UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA

ESCUELA DE INGENIERÍA DE TELECOMUNICACIÓN Y ELECTRÓNICA

PROYECTO FIN DE CARRERA

Localización en WLAN para dispositivos con Windows Mobile

TITULACIÓN: Ingeniería Técnica de Telecomunicación en Telemática
TUTORES : Miguel Ángel Quintana Suárez
David de la Cruz Sánchez Rodríguez
AUTOR : José Julio Báez Redondo
FECHA : Diciembre 2010

UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA

ESCUELA DE INGENIERÍA DE TELECOMUNICACIÓN Y ELECTRÓNICA

PROYECTO FIN DE CARRERA

Localización en WLAN para dispositivos con Windows Mobile

Presidente: Secretario: Vocal:

Tutores: Autor:

NOTA:

TITULACIÓN: **Ingeniería Técnica de Telecomunicación en Telemática**
TUTORES : **Miguel Ángel Quintana Suárez**
David de la Cruz Sánchez Rodríguez
AUTOR : **José Julio Báez Redondo**
FECHA : **Diciembre 2010**

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

GLOSARIO

- ACK (Acknowledgement, acuse de recibo)
- AP (Access Point, Punto de acceso)
- ATM (Asynchronous Transfer Mode, Modo de Transferencia Asíncrono)
- COITT (Colegio Oficial de Ingenieros Técnicos de Telecomunicación)
- CSMA/CA (Carrier-Sense, Múltiple Access, Collision Avoidance)
- CTS (Clear to Send, Listo para enviar)
- DHCP (Dynamic Host Configuration Protocol, Protocolo de Configuración Dinámica de los Equipos)
- DSSS (Direct Sequence Spread Spectrum, Espectro Ensanchado por Secuencia Directa)
- ESSID (Extended Service Set Identifier)
- ETSI (European Telecommunications Standards Institute, Instituto Europeo de Estándares de Telecomunicación)
- FHSS (Frequency Hopping Spread Spectrum, Espectro Ensanchado por Salto de Frecuencia)
- GPS (Global Positioning System, Sistema de Localización Global)
- IEEE (Institute of Electrical and Electronics Engineers, Instituto de Ingenieros Eléctricos y Electrónicos)
- ISO (Internacional Organization for Standardization, Organización Internacional para la Estandarización)
- LAN (Local Area Network, Red de Área Local)
- MAC (Media Access Control, Control de acceso al medio)
- OFDM (Orthogonal Frequency Division Multiplexin)

- OID (Object Identifier, Identificador de objeto)
- PDA (Personal Digital Assistant)
- QoS (Quality of Service, Calidad de Servicio)
- RF (Radio Frecuency, Radio Frecuencia)
- RSSI (Received Signal Strength Intensity)
- RTS (Ready to Send, Listo para enviar)
- UDP (User Datagram Protocol)
- UMTS (Universal Mobile Telecommunications System, Sistema Movil Universal de Telecomunicaciones)
- WEP (Wired Equivalent Privacy)
- WLAN (Wireless Local Area Network, Red Inalámbrica de Área Local)
- WPA (Wi-Fi Protect Access)

MEMORIA

CAPÍTULO 1

INTRODUCCIÓN

1.1 Antecedentes

La proliferación de dispositivos móviles y las tecnologías de acceso inalámbrico, tales como Bluetooth, Wifi o GSM, han impulsado la idea de la localización geográfica usando estas tecnologías.

El sistema de localización más extendido es el GPS, que ofrece la mejor cobertura y la mayor precisión. El único handicap del sistema GPS es que no funciona en interiores de edificios debido a que la señal del satélite se atenúa enormemente.

Los primeros proyectos de localización en interiores se basaban en la tecnología de rayos infrarrojos [Wan92]. Para usar esta tecnología había que realizar un amplio despliegue de dispositivos a lo largo del edificio ya que las señales infrarrojas no atraviesan los muros del edificio. Además, este tipo de señales se ven afectadas enormemente por la radiación solar, así que en zonas con mucha claridad las señales se verían anuladas o distorsionadas por la luz del sol.

