilateración - CDF del error con el cliente orientado hacia todas las direcciones usando los nodos en los vértices

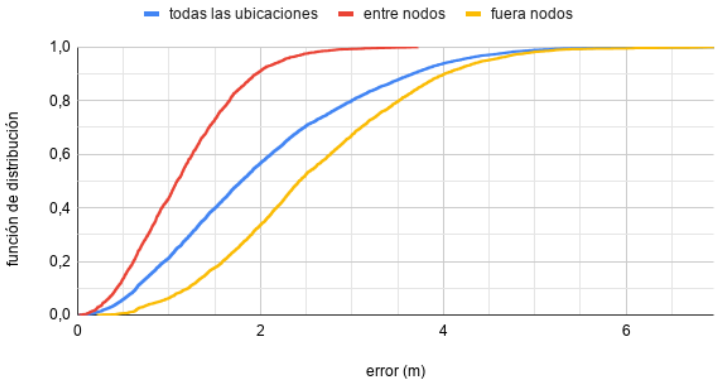


Figura 4.5: Trilateración - CDF del error con el cliente orientado hacia todas las direcciones usando los nodos en las aristas

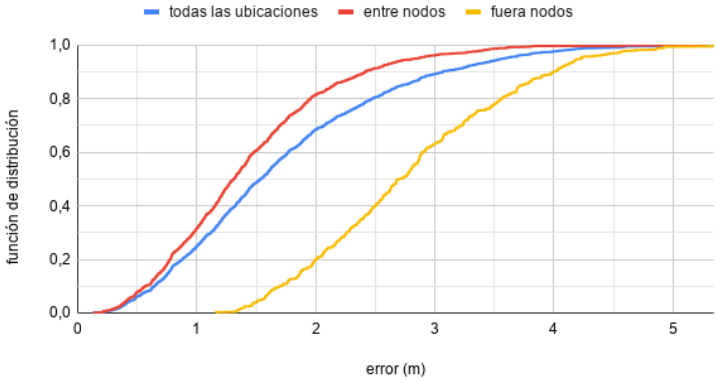


Figura 4.6: Trilateración - CDF del error con el cliente orientado hacia una dirección usando todos los nodos

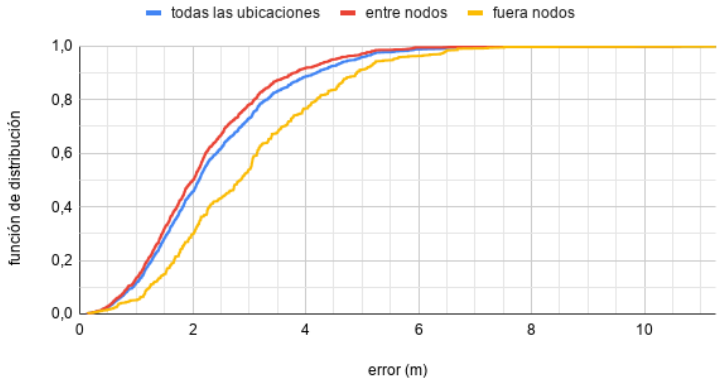


Figura 4.7: Trilateración - CDF del error con el cliente orientado hacia una dirección usando los nodos en los vértices

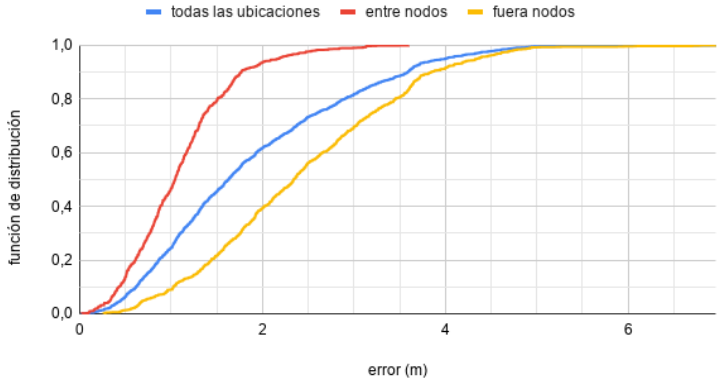


Figura 4.8: Trilateración - CDF del error con el cliente orientado hacia una dirección usando los nodos en las aristas

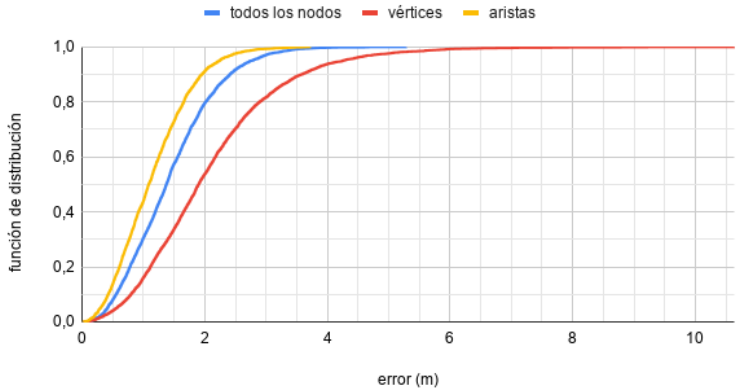


Figura 4.9: Trilateración - CDF del error con el cliente orientado hacia todas las direcciones, para las distintas combinaciones de nodos

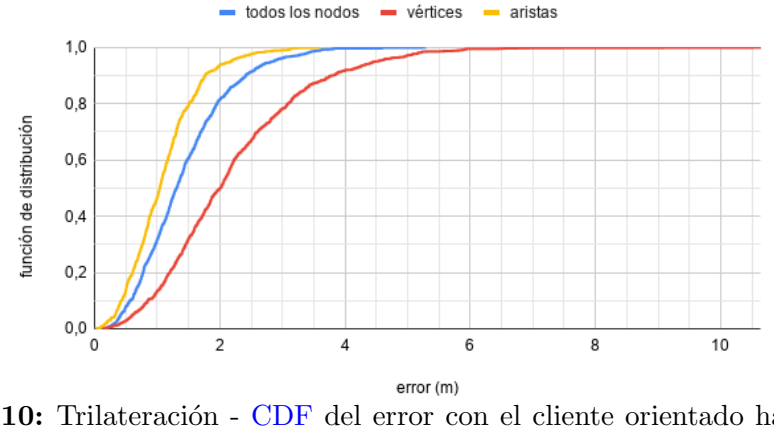


Figura 4.10: Trilateración - CDF del error con el cliente orientado hacia una dirección, para las distintas combinaciones de nodos

Figura 4.11: Trilateración - CDF del error con el cliente orientado hacia todas las direcciones y hacia una dirección

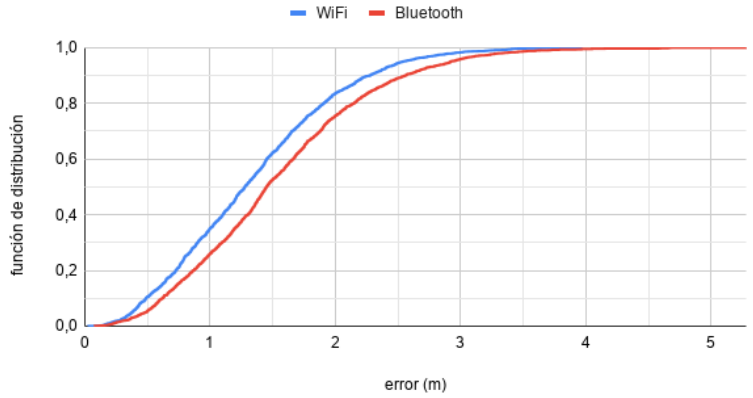


Figura 4.12: Trilateración - CDF del error localizando sólo dispositivos Wi-Fi y sólo dispositivos Bluetooth

los nodos en las aristas) pero abarca una mayor zona del lugar. De todas formas, otra conclusión que podría sacarse de esto es que tener los nodos menos distanciados, mejoraría el desempeño, al disminuir el error.

La figura 4.11 muestra la comparación del error considerando al cliente orientado siempre hacia una misma dirección, y sin considerar la orientación. Se puede ver que los errores son muy similares, de lo que se concluye que para Trilateración la orientación del cliente no afecta al desempeño global del sistema.

Una combinación de nodos adicional que se utilizó para este método, fueron los nodos de las aristas y de los vértices (i.e., todos los nodos menos el del centro). Los resultados de estas pruebas fueron casi idénticos al de todos los nodos, por lo que no se muestran los resultados, pero vale la pena mencionarlo.

Por último, se evaluó el método igual que antes pero para cada interfaz por

	Precisión (m)	P(e ≤ 1,0)	P(e ≤ 1,5)	P(e ≤ 2,0)	Error promedio
Ambas interfaces	1,38	30,58 %	57,45 %	79,54 %	1,44
Wi-Fi	1,28	35,12 %	62,33 %	83,58 %	1,34
Bluetooth	1,45	26,04 %	52,58 %	75,50 %	1,54

Tabla 4.1: Trilateración. Valores del error

separado (Wi-Fi y Bluetooth). Para cada interfaz, las conclusiones respecto a ubicaciones, combinaciones de nodos y orientación del cliente son las mismas que antes. Es decir, el error se comporta de forma similar a los casos antes mostrados. Lo que se pudo ver fue que para Wi-Fi el error es menor que para Bluetooth en todos los casos. La figura 4.12 muestra la comparación del error considerando cada interfaz por separado.

La tabla 4.1 muestra los valores del error para el método Trilateración. Se logra una precisión de 1,38 m si se utilizan ambas interfaces, 1,28 m si es utiliza sólo Wi-Fi y 1,45 m si se utiliza sólo Blueooth. El mejor caso se da para Wi-Fi, con casi el 84 % de las localizaciones con un error por debajo de los 2 m.

4.4.2. Fingerprinting

Además de calcular la exactitud, como se mencionó antes, se calculó el error en metros, ya que la exactitud del sistema podría ser baja pero podría estar prediciendo zonas cercanas a la real. Además, con la CDF del error, y la precisión en metros, es posible comparar la solución propuesta con otros trabajos, ya que en su mayoría miden su desempeño de esta forma, incluso cuando usan el método de Fingerprinting (ver sección 2.7). Como se mencionó en 4.3, el error se calculó como la distancia de la ubicación real del cliente a la ubicación calculada por el sistema (ecuación 4.1). Al tratarse de un método de clasificación, la ubicación calculada es una de las zonas definidas. Por este motivo, y para simplificar el cálculo del error, se tomó como ubicación calculada el centro de la zona determinada por el método. Cabe mencionar entonces que, dadas las posibles ubicaciones del cliente y este cálculo del error, solamente cuando se consideran las zonas de tamaño 1×1 se tendrá un error igual a cero siempre que haya un acierto en la clasificación (recordar consideración 5 de 4.2). Para las zonas de tamaño 3×3 el error será mayor o igual a cero, siendo cero cuando el cliente se encuentra en el centro de la zona. Por otra parte, para las zonas de tamaño 2×2 el error nunca es cero, ya que el centro de la zona en

ese caso es donde se juntan las cuatro esquinas de las baldosas que la forman, y no corresponde al centro de uno de los cuadrantes. La figura 4.13 ilustra los diferentes tamaños de zona con la ubicación que toma el sistema para calcular el error, y las ubicaciones en las que estuvo el cliente en cada zona. Por lo tanto, ante aciertos o fallas, se tendrá un error según los valores de la tabla 4.2. Con lo anterior, la exactitud, al ser el porcentaje de aciertos sobre el total de predicciones, es equivalente a:

- Para división de zonas de 1×1: $exactitud = P(e = 0)$
- Para división de zonas de 2×2: $exactitud = P(e \leq 0,424)$
- Para división de zonas de 3×3: $exactitud = P(e \leq 0,849)$

La tabla 4.3 muestra la exactitud del sistema para las distintas combinaciones de nodos utilizadas y los diferentes tamaños de zonas, cuando el cliente se orientó hacia todas las direcciones. Para cada tamaño de zona y combinación de nodos, se indica el porcentaje de aciertos ("exacto", i.e., la exactitud), el porcentaje de fallas en los que se predijo la zona contigua ("al lado") y el porcentaje de fallas que no es ninguno de los anteriores ("distante"). Para las zonas de tamaño 1×1, además se muestra el porcentaje de fallas en los que se predijo una zona a una zona de por medio a la real ("cerca"). La tabla 4.4 es análoga a la tabla 4.3, pero para el caso en que el cliente estuvo orientado siempre hacia una misma dirección.

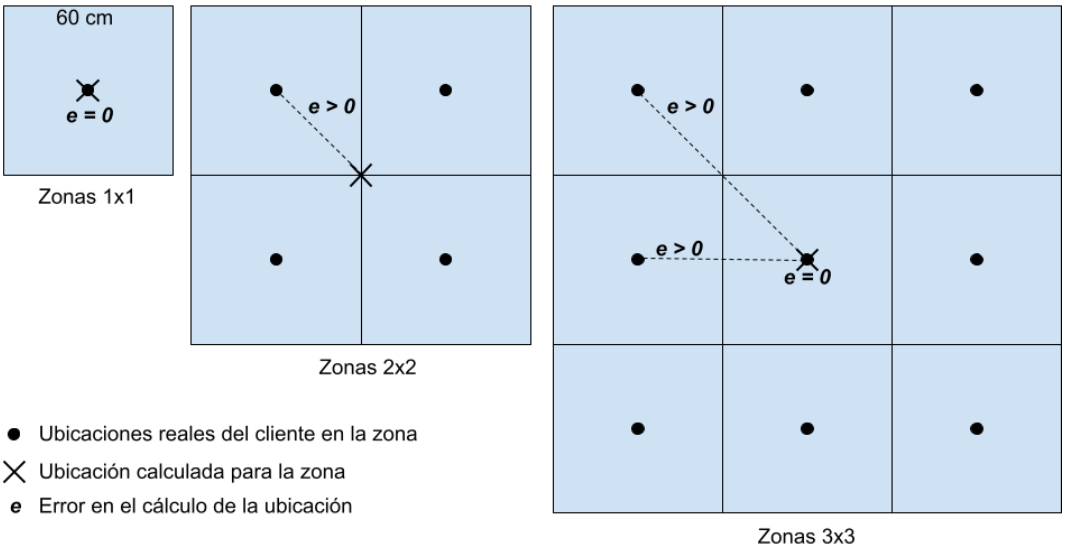


Figura 4.13: Ubicaciones y error en el acierto

	Acierto	Falla
Zonas 1×1	error = 0	error ≥ 0,6
Zonas 2×2	error = 0,424	error ≥ 0,949
Zonas 3×3	error ∈ {0; 0,6; 0,849}	error ≥ 1,2

Tabla 4.2: Error en metros en aciertos y fallas en la clasificación

Zonas 1×1				
	exacto	al lado	cerca	distante
Todos	9,9	28,3	29,6	32,2
Vértices	1,1	8,4	12,2	78,3
Aristas	1,2	8,2	12,2	78,3

-

	Zonas 2×2			**Zonas 3×3**		
	exacto	al lado	distante	exacto	al lado	distante
Todos	27,4	54,3	18,3	43,2	52,0	4,8
Vértices	4,0	22,1	73,9	8,0	39,0	53,0
Aristas	15,9	47,3	36,7	27,7	52,7	19,5

Tabla 4.3: Cliente orientado hacia todas las direcciones

Zonas 1×1				
	exacto	al lado	cerca	distante
Todos	9,6	30,5	22,5	37,4
Vértices	5,7	18,7	23,1	52,5
Aristas	5,0	18,2	26,0	50,8

-

	Zonas 2×2			**Zonas 3×3**		
	exacto	al lado	distante	exacto	al lado	distante
Todos	26,0	49,1	24,9	44,5	48,9	6,6
Vértices	18,0	45,4	36,6	27,5	58,3	14,2
Aristas	13,6	48,5	37,9	25,6	57,1	17,3

Tabla 4.4: Cliente orientado siempre hacia la misma dirección

En las figuras 4.14, 4.15 y 4.16 se muestra la distribución del error para las zonas de tamaño 1×1, 2×2 y 3×3, respectivamente, considerando al cliente orientado hacia todas las direcciones. En cada una de estas figuras se comparan las combinaciones de nodos (todos, sólo vértices y sólo aristas).

