UNIVERSIDAD SAN FRANCISCO DE QUITO

Colegio de Ciencias e Ingeniería

Localización de precisión mediante puntos de acceso Wi-Fi en viviendas

Camilo José Bravo Cabezas

Fernando Sánchez, Ph.D., Director de Tesis

Tesis de grado presentada como requisito para la obtención del título de
Ingeniero de Sistemas

Quito, mayo de 2015

Universidad San Francisco de Quito

Colegio de Ciencias e Ingeniería

HOJA DE APROBACIÓN DE TESIS

Localización de precisión mediante puntos de acceso Wi-Fi en viviendas

Camilo José Bravo Cabezas

Fernando Sánchez, Ph.D.
Director de Tesis _____

Daniel Fellig, M.Sc.
Miembro del Comité de Tesis _____

Mauricio Iturralde, Ph.D.
Miembro del Comité de Tesis _____

Fausto Pasmay, M.Sc.
Director del Programa _____

Ximena Córdova, Ph.D.
Decana de la Escuela de Ingeniería _____
Colegio de Ciencias e Ingeniería

Quito, mayo de 2015

© DERECHOS DE AUTOR

Por medio del presente documento certifico que he leído la Política de Propiedad Intelectual de la Universidad San Francisco de Quito y estoy de acuerdo con su contenido, por lo que los derechos de propiedad intelectual del presente trabajo de investigación quedan sujetos a lo dispuesto en la Política.

Asimismo, autorizo a la USFQ para que realice la digitalización y publicación de este trabajo de investigación en el repositorio virtual, de conformidad a lo dispuesto en el Art. 144 de la Ley Orgánica de Educación Superior.

Firma: _____

Nombre: Camilo José Bravo Cabezas

C.I.: 171857944-2

Lugar: Quito, Ecuador Fecha: mayo de 2015

RESUMEN

El problema del posicionamiento está mayormente resuelto en espacios abiertos por el sistema GPS, mas no en interiores. Actualmente no existe una alternativa generalizada para determinar si un dispositivo está en una habitación u otra de una vivienda.

Esta Tesis pretende evaluar la validez de una alternativa a este problema. El sistema propuesto utiliza los omnipresentes puntos de acceso de Wi-Fi que existen hoy en día en zonas urbanas alrededor del mundo, y valiéndose del hecho de que se puede obtener mediciones de niveles de señal en cualquier momento, sin requerir de ningún "permiso" (técnico o no). Con estos datos, luego de definir manualmente varias "zonas" (por ejemplo habitaciones) dentro de un espacio y almacenar los puntos de acceso medidos por el dispositivo – utilizaremos un teléfono celular con el sistema Android – en una base de datos, se evaluará la confiabilidad de un sistema que intentará reportar la zona en la que se encuentra.

Para determinar la validez del sistema propuesto, se probará aplicando dos algoritmos diferentes para la determinación de la zona actual, y cambiando variables que podrían afectar los resultados (cantidad de zonas a almacenar, granularidad de las zonas, cantidad de puntos de acceso a utilizar en cada medición). Adicionalmente, se intentará encontrar alguna combinación que dé resultados satisfactorios para un uso regular.

ABSTRACT

The problem of positioning is mostly solved by GPS in open spaces, but not in interiors. Currently, there is no widespread alternative to determine if a device is in one room or the next.

This Thesis aims to evaluate the effectiveness of an alternative to this issue. The proposed system uses the ubiquitous Wi-Fi access points online nowadays in urban areas around the world, taking advantage of the fact that one can obtain measurements of the signal levels at any time, without any "permission" (technical or otherwise). With this information, after manually defining several "zones" (e.g. rooms) inside a space and storing the access points seen by the device – we will use an Android phone – into a database, we will evaluate the reliability of a system that will try to report the zone it is in.

In order to determine the validity of the proposed system, we will test two different algorithms for calculating the current zone, and changing variables that might affect the results (how many zones are stored, the granularity of the zones, the amount of access points to be used in each measurement). Additionally, we will attempt to find a combination that yields satisfactory results in regular use.

TABLA DE CONTENIDO

TABLAS

FIGURAS

1. INTRODUCCIÓN

1.1 El problema de la localización

Los sistemas de GPS y AGPS que existen en la actualidad funcionan bien y son útiles para saber dónde se está ubicado en un mapa. Estos sistemas tienen una precisión limitada, normalmente de 2,5 metros (Página GPS Española), lo cual no es problema para determinar la posición al nivel de calle, que es lo que usualmente se necesita. Pero al intentar usar estos sistemas para determinar la ubicación dentro de una casa, por ejemplo, la poca precisión se convierte en un impedimento, pues no podemos utilizarlo de manera confiable para identificar una habitación de otra. Adicionalmente, para que funcione correctamente el sistema GPS se necesita tener línea de vista con los satélites, lo que limita la utilidad de estos sistemas en interiores. En este proyecto intentamos evaluar una alternativa que permita resolver el problema de determinar la ubicación en interiores. El sistema propuesto utiliza redes y puntos de acceso Wi-Fi cercanos al usuario que pueden servir como referencia para calcular la ubicación, aprovechando la gran cantidad de dichas redes en zonas urbanas, lo cual elimina la necesidad de nueva infraestructura. Vale resaltar que para determinar la ubicación no necesitamos acceso a las diferentes redes Wi-Fi, sino solamente estar en la cercanía de los puntos de acceso.

1.2 Ventajas de un sistema más preciso de localización

El poder medir con una precisión mayor a la de los sistemas actuales de GPS o AGPS puede ser útil para diversas aplicaciones. Una de éstas es el monitoreo de pacientes (healthcare), ya que una aplicación puede estar instalada en un teléfono celular, reportando continuamente a un servicio Web la posición de la persona. La aplicación podría incluir un

sistema de alerta que se active cuando el teléfono se aleje de las zonas consideradas "seguras". Otra aplicación podría ser un sistema inteligente donde ciertos dispositivos (iluminación, calefacción, etc.) se activen automáticamente basado en la ubicación de una persona.

Las aplicaciones no se limitan a espacios interiores, ya que se pueden definir zonas en el exterior, por ejemplo en campus universitarios. De este modo se puede tener un monitoreo de la posición de una persona en lugares más amplios.

1.3 Sistema propuesto

Para el proyecto se usará un teléfono con una aplicación que reportará su ubicación dentro de un departamento en la ciudad de Quito. Esto se logrará midiendo los niveles de potencia desde el teléfono hacia todos los puntos de acceso Wi-Fi que se detecten. Se intentará, a través de la técnica de fingerprinting, almacenar los niveles de potencia medidos de cada punto de acceso en cada habitación, y luego usar esos datos para determinar la ubicación del dispositivo en tiempo real.

El sistema propuesto en este proyecto además incluye un Back-end remoto que se encargará de hacer los cálculos y almacenar los datos, al cual se accederá a través de Internet. La aplicación en el teléfono solo será un cliente liviano cuya función es tomar las mediciones y reportar lo devuelto por el Back-end.

Se probarán dos métodos para el cálculo de la ubicación en tiempo real, los cuales se implementarán en el Back-end: el primero es un cálculo geométrico utilizando el teorema de Pitágoras para encontrar la zona almacenada más cercana a la medición; el segundo utiliza una red neuronal para intentar determinar la zona.

Se harán varias mediciones variando la cantidad de puntos de acceso a almacenar y la granularidad de las zonas (habitaciones completas o una cuadrícula), y se intentará determinar una combinación de estos elementos en la que el sistema sea confiable dentro de un margen de error aceptable. Para considerar el proyecto como exitoso, el error en la detección deberá ser de menos del 15%, es decir, se aceptará que hasta el 15% de veces que se reporta una zona, esta sea la zona incorrecta.

A continuación expondremos como fundamento teórico el funcionamiento de los sistemas GPS, posibles aplicaciones de un sistema más preciso para interiores, y el funcionamiento de los elementos relevantes al diseño propuesto posteriormente. Luego describiremos el diseño propuesto y la implementación en particular, y finalmente haremos una evaluación del sistema utilizando el sistema en un espacio real.

2. FUNDAMENTO TEÓRICO

2.1 GPS y AGPS

El sistema de posicionamiento global (GPS por sus siglas en inglés) funciona utilizando satélites puestos en órbita, de los cuales se conoce su posición. El sistema consiste en tres componentes básicos: los satélites, el receptor, y estaciones en tierra que controlan a los satélites. Para obtener la posición, el receptor recibe datos de varios satélites, y calcula la distancia a cada uno tomando el tiempo que demora recibir una señal. (DePriest, 1999) Con estos datos, se puede triangular la posición con cierto error, el cual viene mayormente del paso de ondas por la ionosfera. La precisión que se puede alcanzar normalmente es de 2,5 metros. (Página GPS Española) Para aplicaciones populares como mapas de ciudades, o sistemas de navegación para automóviles, el GPS es suficiente. Para el proyecto actual, sin embargo, se requiere de más precisión ya que 2,5 metros no es suficiente para saber si se está en un cuarto o en otro. Otra limitación importante del sistema GPS es que se necesita tener línea de vista con los satélites, por lo que muchas veces simplemente no funciona en interiores.

Se denomina GPS “asistido” (AGPS) al sistema que utilizan la mayoría de celulares en la actualidad para mejorar al GPS normal, utilizando servidores y la conexión de datos del dispositivo para mejorar la experiencia del usuario. La “asistencia” consiste, por un lado, en leer las posiciones de los satélites desde los servidores, lo cual hace la lectura normal de GPS más rápida, y por otro utilizar al servidor para calcular la posición del dispositivo, teniendo en cuenta que se conocen las posiciones de las torres de celulares a los que se conecta el dispositivo. Esto último también hace que el dispositivo no necesite mucha capacidad de procesamiento, ya que los datos son manejados con facilidad por el

servidor. El sistema GPS asistido también ayuda en el caso de interiores, ya que utilizando información de los servidores no se necesita ver directamente a los satélites, aunque muchas veces tampoco funciona en lugares cerrados. La precisión no mejora notablemente utilizando AGPS, por lo que el sistema no es un buen candidato para el proyecto. (GPS vs. aGPS: A Quick Tutorial, 2009)

2.2 Uso de puntos de acceso Wi-Fi en una zona urbana para localización

En los últimos años el uso de redes Wi-Fi ha aumentado vertiginosamente. En una zona urbana, se puede encontrar fácilmente varios puntos de acceso desde cualquier ubicación con un dispositivo móvil. En un estudio realizado el año 2006 en Chicago, se encontró que en promedio había 6 puntos de acceso disponibles en cada búsqueda. (Nicholson, Chawathe, Chen, Noble, & Wetherall, 2006, pág. 8) Este número probablemente es mucho mayor en la actualidad.

El protocolo de Wi-Fi (802.11) permite leer la dirección única (MAC) de cada punto de acceso, y la potencia medida desde el dispositivo sin tener que unirse a ninguna red. Con estos datos, y considerando que los puntos de acceso están en un lugar fijo, se puede determinar diferentes zonas dentro de una casa utilizando los distintos niveles de señal de puntos de acceso cercanos, y de este modo crear un “mapa”, el cual puede ser usado posteriormente para determinar la posición del dispositivo midiendo los niveles de señal y analizando qué zona es la más cercana.

En este proyecto intentamos determinar qué tan confiable puede ser dicho mapa. Dado que los cambios en los niveles de señal pueden darse por diversos factores, por ejemplo la humedad del aire, estos pueden causar que el error de las mediciones sea

demasiado grande. En este caso, la aplicación no sería confiable al no dar una representación exacta de dónde se encuentra el dispositivo.

2.3 Atenuación

Mientras más lejos se ubique un dispositivo de un punto de acceso Wi-Fi, menor será la señal medida en decibeles. Esto se conoce como atenuación.

La escala de decibeles mide la diferencia entre dos niveles de señal (Young). Es una escala logarítmica, en la que cuando se duplica el nivel de potencia se aumentan 3 dB, y vice versa. La escala empieza desde 0 dB, y solo incluye números negativos. En la práctica se obtienen mediciones de hasta unos -90 dB.

Utilizando dispositivos comunes de Wi-Fi en interiores, un modelo matemático adecuado para predecir el nivel de señal como función de la distancia entre el dispositivo y un punto de acceso, es el dado por la siguiente ecuación: (Faria, 2005)

$$x = Pr\ \ dist\ \ = Pr_0 - 10 \cdot \alpha \cdot \log_{10}\ \ dist\ \ + X_\sigma$$

Donde:

- $Pr\ \ dist\ \ $ es la señal medida en decibeles como función de la distancia $dist$ del punto de acceso.
- Pr_0 es la señal medida a un metro del transmisor.
- α representa qué tantas obstrucciones ocurren en el ambiente; se ha medido empíricamente valores entre 1,8 (ambientes ligeramente obstruidos) y 5 (edificios de varios pisos), para este proyecto usaremos un valor de 4,5 para simular un departamento típico; y,
- X_σ es un valor aleatorio, el cual se omitirá en los cálculos.

Lo importante de esta ecuación es que la relación entre la señal medida en decibeles y la distancia no es lineal, y por lo tanto no es apropiada directamente para el propósito de este proyecto. Entonces, es necesario hacer un ajuste a la señal medida, para que la relación entre este nuevo valor y la distancia real sea lineal. Este ajuste se representa con la siguiente ecuación:

$$x' = -10^{\ -x\ 10\alpha}$$

Esta ecuación se usará para ajustar el nivel de señal medido, el cual por las constantes utilizadas quedará entre 0 y -100 para casos típicos.

2.4 Fingerprinting

El sistema GPS utiliza trilateración, un método matemático en el que se utiliza el conocimiento de las posiciones exactas de los satélites utilizados para determinar la posición. (What is trilateration?) Este método resulta inadecuado para el proyecto actual, ya que no se conoce de antemano la posición de los puntos de acceso utilizados, que serían el análogo de los satélites. Por este motivo se utilizará la técnica de fingerprinting, con la cual se almacenará mediciones de cada zona en una base de datos, la cual luego se podrá consultar para la determinación de la zona en tiempo real.

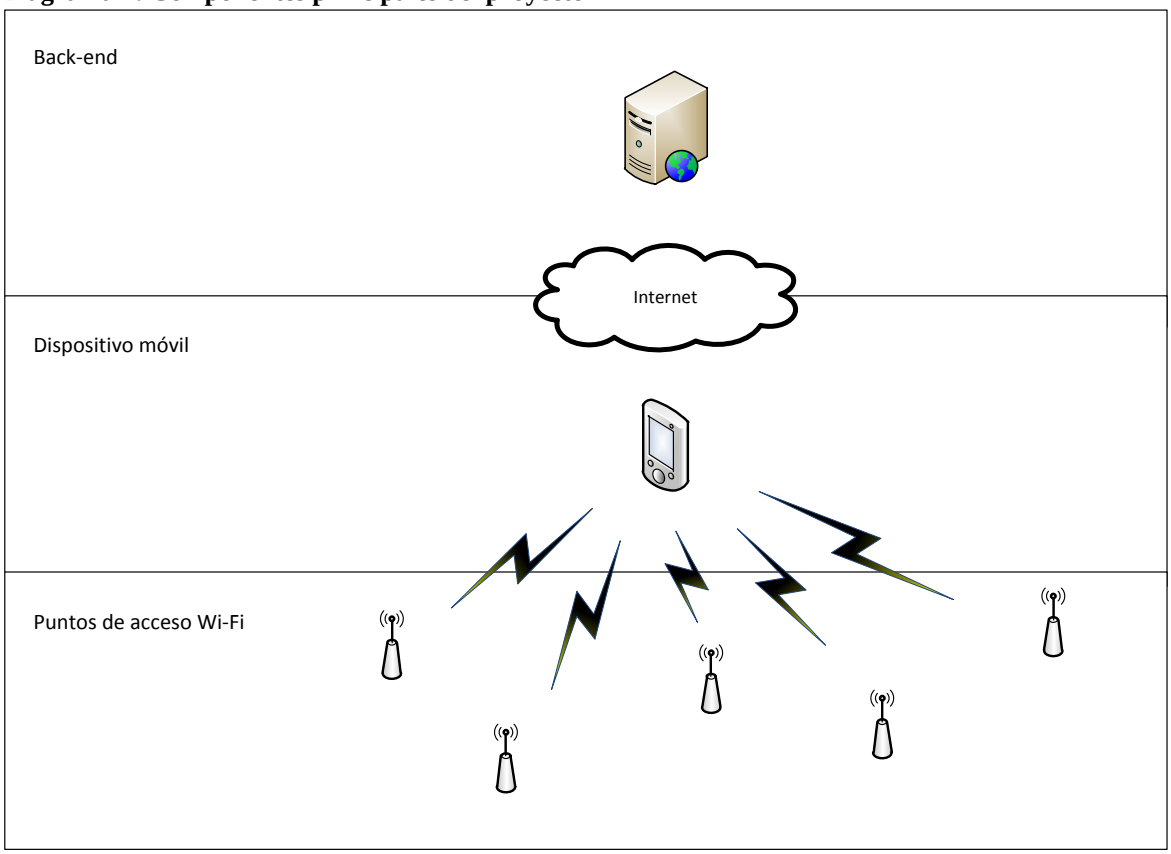
La técnica de fingerprinting se refiere a un proceso inicial en donde se definen las zonas que se utilizará, y se toman las mediciones en cada una para almacenarlas en una base de datos. Nuesto “fingerprint” se compone del conjunto de niveles de señal medidos desde el dispositivo para una zona específica. Este proceso es análogo a tomar huellas digitales de personas y almacenarlas, para luego utilizar la base de datos en el reconocimiento de una huella desconocida; solo después de esta medición inicial se podrá obtener la zona donde se encuentra un dispositivo móvil.

3. DISEÑO

3.1 Componentes principales

El sistema propuesto tiene tres componentes principales, mostrados en el siguiente esquema.

Diagrama 1: Componentes principales del proyecto



A continuación se describe cada componente.

3.1.1 Puntos de acceso Wi-Fi

Estos deberán existir en la cercanía de cada una de las zonas que se quiere identificar. Mientras más puntos de acceso existan, se dispondrá de más datos para los cálculos.