De todas las tecnologías de comunicación inalámbrica, la más extendida es la telefonía móvil. En este caso, se han desarrollado varias aplicaciones (una muy conocida y de libre distribución es la de GoogleMobile). Estos sistemas se basan en la potencia de la señal recibida por la estación base a la que el móvil está conectado. Un método alternativo para la localización con las estaciones de telefonía móvil es el cálculo de las diferencias de tiempo en que una trama es enviada desde una estación y llega al dispositivo móvil. Ya sea el método que se utilice, una vez calculados estos valores se debe acceder a una base de datos donde se encuentran publicadas la localizaciones exactas de las estaciones base; y con estos datos se procedería a calcular la posición del usuario.

Estos sistemas son adecuados para exteriores pero su efectividad en interiores se ve muy limitada por la atenuación y por los múltiples reflejos de la señal (multi-path) [Par00][Tek98].

La siguiente tecnología más extendida es la WirelessLAN (comúnmente llamada "Wifi" ó "Wlan"). Con velocidades de hasta 54Mbps (según la norma 802.11g) estos sistemas han ganado gran popularidad para la conexión de ordenadores de forma inalámbrica. Las redes Wlan ofrecen muchas ventajas sobre el resto de sistemas de localización, como los infrarrojos o el Bluetooth, en términos de cobertura, escalabilidad, despliegue y la posibilidad de conectase a una red de comunicaciones de múltiples usuarios.

1.2 Objetivos

En este PFC se pretende desarrollar una aplicación para dispositivos móviles, que ejecuten WindowsMobile como sistema operativo, para localizar geográficamente al usuario en zonas de interior. Para ello se basará en la lectura de potencias de señal recibidas desde los puntos de acceso Wifi que se encuentren alrededor y se analizarán distintos algoritmos de localización.

Los objetivos que persigue este PFC son:

1. Estudio y análisis de diferentes métodos de localización; comparándolos por su precisión, robustez frente al ruido y consumo de batería del dispositivo.
2. Desarrollo de una aplicación para Windows Mobile que capture la potencia de señal que recibe de los puntos de acceso Wifi a su alrededor y guarde los datos en un fichero de texto para su posterior estudio.
3. Desarrollo de una aplicación para Windows Mobile que, a través de la potencia recibida de los puntos Wifi procese la información necesaria para localizar al usuario geográficamente.

1.3 Metodología seguida en el PFC

El sistema de localización se basa en los puntos de acceso Wifi que ofrecen el acceso a Internet o a la red local desplegados en el edificio de despachos del Departamento de Ingeniería Telemática de la ULPGC. A través de la lectura de la potencia de señal recibida se estimará la distancia a cada punto Wifi y se procederá a calcular la posición del usuario.

Para ello se ha definido la siguiente metodología:

1. Estudio de la propagación de señales electromagnéticas en interiores.
2. Estudio de las señales Wifi del recinto en particular para la obtención de algunos factores físicos necesarios para el sistema de localización.
3. Desarrollo de la aplicación para representar en un mapa la localización del usuario.
4. Análisis de diferentes métodos de localización en diferentes circunstancias medioambientales.

1.4 Estructura de la memoria

La memoria realizada en este PFC se ha estructurado en los siguientes capítulos:

- Memoria

1. **Introducción.**
2. **Tecnología Inalámbrica para Redes de Área Local:** características principales y distintas implementaciones de las redes de área local inalámbricas.
3. **Sistemas de localización:** descripción de varios sistemas de localización, tanto para exteriores como para interiores. Descripción del comportamiento de señales de radiofrecuencia en interiores.
4. **Algoritmos de localización:** descripción de distintos métodos para calcular la posición del usuario.
5. **Entorno de desarrollo:** descripción del entorno de trabajo para la creación de las aplicaciones software.
6. **Análisis previo y funcional:** definición del problema, descripción de las aplicaciones desarrolladas y diseño funcional por bloques.
7. **Análisis Orgánico:** descripción de las clases, métodos y publicación del código fuente más relevante del software.
8. **Caso de estudio:** análisis y comparativas de los distintos algoritmos de localización en el entorno estudiado.
9. **Conclusiones y futuras líneas de trabajo:** conclusiones obtenidas en los apartados anteriores. Mejoras y ampliaciones de la aplicación.
10. **Bibliografía:** bibliografía consultada en este Proyecto Fin de Carrera.

- Pliego de Condiciones: describe los requerimientos de software y hardware necesarios para la puesta en funcionamiento de las aplicaciones implementadas en este PFC.

- Presupuesto: presenta la valoración económica del PFC.