De igual forma, las figuras 4.17, 4.18 y 4.19 muestran la distribución del error cuando el cliente se encuentra orientado siempre hacia la misma dirección.

Para el caso en que se orientó al cliente hacia todas las direcciones se obtuvieron los valores para el error indicados en las tablas 4.5, 4.6 y 4.7, según el tamaño de las zonas, y para el cliente orientado siempre hacia una misma dirección, estos valores están dados en las tablas 4.8, 4.9 y 4.10.

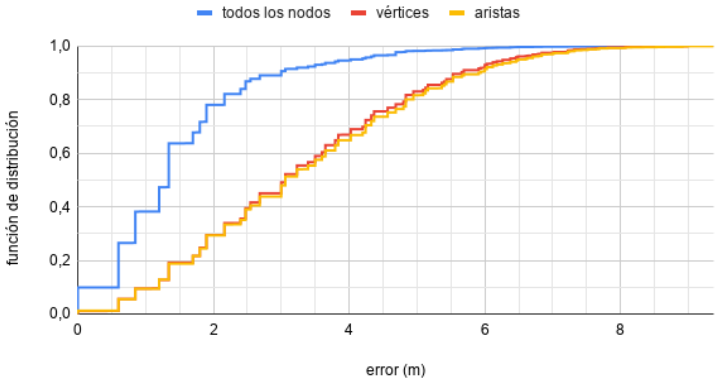


Figura 4.14: Fingerprinting para zonas 1×1 - CDF del error con el cliente orientado hacia todas las direcciones

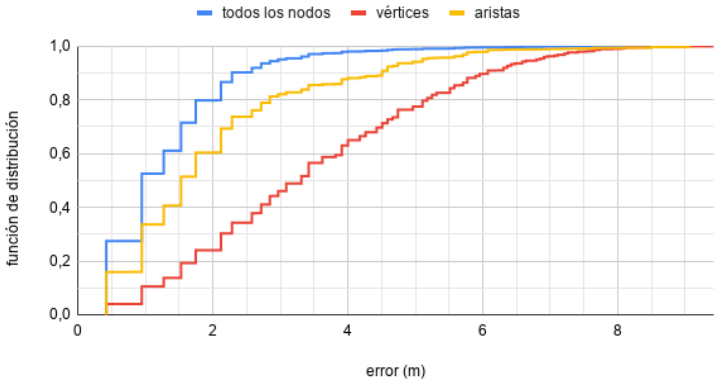


Figura 4.15: Fingerprinting para zonas 2×2 - CDF del error con el cliente orientado hacia todas las direcciones

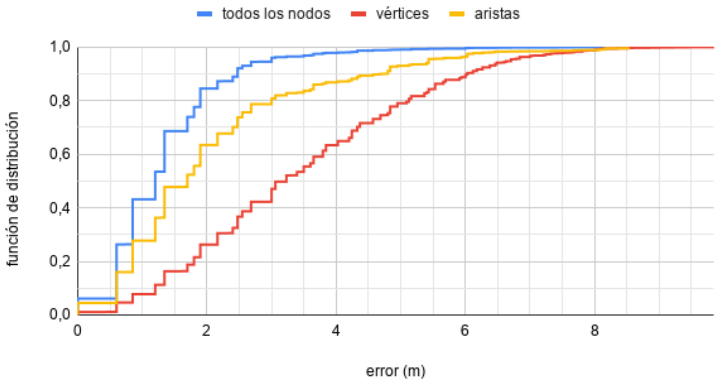


Figura 4.16: Fingerprinting para zonas 3×3 - CDF del error con el cliente orientado hacia todas las direcciones

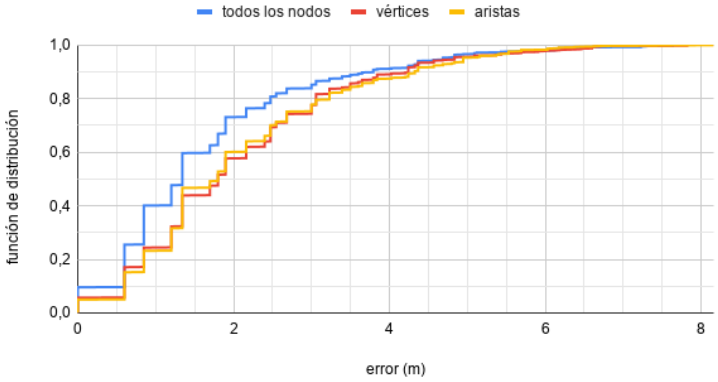


Figura 4.17: Fingerprinting para zonas 1×1 - CDF del error con el cliente orientado hacia una dirección

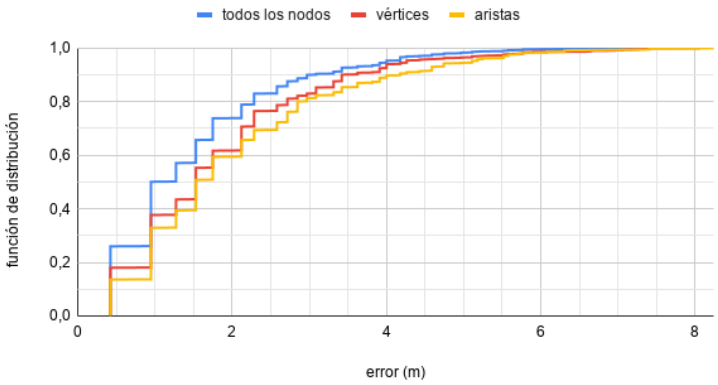


Figura 4.18: Fingerprinting para zonas 2×2 - CDF del error con el cliente orientado hacia una dirección

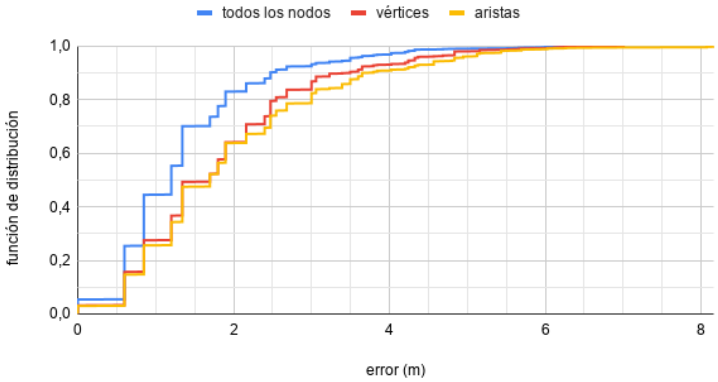


Figura 4.19: Fingerprinting para zonas 3×3 - CDF del error con el cliente orientado hacia una dirección

Zonas 1×1 Todas dirs.	**Precisión (m)**	**P(e ≤ 1,0)**	**P(e ≤ 1,5)**	**P(e ≤ 2,0)**	**Error promedio**
Todos los nodos	1,34	38,21 %	63,71 %	78,08 %	1,50
Nodos en los vértices	3,06	9,50 %	19,14 %	29,46 %	3,24
Nodos en las aristas	3,06	9,42 %	18,79 %	29,33 %	3,31

Tabla 4.5: Zonas 1×1. Cliente hacia todas las direcciones

Zonas 2×2 Todas dirs.	**Precisión (m)**	**P(e ≤ 1,0)**	**P(e ≤ 1,5)**	**P(e ≤ 2,0)**	**Error promedio**
Todos los nodos	0,96	52,56 %	61,10 %	79,85 %	1,34
Nodos en los vértices	3,31	10,56 %	13,73 %	24,03 %	3,46
Nodos en las aristas	1,53	33,64 %	40,67 %	60,41 %	2,01

Tabla 4.6: Zonas 2×2. Cliente hacia todas las direcciones

Zonas 3×3 Todas dirs.	**Precisión (m)**	**P(e ≤ 1,0)**	**P(e ≤ 1,5)**	**P(e ≤ 2,0)**	**Error promedio**
Todos los nodos	1,2	43,17 %	68,62 %	84,53 %	1,34
Nodos en los vértices	3,23	7,78 %	16,34 %	26,27 %	3,44
Nodos en las aristas	1,7	27,74 %	47,80 %	63,47 %	2,07

Tabla 4.7: Zonas 3×3. Cliente hacia todas las direcciones

Zonas 1×1 Una dir.	**Precisión (m)**	**P(e ≤ 1,0)**	**P(e ≤ 1,5)**	**P(e ≤ 2,0)**	**Error promedio**
Todos los nodos	1,34	40,10 %	59,70 %	73,10 %	1,67
Nodos en los vértices	1,8	24,40 %	43,90 %	57,70 %	2,08
Nodos en las aristas	1,8	23,20 %	46,70 %	60,10 %	2,08

Tabla 4.8: Zonas 1×1. Cliente hacia una dirección

Zonas 2×2 Una dir.	**Precisión (m)**	**P(e ≤ 1,0)**	**P(e ≤ 1,5)**	**P(e ≤ 2,0)**	**Error promedio**
Todos los nodos	0,95	50,10 %	57,10 %	73,80 %	1,52
Nodos en los vértices	1,53	37,70 %	43,50 %	61,70 %	1,84
Nodos en las aristas	1,53	32,90 %	39,40 %	59,40 %	2,03

Tabla 4.9: Zonas 2×2. Cliente hacia una dirección

Zonas 3×3 Una dir.	**Precisión (m)**	**P(e ≤ 1,0)**	**P(e ≤ 1,5)**	**P(e ≤ 2,0)**	**Error promedio**
Todos los nodos	1,2	44,50 %	70,10 %	83,00 %	1,37
Nodos en los vértices	1,7	27,50 %	49,30 %	64,10 %	1,84
Nodos en las aristas	1,7	25,60 %	47,50 %	63,80 %	1,97

Tabla 4.10: Zonas 3×3. Cliente hacia una dirección

En las tablas 4.3 y 4.4 se observa que la exactitud es mayor cuando se consideran todos los nodos en lugar de los subconjuntos elegidos. A su vez, de las figuras 4.14, 4.15, 4.16 y 4.17, 4.18, 4.19, se llega a que el error es menor también al utilizar todos los nodos. Por lo tanto, de ahora en más, las

consideraciones que se realizan sobre el error y la exactitud son tomando todos los nodos.

En las figuras 4.20, 4.21 y 4.22 se compara el error cuando el cliente fue orientado hacia todas las direcciones y con éste orientado siempre hacia la misma dirección. Si bien, en general, el error es un poco menor cuando se orientó hacia todas las direcciones, se puede ver que esto no influye significativamente en el error obtenido. Lo mismo se observa para la exactitud en las tablas 4.3 y 4.4. Por lo tanto, en los párrafos que siguen siempre se considera al cliente cuando se orientó hacia todas las direcciones.

Finalmente, la figura 4.23 muestra la comparación del error para los distintos tamaños de zonas. Se observa que el error se distribuye de manera similar para los distintos tamaños, si bien, a priori se podría pensar que debería ser mayor cuanto mayor es el tamaño de las zonas, dado que para zonas mayores los aciertos en su mayoría tienen un error mayor a cero (tabla 4.2). Esto también se observa en las tablas 4.5, 4.6 y 4.7, donde se puede ver que los valores del error son similares para los distintos tamaños de zona, o incluso mejores cuando aumenta. La mejor precisión se obtuvo en las zonas 2×2, siendo 0,96 m. Por otro lado, la mayor cantidad de resultados con error por debajo de los 2 m, se dio para las zonas 3×3, 84,5 %.

Otra observación es que la exactitud en la predicción mejora al aumentar el tamaño de las zonas (tabla 4.3), lo que se esperaría desde un principio. Sin embargo, como ya se dijo, el error en metros tiende a mantenerse. Esto se debe a que, por más que aumente el porcentaje de aciertos, en zonas más grandes no se obtiene una localización más exacta (no se sabe la ubicación exacta dentro de una zona que puede ser grande), y por esto, el error si bien mejora, no lo hace significativamente.

Para ver la distribución de la exactitud en las zonas, se crearon las matrices de confusión a partir de los datos de la evaluación. Para facilitar la visualización de las matrices, a mayores porcentajes de aciertos en la zona se asigna un color más oscuro. Al generarse una diagonal central se describe que lo que se predijo es el valor real, y diagonales a los costados de la central describen que lo que se predijo está cerca del valor real.