3.1.2 Dispositivo móvil

Un dispositivo capaz de obtener una medición de niveles de potencia a los puntos de acceso cercanos, y que tenga conexión a internet. Por ejemplo, un teléfono celular con un sistema que permita conectarse a redes Wi-Fi. Este dispositivo captura los datos y los envía a un Back-end (descrito a continuación), la cual los almacena y reporta los resultados para ser mostrados al usuario.

3.1.3 Back-end

Una aplicación que se ejecuta en un servidor Web, que almacene los datos y realice los cálculos necesarios para reportar la zona en la que se encuentra el dispositivo móvil. Se podría prescindir de este elemento, realizando todo en el dispositivo, pero se ha escogido separar esta funcionalidad por dos razones:

1. Si los datos se guardan remotamente se puede compartir la información entre varios dispositivos. Por ejemplo, varios teléfonos en un mismo espacio podrían reportar su ubicación, sin tener que pasar por el proceso de creación del mapa en cada uno.
2. Los dispositivos móviles en general tienen menos capacidad de procesamiento que un servidor Web. Una de las implementaciones propuestas utiliza redes neuronales, lo que haría costosa la operación en el dispositivo, reduciendo la batería del mismo.

3.2 Limitaciones conocidas

Antes de describir la implementación realizada se debe notar algunas limitaciones del sistema que se conocen de antemano.

3.2.1 Fingerprinting

Se usará la técnica de fingerprinting, con la cual se debe tomar mediciones de cada zona antes de poder conocer la ubicación en tiempo real del dispositivo. Por lo tanto, habrá un período de entrenamiento sin el cual el sistema no se podrá utilizar.

3.2.2 Cambios en puntos de acceso

Ya que las mediciones se hacen sobre puntos de acceso fuera del control del usuario del sistema, cualquier cambio en los mismos podrá causar que el sistema genere resultados incorrectos. Estos cambios incluyen cambios de canal, de posición física, y la activación de puntos de acceso nuevos.

3.2.3 Cambios en infraestructura

Además de los cambios en los puntos de acceso, cualquier cambio en los objetos del lugar puede afectar a los niveles de potencia medidos por el teléfono. Por ejemplo, cambios en la posición de muebles. Esto se da debido a la atenuación de la señal.

4. IMPLEMENTACIÓN

Se ha creado un prototipo para demostrar el proyecto de forma práctica. Este consiste en una aplicación en un dispositivo móvil, la cual reporta los niveles de señal medidos de puntos de acceso de Wi-Fi cercanos; y un Back-end en un servidor remoto que recibe estos datos, almacena información de los espacios físicos en una base de datos, hace los cálculos tomando las entradas del dispositivo móvil, y devuelve información para ser mostrada al usuario.

4.1 Terminología

En este punto definiremos ciertos términos, los cuales se utilizan en el proyecto:

Tabla 1: Terminología

Zona	La identificación de un lugar físico, con un nombre. Por ejemplo: “cocina”.
Espacio	Un conjunto de zonas cercanas. Por ejemplo: “departamento”.
Receptor	Un dispositivo que tiene conexión a internet, y puede leer niveles de señales de Wi-Fi. En el caso del prototipo creado, un teléfono celular con el sistema Android.
Punto de acceso (PA)	Dispositivo Wi-Fi visible públicamente. No es necesario que la red Wi-Fi sea pública, ya que solo se necesitará el nivel de señal medido desde el receptor.
Back-end	Aplicación Web ubicada en un servidor externo. Almacena los espacios y realiza las comparaciones para determinar en qué zona se encuentra el receptor.

4.2 Etapas

El funcionamiento del sistema se compone de dos etapas principales: la creación del mapa, y la posterior utilización del sistema para el reporte en tiempo real de la ubicación del dispositivo basado en el mapa.

4.2.1 Creación del mapa.

El primer paso para el funcionamiento del prototipo es la creación del mapa de un espacio. Esto se hace colocando el receptor en las diferentes zonas de un espacio, y utilizando la función de crear mapa de la aplicación. El proceso completo para cada zona es:

1. Ubicar el receptor físicamente en la zona.
2. Nombrar la zona actual, y el espacio donde se encuentra.
3. Activar la función de guardar, la cual mide los niveles de señal de los puntos de acceso visibles al receptor.
4. El receptor envía lo medido al Back-end.
5. El Back-end almacena en una base de datos las mediciones, junto con el nombre de la zona y espacio.

Para reducir errores se toman varias mediciones en una misma zona para cada punto de acceso encontrado. Este proceso funciona de la siguiente manera:

1. La aplicación en el receptor llama a la función del sistema operativo que mide los niveles de señal.
2. Se envían los niveles medidos hacia el Back-end.
3. Se espera un tiempo corto y se repiten estos pasos hasta tener un número determinado de mediciones.

La siguiente tabla ilustra el proceso, asumiendo que se hacen 3 mediciones. Las mediciones están en decibeles y se les realiza el ajuste descrito en la sección 2.3, por lo tanto son negativas. Valores más cercanos a 0 indican que el punto de acceso tiene un nivel de señal más fuerte.

Tabla 2: Ejemplo de mediciones de niveles de señal de puntos de acceso cercanos

Dirección MAC	Medición 1	Medición 2	Medición 3
12:34:56:78:9a:bc	-15	-25	-5
12:34:56:78:9a:bd	-90		-85
12:34:56:78:9a:be	-30	-20	-22
12:34:56:78:9a:bf	-40	-35	-30
12:34:56:78:9a:c0	-60	-70	

La transmisión de los datos recogidos se hace mediante una petición de HTTP, enviando con el método POST. Para la tercera medición del ejemplo anterior, y asumiendo que el espacio se llama “CASA” y la zona “COCINA”, los datos serían:

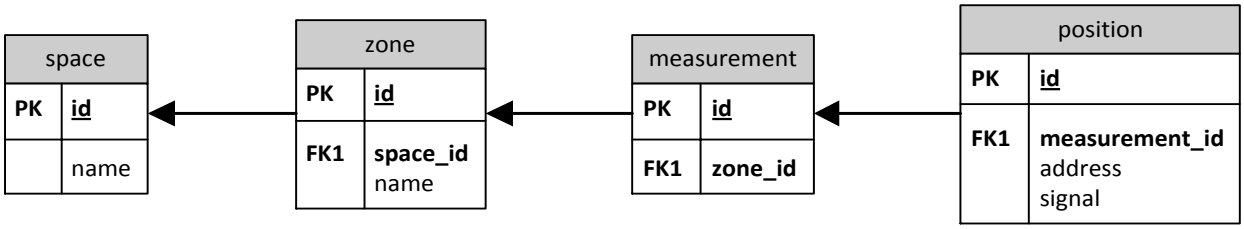
> space=CASA
>
> zone=COCINA
>
> levels[12:34:56:78:9a:bc]=-5
>
> levels[12:34:56:78:9a:bd]=-85
>
> levels[12:34:56:78:9a:be]=-22
>
> levels[12:34:56:78:9a:bf]=-30

Con esto, el Back-end puede almacenar los datos, los que se utilizarán luego para determinar la posición del receptor.

Almacenamiento de datos.

Todos los datos se guardan en el Back-end en una base de datos relacional MySQL. El esquema es el siguiente:

Diagrama 2: Esquema de base de datos utilizada en el Back-end



Las tablas son:

- *space*

  Aquí simplemente se guardan los espacios que define el usuario.

- *zone*

  En esta tabla se guarda cada zona con su nombre. Una zona pertenece a un espacio, por lo cual existe el campo “space_id”.

- *measurement*

  Cada medición almacenada en el proceso de creación del mapa es representada por un registro en esta tabla, relacionado a una zona.

- *position*

  Aquí se guardan los niveles de señal medidos para cada medición, en cada punto de acceso. Se guarda la dirección MAC en el campo “address”, y el nivel de señal en el campo “signal”.

Adicional a la base de datos, en el caso del algoritmo de red neuronal (explicado posteriormente), se almacenan los datos de las redes neuronales en archivos en el servidor.

4.2.2 Reporte de zona actual.

Para mostrar la zona donde se encuentra el receptor, se envían los niveles de señal de puntos de acceso al Back-end, el cual se basa en los datos almacenados anteriormente para devolver el nombre de la zona actual utilizando uno de los algoritmos programados.

Para esto, el receptor mide los niveles de señal y los envía usando POST, de forma similar a cuando se crea el mapa, pero esta vez solo hace una medición ya que se quiere obtener lo más rápido posible el resultado.

El usuario debe seleccionar manualmente el espacio donde se encuentra en el dispositivo. Con esto y los niveles de señal medidos, se hace una petición de HTTP. Esta vez, en el URL se envía el nombre del espacio. Por ejemplo:

http://tesis.com/zone/lookup/space/CASA

Los niveles de señal medidos se envían de forma similar a cuando se crea un mapa. Por ejemplo:

> levels[12:34:56:78:9a:bd]=-45
>
> levels[12:34:56:78:9a:be]=-60
>
> levels[12:34:56:78:9a:bf]=-10

Con esto, el Back-end utiliza uno de los algoritmos explicados más abajo para el cálculo, y devuelve el nombre de la zona más cercana al dispositivo, el cual lo muestra al usuario.

Algoritmos para el cálculo de la zona más cercana.

Se probaron dos algoritmos para calcular la zona más cercana, cuya implementación no cambia de ninguna manera el funcionamiento de la aplicación en el

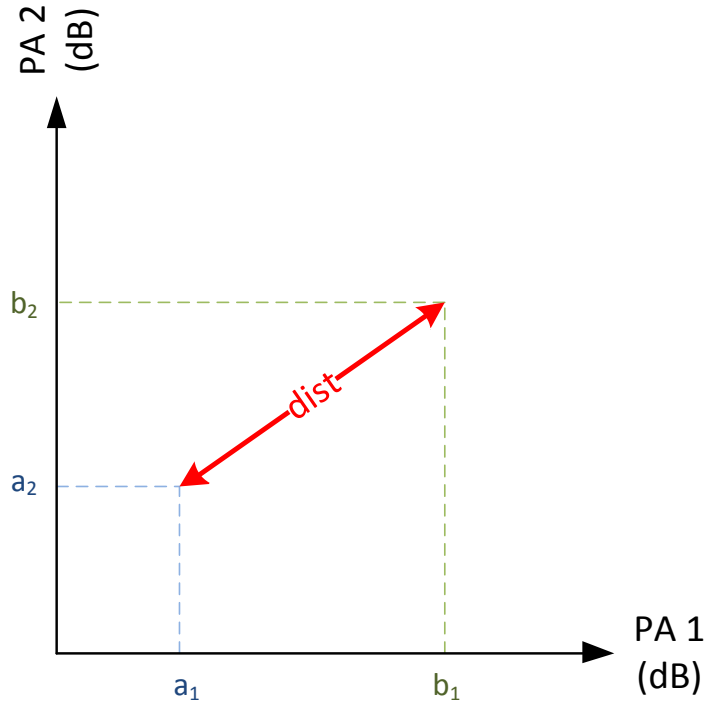
dispositivo ya que está en el Back-end. Por esto, los algoritmos son intercambiables. A continuación se explicará el funcionamiento de cada uno.

Teorema de Pitágoras.

En este algoritmo se calcula una distancia del punto medido hacia cada zona almacenada en la base de datos, y se devuelve la zona cuya distancia sea menor. Ya que en la etapa de creación del mapa se almacenan varias mediciones para cada zona, se calcula un promedio de estas. Si la distancia es mayor a un valor definido, se considera que no hay un resultado correcto y no se devuelve ninguna zona.

Para el cálculo de la distancia, se compara cada punto medido en el dispositivo con los almacenados en la base. Cada punto de acceso representa una dimensión, así, se puede usar el teorema de Pitágoras para determinar una distancia. Recordamos que los valores utilizados son ajustados de su medición original en decibeles. Este ajuste se calcula en el Back-end. A continuación se ilustra esta idea suponiendo que hay dos puntos de acceso medidos (representados entonces en 2 dimensiones), aunque normalmente se utilizarán más, según la cantidad de puntos de acceso medidos en la zona. En caso de no tener la medición en cualquiera de los puntos de acceso, es decir que esté fuera de rango, se utiliza un valor fijo de -100. Además, cualquier valor medido que sea menor se convierte a -100. Se llegó a esta constante tomando una gran cantidad de mediciones, y observando que ninguna era menor.

Diagrama 3: Cálculo de distancia utilizando el teorema de Pitágoras



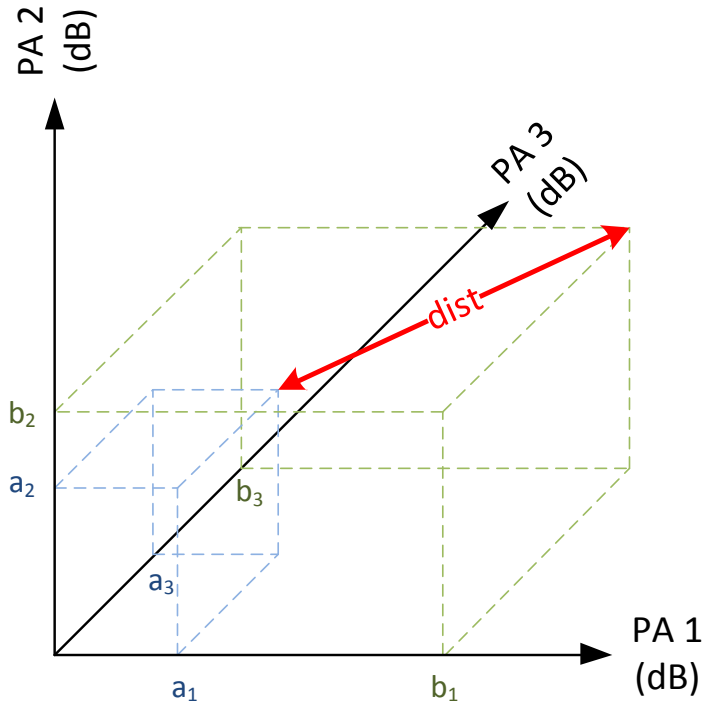
El diagrama muestra dos ejes, correspondientes a los dos puntos de accesos. Las mediciones *a* y *b* corresponden a dos conjuntos de mediciones hechas por el dispositivo móvil (por ejemplo, una zona almacenada anteriormente y una medición hecha en tiempo real). Cada valor representa el valor en decibeles ajustado medido por el dispositivo para el punto de acceso correspondiente, por ejemplo a_1 es el nivel de potencia de la medición *a* para el primer punto de acceso.

En este caso, la distancia entre los dos puntos es:

$$dist = \sqrt{a_1 - b_1\ ^2 + \ a_2 - b_2\ ^2}$$

Esto se puede expandir a más puntos de accesos siguiendo la misma fórmula. Por ejemplo, para 3 puntos de acceso:

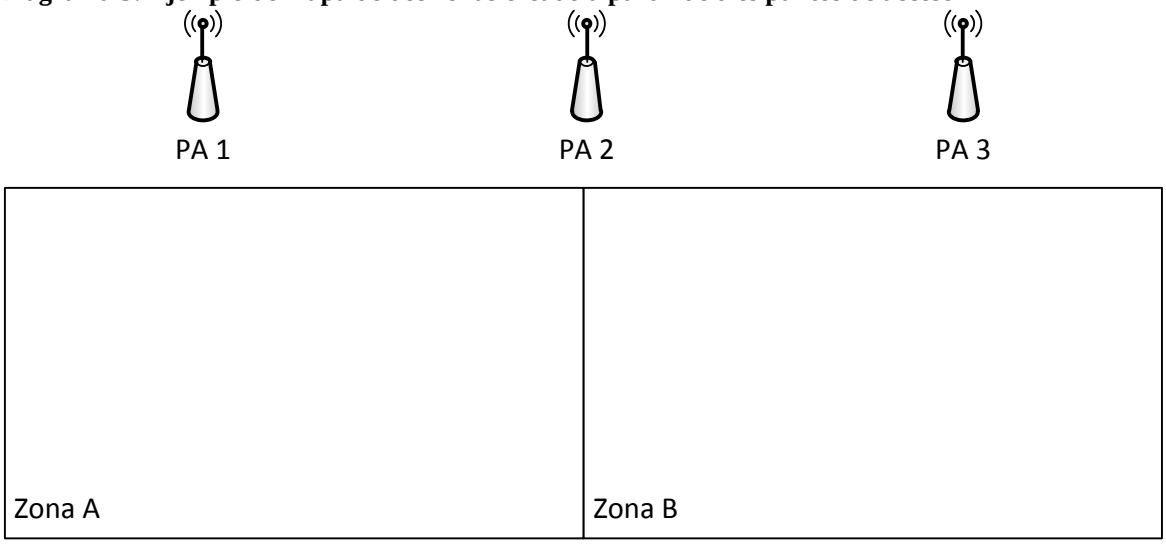
Diagrama 4: Cálculo de distancia para 3 dimensiones



$$dist = \sqrt{a_1 - b_1\ ^2 + \ a_2 - b_2\ ^2 + \ a_3 - b_3\ ^2}$$

Como ejemplo, asumamos que hay dos zonas (A y B), y tres puntos de acceso (1, 2, 3). El siguiente gráfico indica los promedios en cada zona de los valores almacenados en la base de datos:

Diagrama 5: Ejemplo de mapa de dos zonas creado a partir de tres puntos de acceso

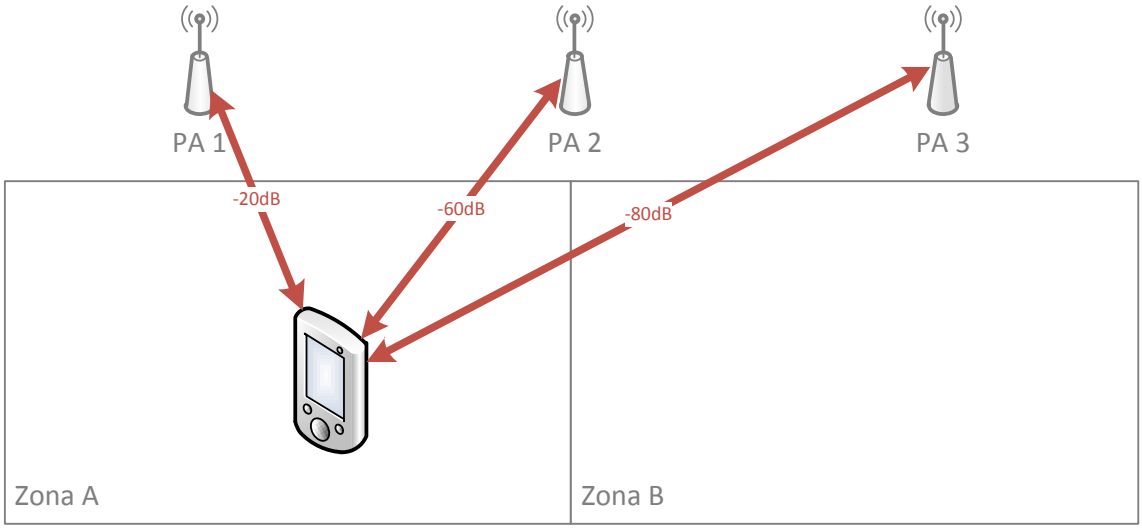


Zona A	Señal
PA 1	-10
PA 2	-40
PA 3	-70

Zona B	Señal
PA 1	-70
PA 2	-40
PA 3	-10

Ahora, ubicando el receptor en la posición que se indica, la siguiente tabla muestra los niveles medidos y las diferencias con los niveles de señal almacenados en cada zona:

Diagrama 6: Ejemplo de medición del receptor ubicado en una de las zonas creadas anteriormente



Punto de acceso	Señal	Diferencia Zona A	Diferencia Zona B
PA 1	-20	10	50
PA 2	-60	20	20
PA 3	-80	10	70

Ahora sólo queda calcular la distancia a cada zona:

$$d_A = \sqrt{10^2 + 20^2 + 10^2} = 24{,}49$$

$$d_B = \sqrt{50^2 + 20^2 + 70^2} = 88{,}32$$

Con esto se obtiene que la zona actual es la zona A, ya que es la que reporta menor distancia al receptor.