CAPÍTULO 2

REDES DE ÁREA LOCAL INALÁMBRICAS

2.1 Introducción

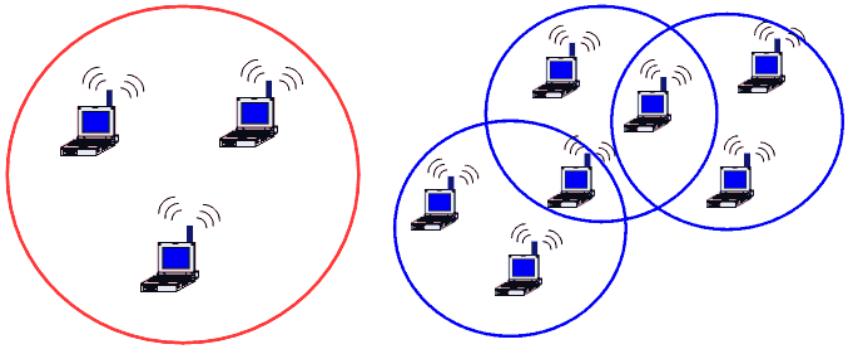
WirelessLAN (WLAN) es una red de área local inalámbrica que conecta dos o más dispositivos electrónicos sin usar cables. Esto permite a los usuarios una movilidad total sin perder conexión con la red.

El consorcio Wi-Fi se originó como un forum de empresas para dar publicidad a esta nueva tecnología hasta el punto que, actualmente, Wi-fi es sinónimo de WLAN. Aunque está extendida la idea de que Wi-fi son iniciales de *Wireless Fidelity*, en realidad es un nombre elegido por una empresa de marketing con el objetivo de que el nombre fuera corto y de fácil pronunciación [Web-1].

2.2 Topologías

Las WLAN tienen dos tipos de topología: ad-hoc o infraestructura.

- En la **estructura ad-hoc**, la red se comunica usuario a usuario. No existe ningún dispositivo de control que medie entre las comunicaciones. Cada estación actúa, simultáneamente, tanto de punto de acceso como de cliente. El grupo de estaciones que estén comunicadas entre sí se denomina *conjunto de servicio básico independiente* (IBSS).



IBSS formada por una única celda

IBSS formada por múltiples celdas

Figura 2.1: Diferentes configuraciones de IBSS

Los equipos que estén comunicados punto a punto con los demás, forman una celda. Se observa en la figura 2.1 un ejemplo de red *ad-hoc* con múltiples celdas, en las que algunos equipos actúan como *puente* para las estaciones que se encuentren en extremos más alejados.

Este es el método más simple para crear una red inalámbrica ya que no se necesita de hardware adicional ni de personal cualificado para gestionar la red.

- En la topología en **infraestructura**, se distribuyen estaciones base (o también llamados puntos de acceso) que ofrecen al usuario una conexión a una red externa. Estos puntos de acceso añaden algunas funcionalidades que las redes ad-hoc no poseen. El servicio DHCP es el que mayor comodidad ofrece al usuario, porque el punto de acceso asigna direcciones IP a cada dispositivo automáticamente.

  Estas estaciones base suelen estar conectadas a una Ethernet cableada por donde se accede a la red externa, como se muestra en la figura 2.2:

Figura 2.2: Diagrama de red básico de una Wireless LAN en infraestructura

Los puntos de acceso también pueden ser configurados en modo *relay*, en el que simplemente funcionan como repetidores de la señal, como indica la siguiente figura 2.3:

Figura 2.3: Punto de acceso en modo 'repetidor' para alcanzar mayor distancia.

2.3 Características de los estándares de WLAN

La IEEE define la Wireles s LAN en diferentes estándares atendiendo a la banda de frecuencias, modulación usada y velocidad de transferencia. Todo estándar 802.11 establece los niveles físico y de acceso al medio. El acceso al medio se basa en un mecanismo similar al de Ethernet (CSMA/CD). En la tabla 2.1 se muestran las características más relevantes de los estándares 802.11, además de Bluetooh e Hiperlan2, que también se explicarán posteriormente.

Las velocidades máximas que se indican en la tabla son las velocidades aceptadas por el canal físico, las cuales siempre son mayores que las velocidades reales para el usuario. Esto se debe a múltiples factores tales como la distancia a la que se encuentra el usuario respecto el punto de acceso, el número de usuarios conectados simultáneamente y el número de obstáculos entre el punto de acceso y el usuario. También hay que tener en cuenta los factores propios de la transmisión digital, como son las diferentes cabeceras y tramas que aumentan la cantidad de bits a enviar, con lo que para cada usuario se tendrá que enviar más bits de información de los que él recibe como datos útiles.