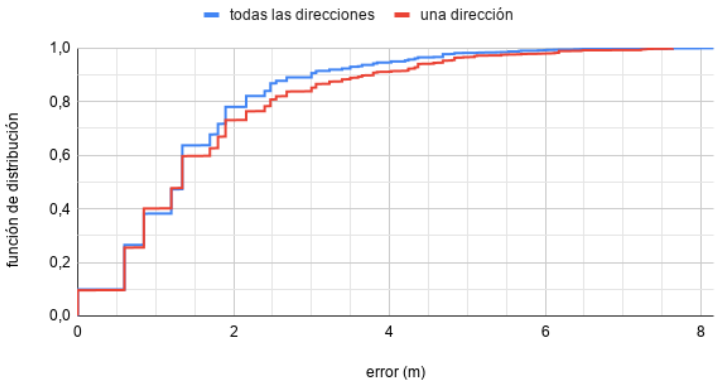


Figura 4.20: Fingerprinting para zonas 1×1 - CDF del error con el cliente orientado hacia todas las direcciones y hacia una dirección

Figura 4.21: Fingerprinting para zonas 2×2 - CDF del error con el cliente orientado hacia todas las direcciones y hacia una dirección

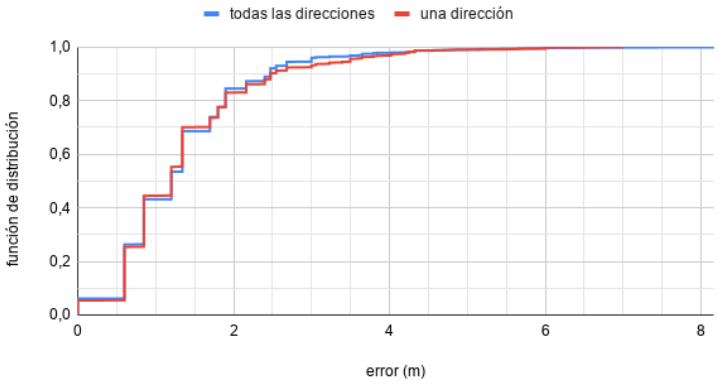


Figura 4.22: Fingerprinting para zonas 3×3 - CDF del error con el cliente orientado hacia todas las direcciones y hacia una dirección

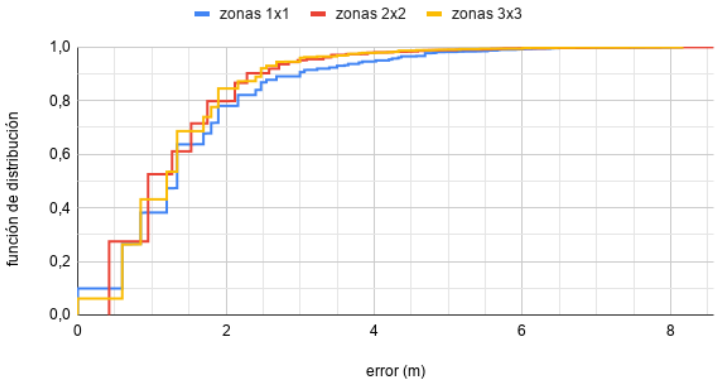


Figura 4.23: Fingerprinting - CDF del error según tamaño de las zonas

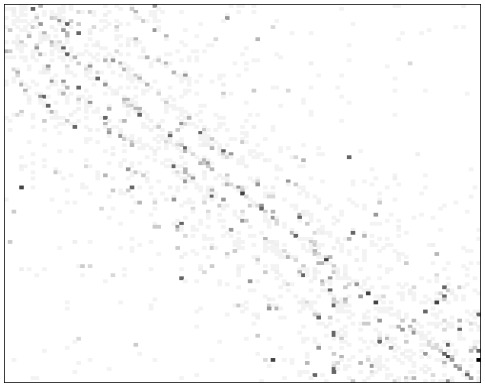


Figura 4.24: Fingerprinting para zonas 1×1 - Matriz de confusión usando todos los nodos

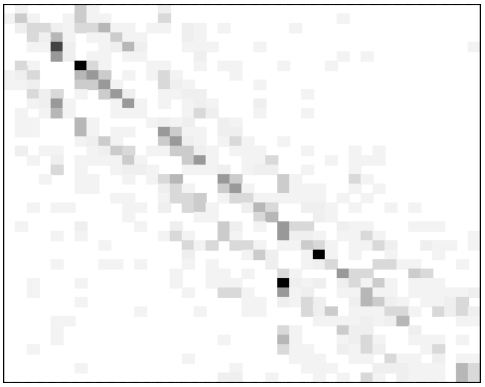


Figura 4.25: Fingerprinting para zonas 2×2 - Matriz de confusión usando todos los nodos

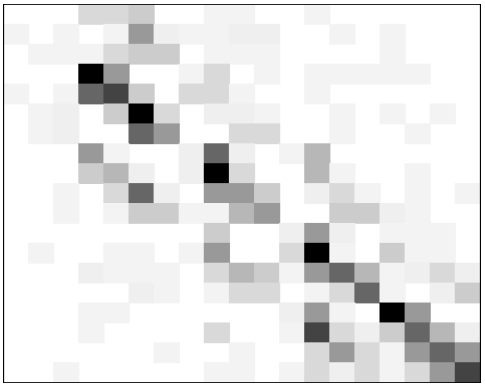


Figura 4.26: Fingerprinting para zonas 3×3 - Matriz de confusión usando todos los nodos

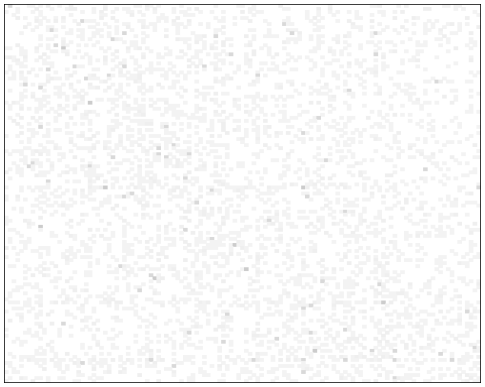


Figura 4.27: Fingerprinting para zonas 1×1 - Matriz de confusión usando los nodos en las aristas

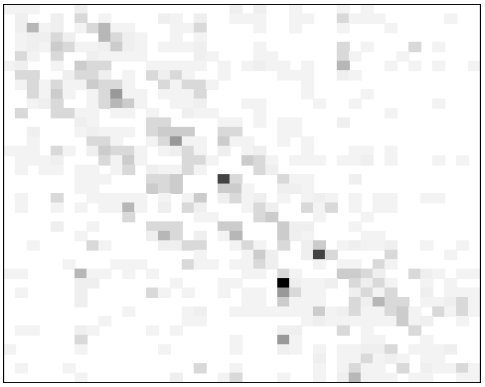


Figura 4.28: Fingerprinting para zonas 2×2 - Matriz de confusión usando los nodos en las aristas

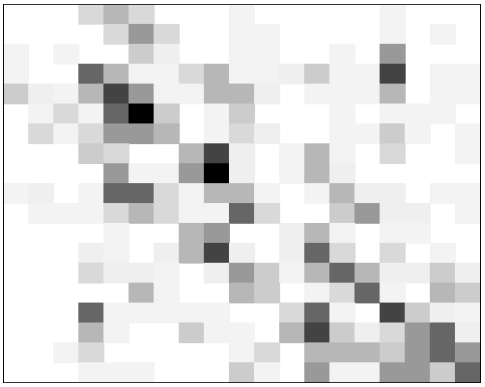


Figura 4.29: Fingerprinting para zonas 3×3 - Matriz de confusión usando los nodos en las aristas

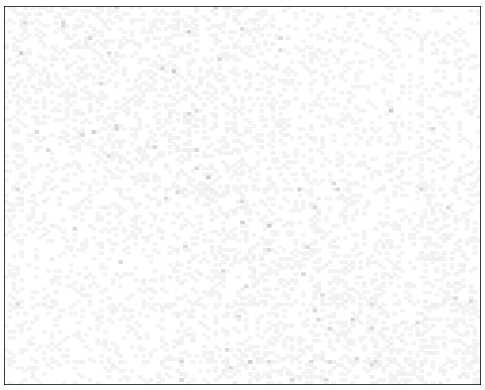


Figura 4.30: Fingerprinting para zonas 1×1 - Matriz de confusión usando los nodos en los vértices

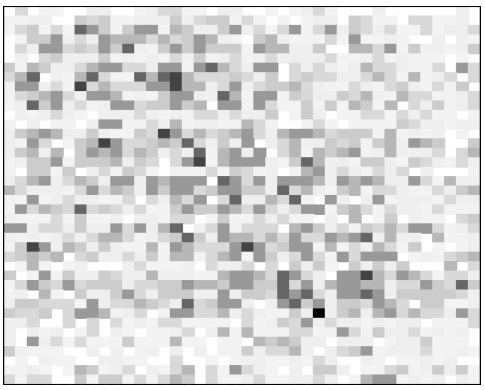


Figura 4.31: Fingerprinting para zonas 2×2 - Matriz de confusión usando los nodos en los vértices

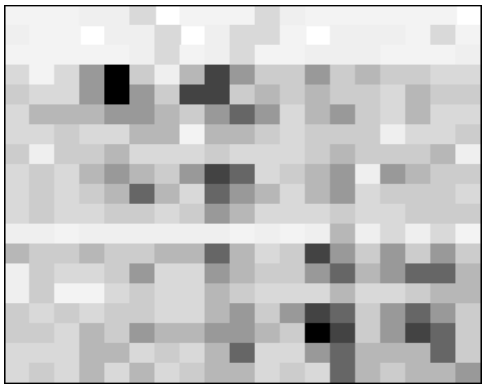


Figura 4.32: Fingerprinting para zonas 3×3 - Matriz de confusión usando los nodos en los vértices

Se puede ver que para el caso de todos los nodos (figuras 4.24, 4.25 y 4.26) se genera una diagonal central oscura, con diagonales en sus costados, para los tres tamaños de zonas, indicando que la mayoría de las predicciones son aciertos o fallas pero cerca del valor real. El peor resultado se obtuvo para los vértices (figuras 4.30, 4.31 y 4.32), donde las predicciones se encuentran dispersas. Para las aristas (figuras 4.27, 4.28 y 4.29), se observa la misma dispersión para las zonas 1×1, y se puede ver que esto mejora para las zonas 2×2 y 3×3. Lo anterior coincide con lo que se muestra en la distribución del error de las figuras 4.14, 4.15, 4.16: el mejor resultado siempre está en todos los nodos, para aristas es algo peor en 2×2 y 3×3, y considerablemente peor en 1×1, y para los vértices es considerablemente peor para los tres tamaños de zona.

Por último se comparó el error al considerar por separado las interfaces Wi-Fi y Bluetooth. La distribución del error para cada tamaño de zona se muestra en las figuras 4.33, 4.34 y 4.35, y los valores de referencia para el error en las tablas 4.11, 4.12 y 4.13. De las figuras se observa que el error se distribuye de forma muy similar para ambas interfaces, y de las tablas, que en general el

error es algo menor al utilizar sólo Wi-Fi. La mejor precisión se da al utilizar sólo Wi-Fi, 0,95 m en las zonas 2×2, y el mayor porcentaje de resultados con error menor a 2 m se da en las zonas 3×3, siendo un 84,9 %, también utilizando la interfaz Wi-Fi.

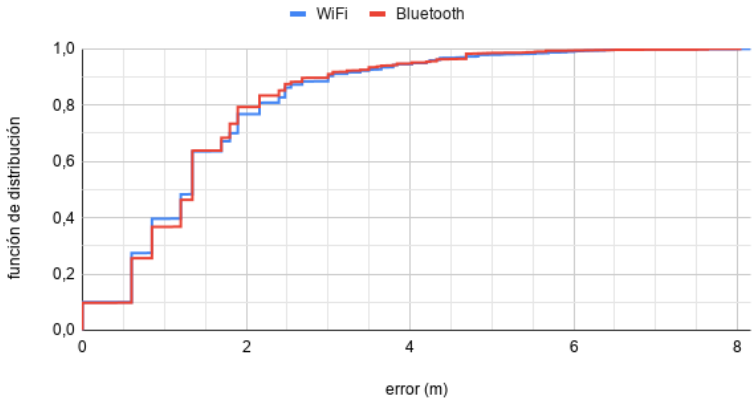


Figura 4.33: Fingerprinting para zonas 1×1 - CDF del error localizando sólo dispositivos Wi-Fi y sólo dispositivos Bluetooth

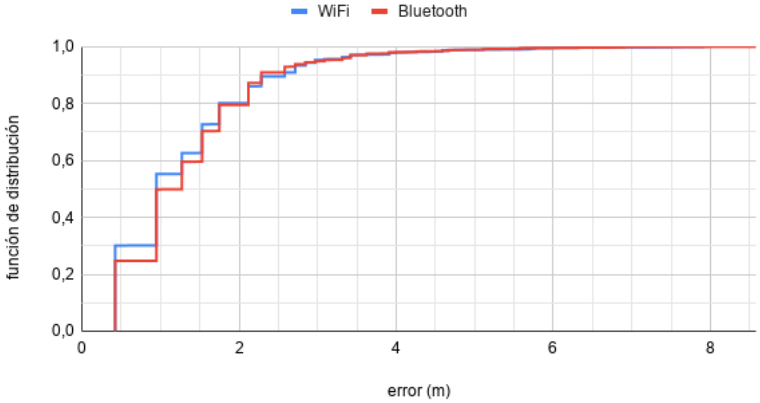


Figura 4.34: Fingerprinting para zonas 2×2 - CDF del error localizando sólo dispositivos Wi-Fi y sólo dispositivos Bluetooth

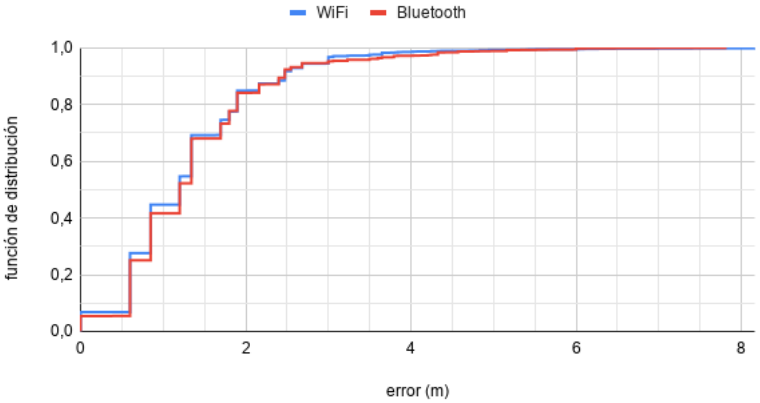


Figura 4.35: Fingerprinting para zonas 3×3 - CDF del error localizando sólo dispositivos Wi-Fi y sólo dispositivos Bluetooth

	Precisión (m)	**P(e ≤ 1,0)**	**P(e ≤ 1,5)**	**P(e ≤ 2,0)**	**Error promedio**
Wi-Fi	1,34	39,66 %	63,58 %	76,81 %	1,51
Bluetooth	1,34	36,77 %	63,84 %	79,35 %	1,49

Tabla 4.11: Zonas 1×1. Wi-Fi vs. Bluetooth

	Precisión (m)	**P(e ≤ 1,0)**	**P(e ≤ 1,5)**	**P(e ≤ 2,0)**	**Error promedio**
Wi-Fi	0,95	55,26 %	62,63 %	80,17 %	1,31
Bluetooth	1,27	49,87 %	59,57 %	79,53 %	1,36

Tabla 4.12: Zonas 2×2. Wi-Fi vs. Bluetooth

	Precisión (m)	**P(e ≤ 1,0)**	**P(e ≤ 1,5)**	**P(e ≤ 2,0)**	**Error promedio**
Wi-Fi	1,2	44,70 %	69,22 %	84,91 %	1,31
Bluetooth	1,2	41,64 %	68,02 %	84,14 %	1,37

Tabla 4.13: Zonas 3×3. Wi-Fi vs. Bluetooth

4.5. Conclusiones

A continuación se realiza un resumen de las conclusiones que surgen de las pruebas realizadas con ambos métodos.

4.5.1. Trilateración

1. Los resultados obtenidos siempre son mejores cuando el cliente se encuentra entre nodos.

2. Funciona mejor cuando se usaron todos los nodos.

3. Cuanto más alejados se encuentran los nodos, el error aumenta, según se pudo ver comparando las combinaciones de nodos en los vértices y en las aristas.

4. La orientación del cliente no influyó significativamente en el error obtenido.

5. Al utilizar las interfaces Wi-Fi y Bluetooth por separado se obtienen resultados similares a utilizarlas a la vez, pero son mejores cuando se utiliza sólo Wi-Fi.

6. La precisión del sistema es 1,38 m al utilizar las tecnologías Wi-Fi y Bluetooth a la vez, y 1,28 m cuando se utilizó sólo Wi-Fi.

7. Con ambas tecnologías a la vez el 30,6 % de las estimaciones tienen un error por debajo del metro y casi el 80 % por debajo de los 2 m. Al utilizar sólo Wi-Fi, estos valores aumentan al 35 % y 83,6 %, respectivamente.

4.5.2. Fingerprinting

1. La localización siempre fue mejor cuando se utilizaron todos los nodos.

2. La orientación del cliente no influyó significativamente en el error obtenido, y tampoco en la exactitud.

3. La exactitud aumenta a medida que el tamaño de las zonas también aumenta.

4. Al aumentar el tamaño de las zonas, el error no varía significativamente.

5. Al utilizar las interfaces Wi-Fi y Bluetooth por separado se obtienen resultados similares a utilizarlas a la vez, pero son mejores cuando se utiliza sólo Wi-Fi.