Red Neuronal.

El segundo algoritmo utilizado es la red neuronal. Una red neuronal consiste en una serie de "neuronas", las cuales tienen un peso (un número de 0 a 1), organizadas de tal manera que hay varias capas de neuronas. Una capa de entradas donde se asignan los valores medidos a neuronas, capas intermedias, y finalmente una capa de salidas, donde se obtienen los resultados.

Para el proyecto se utilizó una librería de PHP llamada Artificial Neural Network (Wien), la cual es una implementación de un tipo de red neuronal artificial llamado perceptrón multicapa, utilizando el algoritmo de propagación hacia atrás (backpropagation) para el entrenamiento.

En este tipo de red neuronal existe un proceso de entrenamiento en el que se utiliza un conjunto de datos donde las salidas de la red, además de las entradas, son conocidas. El tipo de red, perceptrón multicapa, forma una serie de neuronas conectadas entre sí de la forma descrita anteriormente. Las conexiones entre neuronas no forman ciclos (algo que sí es posible en otros tipos de red neuronal, por ejemplo redes neuronales recurrentes). El algoritmo de propagación hacia atrás se utiliza para ajustar los pesos de cada neurona. Esto funciona calculando la salida conocida de cada conjunto de datos de entrenamiento con la salida dada por la red actual, tomando la diferencia si hubiera, y haciendo un pequeño ajuste en la red para acercarse a la salida correcta. Esto se repite muchas veces para todos los datos de entrenamiento, hasta que el error promedio baja a un nivel aceptable.

Se asigna entonces a la red neuronal las mediciones para cada punto de acceso de una zona determinada (capa de neuronas de entrada), y se determina cuál será el resultado (capa de salida). La librería permite añadir "clasificadores" para etiquetar las salidas de la

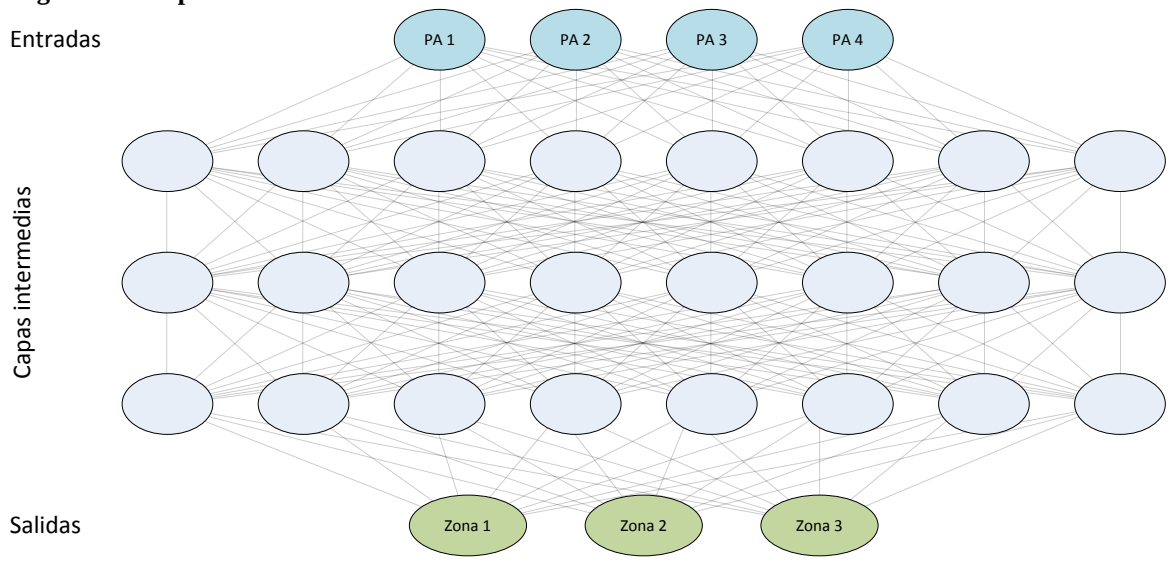
red, es decir las neuronas en la última capa. Esto permite poner el nombre de la zona a las neuronas de salida, de modo que la red neuronal se convierte en un clasificador. Esto se ajusta al proyecto, ya que queremos clasificar las señales de entrada (mediciones de señal de puntos de acceso) y determinar en qué zona se encuentra el receptor. En el proceso de entrenamiento se ingresa a la red neuronal cada medición por separado, al contrario del algoritmo anterior donde se calcula un promedio de las mediciones para cada zona.

Las neuronas de entrada representan los puntos de acceso. Se convierte el valor medido en decibeles y ajustado según lo descrito en la sección 2.3, a un número entre 0 y 1, apropiado para la red neuronal.

Las capas intermedias de neuronas se ajustan automáticamente en el proceso de entrenamiento. Sin embargo, se debe decidir cuántas capas habrá y cuántas neuronas habrá por capa. Se determinó que la cantidad de neuronas sea el máximo entre el doble del número de entradas de la red neuronal (es decir, la cantidad de puntos de acceso), o el doble del número de salidas (la cantidad de zonas definidas en el espacio). La cantidad de capas es una constante de 3.

A continuación se muestra una representación de una red neuronal que podría utilizar el proyecto, con 4 puntos de acceso y 3 zonas definidas. Se muestran todas las conexiones entre neuronas.

Diagrama 7: Representación de una red neuronal



4.3 Back-end web

El Back-end es una aplicación web que expone un API utilizado luego por la aplicación en el receptor. Se encarga de almacenar los datos, y de ejecutar los algoritmos explicados anteriormente para calcular la zona actual del receptor. Para el desarrollo se utilizaron algunas librerías para implementar componentes necesarios para el funcionamiento, pero que no son parte del proyecto en sí.

4.3.1 Librerías utilizadas.

Se escogieron las siguientes librerías para facilitar el desarrollo del Back-end:

- *Zend Framework* (Zend Technologies Ltd.)

  En PHP existen varios frameworks, los cuales implementan funciones comunes para aplicaciones web que no existen en el lenguaje. Se escogió Zend Framework, un framework MVC (modelo-vista-controlador), pero se podría haber utilizado otro sin problema. Se utilizó MySQL para la base de datos.

- *CModel* (Bravo)

  Debido a que Zend Framework no implementa la parte del modelo, se utilizó CModel como base para los modelos del proyecto, explicados posteriormente. CModel consiste en un conjunto de clases abstractas que implementan funcionalidad básica para "entidades", utilizados para cada modelo; "mappers", clases que manejan el acceso a la base de datos; y "colecciones", que representan conjuntos de entidades.

- *Artificial Neural Network for PHP 5.x* (Wien)

  Para implementar el segundo algoritmo se utilizó esta librería de PHP, la cual implementa una red neuronal de tipo perceptrón multicapa, utilizando el algoritmo de propagación hacia atrás (backpropagation) para el entrenamiento.

4.3.2 Estructura.

El Back-end consiste principalmente en los modelos y la interfaz que se provee para interactuar con la aplicación en el receptor (API).

4.3.1 Modelos utilizados.

La aplicación tiene 4 modelos, los cuales se encargan de toda la funcionalidad interna del proyecto, así como del almacenamiento a través de mappers en la base de datos. Un "mapper" es una clase que define el modo de almacenar en una base de datos una "entidad", es decir cada uno de los modelos explicados aquí.

- *Space*

  Este modelo corresponde a los espacios, por ejemplo un departamento. Aparte de un identificador (presente en todos modelos), solo tiene un nombre, y una colección de las zonas que pertenecen a dicho espacio.

  El modelo Space implementa funciones para utilizar la librería de redes neuronales (entrenamiento y uso), ya que se crea una red neuronal diferente para cada espacio, es decir para cada objeto del modelo Space.

- *Zone*

  El modelo Zone representa las zonas de un espacio. Tiene un nombre, una referencia al espacio, y una colección de las medidas almacenadas para esa zona.

  Este modelo tiene funciones para manejar las medidas (objetos del modelo Measurement), es decir para crear nuevas medidas y eliminarlas. También tiene funciones para calcular el promedio de todas las medidas, el cual es útil para el primer algoritmo de cálculo de la zona más cercana, y la función para calcular esta distancia utilizando el teorema de Pitágoras.

- *Measurement*

  Se utilizan objetos del modelo Measurement para almacenar las mediciones hechas en una zona en particular. Este modelo solo se utiliza para conectar los datos medidos (niveles de señal) con las zonas, por lo que solo tiene una referencia a la zona, y una colección de objetos del modelo Position.

  En la clase de la entidad Measurement se encuentran funciones para manejar (eliminar, crear) objetos del modelo Position, así como funciones utilitarias que

devuelven los objetos Position convertidos a un arreglo para facilitar su procesamiento en el modelo Space.

- *Position*

  Este modelo es el que almacena cada nivel de señal medido. Tiene una referencia a un objeto del modelo Measurement, la dirección del punto de acceso leído, y el nivel de señal ajustado de lo medido en decibeles.

  Este modelo no hace nada más que almacenar los datos.

4.3.2 Interfaz API para comunicación con el receptor.

Cada llamada posible es representada por una "acción" en un "controlador". Zend Framework provee la funcionalidad necesaria para implementar los controladores. En este proyecto se utilizaron las rutas predeterminadas dadas por Zend Framework, cuyo URL es "controlador/acción", por ejemplo: http://tesis.cambraca.com/space/all para la acción "all" del controlador "space".

- *Space/all*

  Esta acción devuelve la lista completa de espacios almacenados en la base de datos, en formato JSON.

- *Zone/save*

  Para almacenar una medición de una zona, es decir un conjunto de niveles de señal para los puntos de acceso encontrados, se utiliza esta acción. Recibe el nombre del espacio y de la zona, así como la medición, y un parámetro adicional que sirve para eliminar todas las mediciones anteriores en esa misma zona antes de guardar la medición recibida.

- *Zone/lookup*

  Esta acción recibe como parámetros el nombre de un espacio, y una lista de niveles de señal medidos, con sus respectivas direcciones de puntos de acceso. Devuelve la zona más cercana encontrada, así como un número de 0 a 3 que indica el nivel de confianza. Un nivel de confianza más alto significa que es más seguro que el resultado sea correcto, es decir que el receptor está realmente dentro de la zona.

4.3.3 Otras acciones.

Los controladores tienen algunas acciones adicionales, las cuales sirven para observar el funcionamiento, por ejemplo visualizar una red neuronal para un espacio. Estas se pueden acceder a través de un browser.

- *Space/view*

  Muestra todas las zonas y mediciones de un espacio, cuyo nombre es recibido en el URL. Por ejemplo: /space/view/space/departamento.

- *Space/view-neural-network*

  Muestra datos sobre la red neuronal de un espacio. Utiliza una función de la librería para generar HTML, el cual contiene información como el tiempo que demoró el entrenamiento de la red, así como una representación gráfica de la misma.

4.4 Aplicación en el receptor

La aplicación está hecha en el sistema Android, utilizando el lenguaje de programación Java, pero se podría adaptar a cualquier otro sistema que permita obtener las mediciones de niveles de señal de Wi-Fi en un dispositivo móvil.

4.4.1 Funciones de la aplicación.

La aplicación tiene dos funciones principales:

1. Reportar la "zona" actual en base a un mapa creado anteriormente

   Esta es la operación normal de la aplicación. Constantemente consulta con el Back-end, enviando los niveles de potencia de puntos de acceso cercanos, y muestra al usuario en qué zona se encuentra.

2. Crear un "mapa" (un "espacio" con varias "zonas")

   Durante este proceso la aplicación deberá medir los niveles de potencia de puntos de acceso, y reportarlos al Back-end junto con el nombre de la "zona" actual (por ejemplo, "cocina" o "comedor"). Las zonas se agrupan en un "espacio" (por ejemplo, "departamento").

   Esto se hace una sola vez, y los datos son almacenados en el Back-end. Sin embargo, en caso de que cambien las zonas, o si los puntos de acceso se mueven, lo que ocasiona que los niveles de potencia medidos cambien, el usuario deberá volver a crear el mapa para el espacio.

4.4.2 Estructura interna.

Vistas

La aplicación tiene dos vistas principales, para las dos funciones mencionadas anteriormente:

Diagrama 8: Vistas principales de la aplicación Android

Main: Reporte de zona actual Create_Map: Creación de mapa

- *Vista: Main*

  En esta vista se tiene principalmente el nombre de la zona actual, así como un indicador de confianza (tres estrellas que se activan si la confianza es mayor, reportado por el Back-end). También se tiene un selector de espacios, el cual se llena con la lista de todos los espacios almacenados en el Back-end utilizando la

llamada “Space/all” del API, y un botón para actualizar la zona manualmente. Normalmente se actualiza la zona automáticamente cada cierto tiempo (dado por una constante en el código). Se utiliza la llamada “Zone/lookup” del API del Back-end para obtener la zona actual.

Utilizando el botón de menú de Android, se abre un menú con un único elemento, que sirve para trasladarse a la otra vista.

- *Vista: Create_Map*

  Esta vista sirve para crear el mapa, es decir reportar varias veces los niveles de señal medidos para una zona. Consiste en dos campos de texto para nombrar el espacio y la zona, y un botón que activa la medición y envía los resultados al Back-end. Al activar este botón se hacen 4 mediciones, y en la vista se muestra el progreso. Se utiliza la llamada “Zone/save” del API del Back-end para almacenar los datos.

Clases

Adicional a las vistas, hay dos clases que vale la pena mencionar.

- *RequestHelper*

  Esta clase tiene funciones que ayudan para la comunicación con el Back-end. Las funciones utilizan el API de Android para crear conexiones HTTP y hacer llamadas enviando parámetros.

- *Receiver*

  Esta clase es la que recibe los niveles de señal de puntos de acceso, medidos por el sistema. Se utiliza desde ambas vistas, y hace las llamadas al API del Back-end que sean apropiadas para la función que se está realizando.

5. EVALUACIÓN

Se utilizará como el espacio de prueba un departamento ubicado en una zona central de la ciudad de Quito, para tener suficientes puntos de acceso Wi-Fi disponibles alrededor. Las habitaciones que se usarán se identifican como: dormitorio, cuarto, estudio, baño 1, baño 2, baño 3, sala, comedor, balcón, cocina y lavandería.

Se hará tres conjuntos de pruebas por separado: el primero definiendo solo 3 zonas, correspondientes al dormitorio, cuarto y estudio; el segundo definiendo todas las habitaciones del departamento como las zonas; y un tercero separando el espacio en una cuadrícula con celdas de 2 metros de lado, es decir, ignorando la separación de habitaciones. A cada uno nos referimos como Prueba A, Prueba B y Prueba C, respectivamente.

La razón de hacer estas tres pruebas es intentar identificar qué tanto afecta la efectividad la variación del número de zonas. En particular, se espera que las mediciones sean mejores para la Prueba A que para la Prueba B para las mismas zonas, por el ruido que causa en los cálculos el simple hecho de tener más zonas definidas.