Estándar	Frec.	Nivel físico	Velocidad máxima	Alcance en interiores (m)	Alcance en exteriores (m)	Disponible
802.11	2'4 Ghz	FHSS	2 Mbps	20	100	1997
802.11b	2'4 Ghz	DSSS	11 Mbps	38	140	2000
802.11a	5 Ghz	OFDM	54 Mbps	35	120	2001
802.11g	2'4 Ghz	OFDM/DSSS	54 Mbps	38	140	Fin 2002
802.11n	2'4 /5 Ghz	OFDM	150 Mbps	70	250	Fin 2009
Bluetooth	2'4 Ghz	DSSS/FHSS	1 Mbps	2	25	1994
HiperLAN2	5 Ghz	OFDM	54 Mbps	15	50	2000

Tabla 2.1: Normas 802.11 desarrolladas por el IEEE.

Estudiando la evolución de las normas no sólo hay que centrarse en los datos que aparecen en la tabla anterior, también hay que tener en cuenta otros aspectos como la seguridad y fiabilidad de la comunicación, que se detallan en los apartados siguientes.

Para comparar algunas normas 802.11 con otras tecnologías inalámbricas se muestra a continuación la figura 2.4, donde se realiza una comparativa de la movilidad del usuario y la calidad de la transmisión de datos:

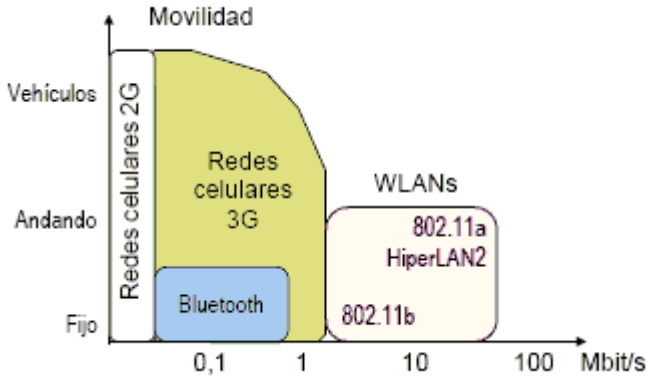


Figura 2.4: Gráfica comparativa de distintas tecnologías inalámbricas.

2.3.1 802.11b

La norma 802.11b ofrece una modulación de espectro ensanchado por secuencia directa (DSSS) sobre la banda de los 2'4 Ghz (2400 Mhz – 2483'5 Mhz). Las velocidades de transmisión aceptadas por esta norma son: 1, 2, 5'5 y 11 Mbps. Esta capacidad de adaptar la velocidad según las características del medio se denomina *Dynamic Rate Shifting* (DRS).

Debido a la banda de trabajo de 2'4 Ghz y su poco espacio para distribuirlo en canales de transmisión este sistema es perceptible de sufrir interferencias con otras redes WLAN y otros dispositivos que emiten en la misma banda, como son los teléfonos inalámbricos.

En cuanto a seguridad, la norma 802.11b sólo permite la posibilidad de encriptar el canal usando el cifrado WEP, para el que se encontró una vulnerabilidad importante con la que se puede descifrar la transmisión del canal en tan sólo unos minutos [Gor02].

Debido a las interferencias que sufre en la banda de trabajo, la norma 802.11b ofrece una baja calidad de servicio. En condiciones favorables la señal puede alcanzar 38 metros en zonas de interior ofreciendo la velocidad máxima disponible de 11 Mbps.

2.3.2 802.11a

Esta norma empezó a desarrollarse, cronológicamente, antes que la 802.11b, pero debido a la complejidad de la modulación OFDM se retrasó su publicación definitiva un año después de haber salido a la luz la 802.11b [Kap02].

La *multiplexación ortogonal por división en frecuenia* (OFDM) es más eficiente y compleja que la modulación DSSS. 802.11a trabaja en las frecuencias más altas de 5 GHz, donde las señales son menos propensas a interferencias que la banda de 2,4 GHz y además permite incluir más canales de transmisión sin posibilidad de solaparse. Sin embargo, en la banda de los 5 Ghz se produce una mayor absorción de las señales por los obstáculos, lo que tiende a reducir el alcance de 802.11a. Gracias a la robustez de OFDM frente a propagaciones multitrayecto, se compensa esta desventaja del alcance en localizaciones en interiores, por lo que en la práctica el alcance de ambas normas 'a' y 'b' es similar, tal como aparece en la tabla 2.1.