6. La precisión con este método, según el tamaño de zona es:

   - en zonas 1×1: 1,34 m, con el 38 % de las estimaciones por debajo del metro y el 78 % debajo de los 2 m.
   - en zonas 2×2: 0,96 m (0,95 m sólo con Wi-Fi), con el 52,6 % de las estimaciones por debajo del metro y casi el 80 % por debajo de los 2 m.
   - en zonas 3×3: 1,2 m, con el 43 % de las estimaciones por debajo del metro y el 84,5 % por debajo de los 2 m.

Cuando se usó la división de zonas 3×3 la precisión de este método (1,2 m) es algo mejor pero muy similar que la de trilateración (1,38 m con ambas interfaces y 1,28 m sólo Wi-Fi). Al utilizar la división de zonas 2×2 la precisión mejora (0,96 m), alcanzando una precisión por debajo del metro.

4.6. Comparación con otros trabajos

En esta sección se realiza una comparación del trabajo realizado con los trabajos existentes encontrados (2.7). Para esto, se realiza la tabla 4.14, donde se indica el trabajo, el método y la tecnología utilizados, y los valores de referencia para el error (precisión, error promedio, $f(1) = P(e \leq 1)$, $f(1,5) = P(e \leq 1,5)$ y $f(2) = P(e \leq 2)$). Notaciones:

- **T.RSSI**: Trilateración a partir de las distancias calculadas con un modelo de propagación a partir del RSSI.
- **T.CSI**: Trilateración a partir de las distancias calculadas con un modelo de propagación a partir del CSI.
- **Fing.RSSI**: Fingerprinting utilizando el RSSI.
- **Fing.CSI**: Fingerprinting utilizando el CSI.
- **W+B**: Wi-Fi y Bluetooth.
- **- (guión)**: Dato no disponible.

Las filas 1 a 5 se corresponden a los resultados de este proyecto. Del resto de los trabajos, sólo el de las filas 6 y 7 (2.7.2.9 [60]) utiliza hardware que se puede considerar de IoT (seis placas Raspberry Pi 3, más una tarjeta Wi-Fi TP-Link TL-WN725N cada una). El resto, o bien utiliza hardware específico, o AP Wi-Fi, o tarjetas de red que deben ser colocadas en una PC, y en todos los casos se requiere conexión a una red de área local para el envío de la información, y una instalación de red eléctrica.

Se puede ver que este trabajo funcionó mejor que el trabajo de las filas 6 y 7, dado que tiene una precisión algo mejor comparando las técnicas de Fingerprinting y Trilateración a partir del RSSI.

De los trabajos que usan el RSSI como medida de la señal, sólo dos tienen una mejor precisión que los de la solución propuesta en el presente proyecto, que son los de las filas 18 (2.7.4.2 [80]) y 22 (2.7.5.2 [45]). El resto de los trabajos con una mejor precisión que la solución propuesta usa AoA o el CSI, lo que requiere hardware con múltiples antenas, o ToF, lo que requiere una alta sincronización entre los nodos y los clientes. Los que requieren hardware con múltiples antenas son los trabajos de las filas 8 (2.7.2.1 [77]), 9 (2.7.2.2 [43]), 13 (2.7.2.6 [73]), 16 (2.7.3.5 [4]), y el de las filas 19 y 20 (2.7.4.5 [76]), y los que utilizan ToF son los de las filas 10 (2.7.2.3 [70]) y 11 (2.7.2.4 [78]).

	Trabajo	Método	Tecnología	Precisión (m)	Error promedio (m)	f(1)	f(1,5)	f(2)
1	*	T.RSSI	Wi-Fi	**1,28**	1,34	35,12	62,33	83,58
2	*	T.RSSI	W+B	1,38	1,44	38,58	57,45	79,54
3	*	Fing.RSSI (1×1)	W+B	1,34	1,50	38,21	63,71	78,08
4	*	Fing.RSSI (2×2)	W+B	**0,96**	1,34	52,56	61,10	79,85
5	*	Fing.RSSI (3×3)	W+B	1,2	1,34	43,17	68,62	84,53
6	2.7.2.9 [60]	T.RSSI	Wi-Fi	2	1,9	35	48	50
7	2.7.2.9 [60]	Fing.RSSI	Wi-Fi	1	1,5	45	55	68
8	2.7.2.1 [77]	AoA	Wi-Fi	0,23	-	95	100	100
9	2.7.2.2 [43]	AoA+ToF	Wi-Fi	0,40	-	63	78	80
10	2.7.2.3 [70]	ToF	Wi-Fi	0,65	-	50	85	93
11	2.7.2.4 [78]	ToF	Wi-Fi	0,9	-	55	75	93
12	2.7.2.5 [34]	AoA	Wi-Fi	2	-	33	48	65
13	2.7.2.6 [73]	Fing.CSI	Wi-Fi	0,95	-	60	75	100
14	2.7.2.7 [11]	Fing.RSSI	W+B	2,32	-	-	-	-
15	2.7.2.8 [5]	Fing.RSSI	Wi-Fi	2,94	-	-	-	-
16	2.7.3.5 [4]	AoA (CSI)	BLE	0,89	-	60	83	96
17	2.7.4.1 [49]	T.RSSI	Wi-Fi	1,76	-	20	48	65
18	2.7.4.2 [80]	Fing.RSSI	Wi-Fi	0,39	-	98	100	100
19	2.7.4.5 [76]	T.CSI	Wi-Fi	0,4	-	97	100	100
20	2.7.4.5 [76]	Fing.CSI	Wi-Fi	0,6	-	75	90	100
21	2.7.4.6 [27]	Fing.RSSI	Wi-Fi	1,22	-	45	72	89
22	2.7.5.2 [45]	Fing.RSSI	W+B	0,77	-	-	-	-

Tabla 4.14: Comparación con otros trabajos

Capítulo 5

Conclusiones y Trabajo a Futuro

En este capítulo se comentan las conclusiones globales sobre el trabajo realizado, además de los trabajos futuros que se podrían realizar a partir del mismo.

5.1. Conclusiones

Se construyó un sistema de localización en interiores utilizando hardware y una arquitectura de IoT y se evaluó su desempeño. Con esto se cumplió con el objetivo general de este trabajo, que consistía en evaluar la viabilidad de utilizar equipamiento de IoT para la localización en interiores. Se implementó una API con servicios REST, que permite la interoperabilidad con cualquier sistema, permitiendo que los datos de localización, además de ser mostrados en un mapa, puedan ser utilizados para análisis de datos más complejos.

Una de las principales ventajas encontradas en la solución propuesta es la facilidad de la implantación del sistema, pudiendo desplegar los nodos sin la necesidad de una red eléctrica ni de una red de área local, dado que los nodos pueden funcionar a batería, y que se cuenta con una tecnología de largo alcance, como lo es LoRa.

En relación a otros trabajos, se pudo desarrollar un sistema con resultados de precisión comparables, en el sentido de su fácil implantación, y el bajo costo de los componentes utilizados. Si bien en los trabajos encontrados hay varios con una mejor precisión, éstos no utilizan hardware de IoT, y en algunos casos

utilizan hardware específico y costoso, y además requieren un despliegue de red eléctrica e infraestructura de red para su funcionamiento.

Si se quisiera lograr una baja latencia en la localización, LoRa parecería no ser la mejor opción para el envío de la información desde los nodos, al menos en su versión original. Esto se debe a que está diseñado, principalmente, para envíos esporádicos y con poca información.

Uno de los principales puntos a tener en cuenta al momento de implantar el sistema, es el algoritmo de localización a utilizar, Fingerprinting o Trilateración. Con Trilateración, en pocas horas es posible tener el sistema funcionando, pero con una precisión algo menor en comparación a Fingerprinting. Con Fingerprinting se gana en precisión, pero requiere un esfuerzo sensiblemente mayor, debido a la recolección de datos para generar el Radio Map para entrenar los algoritmos, que depende del área en donde se va a implementar. Además, ante cambios en el entorno es necesario volver a realizar el aprendizaje en el área (i.e., volver a recolectar los datos para entrenar los algoritmos y calcular la precisión del sistema). Con Trilateración, el esfuerzo necesario para la implantación es menor, ya que requiere volver a tomar el valor del RSSI a 1 m de distancia de los nodos, y calcular el nuevo valor de n del modelo de propagación.

Como se mencionó en la sección 3.3, se observó que, con las antenas con las que cuentan los módulos ESP32 utilizados, se obtienen valores de RSSI distintos a medida que se rota el dispositivo, incluso para la misma distancia y ubicación. En Trilateración, esto puede influir en el error, ya que los valores de RSSI obtenidos pueden no concordar con las distancias a las que se encuentra el cliente de los nodos, aunque se esté en LoS, ya que dependerá de cómo esté posicionado el dispositivo. Al utilizar Fingerprinting, si bien la precisión mejoró, no fue en gran medida, y aunque este método podría estar teniendo esto en cuenta (con la generación del Radio Map), se cree que también puede influir en el error obtenido.

Dados los resultados obtenidos, el sistema podría aplicar a los siguientes casos de uso:

- Localización de dispositivos que formen parte del inventario de una em-

presa. El inventario podría tratarse de dispositivos que cuenten con una interfaz Wi-Fi o Bluetooth (e.g., impresoras, teléfonos celulares, tablets, auriculares bluetooth, notebooks, etc), o cualquier otro objeto (por ejemplo, es posible utilizar beacons Bluetooth, que son pequeños y pueden ser colocados en cualquier objeto).

- Localización de robots, en la que no se requiera una actualización en tiempo real, y que una precisión cercana al metro sea aceptable.
- Localizar dispositivos en un recinto que no tenga una red eléctrica distribuida, dado que con esta solución, al utilizar hardware de IoT, y por las pruebas realizadas, puede funcionar con baterías.
- Localizar dispositivos en un recinto donde no se tenga una buena conexión de red (o que sea nula), ya que mediante el envío con LoRa, esto no sería necesario.
- En un edificio podría ser utilizado como sistema de seguridad, ya que permite detectar dispositivos (es decir, personas) no autorizados a acceder a determinada área.
- Realización de estrategias comerciales en base al recorrido de los clientes en un local (por ejemplo, en un supermercado), para ordenar los productos según su comportamiento.

Por otro lado, el sistema podría no ser adecuado para los siguientes casos de uso:

- Sistemas que requieran localización en tiempo real.
- Sistemas que requieran una precisión bastante menor al metro (por ejemplo, precisión a nivel de decímetros).
- Utilización de robots para reconocimiento de campo, dado que se requiere que los nodos se encuentren cerca entre sí (en las pruebas realizadas, se vio que cuanto más alejados están los nodos, mayor es el error obtenido).

5.2. Trabajo a Futuro

En caso de optar por el uso de Trilateración, una de las posibles mejoras a realizar sería cambiar las antenas de los módulos ESP32 por antenas omnidi-

reccionales (sección 3.3).

Se podría profundizar en la eliminación de las ventanas Rx en la biblioteca LMiC utilizada por LuaRTOS, para que el envío de cada mensaje utilizando LoRaWAN demore menos (problema descrito en 3.5), y así lograr una localización en tiempo real.

Actualmente, los parámetros de los nodos (tiempos de scan y sleep, interfaces a utilizar, filtro de direcciones MAC, etc.) se definen en los scripts Lua. Se podría buscar la forma de configurar los módulos sin tener que modificar estos scripts, y que dicha configuración se replique de manera fácil en todos los nodos.

En el trabajo de la sección 2.7.4.5 se utiliza el CSI con un modelo de propagación, obteniendo una precisión menor a 0,5 m. A su vez, en el trabajo mencionado en 2.7.2.2 se utiliza el CSI para utilizar AoA, con una precisión de 0,4 m. Por lo anterior, resultaría interesante poder obtener el CSI para probar si con éste se logra un mejor desempeño que con el RSSI. Se podría, entonces, evaluar la posibilidad de utilizar hardware que permita obtener el CSI.

Se podría estudiar la posibilidad de definir más de un modelo de propagación según la distancia. Para esto, se podrían definir franjas de valores de RSSI. Para cada franja definida se podría calcular un valor diferente de $RSSI_0$, d_0 y n para la ecuación 2.4. Es decir, para la franja i de RSSI se podría definir los parámetros $RSSI_{0,i}$, $d_{0,i}$ y n_i. La primera franja correspondería a los valores utilizados en el presente proyecto, que corresponden a $d_0 = 1$ m.

A partir de Bluetooth 5.1, el PDU puede contener el campo *Constant Tone Extension* (CTE), con información del ángulo de partida (emisor) o de llegada (receptor) de la señal, para la utilización de AoA. La ventaja de esto es que está incluido en la especificación de Bluetooth, pero requiere hardware que lo soporte (múltiples antenas). Se podría evaluar la posibilidad de utilizar hardware que soporte esta capacidad de BLE, o de investigar la existencia de trabajos que utilicen el campo CTE para determinar su precisión en la localización.

En la visualización de la ubicación calculada (en el mapa) con Trilateración, se vio que muchas veces la ubicación real del cliente quedaba "entre" la ubicación calculada con los datos de la interfaz Wi-Fi y la calculada con los datos de la interfaz Bluetooth. Si esto se diera para la mayoría de los casos, al realizar el promedio de las ubicaciones calculadas, el error global obtenido podría ser menor. Esto no se verificó a partir de los datos, pero se podría evaluar la realización de las modificaciones necesarias para calcular este promedio (promedio de las coordenadas de cada ubicación calculada), y verificar si el error en ese caso mejora al error obtenido para Wi-Fi (en Trilateración, sólo con Wi-Fi se obtuvo la mejor precisión).

Bibliografía

[1] Amazon Web Services, Inc. *FreeRTOS*. URL: https://www.freertos.org/ (visitado 03-02-2021).

[2] *AnyPlace*. URL: https://anyplace.cs.ucy.ac.cy/ (visitado 14-05-2020).

[3] Apple Inc. *Use private Wi-Fi addresses in iOS 14, iPadOS 14, and watchOS 7*. 5 de nov. de 2020. URL: https://support.apple.com/en-us/HT211227 (visitado 23-01-2021).

[4] Roshan Ayyalasomayajula, Deepak Vasisht y Dinesh Bharadia. «BLoc: CSI-Based Accurate Localization for BLE Tags». En: *Proceedings of the 14th International Conference on Emerging Networking EXperiments and Technologies*. CoNEXT '18. Heraklion, Greece: Association for Computing Machinery, 2018, págs. 126-138. ISBN: 9781450360807. DOI: 10.1145/3281411.3281428. URL: https://doi.org/10.1145/3281411.3281428.

[5] P. Bahl y V. N. Padmanabhan. «RADAR: an in-building RF-based user location and tracking system». En: *Proceedings IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (Cat. No.00CH37064)*. Vol. 2. 2000, 775-784 vol.2. DOI: 10.1109/INFCOM.2000.832252.

[6] Juan Barrios. *La matriz de confusión y sus métricas*. 26 de jul. de 2019. URL: https://www.juanbarrios.com/la-matriz-de-confusion-y-sus-metricas/ (visitado 31-01-2021).

[7] Jacob Biehl y col. «LoCo: A ready-to-deploy framework for efficient room localization using Wi-Fi». En: *UbiComp 2014 - Proceedings of the 2014*

ACM International Joint Conference on Pervasive and Ubiquitous Computing (sep. de 2014), págs. 183-187. DOI: 10.1145/2632048.2636083.