A continuación el plano del departamento, y el esquema de zonas para cada prueba:

Diagrama 9: Plano del departamento de referencia

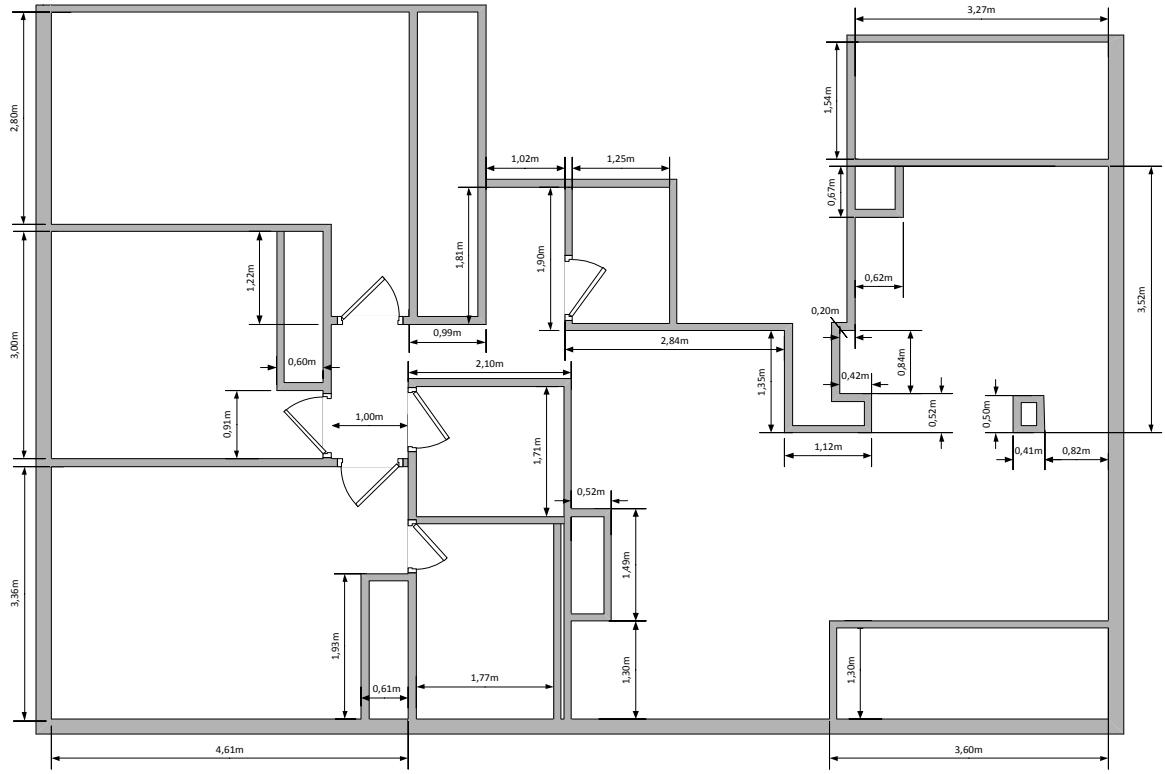


Diagrama 10: Plano con 3 zonas: Prueba A

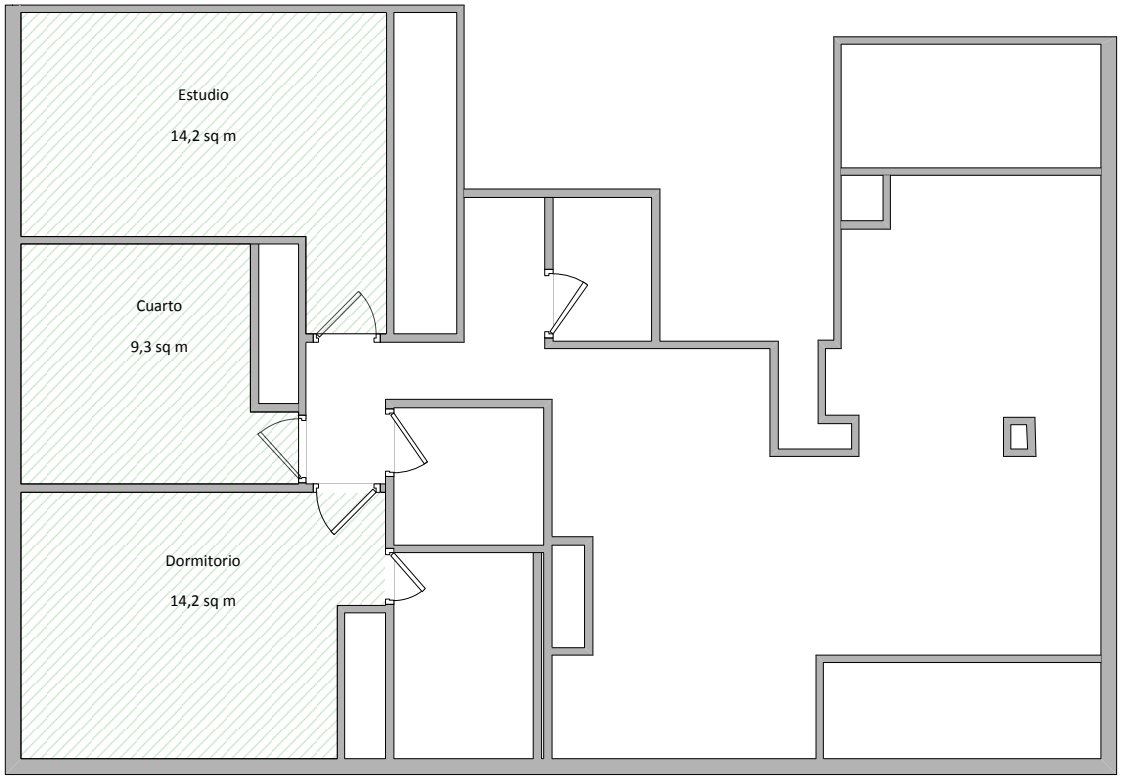


Diagrama 11: Plano con todas las habitaciones como zonas: Prueba B

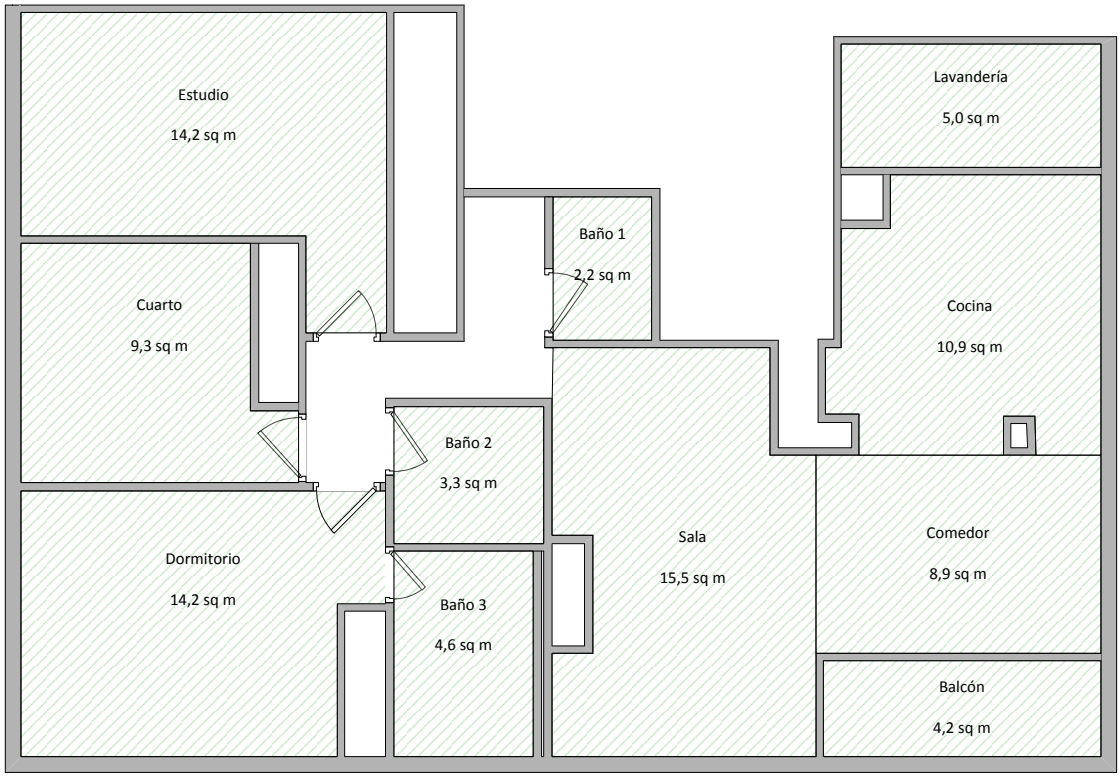
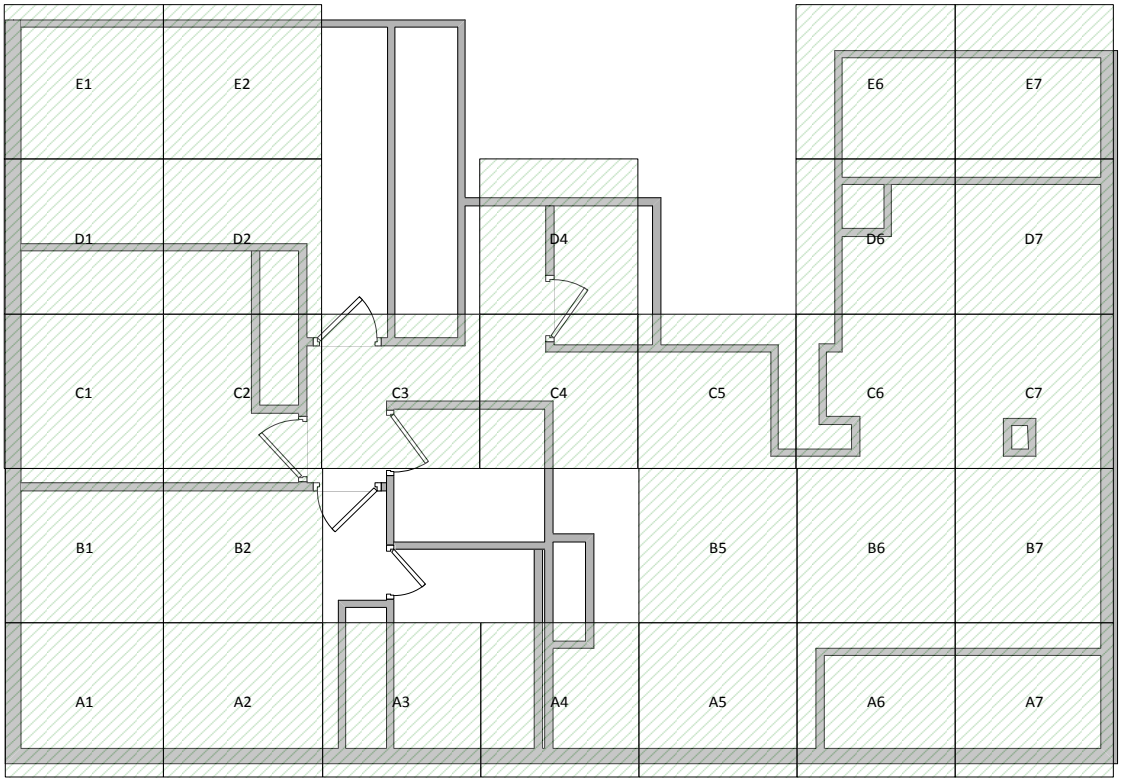


Diagrama 12: Plano con zonas definidas por cuadrícula: Prueba C



La posición del receptor será lo más cercano al centro de cada zona definida. Se omiten algunas zonas (por ejemplo B3 en la Prueba C) por ser de difícil acceso, ya que esta dificultad puede afectar las mediciones.

En cada prueba se hará tres variaciones en las mediciones tomadas. La primera utilizando todos los puntos de acceso que el receptor lee, la segunda limitando a 8 puntos de acceso y una tercera limitando a 4. En los dos últimos esquemas de medición se considerará los puntos de acceso con mayor señal medida.

Para cada combinación de zonas y esquema de selección de puntos de acceso, se creará el mapa correspondiente utilizando la función de crear mapa en el receptor, y luego se procederá a registrar la zona detectada. Se hará 5 intentos en cada zona, y se registrará cada reporte. Adicionalmente, en todas las pruebas, se analizará los resultados utilizando los dos algoritmos presentados anteriormente.

Con todos los datos registrados, se procederá a obtener un promedio del porcentaje de éxito (mediciones correctas) en cada caso. Se calcula también por separado el porcentaje de éxito si la medición es de una zona adyacente. Este cálculo siempre será igual o mayor al primero.

Por ejemplo, en la Prueba A, limitando a 8 puntos de acceso en cada medición, utilizando la red neuronal para el cálculo, se registra la siguiente tabla:

Tabla 3: Ejemplo de registro de errores en mediciones

ID	Zona	8 PAs
A	Dormitorio	· · · · ·
B	Cuarto	C · · X X
C	Estudio	· X X · ·
Estricto		**10/15 = 67%**
Adyacente		**11/15 = 73%**

En la tabla se registra cada medición correcta con un punto; si el algoritmo no llega a ningún resultado se registra una X; en el resto de casos se anota el identificador de la zona reportada. Por ejemplo, la primera medición en el “Cuarto” se calculó como “Estudio”.

El objetivo es lograr una combinación de factores que dé un promedio de error igual o menor al 15%, es decir, que el promedio registrado en la tabla sea de por lo menos 85%.

5.1 Dificultades

Durante la evaluación hubo un número de dificultades, las cuales se enumeran a continuación:

- Las redes neuronales tienen como inputs cada PA, y como outputs cada zona, de manera que cuando se tiene una medición de niveles de PAs, normalmente se activa un output indicando la zona a la que corresponde la medición. Sin embargo, por el funcionamiento de la red no siempre se activa un output. Cuando esto sucede, la aplicación en el teléfono muestra un signo de interrogación indicando que el sistema no pudo determinar la zona actual. Este caso se ha registrado con la letra X en las tablas de resultados, y se considera un error de medición, equivalente a reportar una zona incorrecta no adyacente.
- Para que las redes neuronales generadas logren completar el entrenamiento hubo que limitar los inputs de la red a los PAs que cumplían un mínimo de mediciones. Por ejemplo, en la prueba B, donde se realizaron 44 mediciones para el entrenamiento, si un PA en particular solo aparece en una de las mediciones, no se

lo toma en cuenta. Luego de varios intentos en los que las redes no llegaban a entrenarse al 100%, este número se fijó en 6 (cada PA debe estar presente en por lo menos 6 mediciones), sin importar la cantidad de zonas registradas. Esto asegura que cada PA se haya leído desde por lo menos dos zonas. Esto no es un problema, por supuesto, para el algoritmo de Pitágoras.

- El sistema de red neuronal, durante la Prueba A no tuvo ningún problema durante el entrenamiento, el cual se completó en pocos segundos por la baja cantidad de mediciones (4 en cada zona, 12 en total). En la prueba B, sin embargo, este tiempo subió considerablemente a alrededor de 6 minutos (esta vez fueron 44 mediciones para el entrenamiento). La red nunca se entrenó con la opción de limitar a 8 PAs, razón por la cual no existen estos datos. Lo mismo sucedió en todos los casos de la prueba C.

5.2 Resultados

Los datos medidos en cada caso se anexan al final del documento. La siguiente tabla resume todos los resultados:

Tabla 4: Resumen de resultados

Algoritmo		Teorema de Pitágoras			Red neuronal			Promedio
Límite de PAs		Sin límite	8	4	Sin límite	8	4	
Prueba A (3 zonas)	Estricto	40%	60%	60%	80%	67%	40%	**58%**
	Adyacente	67%	87%	100%	93%	73%	67%	**81%**
Prueba B (11 zonas)	Estricto	25%	42%	62%	35%	N/A	44%	**42%**
	Adyacente	62%	69%	78%	62%	N/A	69%	**68%**
Prueba C (28 zonas)	Estricto	17%	29%	26%	N/A	N/A	N/A	**24%**
	Adyacente	56%	69%	68%	N/A	N/A	N/A	**64%**
Promedio		**45%**	**59%**	**66%**	**68%**	**70%**	**55%**	

Las celdas de la última fila y la última columna muestran promedios de cada columna y fila, respectivamente.

5.2.1 Interpretación de resultados.

Cuando se consideran como correctos los reportes de zonas adyacentes, el porcentaje de éxito sube considerablemente. Esto es interesante sobre todo en la prueba C, donde existe una gran cantidad de zonas. Reportes de zonas adyacentes sugieren que dichos reportes no son aleatorios, sino solo que existe un margen de error en las mediciones.

Como se puede esperar, un aumento en el número de zonas vuelve más complejos los cálculos y más impredecibles los resultados. Esto se ve claramente en casi todos los

casos, donde el porcentaje de mediciones correctas baja de la prueba A a la prueba B, y a la prueba C. Este cambio es menos notorio, aunque sigue presente, cuando se consideran reportes de zonas adyacentes como correctos.

La red neuronal claramente no es apropiada en algunos casos, ya que no se llega a entrenar al 100% (celdas marcadas como "N/A" en la tabla anterior). Adicional a esto, la implementación actual en los casos con 11 zonas (prueba B) es poco práctica ya que el tiempo de entrenamiento de la red es de varios minutos y hace lento el proceso. Esto se podría mejorar teniendo un proceso en background que entrene la red neuronal mientras se sigue utilizando el sistema. Sin embargo, si se toman los valores de los casos que sí funcionaron, y los equivalentes en el otro algoritmo, se obtiene que el promedio utilizando redes neuronales es de 53% (73% con zonas adyacentes), mientras que utilizando el teorema de Pitágoras es de 49% (79% con zonas adyacentes), con lo que no existe un claro "ganador" entre los dos algoritmos utilizados.

El otro factor que se probó fue limitar los PAs utilizados. "Sin límite" significa que se utilizan todas las mediciones, tanto para crear el mapa como para los reportes. En la prueba A es difícil sacar conclusiones ya que el número de zonas es muy bajo, y las variaciones que se ven pueden suceder por otros factores. La prueba C, por su lado, no tiene datos para el algoritmo de redes neuronales, así que usamos solo la prueba B para determinar si limitar los PAs es útil o no. En todos los casos los resultados son mejores limitando los PAs, lo que indica que al estar los mismos más lejos, los niveles de señal medidos son más impredecibles. Poniendo este límite se filtran los PAs más lejanos, lo que hace más confiable el sistema. La red neuronal parece manejar mejor (hay menos variación) una gran cantidad de mediciones lejanas.

6. CONCLUSIONES Y RECOMENDACIONES

Luego de varias pruebas, el sistema de detección de zonas parece funcionar, aunque tiene un margen de error mayor al esperado. La prueba B es la más realista, no tiene muy pocas zonas (lo que haría poco útil al sistema), pero no tantas como para hacer demasiado tedioso el proceso de creación del mapa. Hay que recordar que si los PAs se mueven de posición o si se instala uno nuevo (cambios que no se pueden controlar), el mapa en la base de datos queda obsoleto y es necesario crear uno nuevo desde cero, por lo que tener demasiadas zonas hace difícil el uso prolongado del sistema.

En la introducción se determinó que si el 15% o más de las mediciones son incorrectas, el proyecto no se considera exitoso. Esto se cumple solo en la prueba A, y tomando las zonas adyacentes como correctas. En la prueba B, que es la más realista, este porcentaje es más cercano al 30% (todavía con zonas adyacentes). En conclusión, usando los métodos planteados en este proyecto, el sistema no es lo suficientemente preciso.