En cuanto a velocidades de transmisión esta norma trabaja desde los 6 Mbps hasta los 54 Mbps.

Dos inconvenientes a la hora de implementar esta norma son la incompatibilidad con las redes 802.11b y el alto consumo de batería al implementar el sistema en portátiles y dispositivos móviles.

2.3.3 802.11g

La norma 802.11g ofrece la posibilidad de usar las dos modulaciones OFDM y DSSS. Con OFDM se obtiene la ventaja de reducir el fenómeno del multitrayecto, y con DSSS se consigue que la norma sea compatible con las redes 'b'.

Su banda de trabajo está en el rango de los 2'4 Ghz, con lo que se conseguirá un alcance mayor debido a la menor absorción en la banda de trabajo. Con esto se conseguirá desplegar menos puntos de acceso para obtener el mismo alcance.

Gracias a la combinación del OFDM con la banda de trabajo de 2'4 Ghz se puede conseguir una velocidad de 54 Mbps en zonas de interior.

En cuanto a las desventajas, el estándar 802.11g tiende a tener una velocidad efectiva para el usuario menor que el 802.11a cuando hay una alta densidad de usuarios o de tráfico generado.

2.3.4 802.11n

En el año 2007 comienzan a surgir tarjetas inalámbricas con el logotipo de 802.11 "draft-N", aun sin haberse publicado la versión final de este protocolo.

Muchos fabricantes sacaron al mercado estas tarjetas porque la IEEE aseguraba que a nivel de hardware serían 100% compatibles con la versión final que se publicase. Para solventar problemas de software, se podría actualizar el firmware de la tarjeta en el momento de que saliese a la luz la versión final por el IEEE [Web-2]. Es en octubre de 2009 cuando se publica la versión 11.0 de la norma "n".

La norma "n" aumenta el caudal de información de 54 Mbps a 150 Mbps haciendo uso de hasta cuatro canales de emisión de 40 Mhz (la norma anterior usa canales de 20 Mhz).

En cuanto a compatibilidades, la norma "n" es compatible con las redes "b/g" a costa de reducir su velocidad de transmisión. Esto se debe a que la norma "n" no puede aprovechar correctamente los canales de 40 Mhz y debe adaptarse a los de 20 Mhz de las redes "b/g" [Web-3].

2.3.5 ETSI HiperLAN2

Este sistema de transmisión europeo desarrollado por la ETSI trabaja en la banda de los 5Ghz por lo que compite con el 802.11a. Son muy parecidos en el nivel físico aunque HiperLAN2 tiene ventajas en cuanto a control de potencia y cambio automático de frecuencia en caso de interferencia. Por tanto, se dice que este sistema sí ofrece una calidad de servicio.

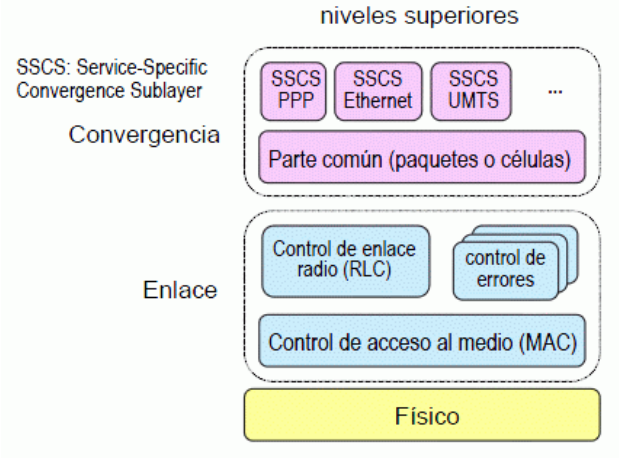


Figura 2.5: Arquitectura de protocolos de HiperLAN2.

El control de acceso al medio del 802.11a funciona mediante el protocolo CSMA/CD (en el caso de que haya habido colisión se ejecuta un plazo de espera hasta volver a retransmitir). En cambio, en HiperLAN2 el acceso MAC es coordinado por el punto de acceso, el cual asigna los recursos del canal de radio a cada equipo móvil que quiera transmitir.