[8] *Bluetooth Core Specification. Version 5.2.* Bluetooth Special Interest Group (SIG). URL: https://www.bluetooth.com/.

[9] *Bluetooth Core Specification. Version 5.2. Volumen 2, Parte B, Sección 6: Paquete BR/EDR.* Bluetooth Special Interest Group (SIG). URL: https://www.bluetooth.com/.

[10] *Bluetooth Core Specification. Version 5.2. Volumen 6, Parte B, Sección 2: Paquete BLE.* Bluetooth Special Interest Group (SIG). URL: https://www.bluetooth.com/.

[11] Philipp Bolliger. «Redpin - Adaptive, Zero-Configuration Indoor Localization through User Collaboration». En: *Proceedings of the First ACM International Workshop on Mobile Entity Localization and Tracking in GPS-Less Environments*. MELT '08. San Francisco, California, USA: Association for Computing Machinery, 2008, págs. 55-60. ISBN: 9781605581897. DOI: 10.1145/1410012.1410025. URL: http://redpin.org/.

[12] Antonio Bracco, Federico Grunwald y Agustn Navcevich. «Localización Indoor Basada en Wi-Fi». Facultad de Ingeniería, Universidad de la República, 1 de mayo de 2019.

[13] Raffaele Bruno y Franca Delmastro. «Design and Analysis of a Bluetooth-Based Indoor Localization System». En: vol. 2775. Sep. de 2003, págs. 711-725. DOI: 10.1007/978-3-540-39867-7_66.

[14] *ChirpStack Application Server*. URL: https://www.chirpstack.io/application-server/ (visitado 26-01-2021).

[15] *ChirpStack Gateway Bridge*. URL: https://www.chirpstack.io/gateway-bridge/ (visitado 26-01-2021).

[16] *ChirpStack Network Server*. URL: https://www.chirpstack.io/network-server/ (visitado 26-01-2021).

[17] *ChirpStack, open-source LoRaWAN Network Server stack*. URL: https://www.chirpstack.io/.

[18] Cisco Systems, Inc. *Cisco Meraki*. URL: https://documentation.meraki.com/ (visitado 11-02-2021).

[19] Waltenegus Dargie y Christian Poellabauer. *Fundamentals of Wireless Sensor Networks: Theory and Practice*. Ene. de 2011. DOI: 10.1002/9780470666388.

[20] Federico Detta. «Aplicacion de IoT con diversas tecnologías inalámbricas». Facultad de Ingeniería, Universidad de la República, 29 de feb. de 2020.

[21] J. J. M. Diaz y col. «Bluepass: An indoor Bluetooth-based localization system for mobile applications». En: *The IEEE symposium on Computers and Communications*. 2010, págs. 778-783. DOI: 10.1109/ISCC.2010.5546506.

[22] Niklas Donges. *A complete guide to the random forest algorithm*. 16 de jun. de 2020. URL: https://builtin.com/data-science/random-forest-algorithm (visitado 19-01-2021).

[23] Dragino Technology Co., LTD. *LPS8 Indoor LoRaWAN Gateway*. URL: https://www.dragino.com/products/lora-lorawan-gateway/item/148-lps8.html (visitado 23-01-2021).

[24] Espressif Systems. *ESP32 Series of Modules*. URL: https://www.espressif.com/en/products/modules/esp32 (visitado 23-01-2021).

[25] Espressif Systems. *Espressif IoT Development Framework (ESP-IDF)*. URL: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/ (visitado 11-02-2021).

[26] Zahid Farid, Rosdiadee Nordin y Mahamod Ismail. «Recent Advances in Wireless Indoor Localization Techniques and System». En: *Journal of Computer Networks and Communications* 2013 (sep. de 2013), pág. 185138. ISSN: 2090-7141. DOI: 10.1155/2013/185138. URL: https://doi.org/10.1155/2013/185138.

[27] Zahid Farid y col. «A WLAN Fingerprinting Based Indoor Localization Technique via Artificial Neural Network». En: (jul. de 2019), pág. 157.

[28] *FIND3*. URL: https://www.internalpositioning.com/doc/ (visitado 16-01-2021).

[29] *FIND3 API*. URL: https://www.internalpositioning.com/doc/api.md (visitado 25-01-2021).

[30] The Apache Software Foundation. *Commons Math: The Apache Commons Mathematics Library*. URL: https://commons.apache.org/proper/commons-math/ (visitado 31-01-2021).

[31] Yoav Freund y Robert E Schapire. «A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting». En: *J. Comput. Syst. Sci.* 55.1 (ago. de 1997), págs. 119-139. ISSN: 0022-0000. DOI: 10.1006/jcss.1997.1504. URL: https://doi.org/10.1006/jcss.1997.1504.

[32] Gerardo Gomez, Francisco Crizul. *RestApiLocalizacion - GitLab - FING*. URL: https://gitlab.fing.edu.uy/francisco.crizul/restapilocalizacion (visitado 23-01-2021).

[33] *GitHub - Trilateration*. URL: https://github.com/lemmingapex/trilateration (visitado 16-01-2021).

[34] Jon Gjengset y col. «Phaser: Enabling Phased Array Signal Processing on Commodity WiFi Access Points». En: *Proceedings of the 20th Annual International Conference on Mobile Computing and Networking*. MobiCom '14. Maui, Hawaii, USA: Association for Computing Machinery, 2014, págs. 153-164. ISBN: 9781450327831. DOI: 10.1145/2639108.2639139. URL: https://doi.org/10.1145/2639108.2639139.

[35] F. J. Gonzalez-Castano y J. Garcia-Reinoso. «Bluetooth location networks». En: *Global Telecommunications Conference, 2002. GLOBECOM '02. IEEE*. Vol. 1. 2002, 233-237 vol.1. DOI: 10.1109/GLOCOM.2002.1188075.

[36] Google LLC. *Eddystone format*. 28 de oct. de 2019. URL: https://developers.google.com/beacons/eddystone (visitado 04-02-2021).

[37] Google LLC. *Privacy: MAC Randomization*. 28 de oct. de 2020. URL: https://source.android.com/devices/tech/connect/wifi-mac-randomization (visitado 23-01-2021).

[38] I. Guvenc y col. «Enhancements to RSS Based Indoor Tracking Systems Using Kalman Filters». En: *In GSPx & International Signal Processing Conference*. 2003.

[39] Henri P. Gavin. *The Levenberg-Marquardt algorithm for nonlinear least squares curve-fitting problems*. 2020. URL: http://people.duke.edu/~hpgavin/ce281/lm.pdf (visitado 31-01-2021).

[40] F. Ijaz y col. «Indoor positioning: A review of indoor ultrasonic positioning systems». En: *2013 15th International Conference on Advanced Communications Technology (ICACT)*. 2013, págs. 1146-1150.

[41] Apple Inc. *iBeacon*. URL: https://developer.apple.com/ibeacon/ (visitado 30-12-2020).

[42] Bashima Islam y col. «Rethinking ranging of unmodified BLE peripherals in smart city infrastructure». En: *Proceedings of the 9th ACM Multimedia Systems Conference*. ACM, jun. de 2018. DOI: 10.1145/3204949.3204950. URL: https://doi.org/10.1145%2F3204949.3204950.

[43] Manikanta Kotaru y col. «SpotFi: Decimeter Level Localization Using WiFi». En: *SIGCOMM Comput. Commun. Rev.* 45.4 (ago. de 2015), págs. 269-282. ISSN: 0146-4833. DOI: 10.1145/2829988.2787487. URL: https://doi.org/10.1145/2829988.2787487.

[44] P. Krishnan y col. «A system for LEASE: location estimation assisted by stationary emitters for indoor RF wireless networks». En: *IEEE INFOCOM 2004*. Vol. 2. 2004, 1001-1011 vol.2.

[45] Pavel Kriz, Filip Maly y Tomáš Kozel. «Improving Indoor Localization Using Bluetooth Low Energy Beacons». En: *Mobile Information Systems* 2016 (abr. de 2016), págs. 1-11. DOI: 10.1155/2016/2083094.

[46] Swarun Kumar y col. «Accurate indoor localization with zero start-up cost». En: *Proceedings of the Annual International Conference on Mobile Computing and Networking, MOBICOM* (sep. de 2014). DOI: 10.1145/2639108.2639142.

[47] Ye-Sheng Kuo y col. «Luxapose: Indoor Positioning with Mobile Phones and Visible Light». En: *Proceedings of the 20th Annual International Conference on Mobile Computing and Networking*. MobiCom '14. Maui, Hawaii, USA: Association for Computing Machinery, 2014, págs. 447-458. ISBN: 9781450327831. DOI: 10.1145/2639108.2639109. URL: https://doi.org/10.1145/2639108.2639109.

[48] James F. Kurose y Keith W. Ross. *Redes de computadoras: Un enfoque descendente*. 5.ª ed. Pearson Education, Inc., 2010. ISBN: 978-84-7829-133-5.

[49] H. Lim y col. «Zero-Configuration, Robust Indoor Localization: Theory and Experimentation». En: *Proceedings IEEE INFOCOM 2006. 25TH IEEE International Conference on Computer Communications*. 2006, págs. 1-12. DOI: 10.1109/INFOCOM.2006.223.

[50] H. Liu y col. «Survey of Wireless Indoor Positioning Techniques and Systems». En: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 37.6 (2007), págs. 1067-1080. DOI: 10.1109/TSMCC.2007.905750.

[51] K. Liu, X. Liu y X. Li. «Guoguo: Enabling Fine-Grained Smartphone Localization via Acoustic Anchors». En: *IEEE Transactions on Mobile Computing* 15.5 (2016), págs. 1144-1156. DOI: 10.1109/TMC.2015.2451628.

[52] LoRa Alliance. *RP2-1.0.1 LoRaWAN Regional Parameters*. URL: https://lora-alliance.org/resource_hub/rp2-101-lorawan-regional-parameters-2/ (visitado 02-02-2021).

[53] LoRa Alliance. *What is LoRaWAN Specification*. URL: https://lora-alliance.org/about-lorawan/ (visitado 10-02-2021).

[54] *LoRa gateway definitions*. URL: https://github.com/brocaar/chirpstack-api/blob/master/protobuf/gw/gw.proto (visitado 16-01-2021).

[55] *LoRaWAN specification*. URL: https://lora-alliance.org/lorawan-for-developers/.

[56] The Things Network. *Limitations: data rate, packet size, 30 seconds uplink and 10 messages downlink per day Fair Access Policy [guidelines]*. URL: https://www.thethingsnetwork.org/forum/t/limitations-data-rate-packet-size-30-seconds-uplink-and-10-messages-downlink-per-day-fair-access-policy-guidelines/1300 (visitado 24-01-2021).

[57] The Things Network. *The Things Gateway*. URL: https://www.thethingsnetwork.org/docs/gateways/gateway/ (visitado 23-08-2020).

[58] Nils J. Nilsson. *MACHINE LEARNING: AN EARLY DRAFT OF A PROPOSED TEXTBOOK*. Robotics Laboratory, Department of Computer Science, StanfordUniversity, Stanford, CA, 1998. URL: http://robotics.stanford.edu/people/nilsson/mlbook.html.

[59] Sebastian Raschka. *Python Machine Learning*. Packt Publishing Ltd., 2015.

[60] Alessandro Redondi y Matteo Cesana. «Building up Knowledge through Passive WiFi Probes». En: *Computer Communications* 117 (dic. de 2017). DOI: 10.1016/j.comcom.2017.12.012.

[61] S. Sadowski y P. Spachos. «RSSI-Based Indoor Localization With the Internet of Things». En: *IEEE Access* 6 (2018), págs. 30149-30161. DOI: 10.1109/ACCESS.2018.2843325.

[62] R. Schmidt. «Multiple emitter location and signal parameter estimation». En: *IEEE Transactions on Antennas and Propagation* 34.3 (1986), págs. 276-280. DOI: 10.1109/TAP.1986.1143830.

[63] scikit-learn developers. *Linear and Quadratic Discriminant Analysis*. URL: https://scikit-learn.org/stable/modules/lda_qda.html (visitado 20-01-2021).

[64] scikit-learn developers. *Naive Bayes*. URL: https://scikit-learn.org/stable/modules/naive_bayes.html (visitado 20-01-2021).

[65] SEMTECH. *An In-depth look at LoRaWAN Class A Devices*. URL: https://lora-developers.semtech.com/library/tech-papers-and-guides/lorawan-class-a-devices/ (visitado 24-01-2021).

[66] SEMTECH. *What are LoRa and LoRaWAN?* URL: https://lora-developers.semtech.com/library/tech-papers-and-guides/lora-and-lorawan/ (visitado 10-02-2021).

[67] SparkFun Electronics. *Sparkfun SPX-14893 (ESP32 LoRa 1-Channel Gateway)*. URL: https://www.sparkfun.com/products/retired/14893 (visitado 23-01-2021).

[68] *Ubuntu 18.04*. URL: https://releases.ubuntu.com/18.04/ (visitado 16-01-2021).

[69] *UDP Packet Forwarder*. URL: https://github.com/Lora-net/packet_forwarder/blob/master/PROTOCOL.TXT.

[70] Deepak Vasisht, Swarun Kumar y Dina Katabi. «Decimeter-Level Localization with a Single WiFi Access Point». En: *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16)*. Santa Clara, CA: USENIX Association, mar. de 2016, págs. 165-178. ISBN: 978-1-931971-29-4. URL: https://www.usenix.org/conference/nsdi16/technical-sessions/presentation/vasisht.

[71] *VirtualBox*. URL: https://www.virtualbox.org/wiki/Downloads (visitado 16-01-2021).

[72] VirtualBox. *Chapter 6. Virtual Networking*. URL: https://www.virtualbox.org/manual/ch06.html (visitado 21-02-2021).

[73] X. Wang y col. «CSI-Based Fingerprinting for Indoor Localization: A Deep Learning Approach». En: *IEEE Transactions on Vehicular Technology* 66.1 (2017), págs. 763-776. DOI: 10.1109/TVT.2016.2545523.

[74] Y. Wang y col. «Bluetooth positioning using RSSI and triangulation methods». En: *2013 IEEE 10th Consumer Communications and Networking Conference (CCNC)*. 2013, págs. 837-842. DOI: 10.1109/CCNC.2013.6488558.

[75] Whitecat. *LuaRTOS*. URL: https://github.com/whitecatboard/Lua-RTOS-ESP32 (visitado 23-01-2021).

[76] K. Wu y col. «CSI-Based Indoor Localization». En: *IEEE Transactions on Parallel and Distributed Systems* 24.7 (2013), págs. 1300-1309. DOI: 10.1109/TPDS.2012.214.

[77] Jie Xiong y Kyle Jamieson. «ArrayTrack: A Fine-Grained Indoor Location System». En: abr. de 2013, págs. 71-84.

[78] Jie Xiong, Karthikeyan Sundaresan y Kyle Jamieson. «ToneTrack: Leveraging Frequency-Agile Radios for Time-Based Indoor Wireless Localization». En: *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*. MobiCom '15. Paris, France: Association for Computing Machinery, 2015, págs. 537-549. ISBN: 9781450336192. DOI: 10.1145/2789168.2790125. URL: https://doi.org/10.1145/2789168.2790125.