Una opción para futuros estudios es trabajar en algún método que calcule un "promedio" entre varias de las últimas mediciones, no solo en el proceso de creación del mapa sino en la detección en tiempo real. También sería útil detectar y descartar mediciones muy lejanas (outliers) que puedan afectar a dicho promedio. Esto podría estabilizar los datos, con lo que los cálculos pueden llegar a ser más precisos. Una desventaja es que se demoraría un momento en detectar cuando el dispositivo cambia de zona.

Algo que también se podría medir es cuánto afecta la interferencia de objetos de varios materiales. En este proyecto se ha asumido una constante de 4,5 para la ecuación dada en la sección 2.3, la cual corresponde a la cantidad de obstrucciones. Esto es algo que

no se puede conocer en un espacio real, donde los PAs que se miden pueden estar desde en la misma habitación (ninguna obstrucción) hasta en departamentos lejanos separados por distintos tipos de material de construcción. Algo que valdría la pena explorar es la variación de esta constante, tal vez preguntando al usuario cuáles PAs son muy cercanos (se encuentran dentro del departamento donde se mide, por ejemplo), y definir para esos una constante diferente. Al ser más cercanos, la señal medida es mucho más fuerte, y hay mayor probabilidad de que ese PA aparezca en la mayoría de zonas, por lo tanto esos PAs son importantes para la medición en general.

Por último, sería interesante hacer pruebas en espacios distintos. Por ejemplo, en zonas urbanas y zonas rurales, donde la cantidad de redes Wi-Fi es muy diferente, y determinar si este factor afecta a los resultados.

REFERENCIAS

GPS vs. aGPS: A Quick Tutorial. (3 de Enero de 2009). Recuperado el 22 de Abril de 2013, de http://www.wpcentral.com/gps-vs-agps-quick-tutorial

Bravo, C. (s.f.). Obtenido de CModel: http://cambraca.github.io/CModel/

DePriest, D. (12 de Agosto de 1999). *How Your GPS Works*. Recuperado el 22 de Abril de 2013, de http://www.gpsinformation.org/dale/theory.htm

Faria, D. B. (2005). *Modeling Signal Attenuation in IEEE 802.11 Wireless LANs - Vol. 1.* Stanford University.

Nicholson, A. J., Chawathe, Y., Chen, M. Y., Noble, B. D., & Wetherall, D. (2006). *Improved Access Point Selection.* Uppsala, Sweden.

Página GPS Española. (s.f.). *Test que Predice la Precisión de un Receptor Portátil*. Recuperado el 22 de Abril de 2013, de http://www.elgps.com/documentos/consejos/PrecisionGarmin.html

What is trilateration? (s.f.). Recuperado el 15 de 12 de 2014, de Mio Technology: http://www.mio.com/technology-trilateration.htm

Wien, T. (s.f.). Recuperado el 26 de Abril de 2013, de Artificial Neural Network for PHP 5.x: http://ann.thwien.de

Young, M. F. (s.f.). *Understanding Decibels and Their Use in Radio Systems.* Obtenido de Wireless Telecommunications Bureau: http://wireless.fcc.gov/outreach/2004broadbandforum/comments/YDI_understandingdb.pdf

Zend Technologies Ltd. (s.f.). Obtenido de Zend Framework: http://framework.zend.com/

ANEXO A: CÓDIGO FUENTE BACK-END

Se muestra solo el código relevante al proyecto. El resto es generado por Zend Framework, y se omite acá.

Script para la creación de la base de datos MySQL

```
CREATE TABLE `space` (
  `id` char(6) NOT NULL DEFAULT '',
  `name` varchar(255) NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB;

CREATE TABLE `zone` (
  `id` char(6) NOT NULL DEFAULT '',
  `space_id` char(6) DEFAULT NULL,
  `name` varchar(255) NOT NULL,
  PRIMARY KEY (`id`),
  KEY `fk_zone_space` (`space_id`),
  CONSTRAINT `zone_ibfk_1` FOREIGN KEY (`space_id`) REFERENCES `space` (`id`) ON DELETE
CASCADE
) ENGINE=InnoDB;

CREATE TABLE `measurement` (
  `id` char(6) NOT NULL,
  `zone_id` char(6) NOT NULL,
  PRIMARY KEY (`id`),
  KEY `fk_measurement_zone` (`zone_id`),
  CONSTRAINT `measurement_ibfk_1` FOREIGN KEY (`zone_id`) REFERENCES `zone` (`id`) ON
DELETE CASCADE
) ENGINE=InnoDB;

CREATE TABLE `position` (
  `id` char(6) NOT NULL DEFAULT '',
  `measurement_id` char(6) NOT NULL,
  `address` char(17) NOT NULL,
  `signal` decimal(5,2) NOT NULL,
  PRIMARY KEY (`id`),
  KEY `fk_position_measurement` (`measurement_id`),
  CONSTRAINT `position_ibfk_1` FOREIGN KEY (`measurement_id`) REFERENCES `measurement`
(`id`) ON DELETE CASCADE
) ENGINE=InnoDB;
```

Modelos

Los modelos son las clases que contienen la lógica principal de la aplicación.

/application/models/Config.php

```
/**
* Config class. Stores configuration in a text file.
*/
class Config {
    private static $defaults = array(
      'Algorithm' => 'Neural Network',
      'AP Limit' => 'No limit',
    );

    /**
    * Returns the possible changes to configuration variables.
    */
    static function changeMatrix() {
        return array(
          'Algorithm' => array(
            'Neural Network' => array('Pythagoras'),
            'Pythagoras' => array('Neural Network'),
          ),
          'AP Limit' => array(
            'No limit' => array('4', '8'),
            '4' => array('No limit', '8'),
            '8' => array('No limit', '4'),
          ),
        );
    }

    /**
    * Retrieves a configuration variable.
    */
    static function get($name) {
        if (!file_exists(self::filename()))
          return self::$defaults[$name];
        $data = json_decode(file_get_contents(self::filename()), TRUE);
        if (isset($data[$name]))
          return $data[$name];
        else
          return self::$defaults[$name];
    }

    /**
    * Stores a configuration variable into a JSON text file.
```

```
    */
    static function set($name, $value) {
        $data = array();
        if (file_exists(self::filename()))
          $data = json_decode(file_get_contents(self::filename()), TRUE);

        if (!is_array($data))
          $data = array();

        $data[$name] = $value;
        file_put_contents(self::filename(), json_encode($data));
    }

    /**
    * Reads all configuration variables stored in the file.
    */
    static function getAll() {
        if (!file_exists(self::filename()))
          return self::$defaults;
        return json_decode(file_get_contents(self::filename()), TRUE)
          + self::$defaults;
    }

    /**
    * Returns an absolute path for the configuration file.
    */
    private static function filename() {
        return realpath(APPLICATION_PATH.'/..').'/project.conf';
    }
}
```

/application/models/Space.php

```
use ANNNetwork;
use ANNClassification;
use ANNValues;
use ANNInputValue;

/**
* Space class.
*
* @property string $id The unique id of this space;
* @property string $name The name of this space;
* @property-read Zone[] $zones All the zones defined in the current space;
*/
class Space extends CModel_Entity {
    const NEURAL_NETWORK_ERROR_TOLERANCE = 0.1; //default is .02, less precise is 0.1
    const NEURAL_NETWORK_LAYERS = 2;
```

```
protected $_data = array(
  'id'    => NULL,
  'name'    => '',
  'zones'    => NULL,
);

private $_addresses_cache = array();

public function __construct($data = NULL) {
    parent::__construct($data);
    $this->setCollection('zones', 'Zone', 'space_id');
}

public static function byName($name) {
    $space = SpaceMapper::singleton()->findByName($name);
    if (!$space) {
        $space = new Space(array('name' => $name));
        SpaceMapper::singleton()->save($space);
    }
    return $space;
}

public function zoneByName($name)
{
    $zone = ZoneMapper::singleton()->findByName($name, $this);
    if (!$zone) {
        $zone = new Zone(array('name' => $name, 'space' => $this));
        ZoneMapper::singleton()->save($zone);
    }
    return $zone;
}

/**
* Performs a lookup.
* @param Array $levels keyed by MAC address, values are signal levels
* @return Array first element is confidence (int, 0 to 3, 0 means no zone
* was found), second is Zone object
*/
public function lookupZone($levels) {
    foreach ($levels as &$level)
    $level = intval($level);

    if (Config::get('Algorithm') == 'Neural Network')
      return $this->_lookupZoneNeuralNetwork($levels);
    else
      return $this->_lookupZonePythagoras($levels);
```

```
}

public function printNeuralNetwork($notrain = FALSE) {
    list($objNetwork, $objValues, $objClassification) = $this->_loadNeuralNetwork();
    if (!$notrain)
      $objNetwork->train();
    $objNetwork->saveToFile($this->_nnFilenamePrefix().'strings.dat');
    $objNetwork->printNetwork();
}

private function _loadNeuralNetwork() {
    if (!file_exists($this->_nnFilenamePrefix().'strings.dat'))
      $this->_trainNN();

    $objNetwork = Network::loadFromFile(
      $this->_nnFilenamePrefix().'strings.dat'
    );
    $objNetwork->setOutputErrorTolerance(self::NEURAL_NETWORK_ERROR_TOLERANCE);
    $objValues = Values::loadFromFile(
      $this->_nnFilenamePrefix().'values_strings.dat'
    );
    $objClassification = Classification::loadFromFile(
      $this->_nnFilenamePrefix().'classifiers_strings.dat'
    );

    return array($objNetwork, $objValues, $objClassification);
}

private function _lookupZoneNeuralNetwork($levels) {
    list($objNetwork, $objValues, $objClassification) = $this->_loadNeuralNetwork();

    $objValues = new Values();
    $objValues->input(self::_nnInputValue(
      Measurement::signalsArray2($this->getAllAddresses(), $levels)
    ));
    $objNetwork->setValues($objValues);
    $arrOutputs = $objNetwork->getOutputs();

    $outs = array();
    foreach ($arrOutputs as $arrOutput) {
        $out = $objClassification->getRealOutputValue($arrOutput);
        if (count($out) < 1)
          continue;

        $out = $out[0];

        if (array_key_exists($out, $outs))
```

```
            $outs[$out]++;
          else
            $outs[$out] = 1;
      }
      arsort($outs);

      if (count($outs) == 0)
        return array('?', 0);

      if (count($outs) == 1)
        return array(ZoneMapper::singleton()->find(key($outs)), 3);

      $temp = array_values($outs);
      rsort($temp);
      if ($temp[0] == $temp[1]) //a tie between two zones
        return array('?', 0);

      return array(ZoneMapper::singleton()->find(key($outs)), max(1, 4-count($outs)));
  }

  /**
  * Returns the base path for neural network files, e.g. "/path/to/spaceid_".
  */
  private function _nnFilenamePrefix() {
      $dir = realpath(APPLICATION_PATH.'/..').'/nn_data';
      @mkdir($dir, 0777, TRUE);
      return $dir.'/'.$this->id.'_';
  }

  /**
  * Deletes all cache data for this Space, including its neural network, if any.
  */
  public function invalidateCaches() {
      $this->_addresses_cache = NULL;
      if ($this->id) {
          unlink($this->_nnFilenamePrefix().'strings.dat');
          unlink($this->_nnFilenamePrefix().'classifiers_strings.dat');
          unlink($this->_nnFilenamePrefix().'values_strings.dat');
      }
  }

  private function _trainNN()
  {
      if (!$this->id)
        throw new Exception();
      try {
          $objNetwork = Network::loadFromFile(
```

```
            $this->_nnFilenamePrefix().'strings.dat'
        );
        $objNetwork->setOutputErrorTolerance(self::NEURAL_NETWORK_ERROR_TOLERANCE);
    } catch (Exception $e) {
        $zone_count = count($this->zones);

        $objClassification = new Classification($zone_count);
        foreach ($this->zones as $zone)
          $objClassification->addClassifier($zone->id);

        $objClassification->saveToFile(
          $this->_nnFilenamePrefix().'classifiers_strings.dat'
        );

        $objNetwork = new Network(
          self::NEURAL_NETWORK_LAYERS,
          2 * max($zone_count, count($this->getAllAddresses())),
          $zone_count
        );
        $objNetwork->setOutputErrorTolerance(self::NEURAL_NETWORK_ERROR_TOLERANCE);

        $objValues = new Values;
        $objValues->train();

        $zz = $this->zones;
        $all = $this->getAllAddresses();

        foreach ($this->zones as $zone) {
            foreach ($zone->measurements as $measurement)
                $objValues
                  ->input(self::_nnInputValue($measurement->signalsArray($all)))
                  ->output($objClassification($zone->id));
        }

        $objValues->saveToFile($this->_nnFilenamePrefix().'values_strings.dat');
        unset($objValues);
    }

    $objValues = Values::loadFromFile(
      $this->_nnFilenamePrefix().'values_strings.dat'
    );

    $objNetwork->setValues($objValues);
    $objNetwork->train();
    $objNetwork->saveToFile(
      $this->_nnFilenamePrefix().'strings.dat'
```

```
        );
    }

    /**
    * @param array|number $mixed
    * @return mixed
    */
    private static function _nnInputValue($mixed) {
        $ret = array();
        $input_value = new InputValue(Position::MIN_SIGNAL, 0, TRUE);
        foreach (is_array($mixed) ? $mixed : array($mixed) as $value)
          $ret[] = $input_value->getInputValue($value);
        return is_array($mixed) ? $ret : $ret[0];
    }

    /**
    * Gets a list of all the MAC addresses associated to zones in this space.
    * @param bool $all if FALSE, only returns APs that are "used" by more than half the
zones in this space
    */
    public function getAllAddresses($all = FALSE) {
        if (isset($this->_addresses_cache[$all ? 1 : 2]))
          return $this->_addresses_cache[$all ? 1 : 2];

        $zone_count = count($this->zones);
        $threshold = 6; //used when $all is FALSE

        $counts = array();
        $ret = array();
        foreach ($this->zones as $zone) {
            foreach ($zone->measurements as $measurement) {
                foreach ($measurement->positions as $position) {
                    if ($position->signal > Position::MIN_SIGNAL)
                        $counts[$position->address]++;
                }
            }
        }
        foreach ($counts as $address => $count) {
            if ($all || $count >= $threshold)
            $ret[] = $address;
        }

        $this->_addresses_cache[$all ? 1 : 2] = $ret;
        return $this->_addresses_cache[$all ? 1 : 2];
    }
```

```
    private function _lookupZonePythagoras($levels) {
        $distance = -Position::MIN_SIGNAL + 1; $zone = NULL;
        foreach ($this->zones as $current_zone) {
            $current_distance = $current_zone->distanceTo($levels);
            if (!is_null($current_distance) && $current_distance < $distance) {
                $distance = $current_distance;
                $zone = $current_zone;
            }
        }

        $confidence = 0;
        if ($zone && $distance <= 10)
          $confidence = 3;
        elseif ($zone && $distance <= 20)
          $confidence = 2;
        elseif ($zone && $distance <= 30)
          $confidence = 1;

        return array($zone, $confidence);
    }
}
```

/application/models/Zone.php

```
/**
* Zone class.
*
* @property string $id The unique id for this zone;
* @property Space $space The space this zone belongs to;
* @property string $name The name of this zone;
* @property Measurement[] $measurements A list of measurements defined in this zone;
*/
class Zone extends CModel_Entity {
    protected $_data = array(
      'id'            => NULL,
      'space'            => NULL,
      'name'           => '',
      'measurements'    => NULL,
    );

    public function __construct($data = NULL) {
        parent::__construct($data);
        $this->setCollection('measurements', 'Measurement', 'zone_id');
    }
    public function __get($name) {
        switch ($name)
        {
            case 'space':
```

```
            if ($this->getReferenceId($name) && !$this->_data[$name] instanceof Space)
              $this->_data[$name] = SpaceMapper::singleton()->find(
                $this->getReferenceId($name)
              );
            break;
          default:
      }
      return parent::__get($name);
  }

  public function clearMeasurements() {
      $mapper = MeasurementMapper::singleton();
      foreach ($this->measurements as $m)
        $mapper->delete($m);

      $this->setCollection('measurements', 'Measurement', 'zone_id');
  }

  /**
  * Creates and returns a new Measurement object for this zone.
  * @return Measurement
  */
  public function newMeasurement() {
      if (!$this->id)
        throw new Exception();
      $ret = new Measurement(array('zone' => $this));
      MeasurementMapper::singleton()->save($ret);
      return $ret;
  }

  /**
  * Returns the average of all stored measurements for this zone.
  * The returned object is never stored in the database.
  * @return Measurement
  */
  public function averagePositions() {
      $ret = array();

      foreach ($this->measurements as $measurement) {
          foreach ($measurement->positions as $position)
            $ret[$position->address][] = $position->signal;
      }

      foreach ($ret as &$item)
        $item = intval(array_sum($item) / count($item));

      return $ret;
```

```
    }

    /**
    * Returns the distance to this zone. Calculate using Pythagoras' theorem.
    * @param array $levels keyed by MAC address, values are signal levels
    * @return float|NULL returns NULL if no MAC addresses match stored values
    */
    public function distanceTo($levels) {
        $positions = $this->averagePositions();
        $distance = 0; $return_null = TRUE;
        foreach ($positions as $pos_address => $pos_signal) {
            $found = FALSE;
            foreach ($levels as $address => $signal) {
                if ($pos_address == $address) {
                    $found = TRUE;
                    $return_null = FALSE;
                    $distance += pow($signal - $pos_signal, 2);
                    break;
                }
            }
            if (!$found) {
                $return_null = FALSE;
                $distance += pow(Position::MIN_SIGNAL - $pos_signal, 2);
                break;
            }
        }
        foreach ($levels as $address => $signal) {
            $found = FALSE;
            foreach ($positions as $pos_address => $pos_signal) {
                if ($pos_address == $address) {
                    $found = TRUE;
                    break;
                }
            }
            if (!$found) {
                $return_null = FALSE;
                $distance += pow($signal - Position::MIN_SIGNAL, 2);
                break;
            }
        }
        return $return_null ? NULL : sqrt($distance);
    }
}
```

/application/models/Measurement.php

```
/**
* Measurement class. Represents a measurement for a zone (a set of Position objects).
*
* @property string $id The unique id for this measurement;
* @property Zone $zone The zone this measurement is related to;
* @property Position[] $positions A list of the access points defined in this measuremen
t;
*/
class Measurement extends CModel_Entity {
    protected $_data = array(
      'id'        => NULL,
      'zone'        => NULL,
      'positions'    => NULL,
    );

    public function __get($name) {
        switch ($name) {
            case 'zone':
              if ($this->getReferenceId($name) && !$this->_data[$name] instanceof Zone)
                $this->_data[$name] = ZoneMapper::singleton()->find(
                  $this->getReferenceId($name)
                );
              break;
            case 'positions':
              $ap_limit = Config::get('AP Limit');
              if (is_numeric($ap_limit) && $ap_limit > 0) {
                //has limit
                return PositionMapper::singleton()->findStrongest($this, $ap_limit);
              } else {
                //has no limit
                return new CModel_Collection('Position', 'measurement_id', $this);
              }
              break;
            default:
        }
        return parent::__get($name);
    }

    public function clearPositions() {
        $mapper = PositionMapper::singleton();
        foreach ($this->positions as $position)
          $mapper->delete($position);
    }
```

```
    public function setPosition($address, $signal) {
        $found = FALSE;
        foreach ($this->positions as $position)
        if ($position->address == $address) {
            $found = TRUE;
            break;
        }

        if (!$found)
          $position = new Position(array('measurement' => $this));

        $position->address = $address;
        $position->signal = $signal;

        PositionMapper::singleton()->save($position);
    }

    /**
    * Convenience method.
    * @param Array $addresses for ex. array('12:34:56:67','64:a3:b8:22:31'...)
    * @return Array signals in the same order as $addresses
    */
    public function signalsArray($addresses) {
        $signals = array();
        foreach ($this->positions as $position)
          $signals[$position->address] = $position->signal;

        return self::signalsArray2($addresses, $signals);
    }

    /**
    * Convenience method.
    */
    public static function signalsArray2($addresses, $signals) {
        $ret = array();
        foreach ($addresses as $address) {
            if (array_key_exists($address, $signals))
              $ret[] = $signals[$address];
            else
              $ret[] = Position::MIN_SIGNAL;
        }
        return $ret;
    }
}
```

/application/models/Position.php

```
/**
* Position class. Represents a single access point, and stores its signal level
* in the specified measurement for a zone.
*
* @property string $id The unique id for this position;
* @property Measurement $measurement The measurement this position is related to;
* @property string $address The MAC address of the access point;
* @property float $signal The measured signal level for this access point;
*/
class Position extends CModel_Entity {
    const MIN_SIGNAL = -100;

    protected $_data = array(
      'id'            => NULL,
      'measurement'    => NULL,
      'address'        => '',
      'signal'        => '',
    );

    public function __get($name) {
        switch ($name) {
            case 'measurement':
              if ($this->getReferenceId($name)
                && !$this->_data[$name] instanceof Measurement)
                $this->_data[$name] = MeasurementMapper::singleton()->find(
                  $this->getReferenceId($name)
                );
              break;
            default:
        }
        return parent::__get($name);
    }

    /**
    * Perform the necessary adjustment to the signal in decibels.
    * The formula is:
    *         x' = -pow(10, (-x / 10a))
    * Where "a" is 4.5 and represents how obstructed the environment is.
    *
    * @param int $signal
    */
    public static function adjust($signal) {
        return max(self::MIN_SIGNAL, intval(-pow(10, $signal / -45)));
    }
}
```

Mappers

Los “mappers” son clases que se dedican a interactuar con la base de datos para crear, editar y eliminar registros, los cuales se mapean a los modelos. Así se logra tener la lógica de los modelos separada de la de almacenaje.

/application/models/SpaceMapper.php

```
/**
* Space Mapper class.
*/
class SpaceMapper extends CModel_Mapper {
    protected $_tableName = 'space';
    protected $_entityClass = 'Space';

    public function save(Space $space) {
        if (!$space->id || $space->hasChanged()) {
            $space->validateRequiredProperties();
            $space->saving();
            $data = array(
              'name'    => $space->name,
            );
            if (!$space->id) {
                $space->id = $this->_getGateway()->insert($data);
                $this->_setIdentity($space->id, $space);
            } else {
                $where = $this->_getGateway()->getAdapter()
                  ->quoteInto('id = ?', $space->id);
                $this->_getGateway()->update($data, $where);
            }
            $space->saved();
        }
    }

    public function find($id) {
        if (is_object($id)) {
            $result = $id;
            $id = $result->id;
        }
        if ($this->_getIdentity($id)) {
            return $this->_getIdentity($id);
        }
        if (!isset($result))
          $result = $this->_getGateway()->find($id)->current();
        $space = new $this->_entityClass(array(
```

```
            'id'      => $result->id,
            'name'    => $result->name,
        ));

        $this->_setIdentity($id, $space);
        return $space;
    }

    public function findByName($name) {
        $table = $this->_getGateway();

        $select = $table->select(TRUE)
          ->reset(Zend_Db_Table::COLUMNS)
          ->columns('id')
          ->where('name = ?', strtoupper(trim($name)));

        $row = $table->fetchRow($select);
        return $row ? $this->find($row->id) : NULL;
    }

    public function delete(Space $object) {
        if ($object->id) {
            $where = $this->_getGateway()->getAdapter()
            ->quoteInto('id = ?', $object->id);
            $this->_getGateway()->delete($where);
        }
    }
}
```

/application/models/ZoneMapper.php

```
/**
* Zone Mapper class.
*/
class ZoneMapper extends CModel_Mapper {
    protected $_tableName = 'zone';
    protected $_entityClass = 'Zone';

    public function save(Zone $zone) {
        if (!$zone->id || $zone->hasChanged()) {
            $zone->validateRequiredProperties();
            $zone->saving();
            $data = array(
              'space_id'    => $zone->space->id,
              'name'        => $zone->name,
            );
            if (!$zone->id) {
```

```
            $zone->id = $this->_getGateway()->insert($data);
            $this->_setIdentity($zone->id, $zone);
        } else {
            $where = $this->_getGateway()->getAdapter()
              ->quoteInto('id = ?', $zone->id);
            $this->_getGateway()->update($data, $where);
        }
        $zone->saved();
    }
    $zone->space->invalidateCaches();
}

public function find($id) {
    if (is_object($id)) {
        $result = $id;
        $id = $result->id;
    }
    if ($this->_getIdentity($id)) {
        return $this->_getIdentity($id);
    }
    if (!isset($result))
      $result = $this->_getGateway()->find($id)->current();
    $zone = new $this->_entityClass(array(
      'id'        => $result->id,
      'name'      => $result->name,
    ));
    $zone->setReferenceId('space', $result->space_id);

    $this->_setIdentity($id, $zone);
    return $zone;
}

public function findByName($name, $space) {
    $table = $this->_getGateway();

    $select = $table->select(TRUE)
      ->reset(Zend_Db_Table::COLUMNS)
      ->columns('id')
      ->where('space_id = ?', $space->id)
      ->where('name = ?', trim(strtoupper($name)));

    $row = $table->fetchRow($select);
    return $row ? $this->find($row->id) : NULL;
}

public function delete(Zone $object) {
    if ($object->id) {
```

```
            $where = $this->_getGateway()->getAdapter()
              ->quoteInto('id = ?', $object->id);
            $this->_getGateway()->delete($where);
        }
    }
}
```

/application/models/MeasurementMapper.php

```
/**
* Measurement Mapper class.
*/
class MeasurementMapper extends CModel_Mapper {
    protected $_tableName = 'measurement';
    protected $_entityClass = 'Measurement';

    public function save(Measurement $measurement) {
        if (!$measurement->id || $measurement->hasChanged()) {
            $measurement->validateRequiredProperties();
            $measurement->saving();
            $data = array(
              'zone_id'    => $measurement->zone->id,
            );
            if (!$measurement->id) {
                $measurement->id = $this->_getGateway()->insert($data);
                $this->_setIdentity($measurement->id, $measurement);
            } else {
                $where = $this->_getGateway()->getAdapter()
                  ->quoteInto('id = ?', $measurement->id);
                $this->_getGateway()->update($data, $where);
            }
            $measurement->saved();
        }
        $measurement->zone->space->invalidateCaches();
    }

    public function find($id) {
        if (is_object($id)) {
            $result = $id;
            $id = $result->id;
        }
        if ($this->_getIdentity($id)) {
            return $this->_getIdentity($id);
        }
        if (!isset($result))
          $result = $this->_getGateway()->find($id)->current();
        $measurement = new $this->_entityClass(array(
```

```
            'id'        => $result->id,
        ));
        $measurement->setReferenceId('zone', $result->zone_id);

        $this->_setIdentity($id, $measurement);
        return $measurement;
    }

    public function delete(Measurement $object) {
        if ($object->id) {
            $where = $this->_getGateway()->getAdapter()
            ->quoteInto('id = ?', $object->id);
            $this->_getGateway()->delete($where);
        }
    }
}
```

/application/models/PositionMapper.php

```
/**
* Position Mapper class.
*/
class PositionMapper extends CModel_Mapper {
    protected $_tableName = 'position';
    protected $_entityClass = 'Position';

    public function save(Position $position) {
        if (!$position->id || $position->hasChanged()) {
            $position->validateRequiredProperties();
            $position->saving();
            $data = array(
              'measurement_id'    => $position->measurement->id,
              'address'           => $position->address,
              'signal'            => $position->signal,
            );
            if (!$position->id) {
                $position->id = $this->_getGateway()->insert($data);
                $this->_setIdentity($position->id, $position);
            } else {
                $where = $this->_getGateway()->getAdapter()
                  ->quoteInto('id = ?', $position->id);
                $this->_getGateway()->update($data, $where);
            }
            $position->saved();
        }
        $position->measurement->zone->space->invalidateCaches();
    }
```

```
public function find($id) {
    if (is_object($id)) {
        $result = $id;
        $id = $result->id;
    }
    if ($this->_getIdentity($id)) {
        return $this->_getIdentity($id);
    }
    if (!isset($result))
      $result = $this->_getGateway()->find($id)->current();
    $position = new $this->_entityClass(array(
      'id'         => $result->id,
      'address'    => $result->address,
      'signal'     => intval($result->signal),
    ));
    $position->setReferenceId('measurement', $result->measurement_id);

    $this->_setIdentity($id, $position);
    return $position;
}

public function delete(Position $object) {
    if ($object->id) {
        $where = $this->_getGateway()->getAdapter()
        ->quoteInto('id = ?', $object->id);
        $this->_getGateway()->delete($where);
    }
}

/**
* @param Measurement $measurement
* @param int $limit
* @return CModel_Entity[]
*/
public function findStrongest($measurement, $limit) {
    $table = $this->_getGateway();

    $select = $table->select(TRUE)
      ->reset(Zend_Db_Table::COLUMNS)
      ->columns('id')
      ->where('measurement_id = ?', $measurement->id)
      ->order('signal DESC')
      ->limit($limit);

    $rows = $table->fetchAll($select);
    $ret = array();
```

```
        foreach ($rows as $row)
          $ret[] = $this->find($row->id);
        return $ret;
    }
}
```

Controllers

Los controladores son clases que definen “acciones”. Estos representan páginas o endpoints para la aplicación Android, y básicamente son intermediarios entre vistas y modelos.

/application/controllers/IndexController.php

```
class IndexController extends Zend_Controller_Action {
    /**
    * See a list of spaces, config options, and links.
    */
    public function indexAction()
    {
        $this->view->spaces = new CModel_Collection('Space');
        $this->view->config = Config::getAll();
    }

    /**
    * Changes a configuration setting. Clears all Space caches (e.g. neural networks)
    */
    public function configAction() {
        foreach (new CModel_Collection('Space') as $space)
          $space->invalidateCaches();

        Config::set($this->_getParam('name'), $this->_getParam('value'));
        $this->_helper->getHelper('Redirector')->gotoUrl('/');
    }
}
```

/application/controllers/SpaceController.php

```
class SpaceController extends Zend_Controller_Action {
    /**
    * Defines JSON actions.
    */
    public function init() {
        $ajaxContext = $this->_helper->getHelper('AjaxContext');
        $ajaxContext
        ->addActionContext('all', 'json')
        ->initContext();
    }
    /**
    * Returns names for all the Spaces stored.
    */
    public function allAction() {
        $this->view->spaces = array();
        foreach (new CModel_Collection('Space') as $space)
          $this->view->spaces[] = trim(strtoupper($space->name));
        $this->getHelper('json')->sendJson($this->view);
    }
    /**
    * View a Space (table with zones, measurements, etc.)
    */
    public function viewAction() {
        $this->view->space = Space::byName($this->_getParam('space'));
    }
    /**
    * View a Space's neural network.
    */
    public function viewNeuralNetworkAction() {
        set_time_limit(
          $this->_hasParam('time') ? intval($this->_getParam('time')) : 300
        );
        $space = Space::byName($this->_getParam('space'));
        $space->printNeuralNetwork($this->_hasParam('notrain'));
        exit;
    }
    /**
    * Delete a single Space.
    */
    public function deleteAction() {
        SpaceMapper::singleton()->delete(Space::byName($this->_getParam('space')));
        $this->_helper->getHelper('Redirector')->gotoUrl('/');
    }
}
```

/application/controllers/ZoneController.php

```
class ZoneController extends Zend_Controller_Action {
    /**
    * Performs a Zone lookup. Receives a measurement.
    * Returns a confidence number (single character) and a zone name.
    */
    public function lookupAction() {
        $space = Space::byName(trim(strtoupper($this->_getParam('space'))));
        $levels = $this->_getParam('levels');
        $adjusted_levels = array();
        foreach ($levels as $k => $level)
          $adjusted_levels[$k] = Position::adjust($level);

        list($zone, $confidence) = $space->lookupZone($adjusted_levels);

        $ret = intval($confidence);
        if (!$zone || (is_object($zone) && !trim($zone->name)))
          $ret .= '?';
        elseif (is_object($zone) && is_a($zone, 'Zone'))
          $ret .= trim(strtoupper($zone->name));
        else
          //a plain string was returned
          $ret .= trim(strtoupper($zone));

        echo $ret;
        exit;
    }

    /**
    * Stores a single Measurement for a Zone into the database.
    */
    public function saveAction()
    {
        $space = Space::byName(trim(strtoupper($this->_getParam('space'))));
        $zone = $space->zoneByName(trim(strtoupper($this->_getParam('zone'))));
        if ($this->_hasParam('clear') && $this->_getParam('clear'))
            $zone->clearMeasurements();

        $measurement = $zone->newMeasurement();
        foreach ($this->_getParam('levels') as $address => $level)
          $measurement->setPosition($address, Position::adjust($level));
    }
}
```

Views

Las vistas se utilizan para generar el HTML visto en el navegador cuando se abre una página definida por una acción de un controlador. La mayoría de acciones en este proyecto no tienen una vista correspondiente ya que su output es muy básico.

/application/views/index/index.phtml

```
<h1>WI-FI POSITIONING PROJECT: BACK-END</h1>
<h2>Config</h2>
<table>
  <tr><th>Name</th><th>Value</th><th>Change to</th></tr>
  <?php foreach (Config::changeMatrix() as $name => $changes): ?>
  <tr>
    <td><?= $name ?></td><td><?= $this->config[$name] ?></td>
    <td>
      <?php
        $first = TRUE;
        foreach ($changes[$this->config[$name]] as $change):
          if ($first) $first = FALSE; else print '|'; ?>
          <a href="/index/config/name/<?= $name ?>/value/<?= $change ?>">
            <?= $change ?>
          </a>
      <?php endforeach ?>
    </td>
  </tr>
  <?php endforeach ?>
</table>
<h2>Spaces</h2>
<table>
  <tr><th>ID</th><th>Name</th><th>Zones</th></tr>
  <?php foreach ($this->spaces as $space): ?>
  <tr>
    <td><?= $space->id ?></td><td><?= $space->name ?></td>
    <td><?= count($space->zones) ?></td>
    <td>
      <a href="/space/view/space/<?= $space->name ?>">view</a>
    | <a href="/space/view-neural-network/space/<?= $space->name ?>">neural network</a>
    | <a href="/space/delete/space/<?= $space->name ?>">delete!</a>
    </td>
  </tr>
  <?php endforeach ?>
</table>
```

/application/views/space/view.phtml

```
<h1>Space: <?= $this->space->name ?></h1>

<table>
   <tr>
      <th>ZONE<br/><em>Measurement</em></th>
      <?php
        $within_threshold = $this->space->getAllAddresses();
        foreach ($this->space->getAllAddresses(TRUE) as $address): ?>
          <?php if (in_array($address, $within_threshold)): ?>
            <th style="text-decoration: underline;"><?= $address ?></strong></th>
          <?php else: ?>
            <th><?= $address ?></th>
          <?php endif ?>
      <?php endforeach ?>
   </tr>
   <?php foreach ($this->space->zones as $zone): ?>
      <tr>
         <td><strong><?= $zone->name ?></strong></td>
      </tr>
      <?php foreach ($zone->measurements as $measurement): ?>
      <tr>
         <td><?= $measurement->id ?></td>
         <?php foreach ($this->space->getAllAddresses(TRUE) as $address): ?>
         <td>
         <?php
            $found = FALSE;
            foreach ($measurement->positions as $position) {
               if ($position->address == $address) {
                  echo $position->signal;
                  $found = TRUE;
               }
            }
            if (!$found)
               echo '<span style="opacity: .3">'.Position::MIN_SIGNAL.'</span>';
         ?>
         </td>
         <?php endforeach ?>
      </tr>
      <?php endforeach ?>
   <?php endforeach ?>
</table>
```

CÓDIGO FUENTE PARA APLICACIÓN DE ANDROID

Android Manifest

Este archivo contiene una descripción de alto nivel de la aplicación. Es común para todas las aplicaciones hechas en este sistema.