[79] Zheng Yang, Zimu Zhou y Yunhao Liu. «From RSSI to CSI: Indoor Localization via Channel Response». En: *ACM Comput. Surv.* 46.2 (dic. de 2013). ISSN: 0360-0300. DOI: 10.1145/2543581.2543592. URL: https://doi.org/10.1145/2543581.2543592.

[80] Moustafa Youssef y Ashok Agrawala. «The Horus WLAN Location Determination System». En: *Proceedings of the 3rd International Conference on Mobile Systems, Applications, and Services*. MobiSys '05. Seattle, Washington: Association for Computing Machinery, 2005, págs. 205-218. ISBN: 1931971315. DOI: 10.1145/1067170.1067193. URL: https://doi.org/10.1145/1067170.1067193.

[81] F. Zafari, A. Gkelias y K. K. Leung. «A Survey of Indoor Localization Systems and Technologies». En: *IEEE Communications Surveys Tutorials* 21.3 (2019), págs. 2568-2599. DOI: 10.1109/COMST.2019.2911558.

[82] F. Zafari e I. Papapanagiotou. «Enhancing iBeacon Based Micro-Location with Particle Filtering». En: *2015 IEEE Global Communications Conference (GLOBECOM)*. 2015, págs. 1-7. DOI: 10.1109/GLOCOM.2015.7417504.

[83] Faheem Zafari. «iBeacon Based Proximity and Indoor Localization System». Tesis de mtría. Purdue University, 2016. URL: https://docs.lib.purdue.edu/dissertations/AAI10146475/.

Lista de siglas

Lista de siglas

ANN Red Neuronal Artificial 45

AP Access Point 16, 17, 18, 33, 34, 36, 41, 43, 45, 48, 50, 97

BLE Bluetooth Low Energy 18, 19, 25, 40, 41, 58, 59, 60, 71, 72, 73, 74, 102

BR/EDR Basic Rate/Enhanced Data Rate 18, 58, 59, 60, 71, 72

CDF Función de Distribución Acumulada 32, 33, 34, 35, 36, 37, 38, 40, 41, 43, 45, 77, 80, 81, 82, 83, 86, 87, 90, 94, 116, 117, 118

CFR Respuesta de Frecuencia del Canal 6

CIR Respuesta de Impulso del Canal 6

CSI Información de Estado del Canal 6, 11, 34, 36, 40, 41, 43, 55, 97

DBL Localización Basada en Dispositivo 5, 32, 50, 55

ISM Industriales, Científicas y Médicas 16, 18, 28

IoT Internet de las Cosas 16, 46

LED Diodo Emisor de Luz 26, 27

LoS Line of Sight 9, 10, 27, 28, 34, 41, 100

MAC Control de Acceso al Medio 18, 25

MBL Localización Basada en Monitor 5, 32, 41, 47, 50, 51, 55

MLP Perceptrón Multi-Capa 45

NLoS Non-Line of Sight 34, 41

RF Radio Frequency 25, 27

RFID Dispositivo de Identificación por Radio Frecuencia 25, 26

RN Nodo de Referencia 16, 27, 38, 39, 45, 46

RSS Received Signal Strength 6, 8, 16, 18

RSSI Received Signal Strength Indicator 6, 8, 11, 12, 18, 34, 38, 39, 40, 41, 45, 46, 48

SVM Support Vector Machine 36, 52

UHF Frecuencia Ultra Alta 26

UWB Banda Ultra Ancha - Ultra Wideband 26

VLC Comunicación por Luz Visible 26

WSN Redes de Sensores Inalámbricos 25

kNN k-Nearest Neighbors 36, 38, 46, 50, 52

Lista de figuras

APÉNDICES

Apéndice A

Modificaciones realizadas a LuaRTOS

Para este trabajo fue necesario agregar funcionalidades a la API Lua de LuaRTOS (sección 2.6.3). Estas funciones se agregan en la capa 1 (*Top Layer)*, y la implementación se realiza en la capa 2 (*Middle Layer*), haciendo uso del framework ESP-IDF (capa 3, *Bottom Layer*), que interactúa directamente con el hardware. Las funcionalidades agregadas corresponden a los módulos *bt* (Bluetooth) y *net.wf* (Wi-Fi) de la API. También se hizo una modificación en el módulo *lora* (LoRa y LoRaWAN), para que trabaje en el plan de frecuencias AU915-928 MHz.

Los archivos modificados para cada módulo son:

- Bluetooh (*bt*):
  - Capa 1: *components/lua/modules/bluetooth/bluetooth.c.*
  - Capa 2: *components/sys/drivers/bluetooth.c* y *components/sys/drivers/bluetooth.h.*
- Wi-Fi (*net.wf*):
  - Capa 1: *components/lua/modules/net/net_wifi.inc.*
  - Capa 2: *components/sys/drivers/wifi.c.*
- LoRa (*lora*): Capa 2: *components/lora/node/lmic/lorabase.h* y *components/lora/gateway/single_channel/gateway.c.*

Se partió del código fuente de LuaRTOS con las modificaciones realizadas en [20], proyecto de grado de un año anterior en el que se agregó la función *radar* al módulo *net.wf* de la API de LuaROTS. El código fuente inicial fue descargado de https://github.com/fdetta/Lua-RTOS-ESP32. El código fuente

con las modificaciones realizadas para el presente proyecto queda disponible en https://gitlab.fing.edu.uy/gerardo.gomez.detjen/Lua-RTOS-ESP32.

Para hacer referencia a las funciones agregadas a la API Lua se dirá "*funciones Lua*", y serán escritas en *itálica*. De igual forma, se dirá "*funciones de capa 1*" y "*funciones de capa 2*" para hacer referencia a las funciones implementadas en la capa 1 y la capa 2, respectivamente, de cada módulo, y serán escritas con la fuente `Courier`. Para indicar los tipos de los parámetros y de retorno de las funciones Lua se usará la sintaxis de *C*.

Algo en común a la definición de todas las funciones de capa 1 es que reciben como único parámetro un puntero al stack de Lua (`lua_State`). Las funciones de capa 1 toman de este stack los parámetros de entrada (pasados a la función Lua) e introducen en éste los parámetros de salida (lo que devuelve la función Lua).

A.1. Bluetooth (bt)

A continuación se listan las funciones Lua agregadas, y se indican las funciones de capa 1 y capa 2 que la implementan.

***char* bt.getbdaddr()*:** Devuelve la BD_ADDR del dispositivo.
Se implementa en las funciones `lbt_get_bdaddr(lua_State*)` de capa 1 y `bt_get_bdaddr(uint8_t *bd_addr)` de capa 2.

***void bt.attachdual(uint8_t bt_mode, uint8_t scan_bredr_mode, char *dev_bredr_name)*:** Inicializa el módulo Bluetooth en el modo deseado. Es similar a la función Lua existente *bt.attach(...)*, pero permite indicar si se quiere utilizar el módulo Bluetooth en el modo BLE, BR/EDR o ambos a la vez, liberando la memoria del modo no utilizado. Parámetros de entrada:

- *bt_mode*: Modo (0: BLE, 1: BR/EDR, 2: Dual).
- *scan_bredr_mode*: Mode de scan para BR/EDR:
  - 0: *Neither discoverable nor connectable*. No permite ser descubierto ni acepta conexiones por otros dispositivos BR/EDR.
  - 1: *Connectable but not discoverable*. Acepta conexiones de otros dispositivos BR/EDR pero no permite ser descubierto.
  - 2: *Both discoverable and connectable*. Permite ser descubierto y acepta conexiones de otros dispositivos BR/EDR.

- *dev_bredr_name*: Nombre del dispositivo si está habilitado el modo BR/EDR.

Se implementa en la funciones `lbt_attach_dual(lua_State*)` de capa 1 y `bt_setup_dual(uint8_t bt_mode, uint8_t scan_bredr_mode, char *dev_bredr_name)` de capa 2. En la función de capa 2 se inicializan los componentes y los controladores de Bluetooth, y se configuran las funciones de callback que son utilizadas en los scan de BLE y en los descubrimientos de BR/EDR. Las funciones de callback (una para BLE y otra para BR/EDR) son invocadas por ESP-IDF por cada paquete Bluetooth recibido, y éstas comunican a la capa 1 enviando los datos a una cola de mensajes, que es leída desde una tarea que ejecuta de forma concurrente (la cola de mensajes y la tarea son creadas en esta función). Para BR/EDR se utilizó la misma cola de mensajes que ya se utilizaba para BLE. Esto facilita la sincronización en la capa 1 y permite utilizar la misma función de callback Lua, que se explicará más adelante.

***void bt.freemem()*:** Libera toda la memoria del componente Bluetooth. Si se ejecuta esta función, es necesario reiniciar la placa para poder utilizar el componente Bluetooth. Como pre-condición, el módulo no puede haber sido inicializado. Se implementa en las funciones `lbt_free_mem(lua_State*)` de capa 1 y `bt_free_mem()` de capa 2.

***void bt.scan.setLeDuration(uint32_t ble_duration, uint16_t scan_interval, uint16_t scan_window)*:** Configura los tiempos para el scan LE. Parámetros de entrada:

- *ble_duration*: Tiempo en segundos que durará un un scan LE desde que es iniciado. Si es 0, el scan no terminará salvo que se invoque la función para detener el scan, que se explicará luego.
- *scan_interval*: Intervalo de tiempo entre el comienzo de una búsqueda LE y la siguiente. El rango válido es [0x0004, 0x4000], y su valor es multiplicado por 0,625 ms (2,5 ms a 10,24 s). En caso de no ser asignado, su valor es 0x50.
- *scan_window*: Duración de la búsqueda LE, dentro de *scan_interval*. Se debe cumplir $scan\_window \leq scan\_interval$. El rango válido es [0x0004, 0x4000], y su valor es multiplicado por 0,625 ms (2,5 ms a 10,24 s). En caso de no ser asignado, su valor es 0x30.

Es decir que, una vez iniciado un scan BLE, durante *ble_duration* segundos, se ejecutará un scan de $scan\_window \times 0,625$ ms cada $scan\_interval \times 0,625$ ms. Esta función se implementa en las funciones `lbt_scan_set_le_duration(lua_State*)` de capa 1 y `bt_scan_set_le_duration(uint32_t ble_duration, uint16_t scan_interval, uint16_t scan_window)` de capa 2.
En la versión original de LuaRTOS, estos valores están fijos y no permite modificarlos ($ble\_duration = 0$, $scan\_interval =$ 0x50 y $scan\_window =$ 0x3).

***void bt.scan.setBrEdrDuration(uint8_t bredr_inq_len)*:** Configura la duración del descubrimiento BR/EDR, que será $bredr\_inq\_len \times 1,28$ segundos. Se implementa en las funciones `lbt_scan_set_bredr_duration(lua_State*)` de capa 1 y `bt_scan_set_bredr_duration(uint8_t bredr_inq_len)` de capa 2.

***void bt.scan.setMinRssi(int min_rssi)*:** Permite configurar un valor de RSSI mínimo a ser considerado en el scan LE y el descubrimiento BR/EDR. Los paquetes recibidos en una señal con un RSSI menor al mínimo serán descartados. 0 indica que no hay mínimo. Se implementa en las funciones `lbt_scan_set_min_rssi(lua_State*)` de capa 1 y `bt_scan_set_min_rssi(int min_rssi)` de capa 2.

***void bt.scan.startLe(callback)*:** Inicia un scan LE. Por cada paquete recibido, se invocará a la función Lua *callback(data)*, que se explicará más adelante. Se implementa en las funciones `lbt_scan_start_le(lua_State*)` de capa 1 y `bt_scan_start_le(bt_scan_callback_t cb, int cb_id)` de capa 2. La función `bt_scan_start_le`, de capa 2, recibe un puntero a una función de callback de capa 1, que es a través de la cual la capa 2 envía los datos de los paquetes Bluetooth recibidos a la capa 1. A su vez, la función de callback de capa 1, envía la información al hilo de Lua a través de la función Lua que se pasó como parámetro a *bt.scan.startLe*.
Esta función es similar a la función Lua pre-existente *bt.scan.start(callback)*, pero no utiliza las estructuras de datos de la función original y considera una duración de scan no indefinida ($ble\_duration \neq 0$).
El scan LE puede ser detenido con la funicón Lua pre-existente *bt.scan.stop()*.

***void bt.scan.startBrEdr(callback)*:** Inicia un descubrimiento BR/EDR.

Por cada paquete recibido, se invocará a la función Lua *callback(data)*, que se explicará más adelante. Se implementa en las funciones `lbt_scan_start_bredr(lua_State*)` de capa 1 y `bt_scan_start_bredr(bt_scan_callback_t cb, int cb_id)` de capa 2. Funciona con el mismo esquema de funciones de callback que *bt.scan.startLe*, pero la función de callback de capa 2 es exclusiva de BR/EDR.

***void bt.scan.stopBrEdr()*:** Permite detener un descubrimiento BR/EDR antes de que haya transcurrido el tiempo configurado con *bt.scan.setBrEdrDuration*. Se implementa en las funciones `lbt_scan_stop_bredr(lua_State*)` de capa 1 y `bt_scan_stop_bredr()` de capa 2.

***bool bt.isLeScanning()*:** Indica si se está ejecutando un scan LE (previamente iniciado). Se implementa en las funciones `lbt_is_le_scanning(lua_State*)` de capa 1 y `bt_is_le_scanning(bool *isLeScanning)` de capa 2.

***bool bt.isBrEdrScanning()*:** Indica si se está ejecutando un descubrimiento BR/EDR (previamente inicado). Se implementa en las funciones `lbt_is_bredr_scanning(lua_State*)` de capa 1 y `bt_is_bredr_scanning(bool *isBrEdrScanning)` de capa 2.

`struct bt_adv_frame_t`**:** Este estructurado (definido en *components/sys/drivers/bluetooth.h*) es el que se utiliza para que la capa 2 le envíe los datos de los paquetes Bluetooth a la capa 1. Fue necesario agregar a este estructurado un campo para la BD_ADDR del dispositivo escaneado (BLE) o descubierto (BR/EDR):

```
uint8_t bd_addr[ESP_BD_ADDR_LEN]; // BD_ADDR del
    ↪ dispositivo emisor
```

Se usa el mismo estructurado para BLE y para BR/EDR, ya que la función de callback de capa 1 es la misma para ambos modos.

`enum bt_adv_frame_type_t`**:** Indica el tipo del frame del paquete Bluetooth. LuaRTOS reconoce si un paquete Bluetooth corresponde a uno de los frames UID o URL de Eddystone, definidos por Google para BLE [36]. Se agregó un

nuevo valor al enumerado para indicar que se trata de un paquete BR/EDR:

BTAdvBrEdrInqRsp = 3 *// BR/EDR Inquiry Response*

***bt.frameType*:** Es un enumerado en la API Lua, para conocer el tipo del frame del paquete Bluetooth. Se corresponde con el enumerado anterior. Se agregaron los valores *OtherAdvertisement*, para indicar que es un paquete BLE (no Eddystone) y *BR_EDRInqRsp* para indicar que es un paquete BR/EDR.

`void gap_cb(esp_gap_ble_cb_event_t event, esp_ble_gap_cb_param_t* param):` Es la función de callback de capa 2 correspondiente a BLE (invocada por ESP-IDF). Ya estaba implementada, pero se le hicieron las modificaciones necesarias para:

- Obtener la BD_ADDR del dispositivo emisor.
- Considerar la duración máxima del scan BLE, *ble_duration* (ya que, al configurar los parámetros de scan, se dispara un evento capturado en esta función).
- Manejar el estado de si existe un scan BLE en ejecución.
- Considerar el valor RSSI mínimo.
- Manejar el evento cuando el scan finalizó por tiempo ($ble\_duration \neq 0$)

`void bt_bredr_cb(esp_bt_gap_cb_event_t event, esp_bt_gap_cb_param_t *param):` Es la función de callback de capa 2 correspondiente a BR/EDR. Se agregó para manejar los eventos BR/EDR disparados por ESP-IDF.

Función de callback Lua: Al iniciar un scan BLE o un descubrimiento BR/EDR, para poder obtener los datos de los paquetes BLE capturados en las capas inferiores, es necesario definir una función Lua de callback. La definición de esta función es "*function scan_callback(data)*", donde *data* es un estructurado, con los siguientes campos:

type: int -- Tipo del frame (*bt.frameType.BR_EDRInqRsp*, ...)
rssi: int -- RSSI
raw: string -- Campo *AdvData* del Payload del paquete BLE
len: int -- Largo de *raw*
bdaddr: string -- BD_ADDR del emisor
... -- Otros campos cuando es Eddystone

A.2. Wi-Fi (net.wf)

A continuación se listan las funciones Lua agregadas, y se indican las funciones de capa 1 y capa 2 que la implementan.

***char* net.wf.getmacaddr()*:** Devuelve la dirección MAC de la interfaz Wi-Fi del modo Station. Se implementa en las funciones `lwifi_get_mac_address(lua_State*)` de capa 1 y `wifi_get_mac_address(uint8_t *mac)` de capa 2. Se implementó esta función ya que es más directa que la que ofrece la API Lua, en la que es necesario configurar e iniciar el módulo Wi-Fi, obtener el estado, y de este último la dirección MAC. Mediante esta nueva función, se obtiene directamente del hardware, a través del framework ESP-IDF.

***table net.wf.scanch(bool table, uint8_t channel)*:** Realiza un scan Wi-Fi en el canal especificado en *channel* (0 para todos los canales). El parámetro *table* indica si se quiere obtener la información en un estructurado (*true*), o si se quiere imprimir en consola (*false*) la información de los AP que contestaron a los mensajes *Probe request*. Se implementa en las funciones `lwifi_scan_channel(lua_State*)` de capa 1 y `wifi_scan_channel(uint8_t channel, uint16_t *count, wifi_ap_record_t **list)` de capa 2. Esta función es igual a la función Lua *net.wf.scan* pre-existente, pero se agrega la posibilidad de enviar los mensajes *Probe Request* a un único canal.
Se hizo esta función con el fin de poder emitir tramas Wi-Fi en un único canal, sin la necesidad de haber establecido una conexión a una red.

***net.wf.radar(bool table, uint8_t secsChannel, int8_t minRssi, bool onlyData, uint8_t channel, uint8_t rssiPolicy)*:** Función agregada en [20] (proyecto de grado de un año anterior). Cuenta los dispositivos Wi-Fi que se tiene al alcance, con la posibilidad de obtener la lista de las direcciones MAC detectadas, junto con el RSSI asociado a cada una. A esta función se le agregó la posibilidad de detectar los dispositivos en un único canal, y también elegir la forma en que se devolverá el RSSI de cada dispositivo detectado. Parámetros de entrada:

- *table*: Indica si devolver la lista de direcciones MAC junto con el RSSI (*true*) o si devolver únicamente la cantidad de dispositivos detectados (*false*).

- *secsChannel*: Cantidad de segundos que escaneará cada canal.
- *minRssi*: Valor de RSSI mínimo a considerar (tramas obtenidas en una señal con un RSSI menor a este valor, serán descartadas).
- *onlyData*: Indica si tomar únicamente tramas de clientes conectados (*true*) o todos (*false*). En caso de ser *false*, no sólo se detectarán clientes sin conectarse a una red inalámbrica, sino que también se detectarán las tramas de tramas de administración enviadas por los AP a otros dispositivos en una BSS (i.e., tramas con las flags *DesdeAP* y *HaciaAP* en 0).
- *channel*: Canal a escanear. 0 indica todos los canales.
- *rssiPolicy*: Política de RSSI. Durante el scan se reciben muchas tramas de cada cliente. Para cada cliente, el RSSI devuelto depende del valor de este parámetro:
  - 0: Primer valor obtenido del cliente
  - 1: Último valor obtenido del cliente
  - 2: Promedio de los valores obtenidos para el ciente

Se implementa en las funciones `lwifi_radar(lua_State*)` de capa 1 y `wifi_radar(int8_t *minsen, mac_t **list, uint16_t *count, int segs, u8_t onlyData, uint8_t channel, uint8_t rssi_policy)` de capa 2. Esta función es bloqueante, es decir que el hilo en que ejecuta queda bloqueado hasta que finaliza. Cada dirección MAC en la lista devuelta ocurre una única vez y el valor del RSSI asociado habrá sido calculado según el parámetro *rssiPolicy*.

A.3. LoRa (lora)

A continuación se describen las modificaciones realizadas en el módulo *lora*. Las modificaciones fueron realizadas en la capa 2 de LuaRTOS.

Archivo "*components/lora/gateway/single_channel/gateway.c*": Si bien no se usaron las placas como gateways por medio de LuaRTOS, al configurar el framework con el tipo de hardware SX1276 y para trabajar en el plan de frecuencias AU915-928 (en el framework se configura *United States - 915 Mhz*), hace que no compile. Se debe quitar la macro:

```
#error "Not supported US915"
```

y definir los arreglos:

```
static const uint8_t sf[] = {7, 8, 9, 10, 11, 12};
static const uint32_t freq[] = { 923300000, 923300000,
923300000, 923300000, 923300000, 923300000, 923300000,
923300000, 923300000 };
```

No se probó si los valores son los correctos para utilizar las placas como un gateway, simplemente se definieron para poder compilar.

Archivo "*components/lora/node/lmic/lorabase.h*": Este archivo fue necesario modificarlo para definir las frecuencias en las que se quiere que LoRa trabaje. Se agregó la macro:

```
#define MULTICANAL_AU915 1
```

y se dejaron como comentarios las líneas en las que se define la macro:

```
#define UN_CANAL_FRECUENCIA 923300000 /* Un canal: Canal Downlink
0 */
```

La idea es definir sólo una de estas macros, *MULTICANAL_AU915* con cualquier valor, para indicar que se trabajará en el plan de frecuencias AU915-928 MHz, ó *UN_CANAL_FRECUENCIA* con el valor de la frecuencia que se quiere utilizar cuando se usará LoRaWAN en un único canal. Luego, la siguiente porción de código utilizará una opción o la otra:

```
#if defined(UN_CANAL_FRECUENCIA)
// Frecuencias para un único canal:
enum { US915_125kHz_UPFBASE = UN_CANAL_FRECUENCIA,
       US915_125kHz_UPFSTEP =      0,
       US915_500kHz_UPFBASE = UN_CANAL_FRECUENCIA,
       US915_500kHz_UPFSTEP =      0,
       US915_500kHz_DNFBASE = 923300000,
       US915_500kHz_DNFSTEP =      0
};
#elif defined(MULTICANAL_AU915)
// Default frequency plan for AU 915MHz
enum { US915_125kHz_UPFBASE = 915200000,
       US915_125kHz_UPFSTEP =    200000,
       US915_500kHz_UPFBASE = 915900000,
       US915_500kHz_UPFSTEP =   1600000,
       US915_500kHz_DNFBASE = 923300000,
       US915_500kHz_DNFSTEP =    600000
```

```
};
#endif
```

Apéndice B

Configuración de la REST API de localización

En este apéndice se explicará los requerimientos, funcionamiento y configuración de la aplicación web cuyo código fuente se encuentra en el repositorio GitLab [GitLab 32, RestApiLocalizacion].

B.1. Requerimientos

Esta aplicación tiene los siguientes requerimientos:

- `Base de datos MySQL` $\geq$ `5.7`
- `Apache Tomcat 7`
- `Chirpstack Application Server 3.13`

El desarrollo se realizó usando el IDE Eclipse 2020-03

B.2. Configuración

La API REST accede a una base de datos MySQL en donde se tiene toda la configuración del sistema, además es donde se guardan las mediciones enviadas por los nodos ancla y las ubicaciones calculadas por el método seleccionado. A continuación se listan los parámetros del sistema, su descripción y los valores posibles:

Parámetro	Descripción	Valores	Algoritmo
APATAR	Apaga la tarea programada	Booleano	Ambos
FINDFAMILY	Familia principal, zona 1	Texto	Ambos
INTSEGLOC	Intervalo de loc	Numérico	Ambos
METLOC	Algoritmo Localización	Texto	Ambos
REALPOS	Posicion real	JSON	Ambos
RSSIMINWF	RSSI mínimo aceptado para procesar	Numérico	Ambos
FAMILIES	Familias auxiliares	JSON	FIND3
FINDURL	URL base FIND3	URL	FIND3
FINDWIN	Ventana de aprendizaje de FIND3	Numérico	FIND3
CANTRSSI	Cantidad máxima de RSSI para el cálculo	Numérico	Trilateración

Tabla B.1: Parámetros del sistema

- FAMILIES: En este parámetro se almacena los nombre de las familias auxiliares, con su tamaño, con la siguiente estructura JSON.

```
[
    {
        "name":"inco_zona_2_all",
        "zone":2
    },
    {
        "name":"inco_zona_3_all",
        "zone":3
    }
]
```

- REALPOS: En este parámetro se almacena los datos de la posición real del cliente, con las coordenadas “x” e “y” en metros y el nombre del location de FIND3. El parámetro tiene la siguiente estructura en JSON:

```
{
    "x":4.7,
    "y":4.5,
    "loc":"1_1_2"
}
```

- METLOC: Con este parámetro se especifica el algoritmo a utilizar, “find3” o “trilateracion”

B.3. Endpoints

- `POST /rest/lora_service`

  Este web service es utilizado para recibir mensajes de los nodos por medio de ChirpStack Application Server. El web service recibe la siguiente

estructura:

```
{
    "applicationID":"1",
    "applicationName":"PGrado",
    "deviceName":"NodoG-01-30ae-532c",
    "devEUI":"3837353730143d75",
    "rxInfo":[
        {
            "gatewayID":"840d8efffe074d90",
            "uplinkID":"9346c83c-cffc-4d0b-ac10-9936bae80a79",
            "name":"GatewayGG",
            "rssi":-23,
            "loRaSNR":10,
            "location":{
                "latitude":52,
                "longitude":5,
                "altitude":1
            }
        }
    ],
    "txInfo":{
        "frequency":923299987,
        "dr":0
    },
    "adr":false,
    "fCnt":9,
    "fPort":1,
    "data":"AUB3ZnwzMEFFQTRCRTUzMkN8MUNDQ0Q2QkExNzNBfC00MQA="
}
```

El campo "data" es la información que envían los nodos, información en formato base64 que dentro puede tener alguna de estas dos estructuras:

1. `<interfaz>|<mac_nodo>|<mac_device>|<RSSI>`

   Este datos son enviados cuando se obtiene intensidad de señal por los clientes, donde la interfaz tomas los valores wf o bt.

2. `init|<interfaz>|<mac_nodo>`

   Estos datos son enviados periódicamente por los nodos para comunicar que está "en línea"

- `POST /rest/localizacion`

  Este web service es para obtener información de los clientes localizados, recibe la siguiente estructura:

```
{
    "interfaz": "",
    "mac":"",
    "fecha": "2020-08-09T14:17:00-02:00"
```

```
}
```

En donde se puede filtrar la salida según esos campos de entrada, si los campos de "mac" o "interfaz" son vacíos retorna los datos sin aplicar esos filtros. Si la fecha es vacía se retorna los datos actuales. La salida es un JSON que tiene los datos de todos los clientes localizados en los últimos INTSEGLOC (parámetros B.1) segundos.

- `POST /rest/plano`

  El servicio retorna información del plano, espera un JSON con la siguiente estructura:

```
{
    "algoritmo": ""
    "nodos": true
}
```

  Los campos "algoritmo" y "nodos" son opcionales.

  Retorna un JSON con todos los datos necesarios para visualizar el mapa y los nodos para cualquier algoritmo.

- `POST /rest/aprendizaje`

  Este servicio es para activar o desactivar el aprendizaje de FIND3, recibe la siguente estructura:

```
{
    "active": true,
    "mac": "30AEA4BE532E",
    "interface":"bt",
    "location" : ""
}
```

  Realiza un POST a FIND3 comunicando la acción.

- `POST /rest/aprendizaje/offline`

  Este web service comunica a FIND3 mediante POST los datos guardados de ubicaciones reales, simulando que los nodos están recolectando datos. Los datos comunicados tienen que ser almacenados de forma previa para su posterior simulación. El servicio recibe la siguiente estructura:

```
{
    "date_ini": "2020-10-25T00:00:00-00:00",
    "date_fin": "2020-11-17T00:00:00-00:00",
```

```
    "w_find3": 30,
    "w_fingerprints": 30,
    "evaluacion": true,
    "limitTime": 0
}
```

Además, el servicio recibe el parámetro "evaluacion" que envía los datos con el fin de evaluar el sistema

B.4. Estructura de la base de datos

A continuación si listan las tablas del sistema detallando el fin de cada una.

- `nodes`

  Se registran todos los nodos ancla del sistema

  `devices`

  Se registran todos los dispositivos reconocidos por el sistema

- `ubicaciones`

  Almacena la información de la ubicación calculada por el algoritmo seleccionado

- `error`

  Guarda datos del error según la ubicación del cliente al momento de realizar la localización, cada registro de la tabla `ubicaciones` con tiene un registro en la tabla `error`.

- `parameters`

  Contiene la configuración del sistema

- `fingerprints`

  Registra conjuntos de RSSI para diferentes clientes, recolectados por los nodos anclas. Estos registros son utilizados para calcular las ubicaciones

- `locations_f3`

  Registra los mapeos entre las "locations" de FIND3 y posición "x" e "y", necesario para dibujar en el mapa la ubicación del cliente

- `planos`

  Registra la información de los planos

- `ubicaciones_reales`

  Registra los datos de la ubicación real de los clientes en un rango de tiempo

B.5. Cálculo de ubicaciones

El sistema usa un algoritmo de localización a la vez, FIND3 o Trilateración. Para Trilateracion, el sistema almacena las mediciones recibidas en la tabla `fingerprints` para su posterior cálculo. Para FIND3, todo lo que recibe de los nodos se almacena en la tabla `fingerprints`, además todos esos datos se reenvían a la API [29] de FIND3 para su clasificación.

El sistema tiene una tarea programada que ejecuta cada `INTSEGLOC` segundos (ver tabla de parámetros B.1). Cuando el algoritmo es trilateración se calcula la ubicación con los datos de los últimos `INTSEGLOC` segundos. Cuando ejecuta FIND3, se le consulta a FIND3 mediante su API la ubicación del cliente según el resultado de la clasificación. El resultado de los dos algoritmos se almacena en la tabla `ubicaciones`.

Apéndice C

Configuración del Gateway Dragino

Para tener acceso al gateway, conectamos el mismo mediante el puerto Ethernet a una PC, en la PC se configura la subred configurada en el gateway con IP: 172.31.255.253 y máscara de red: 255.255.255.252.