/app/src/main/AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="com.cambraca.tesis"
      android:versionCode="1"
      android:versionName="1.0">
    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
    <uses-permission android:name="android.permission.CHANGE_WIFI_STATE" />
    <uses-permission android:name="android.permission.INTERNET" />

    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".Tesis14929Activity"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <activity android:name="com.cambraca.tesis.CreateMapActivity"
            android:label="@string/title_activity_create_map"
            android:parentActivityName="com.cambraca.tesis.Tesis14929Activity">
            <meta-data android:name="android.support.PARENT_ACTIVITY"
                android:value="com.cambraca.tesis.Tesis14929Activity" />
        </activity>
    </application>
</manifest>
```

Activities

Acá se muestra el código de las dos vistas de la aplicación Android, descritas en la sección 4.4.2.

/app/src/main/java/com/cambraca/tesis/Tesis14929Activity.java

```
public class Tesis14929Activity extends Activity {
   private Space space;
   private Receiver receiver;

   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);

        space = new Space(getApplicationContext());
        receiver = Receiver.getInstance(getApplicationContext());
        receiver.setMainActivity(this);

        ((Spinner) findViewById(R.id.space_spinner)).setOnItemSelectedListener(
            new AdapterView.OnItemSelectedListener() {
                public void onItemSelected(
                  AdapterView<?> parent, View view, int position, long id) {
                    receiver.read(((TextView) view).getText().toString());
                }

                public void onNothingSelected(AdapterView<?> parent) {
                    receiver.read("");
                }
            }
        );

        updateSpaces(null);
   }

    @Override
    protected void onStop() {
        unregisterReceiver(receiver);
        super.onStop();
    }

    @Override
    protected void onStart() {
        registerReceiver(receiver, new IntentFilter(
          WifiManager.SCAN_RESULTS_AVAILABLE_ACTION
```

```
        ));
        super.onStart();
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
       MenuInflater inflater = getMenuInflater();
       inflater.inflate(R.menu.main_menu, menu);
       return super.onCreateOptionsMenu(menu);
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
       switch (item.getItemId()) {
       case R.id.debug_wifi:
          String msg;
          msg = space.tempTest();
          Toast.makeText(getApplicationContext(), msg, Toast.LENGTH_SHORT).show();
          return true;
       case R.id.create_map:
          Intent intent = new Intent(this, CreateMapActivity.class);
          startActivity(intent);
          return true;
       default:
          return super.onOptionsItemSelected(item);
       }
    }

    public void setCurrentLocation(String zone, int confidence) {
        ((TextView) findViewById(R.id.zone)).setText(zone);
        ((RatingBar) findViewById(R.id.confidence)).setRating(confidence);
        Toast.makeText(
          getApplicationContext(),
          "Received: " + zone, Toast.LENGTH_SHORT
        ).show();
    }
    public void reportError(String response) {
        Toast.makeText(
          getApplicationContext(),
          "Error: " + response, Toast.LENGTH_SHORT
        ).show();
    }
    public void showLastHash(int hash) {
        ((TextView) findViewById(R.id.last_hash)).setText("Last hash: " + hash);
    }
    public void updateSpaces(View view) {
        receiver.startScan();
```

```
        try {
            JSONObject jsonObject = new JSONObject(
              RequestHelper.get(getString(R.string.backend_url) + "/space/all")
            );
            JSONArray spacesJson = jsonObject.getJSONArray("spaces");
            ArrayList<String> spaces = new ArrayList<String>();

            for (int i = 0; i < spacesJson.length(); i++) {
                spaces.add((String) spacesJson.get(i));
            }

            ArrayAdapter<String> adapter = new ArrayAdapter<String>(
              this,
              android.R.layout.simple_spinner_item,
              spaces
            );
            Spinner spinner = (Spinner) findViewById(R.id.space_spinner);
            spinner.setAdapter(adapter);
        } catch (Exception e) {
        }
    }
}
```

/app/src/main/java/com/cambraca/tesis/CreateMapActivity.java

```
public class CreateMapActivity extends Activity {
    Receiver receiver;

    @Override
    protected void onStop() {
        unregisterReceiver(receiver);
        super.onStop();
    }

    @Override
    protected void onStart() {
        registerReceiver(
          receiver,
          new IntentFilter(WifiManager.SCAN_RESULTS_AVAILABLE_ACTION)
        );
        super.onStart();
    }


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_create_map);
```

```
        receiver = Receiver.getInstance(getApplicationContext());
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.menu_create_map, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        int id = item.getItemId();
        if (id == R.id.action_settings)
            return true;

        return super.onOptionsItemSelected(item);
    }

    public void measuring(int current) {
        ((TextView) findViewById(R.id.message_label))
                .setText("Grabando... " + current + " / " + Receiver.ZONE_MEASUREMENTS);
    }

    public void doneMeasuring() {
        ((TextView) findViewById(R.id.message_label))
                .setText("");
        ((Button) findViewById(R.id.action))
                .setText(getString(R.string.store));
        findViewById(R.id.instructions_label).setVisibility(View.VISIBLE);
        findViewById(R.id.recording_instructions_label).setVisibility(View.INVISIBLE);
    }

    public void Action(View view) {
        Button action = ((Button) findViewById(R.id.action));
        switch (receiver.status) {
            case STOPPED:
            case READING:
                String space = ((EditText) findViewById(R.id.space))
                  .getText().toString();
                String zone = ((EditText) findViewById(R.id.zone)).getText().toString();
                receiver.beginMeasuringZone(this, space, zone);
                action.setText(getString(R.string.stop));
                findViewById(R.id.instructions_label)
                  .setVisibility(View.INVISIBLE);
                findViewById(R.id.recording_instructions_label)
                  .setVisibility(View.VISIBLE);
```

```
                break;
            case MEASURING_ZONE:
                receiver.stop();
                action.setText(getString(R.string.store));
                findViewById(R.id.instructions_label)
                  .setVisibility(View.VISIBLE);
                findViewById(R.id.recording_instructions_label)
                  .setVisibility(View.INVISIBLE);
                break;
        }
    }
}
```

Otras clases

Algunas de estas clases se explican en la sección 4.4.2.

/app/src/main/java/com/cambraca/tesis/RequestHelper.java

```
public class RequestHelper {
    public static String post(String url, ArrayList<NameValuePair> parameters) {
        DefaultHttpClient client = new DefaultHttpClient(new BasicHttpParams());
        HttpPost post = new HttpPost(url);

        InputStream input = null;
        try {
            post.setEntity(new UrlEncodedFormEntity(parameters));

            HttpResponse response = client.execute(post);
            HttpEntity entity = response.getEntity();

            input = entity.getContent();
            BufferedReader reader = new BufferedReader(
              new InputStreamReader(input, "UTF-8"),
              8
            );
            StringBuilder sb = new StringBuilder();

            String line = null;
            while ((line = reader.readLine()) != null) {
                sb.append(line + "\n");
            }
            return sb.toString();
        } catch (Exception e) {
        } finally {
            try {
                if (input != null)
```

```
                    input.close();
                } catch (Exception e) {
                }
            }
            return null;
        }

        public static String get(String url) {
            DefaultHttpClient client = new DefaultHttpClient(new BasicHttpParams());
            HttpGet get = new HttpGet(url);
            get.setHeader("Content-type", "application/json");

            InputStream input = null;
            try {
                HttpResponse response = client.execute(get);
                HttpEntity entity = response.getEntity();

                input = entity.getContent();
                BufferedReader reader = new BufferedReader(
                  new InputStreamReader(input, "UTF-8"),
                  8
                );
                StringBuilder sb = new StringBuilder();

                String line = null;
                while ((line = reader.readLine()) != null) {
                    sb.append(line + "\n");
                }
                return sb.toString();
            } catch (Exception e) {
            } finally {
                try {
                    if (input != null)
                        input.close();
                } catch (Exception e) {
                }
            }
            return null;
        }
    }
```

/app/src/main/java/com/cambraca/tesis/Receiver.java

```
public class Receiver extends BroadcastReceiver {
    public enum ReceiverStatus {
        STOPPED, READING, MEASURING_ZONE
    }

    private WifiManager wifi;
    private Tesis14929Activity mainActivity;
    private static Receiver instance = null;
    public ReceiverStatus status = ReceiverStatus.STOPPED;
    public static final int ZONE_MEASUREMENTS = 4; // store 4 measurements for each zone
    private CreateMapActivity createMapActivity;
    private String currentZone;
    private String currentSpace;
    private ArrayList<List<ScanResult>> measurements;

    protected Receiver(Context c) {
        wifi = (WifiManager) c.getSystemService(
                Context.WIFI_SERVICE);
    }

    public static Receiver getInstance(Context c) {
        if (instance == null)
            instance = new Receiver(c);
        return instance;
    }

    public void setMainActivity(Tesis14929Activity a) {
        mainActivity = a;
    }

    public void stop() {
        this.status = ReceiverStatus.STOPPED;
    }

    public void read() {
        if (this.currentSpace == null)
            this.stop();
        else {
            this.status = ReceiverStatus.READING;
            wifi.startScan();
        }
    }

    public void startScan() {
        if (wifi != null)
```

```
        wifi.startScan();
}

public void read(String space) {
    if (space.equals("")) {
        this.currentSpace = null;
        this.status = ReceiverStatus.STOPPED;
        return;
    }
    this.status = ReceiverStatus.READING;
    this.currentSpace = space;
    wifi.startScan();
}

public void beginMeasuringZone(CreateMapActivity a, String space, String zone) {
    this.status = ReceiverStatus.MEASURING_ZONE;
    this.measurements = new ArrayList<List<ScanResult>>();
    this.createMapActivity = a;
    this.currentSpace = space;
    this.currentZone = zone;
    wifi.startScan();
}

@Override
public void onReceive(Context context, Intent intent) {
    List<ScanResult> aps = wifi.getScanResults();

    switch (status) {
        case READING:
            ArrayList<NameValuePair> params = new ArrayList<NameValuePair>();
            int hash = 0;
            for (ScanResult scanResult : aps) {
                params.add(new BasicNameValuePair("levels[" + scanResult.BSSID
                        + "]", Integer.toString(scanResult.level)));
                hash += scanResult.level;
                mainActivity.showLastHash(hash);
            }
            new QueryCurrentZone().execute(
                    context.getString(R.string.backend_url) + "/zone/lookup/space/"
                      + currentSpace,
                    params);
            break;
        case MEASURING_ZONE:
            // record measurement in array
            this.measurements.add(aps);
            createMapActivity.measuring(this.measurements.size());
```

```
            // did we do the necessary measurements?
            if (this.measurements.size() == Receiver.ZONE_MEASUREMENTS) {
                this.stop();
            } else {
                Timer myTimer = new Timer();
                myTimer.schedule(new TimerTask() {
                    @Override
                    public void run() {
                        wifi.startScan();
                    }
                }, 1000);
            }

            List<NameValuePair> params1 = new ArrayList<NameValuePair>();

            for (ScanResult scanResult : aps) {
                params1.add(new BasicNameValuePair("levels["
                        + scanResult.BSSID + "]",
                    Float.toString(scanResult.level)
                ));
            }

            params1.add(new BasicNameValuePair("zone", currentZone));
            params1.add(new BasicNameValuePair("space", currentSpace));
            if (this.measurements.size() == 1)
                //first measurement, clear all previous ones stored in the backend
                params1.add(new BasicNameValuePair("clear", "1"));

            // save to backend
            new SaveZoneMeasurements().execute(
                    context.getString(R.string.backend_url) + "/zone/save",
                    params1);
            break;
        default:
            break;
    }

}

private class SaveZoneMeasurements extends AsyncTask<Object, Void, String> {
    @SuppressWarnings("unchecked")
    @Override
    protected String doInBackground(Object... params) {
        return RequestHelper.post((String) params[0],
                (ArrayList<NameValuePair>) params[1]);
    }
```

```
        @Override
        protected void onPostExecute(String result) {
            if (measurements.size() == ZONE_MEASUREMENTS)
                createMapActivity.doneMeasuring();
        }
    }

    private class QueryCurrentZone extends AsyncTask<Object, Void, String> {
        @SuppressWarnings("unchecked")
        @Override
        protected String doInBackground(Object... params) {
            return RequestHelper.post((String) params[0],
                    (ArrayList<NameValuePair>) params[1]);
        }

        @Override
        protected void onPostExecute(String result) {
            if (result == null || !result.substring(0, 1).matches("-?\\d+(\\.\\d+)?"))
                mainActivity.reportError(result != null ? result : "[no response]");
            if (result.length() >= 2)
                mainActivity.setCurrentLocation(
                  result.substring(1).trim(),
                  Integer.parseInt(result.substring(0, 1))
                );
        }
    }
}
```

/app/src/main/java/com/cambraca/tesis/Space.java

```
public class Space {
   Context context;
   WifiManager wifi;

   public Space(Context c) {
      context = c;
      wifi = (WifiManager) c.getSystemService(Context.WIFI_SERVICE);
   }

   public String tempTest() {
      String ret = "";
      ret = ret + "wifi ";
      ret = ret + (wifi.isWifiEnabled() ? "active" : "inactive");
      if (wifi.isWifiEnabled()) {
         ret += "\n" + (wifi.startScan() ? "scan" : "no scan");
         List<ScanResult> aps = wifi.getScanResults();
         for (ScanResult scanResult : aps) {
            ret = ret + "\n" + scanResult.BSSID + ": " + scanResult.level;
         }
      }
      return ret;
   }
}
```

TABLAS DE RESULTADOS

La primera tabla en cada prueba muestra las zonas, cuántos puntos de acceso se leyeron desde cada zona (los dos valores representan los PAs que se detectaron en todas las 4 mediciones, y los PAs que se detectaron en cualquiera de las mediciones, respectivamente), y qué zonas se consideran adyacentes.

Las dos tablas siguientes corresponden a los dos algoritmos. Las tres columnas del lado derecho representan las opciones de limitar los puntos de acceso en cada medición (por ejemplo, tomar solo las 4 mediciones de señal más fuertes), y en cada celda se muestra la zona reportada en 5 mediciones. Si la zona reportada no es la correcta, se muestra subrayada cuando es adyacente. Para mayor claridad, cuando la zona reportada coincide con la zona real, se representa con un punto. En el algoritmo de red neuronal, si el reporte no es de ninguna zona, se representa con la letra X. Al final de cada tabla se muestra el porcentaje de mediciones correctas, primero estrictamente y luego considerando como correcto un reporte de una zona adyacente a la que se está.

Por último se muestra una tabla con todas las mediciones realizadas. Las columnas representan los PAs (se muestra su dirección MAC), y las filas cada medición hecha desde cada zona (4 filas por cada zona).

Prueba A

La prueba A utiliza tres habitaciones del departamento utilizado en las pruebas como zonas.

Zonas.

ID	Zona	PAs	PAs max	Adyacentes
A	Dormitorio	2	8	B
B	Cuarto	4	6	A C
C	Estudio	4	8	B

Teorema de Pitágoras.

ID	Zona	Sin límite de PA	8 PAs	4 PAs
A	Dormitorio	C C C C C	C C B B ·	· · B B B
B	Cuarto	C · A C C	· C · · ·	A · · · ·
C	Estudio	· · · · ·	· · · · B	B B · · ·
Estricto		**6/15 = 40%**	**9/15 = 60%**	**9/15 = 60%**
Adyacente		**10/15 = 67%**	**12/15 = 87%**	**15/15 = 100%**

Red neuronal.

ID	Zona	Sin límite de PA	8 PAs	4 PAs
A	Dormitorio	· · · · ·	· · · · ·	C · B B C
B	Cuarto	C · · · C	C · · X X	· C · C X
C	Estudio	· · · · X	· X X · ·	· A · · A
Estricto		**12/15 = 80%**	**10/15 = 67%**	**6/15 = 40%**
Adyacente		**14/15 = 93%**	**11/15 = 73%**	**10/15 = 67%**

Mediciones.

	58:6d:8f:db:bb:71	dc:9f:db:6e:08:ec	00:90:a9:f1:5c:9d	00:18:e7:ee:85:c5	00:27:22:94:c9:f4	ac:e2:15:f1:de:2c	b0:a7:37:76:49:cb	ec:1a:59:2b:46:c7	00:1d:7e:18:15:58	20:aa:4b:88:c9:24
Dormitorio			-23	-29	-54	-51		-73	-81	
			-19	-32				-81		-73
			-25	-26	-54	-48		-73		
		-85	-23	-35		-56			-81	
Cuarto	-95		-21	-15	-51					
	-69		-12	-10	-48					-100
	-73		-17	-11	-59					-100
	-69		-21	-12	-59		-100			
Estudio	-73	-90	-14	-15	-66					-100
	-81		-12	-12	-54					-100
	-69	-90	-10	-13	-95	-77	-69			
	-63	-90	-9	-15	-54	-77	-69			

Prueba B

La prueba B utiliza como zonas las habitaciones del departamento utilizado en las pruebas.

Zonas.

ID	Zona	PAs	PAs max	Adyacentes
1	Dormitorio	2	6	2 5 6
2	Cuarto	2	9	1 3
3	Estudio	2	7	2
4	Baño 1	3	4	7
5	Baño 2	2	7	1 6 7
6	Baño 3	3	4	1 5 7
7	Sala	3	9	4 5 6 8 9
8	Comedor	4	15	7 9 A
9	Balcón	5	28	7 8
A	Cocina	2	11	8 B
B	Lavandería	4	9	A

Teorema de Pitágoras.

ID	Zona	Sin límite de PA					8 PAs					4 PAs				
1	Dormitorio	2	2	2	2	2	2	6	4	4	4	4	4	4	4	4
2	Cuarto	·	·	4	4	·	3	3	3	3	3	3	3	3	·	3
3	Estudio	2	2	2	2	4	·	·	·	·	·	·	·	·	·	·
4	Baño 1	·	·	·	·	5	·	5	5	5	5	5	·	5	·	5
5	Baño 2	·	2	2	4	4	·	3	·	·	·	·	·	·	·	·
6	Baño 3	A	A	2	2	2	·	9	9	7	·	·	5	·	·	·
7	Sala	A	A	A	·	A	5	·	A	·	·	·	·	B	·	·
8	Comedor	A	A	A	A	A	7	7	B	5	B	7	7	5	7	5
9	Balcón	A	8	8	A	A	·	A	B	7	·	·	·	·	·	·
A	Cocina	9	B	B	·	·	·	·	B	B	·	·	·	·	7	·
B	Lavandería	·	·	·	A	A	7	·	·	·	A	·	·	·	·	A
Estricto		**14/55 = 25%**					**23/55 = 42%**					**34/55 = 62%**				
Adyacente		**34/55 = 62%**					**38/55 = 69%**					**43/55 = 78%**				

Red neuronal.

Esta prueba no fue posible con límite de 8 PAs, ya que la red neuronal nunca llegó al 100% de entrenamiento.

ID	Zona	Sin límite de PA					4 PAs				
1	Dormitorio	6	X	·	·	·	·	5	X	5	5
2	Cuarto	X	3	X	·	·	1	X	·	·	3
3	Estudio	5	2	X	2	·	2	·	·	·	2
4	Baño 1	·	5	1	·	3	X	·	1	1	1
5	Baño 2	·	4	2	2	1	1	4	4	4	1
6	Baño 3	·	·	1	X	1	2	·	·	·	·
7	Sala	6	5	·	6	4	8	·	X	X	9
8	Comedor	X	7	9	A	7	7	X	6	1	9
9	Balcón	·	A	A	2	·	7	X	X	·	·
A	Cocina	7	7	·	9	X	·	·	·	·	·
B	Lavandería	9	·	·	·	·	·	·	·	·	·
Estricto		**19/55 = 35%**					**24/55 = 44%**				
Adyacente		**34/55 = 62%**					**38/55 = 69%**				

Mediciones.

	00:14:d1:6d:40:91	9c:d6:43:cb:cf:94	68:7f:74:a9:98:86	00:90:a9:f1:5c:9d	00:02:6f:e1:75:d8	28:c6:8e:92:27:5f	88:c9:d0:b3:d7:f3	a0:f3:c1:ba:6e:ee	e8:94:f6:ee:d2:b5	c0:a0:bb:c6:e0:ac	dc:9f:db:6e:08:ec	00:27:22:94:c9:f4	00:18:e7:ee:85:c5	d4:ca:6d:11:cf:c5	88:9f:fa:4a:7e:0e	20:aa:4b:88:c9:24	ac:e2:15:f1:de:2c	64:66:b3:c8:73:de	f4:b7:e2:5f:63:b2	4c:bc:a5:87:03:15	f8:d1:11:5a:59:89	00:e0:4d:d3:e9:28	00:14:d1:ba:75:9a	00:16:b6:14:e6:35	ec:1a:59:2b:46:c7	ec:1a:59:2b:46:c6	58:6d:8f:db:bb:71	b0:a7:37:76:49:cb	00:21:63:de:7a:5e	24:a4:3c:80:06:0a	02:15:6d:69:a5:60	02:15:6d:69:a5:6d	00:10:e7:b5:9c:11	02:00:00:aa:00:00	02:15:6d:69:a5:5f	0a:0e:dc:42:fb:d3	02:15:6d:69:a5:63	02:15:6d:69:a5:61	02:15:6d:69:a5:65	02:15:6d:69:a5:6f	f8:d1:11:6f:3d:14	00:18:e7:e5:72:c0	00:15:6d:55:e2:4e	00:0c:42:6c:21:aa	c8:b3:73:1e:8f:d9	00:21:63:df:cc:83	64:70:02:7a:e8:fc
Dormitorio				-19									-41		-90	-95	-54																												
				-18								-29	-39			-95																													
				-19									-37		-81																															
				-21									-56		-90		-54																													
Cuarto				-12								-37	-15															-95	-51																
				-13								-37	-59															-95	-51																
				-12									-27											-100		-66			-85				-100												
				-13									-25																																
Estudio				-10									-13				-85																												
				-9									-10															-81																
				-11									-8		-81											-63	-59	-90																
				-9									-15				-85											-81																
Baño 1	-100			-17					-100				-54																														
	-95			-15									-54																																
	-81			-12									-59																																
	-100			-14									-51																																
Baño 2	-81			-13									-23													-100																		
				-15									-26																																
	-73			-17								-66	-32												-95	-69	-48																	
	-81			-13									-25													-100																		
Baño 3				-26								-77	-35	-85																														
				-29								-54	-41																																
				-20								-77	-46																																
				-20								-54	-48																																
Sala	-46			-69	-73	-63			-73		-90	-69																																
	-51	-95		-25	-90				-69			-66	-81																																
	-39			-29					-63		-90	-69	-51																																
	-46			-25	-73	-63			-69			-59	-81																																
Comedor				-29	-48	-77			-59	-100								-85	-85	-81	-81																						
	-63	-69		-27	-69	-77			-69	-73		-77					-95		-85		-81																						
	-56	-63		-30	-44				-69	-56													-73	-77																				
	-56	-81		-30	-44				-81	-73		-77					-95							-81																				
Balcón	-63	-85		-39		-81			-73										-59	-100	-100										-85	-95	-90	-90	-95	-85	-85	-85	-90	-90	-85			-100			
		-81		-48		-59						-51						-81	-56	-85														-63									
		-81		-48		-73						-51						-90		-85														-63								-90	-63
		-85		-39		-81													-56	-85											-85	-95	-90		-95	-85		-95	-90	-90	-85		-95	-95	-95	-90			
Cocina	-63	-73	-95	-51	-59								-100																																
		-56		-48	-56	-90	-90	-95	-81																																		
	-66	-56		-77	-56	-90	-90	-95	-81				-100																																
		-85		-56	-69		-81			-81																																		
Lavandería		-69		-90	-69					-66			-95																																
		-85	-100		-54				-95	-85			-90	-90																															
		-85	-100		-81					-63			-100				-100																											
		-90			-69					-73			-95				-100																											

Prueba C

En la prueba B se divide el departamento en celdas de 2 metros de lado. Se identifican estas celdas con una letra y un número, similar a una hoja de cálculo.

Zonas.

Zona	PAs	PAs max	Adyacentes
A1	5	9	A2 B1 B2
A2	2	11	A1 A3 B1 B2
A3	3	6	A2 A4 B2
A4	4	12	A3 A5 B5
A5	3	11	A4 A6 B5 B6
A6	7	14	A5 A7 B5 B6 B7
A7	7	13	A6 B6 B7
B1	4	9	A1 A2 B2 C1 C2
B2	3	9	A1 A2 A3 B1 C1 C2 C3
B5	6	13	A4 A5 A6 B6 C4 C5 C6
B6	6	12	A5 A6 A7 B5 B7 C5 C6 C7
B7	6	14	A6 A7 B6 C6 C7
C1	5	12	B1 B2 C2 D1 D2
C2	2	8	B1 B2 C1 C3 D1 D2
C3	3	6	B2 C2 C4 D2 D4
C4	3	10	B5 C3 C5 D4
C5	4	10	B5 B6 C4 C6 D4 D6
C6	5	15	B5 B6 B7 C5 C7 D6 D7
C7	4	13	B6 B7 C6 D6 D7
D1	3	8	C1 C2 D2 E1 E2
D2	3	8	C1 C2 C3 D1 E1 E2
D4	2	7	C3 C4 C5
D6	3	8	C5 C6 C7 D7 E6 E7
D7	2	9	C6 C7 D6 E6 E7
E1	4	6	D1 D2 E2
E2	4	6	D1 D2 E1
E6	2	8	D6 D7 E7
E7	3	12	D6 D7 E6

Teorema de Pitágoras.

Zona	Sin límite de PA					8 PAs					4 PAs				
A1	B1	B1	B1	B1	B1	C2	C2	C2	A3	A3	D4	D4	D4	D4	·
A2	B2	B2	B1	B1	B1	D4	A3	A3	A3	A3	C2	B1	D4	D4	C2
A3	B1	B1	E7	B1	B1	·	·	·	·	·	·	·	·	·	·
A4	C6	C6	C4	A5	C4	C7	A3	·	A3	A3	·	·	·	·	D7
A5	B6	C6	E7	B1	·	A4	D4	C4	·	C5	·	D6	D6	D7	D6
A6	D1	D7	A7	A7	·	D7	E6	D6	D7	A4	D6	C7	B6	D7	D7
A7	·	·	E7	·	·	A5	D7	D7	D7	D7	E6	D7	E6	A6	E6
B1	·	·	D1	D1	D1	A3	C2	C2	C2	D4	D4	D4	D1	D1	C2
B2	·	D1	A2	A2	·	C2	C2	C2	C2	C2	C2	D1	C2	C2	D4
B5	D7	C6	C6	C5	C6	A3	A4	A4	A4	A4	·	·	·	A5	·
B6	E6	B1	B1	B1	D4	A4	A4	A4	·	·	D7	C5	·	C5	A4
B7	C4	C6	D7	A5	D4	E7	A3	E7	E7	A4	A3	A3	B5	C6	A4
C1	C2	A2	B1	B1	D1	D1	E2	C2	C2	C2	C2	C2	E2	E2	C2
C2	B1	B1	B1	B1	B2	·	D1	E2	D1	·	·	·	D1	D4	D1
C3	B1	D4	D4	D4	D4	·	·	E2	E2	·	C2	·	·	·	·
C4	B5	D7	D4	A5	D4	·	D4	C5	C5	D4	C5	D4	B5	B5	A3
C5	D4	B5	B5	B5	B1	D4	A4	·	A3	A4	B5	B5	B5	B5	D4
C6	A5	B1	B1	E7	E7	D7	·	·	C7	·	D7	D7	D6	E7	D7
C7	E6	E7	E7	C6	C6	C6	D7	D7	·	D6	D7	D6	D7	D6	D6
D1	·	B1	B1	C2	C2	C3	E1	E2	E2	E1	D2	C2	D2	E2	C2
D2	B1	B1	D1	D1	B1	C2	E2	·	·	·	C3	C2	C2	C2	C2
D4	·	B1	B1	·	·	A2	·	·	·	·	E2	A3	·	·	·
D6	C6	E7	E7	E7	D7	·	·	·	·	·	·	·	·	·	·
D7	·	·	B1	·	·	·	C6	D6	E7	E7	D6	E7	E7	E7	E7
E1	B1	B2	B2	B2	B2	C2	D1	·	C2	D1	C2	D2	·	·	C2
E2	A1	A1	B2	B2	B1	C2	D1	D2	D2	·	D1	D1	D1	C2	C2
E6	E7	D7	D7	·	·	D6	·	·	D7	D7	D7	D6	D7	D7	C7
E7	B6	·	·	·	·	D7	·	·	·	·	·	·	·	D6	·
Estricto		**24/140 = 17%**					**41/140 = 29%**					**36/140 = 26%**			
Adyacente		**79/140 = 56%**					**97/140 = 69%**					**95/140 = 68%**			

Red neuronal.

La prueba C utilizando el algoritmo de red neuronal no se pudo realizar ya que, al igual que en la prueba B con límite de 8 PAs, las redes neuronales no se llegaron a entrenar.

Mediciones.

	00:14:d1:6d:40:91	e8:94:f6:ee:d2:b5	00:90:a9:f1:5c:9d	00:18:e7:ee:85:c5	9c:d6:43:cb:cf:94	00:27:22:94:c9:f4	00:02:6f:e1:75:d8	28:c6:8e:92:27:5f	f4:b7:e2:5f:63:b2	58:6d:8f:d1:50:c6	58:6d:8f:d1:50:c4	c0:a0:bb:c6:e0:ac	f8:d1:11:5a:59:89	64:70:02:7a:e8:fc	f8:d1:11:ba:cc:94	2c:e6:cc:6b:c5:a8	dc:9f:db:6e:08:ec	58:6d:8f:db:bb:71	ec:1a:59:2b:46:c6	78:54:2e:5a:bf:48	ac:e2:15:f1:de:2c	a0:f3:c1:51:75:1a	68:7f:74:a9:98:86	ac:7f:3e:ec:64:84	d4:6e:5c:15:38:04	0a:0e:dc:44:8b:56	ec:1a:59:2b:46:c7	90:f6:52:a2:de:90	00:14:d1:ba:bd:82	88:9f:fa:4a:7e:0e	84:1b:5e:e1:4f:d8	28:c6:8e:92:27:5e	14:b9:68:c8:ac:40	00:e0:4d:d3:e9:28	00:14:d1:ba:75:9a	00:22:75:e6:6c:35	00:10:e7:b5:9c:11	a0:f3:c1:35:7d:38	f8:3d:ff:1f:e1:50
A1			-34	-32														-59	-51		-46						-51		-100	-85								
			-25	-41		-59												-73	-63								-56										
			-37	-51		-51												-73	-54								-54										
			-23	-32														-73	-54		-46						-51		-100	-85								
A2			-22	-34				-100									-66		-100														-85	-77					
			-16	-69		-56												-81			-77													-77				
			-23	-35		-56		-100																														
			-15	-30														-81			-77				-66								-85					
A3	-85		-19	-59		-95																																
	-100	-100	-18	-51		-95																																
	-100		-19	-56																																		
	-100	-100	-20	-59																											-100							
A4		-77	-37	-90		-77	-54	-100		-100	-100																				-95							
	-39	-59	-34	-95		-77		-100	-100	-100																												
		-73	-35	-95	-95	-77	-56		-100																													
		-66	-48	-81	-95	-81	-54		-100		-100																				-95							
A5	-46	-73	-30	-81	-85								-100																									
	-54	-77	-41			-48	-46	-85	-95	-100																												
	-54	-73	-29	-81	-100		-48	-85	-77	-81			-100																									
	-48	-77	-29		-85	-51	-44	-85		-100																												
A6	-73		-35			-51	-39		-69	-73	-81	-69	-63																									
	-48	-81	-27			-51	-44		-95	-95	-100		-54	-90	-69																						
	-77	-81	-34			-48	-39		-69	-73	-69		-73																									
	-73	-100	-51		-77	-51	-44		-66		-81	-69	-63			-90																						
A7			-32			-48	-44	-56	-77	-66	-95		-85													-95													
	-69		-59			-37	-77	-59	-51	-85	-95	-81	-56												-90													
			-35			-48	-44	-66	-77	-66	-95		-85		-100																								
	-69		-37			-46	-54	-66		-95	-85		-95		-100																								
B1			-23	-29		-41				-100	-100						-69	-77																				
			-29	-30													-69	-85	-54																			
			-22	-27		-46				-100	-100						-59	-77																				
			-23	-29		-54											-66	-81								-100												
B2			-19	-23		-63											-77				-51															-81			
			-22	-23		-56													-66		-59						-66										-85		
			-20	-26																	-48																	
			-21	-23		-56											-77		-81		-59						-77									-81			
B5	-44	-77	-19	-66		-81	-51	-81	-81																												-100		-95
	-41	-66	-19		-100	-73	-59		-90		-77																												
	-69	-77	-18	-69		-81	-51		-81	-77	-81																												-100
	-46	-69	-19	-69		-73	-46		-100	-77	-73																												-100
B6	-66	-81	-25	-90	-81		-69		-77																														
	-48		-20	-90	-59		-51		-77				-100																								-100		
	-54	-63	-30	-77	-56		-48	-73	-100	-100	-100		-95																								-100		
	-51	-90	-25	-77	-81		-69	-73	-95	-100	-100		-95																										
B7	-54	-69	-19		-48		-44	-100	-77	-59	-59	-85									-100										-100							
	-73	-69	-26	-90	-48		-54		-77	-59	-59																												
	-73	-66	-18	-63	-81	-85	-51	-54		-73																													
	-73	-66	-27		-46	-85	-51	-100	-95	-56	-56	-85									-100										-100							
C1			-21	-20		-63												-77	-66		-39						-69	-63										
			-21	-15		-63												-77	-48		-35						-69	-48	-81	-95	-90							
			-21	-15		-63													-66		-39							-51										
			-23	-15														-81	-51	-100	-41							-59										

C2			-15	-23		-66																								
			-16	-37		-66											-95													
			-17	-20													-90	-51	-100	-41				-59						
			-20	-16													-90													
C3	-77		-8	-32		-100																			-95					
	-69		-9	-29				-100																						
	-100		-9	-32				-100																						
	-69		-9	-30				-100																						
C4	-51		-9	-41				-77														-100								
	-66	-85	-10	-37			-73	-100	-85																				-90	
		-100	-8	-56	-100		-66	-77	-81																					
		-100	-8	-44	-100		-66	-77	-81																					
C5	-46	-85	-18	-56		-81																								
	-48	-100	-19	-66			-90	-100							-100	-81														
	-35	-90	-27	-73		-81									-100	-81														
	-48	-81	-26	-66		-100	-90	-100	-77						-95															
C6		-100	-66		-56		-69	-100		-90	-90		-100		-100											-81	-100	-100		
	-66	-81	-81		-56		-69	-100		-90	-90		-100		-100											-81				
			-54		-63	-95	-41					-100	-85													-63		-100		
	-66	-81	-54		-56	-95	-41						-85													-81		-100		
C7	-100		-51							-66			-69													-85	-90			-100
	-77	-85	-59		-66	-81	-48			-73	-77															-81				
	-77	-85	-59		-66	-81	-51			-69	-77	-85	-69													-85	-90			-100
	-77		-54				-46			-69	-77	-85	-69													-85	-90			-100
D1			-15	-6													-77								-100					
			-20	-6		-69											-85								-100					
			-18	-5														-90		-63				-81	-100					
			-23	-6		-69											-85	-90						-81	-100					
D2			-14	-11		-63											-90								-100					
			-10	-12		-59											-95	-95						-100						
			-11	-16													-77	-95	-95					-100						
			-12	-13		-63											-95	-95							-100					
D4	-100		-21	-35				-100		-100	-100																			
			-23	-37																										
			-35	-37																									-100	
			-26	-35																									-100	
D6			-54				-41						-77																	-100
	-56		-59		-81	-100	-41						-81										-100							
			-48		-81	-100	-41						-81																	
	-56		-56				-41						-77										-100							
D7			-48		-63																									
	-85		-51		-63		-54	-85			-100	-95																		
			-48		-69	-100	-73							-95																
	-56		-56		-59	-100	-73							-95																
E1			-10	-11		-56											-95			-56					-100					
			-11	-11		-48											-77								-100					
			-11	-12		-44											-73			-56					-100					
			-10	-11		-54											-66													
E2			-10	-17		-100											-81													
			-11	-18		-77											-85								-100					
			-9	-15		-77											-85			-56					-100					
			-10	-14		-85											-90													
E6			-46	-59	-90		-51						-100																-100	
			-46	-63	-100		-51																						-100	
			-54	-95	-90		-69																							
					-51		-56					-73	-100	-95																
E7	-100		-85		-48		-66	-100				-81								-85	-95	-85								
	-100				-56		-63	-100				-73	-81							-81		-90	-95							
					-56		-54	-100				-73	-100			-100				-69		-90	-95							
			-85		-51		-66													-85	-95		-100							