Una vez configurado lo anterior, podremos acceder al gateway mediante la url http://172.31.255.254:8000. Estando dentro, se configura la red Wi-Fi para conectarnos al AP y tener acceso a las máquinas virtuales del sistema, además se configura la IP del servidor LoRa.

Apéndice D

Modificaciones realizadas a FIND3

D.1. Instalación

La instalación se realizó en el sistema operativo Ubuntu Server 18.04.4 LTS.

```
$ sudo apt install g++
$ sudo apt install mosquitto-clients mosquitto

# Descargar e instalar la ultima versión de Go.
# Para la versión 1.14.3 (ver versión acorde al SO en https://golang.org/dl/):
$ wget https://dl.google.com/go/go1.14.3.linux-amd64.tar.gz
# Para la instalación se siguieron los pasos indicados en la página de Golang.
# Se extrae el contenido del archivo anterior en /usr/local:
$ sudo tar -C /usr/local -xzf go1.14.3.linux-amd64.tar.gz

# Agregar el directorio go al PATH
# Editar el archivo /etc/profile (configuracion global) o ~/.profile :
# export PATH=$PATH:/usr/local/go/bin
# Volver a iniciar sesión para que el cambio en el PATH surta efecto.

# Descargar los fuentes de FIND3 y compilarlos:
go get -u -v github.com/schollz/find3/...

# Definir la variable GOPATH (archivo /etc/profile o ~/.profile)
# export GOPATH=/home/usuario/go
# usuario es el nombre del usuario. Volver a iniciar sesión para que exista la
    ↪ variable.

# Instalar el módulo pip de Python:
$ sudo apt install python3-pip

# Instalar las dependencias Python:
$ cd $GOPATH/src/github.com/schollz/find3/server/ai
$ python3 -m pip install -r requirements.txt
```

```
# Ejecutar el AI server:
$ cd $GOPATH/src/github.com/schollz/find3/server/ai
$ make
```

En otra terminar ejecutar lo siguiente:

```
# Compilar el Main server:
$ cd $GOPATH/src/github.com/schollz/find3/server/main
$ go build -v

# Para ejecutar el Main server:
$ cd $GOPATH/src/github.com/schollz/find3/server/main
$ ./main -port 8005
```

D.2. Modificaciones

El código fuente de FIND3 con las modificaciones realizadas para el presente proyecto se encuentra disponible en el siguiente repositorio: https://gitlab.fing.edu.uy/gerardo.gomez.detjen/find3.

D.2.1. Remoción de algoritmos de Machine Learning

Para quitar o agregar algoritmos de machine learning a entrenar y utilizar, es necesario editar el archivo "*server/ai/src/learn.py*". La variable *names* tiene los nombres de los algoritmos utilizados (a modo descriptivo), y la variable *classifiers* los algoritmos que serán invocados.

D.2.2. Nuevas flags al servidor principal

Se modificó el archivo "*server/main/main.go*" para que el ejecutable del Main Server acepte las siguientes flags:

- `csvdir (string)`: Directorio para guardar una copia del archivo CSV que contiene los fingerprints utilizados para entrenar los algorimos de ML.

- `archcalib (string)`: Ruta de archivo de calibración (entrenamiento). Cuando a FIND3 le llegan fingerprints de aprendizaje, cada cierto tiempo ejecuta una calibración de forma automática, que causa un enlentecimiento del sistema. Para facilitar el prendido y apagado de la calibración,

se configura una ruta de archivo, cuya existencia indica que se debe calibrar. La calibración (automática o manual) se ejecuta únicamente si existe este archivo.

- `separarDatosLearnTest (bool)`: Al calibrar los algoritmos de ML, FIND3 realiza una evaluación del sistema (i.e., calcula la exactitud). Para esto separa los fingerprints de forma aleatoria, tomando un subconjunto para el entrenamiento y otro para la evaluación. Si esta flag es `true`, se permitirá realizar esta separación de datos para entrenamiento y evaluación. En caso de ser `false`, no se realizará una separación y se utilizarán todos los datos para entrenamiento.

- `usarDatosTest`: Aplica únicamente cuando `separarDatosLearnTest` es `false`, es decir, se utilizan todos los datos para entrenamiento. Si `usarDatosTest` es `true`, se utilizará un subconjunto de los datos para la evaluación (incluidos en los datos de entrenamiento). En caso de ser `false`, no se realizará evaluación, y en la consola de FIND3, la exactitud indicará 0 %.

Las flags anteriores se guardan como variables globales del package `api`, que se encuentran declaradas en al archivo "*server/main/src/api/calibration.go*". En ese archivo se encuentra la función `Calibrate`, que es la que ejecuta el entrenamiento de los algoritmos. Allí se implementa la utilización de las flags.

D.2.3. Otras modificaciones

Otras modificaciones al archivo "*server/main/src/api/calibration.go*":

- La función `splitDataForLearning` es la que realiza la separación de datos para calibración y evaluación. Se modificó para que use todos los fingerprints generados en lugar de limitar este número a 1000.

- Se agregó la función `quitarIgnorados`, que permite quitar de los fingerprints los datos de los nodos que no se quiera utilizar (para las distintas combinaciones de nodos de la evaluación). Para esto utiliza la función `considerarSensor` del archivo "*server/main/src/api/sensoresConsiderar.go*", que recibe el sensor (en formato "*dirección_mac-interfaz*") y devuelve un booleano si se debe considerar. En la implementación de

esta función, los nodos a considerar están hardcoded (la versión actual en el repositorio acepta todos los nodos).

Apéndice E

Instalación de Herramientas y Ambiente de Ejecución

E.1. ESP-IDF y LuaRTOS

A continuación se describen los comandos necesarios para la instalación de LuaRTOS:

```
# Permisos de escritura al usuario en el puerto USB
$ sudo usermod -a -G dialout $USER

# Instalación de paquetes necesarios
$ sudo apt install git wget flex bison gperf python python-setuptools python-
    ↪cryptography python-pyparsing python-pyelftools cmake ninja-build ccache
    ↪ picocom libncurses5-dev

# Instalación de python-pip
#   Agregar el repositorio universe
$ sudo add-apt-repository universe
#   Descargar script get-pip.py
$ curl https://bootstrap.pypa.io/get-pip.py --output get-pip.py
#   Ejecutar el script get-pip.py
$ sudo python2 get-pip.py
#   Instalar módulos necesarios de python
$ pip install pyserial future click

# Clonar el repositorio de esp-idf
$ git clone --recursive https://github.com/espressif/esp-idf.git

# Clonar el repositorio de LuaRTOS para ESP32
$ git clone --recursive https://gitlab.fing.edu.uy/gerardo.gomez.detjen/Lua-
    ↪RTOS-ESP32.git

# Descargar el ESP32 toolchain que es necesario para compilar LuaRTOS
# Estando en el directorio home
```

```
$ wget https://dl.espressif.com/dl/xtensa-esp32-elf-linux64-1.22.0-80-g6c4433a
    ↪-5.2.0.tar.gz
$ mkdir esp
$ cd esp
$ tar -xzf ~/xtensa-esp32-elf-linux64-1.22.0-80-g6c4433a-5.2.0.tar.gz

# Agregar directorio al PATH, agregando la siguiente linea a ~/.profile
$ export PATH=~/esp/xtensa-esp32-elf/bin:$PATH
# Volver a iniciar sesión para tomar los cambios

# Editar el archivo ~/Lua-RTOS-ESP32/env, comentando la modificación anterior.
# Además corroborar que la variable IDF_PATH está correctamente asignada

# El siguiente comando da error si nunca se hizo un build
$ make clean
$ make -j5
$ make flash

# Descargar wcc (whitecat console) en "/usr/bin"
$ cd /usr/bin
$ sudo wget http://downloads.whitecatboard.org/console/linux/wcc
$ sudo chmod -v 755 wcc
```

E.2. Chirpstack

E.2.1. Instalación

A continuación se detallan los comandos para la instalación de todo el stack necesario de Chirpstack. La instalación está basada en un Ubuntu Server 18.04.4LTS sin ninguna otra instalación.

E.2.1.1. Instalación de requerimientos

MQTT 3.1.1

```
$ sudo apt-get install mosquitto
```

PostgreSQL 9.5+

```
$ sudo apt install postgresql
```

E.2.1.2. Instalación de ChirpStack Gateway Bridge

A continuación se resumen los comandos ejecutados para la instalación [15]

```
# Agregar el repositorio de ChirpStack 3.x
$ sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys 1
    ↪CE2AFD36DBCCA00
```

```
$ sudo echo "deb https://artifacts.chirpstack.io/packages/3.x/deb stable
    ↪main" | sudo tee
/etc/apt/sources.list.d/chirpstack.list
$ sudo apt update

# Instalación del gateway bridge
$ sudo apt install chirpstack-gateway-bridge
```

El archivo de configuración se encuentra en la ruta:
`etc/chirpstack-gateway-bridge/chirpstack-gateway-bridge.toml`

E.2.1.3. Instalación de ChirpStack Network Server

A continuación se resumen los comandos ejecutados para la instalación [16]

```
# Creación de base de datos y usuario para la misma
$ sudo -u postgres psql

-- create the chirpstack_ns user with password 'dbpassword'
create role chirpstack_ns with login password 'dbpassword';

-- create the chirpstack_ns database
create database chirpstack_ns with owner chirpstack_ns;

-- exit the prompt
\q

# Instalación de network server
$ sudo apt install chirpstack-network-server
```

El archivo de configuración se encuentra en la ruta:
`/etc/chirpstack-network-server/chirpstack-network-server.toml`

E.2.1.4. Instalación de ChirpStack Application Server

A continuación se resumen los comandos ejecutados para la instalación [14]

```
# Extension pq_trgm y hstore
$ sudo -u postgres psql

-- create the chirpstack_as user
create role chirpstack_as with login password 'dbpassword';

-- create the chirpstack_as database
create database chirpstack_as with owner chirpstack_as;

-- enable the trigram and hstore extensions
\c chirpstack_as
create extension pg_trgm;
create extension hstore;

-- exit the prompt
```

```
\q

# Instalación de network server
$ sudo apt-get install chirpstack-application-server
```

El archivo de configuración se encuentra en `/etc/chirpstack-application-server/chirpstack-application-server.toml`

E.2.2. Comandos útiles

```
# (re)start and stop ChirpStack Gateway Bridge
$ sudo systemctl [start|stop|restart|status] chirpstack-gateway-bridge

# (re)start and stop ChirpStack Network Server
$ sudo /etc/init.d/chirpstack-network-server [start|stop|restart|status]

# (re)start and stop ChirpStack Application Server
$ sudo /etc/init.d/chirpstack-application-server [start|stop|restart|
    ↪status]

# Display logs
$ sudo journalctl -f -n 100 -u chirpstack-gateway-bridge
$ sudo journalctl -f -n 100 -u chirpstack-network-server
$ sudo journalctl -f -n 100 -u chirpstack-application-server
```

Apéndice F

Consumo de Energía de los Nodos

Se realizaron pruebas específicas para determinar el consumo de energía por parte de los nodos. Para esto, se usó un powerbank de 20.000 mAh (12.000 mAh de capacidad real) en un nodo que estuvo escaneando utilizando Wi-Fi, BLE y BR/EDR en paralelo y de forma continua, sin realizar filtro de direcciones MAC (es decir, se envió vía LoRaWAN todas las direcciones MAC y RSSI que detectó). La duración de la batería fue de 5 días y 6 horas.

Si bien LoRa está diseñado para consumir poca energía, el uso de Wi-Fi y Bluetooth al mismo tiempo y de forma continua ocasiona un mayor consumo.

Apéndice G

Despliegue de Nodos en las Pruebas

En este apéndice se muestra el despliegue realizado de los nodos, en el hall de la planta baja del InCo. Las figuras G.1 y G.2 muestran los nodos como fueron colocados para realizar las pruebas, sin disponer de una foto con los nodos 8 y 9 (según la numeración mostrada en la figura 4.1), con el cliente marcado con una circunferencia roja.

Luego, las figuras G.3 y G.4 muestran las ubicaciones de todos los nodos, si bien no están colocados sobre las cajas exactamente como se usaron para las pruebas. Estas fotos fueron tomadas en un momento anterior a la realización de las pruebas, y en este caso, el cliente es la placa que se encuentra sobre la caja transparente.

Finalmente, la figura G.5 muestra el Gateway LoRaWAN, que se colocó en el pasillo de la misma planta donde se encuentran los nodos.

Figura G.1: Despliegue de nodos utilizados en las pruebas (1)

Figura G.2: Despliegue de nodos utilizados en las pruebas (2)

Figura G.3: Ubicaciones de los nodos (1)

Figura G.4: Ubicaciones de los nodos (2)

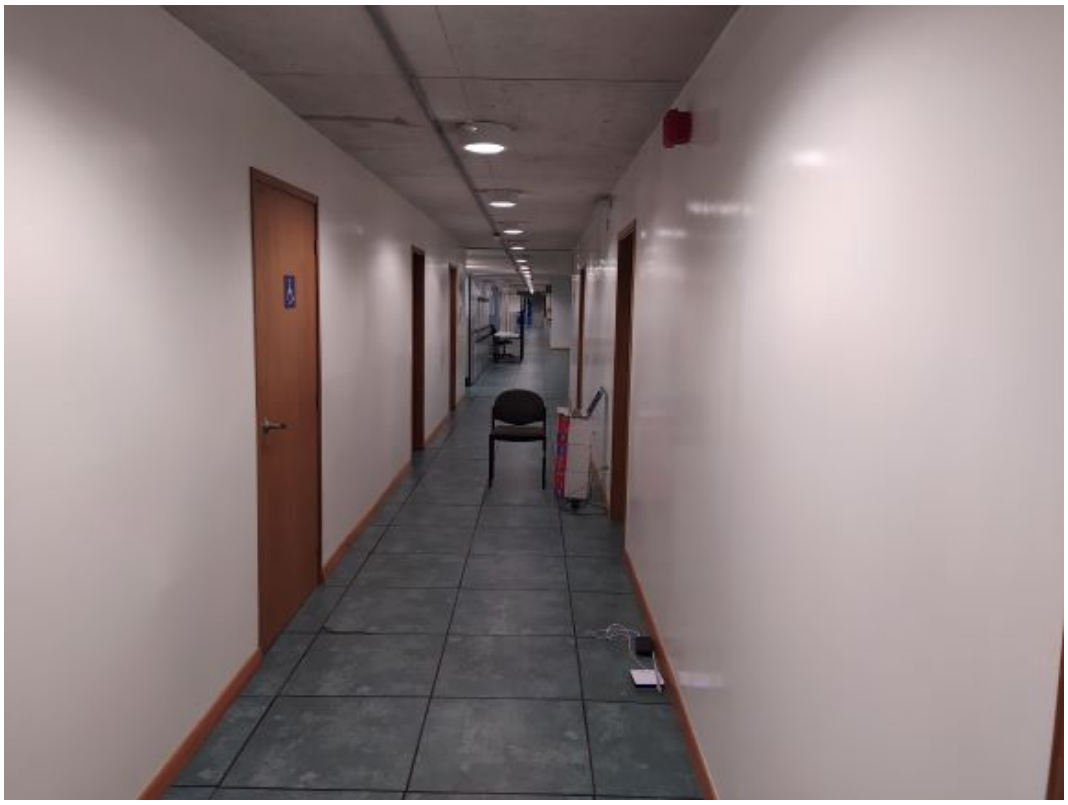


Figura G.5: Ubicación del Gateway LoRaWAN