Como último nivel en la arquitectura de HiperLAN2 se encuentra el *nivel de convergencia*, cuya misión es facilitar su interconexión con distintas redes. Como se muestra en la figura anterior, aparecen bloques específicos para Ethernet, UMTS, PPP y Firewire; y luego una parte común basada en paquetes (para redes IP) y otra basada en células (para redes ATM).

2.3.6 Bluetooth

Es un sistema de radio de corto alcance destinado a breves transmisiones entre dispositivos. Fue creada por Ericsson en el año 1994 como alternativa al RS-232 de forma inalámbrica [Web-4].

Bluetooth trabaja en la banda de libre acceso entre 2'4 Ghz y 2'480 Ghz. Y según su implementación puede abarcar hasta 25 metros a cielo abierto [Web-5].

En una comunicación por Bluetooh uno de los dispositivos toma el papel de 'maestro' y será el que gobierne el medio, pudiendo comunicarse hasta con siete dispositivos.

En su versión original podía transmitir hasta 800 Kbps. Pasando a la actual versión 3.0 con un ancho de banda de hasta 15 Mbps [Web-4].

CAPÍTULO 3

SISTEMAS DE LOCALIZACIÓN

3.1 Introducción

Un sistema de localización o posicionamiento consiste en la combinación de las tecnologías necesarias para la localización geográfica de los usuarios, un medio de comunicación para transmitir y recibir información entre la unidad móvil y un centro de control, y, finalmente, un software con capacidad de procesamiento de cartografía.

A continuación se van a mostrar distintos tipos de sistemas de localización, cuáles son sus ventajas e inconvenientes de usarlos en una zona de interiores.

3.2 Localización en exteriores

Los siguientes sistemas fueron diseñados para el uso en zonas al aire libre. Cuando se han intentado utilizar en zonas de interior los principales inconvenientes que han surgido han sido la atenuación de señales que el sistema necesita tomar como referencia, debido a la absorción de muros y paderes del inmueble; y el efecto de *multipath*, que modifica las lecturas que realizan los dispositivos móviles.

3.2.1 GPS

El sistema de localización GPS fue desarrollado por el gobierno militar norteamericano en los años 70. El sistema consiste en 25 satélites que orbitan alrededor de la Tierra cubriendo toda su extensión, más 6 satélites auxiliares para ir renovando la flota, ya que cada satélite tiene un tiempo de vide de unos siete años.

Cada usuario dispondrá de un receptor GPS, el cual detectará las señales que se reciban de los satélites orbitando alrededor. El receptor GPS necesita un mínimo de tres satélites para ofrecer un punto de localización aproximado (mientras más satélites detecte mayor será su precisión). De cada satélite recibirá un mensaje con los siguientes datos: un *timestamp* (hora exacta en la que se envió el mensaje) y la posición de ese mismo satélite. El receptor calculará la distancia a la que se encuentra cada satélite. Con estos datos, el localizador GPS calcula por triangulación su posición en el campo tridimensional (latitud, longitud, altitud). Un diagrama básico de triangulación se muestra en la siguiente figura:

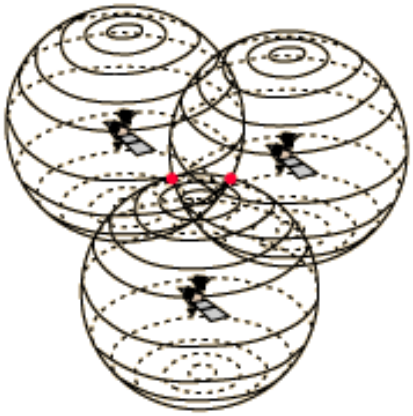


Figura 3.1: Tres esferas imaginarias intersectan en dos posibles puntos.

Se comprueba que las tres esferas imaginarias de los satélites coinciden en dos puntos, pero al usuario hay que indicarle solamente uno. Para resolver el sistema fácilmente suele ocurrir que uno de los puntos calculados se encuentra sobre la Tierra y el otro sobre el espacio. Por tanto, sólo queda excluir el del espacio para mostrar en pantalla la localización aproximada del usuario. En el caso que el receptor GPS detectara cuatro o más satélites sí obtendría un único punto posible y con mayor precisión.

Un inconveniente de los sistemas satelitales es que el usuario necesita visión directa con los satélites, por lo que no se podría usar el sistema en ciudades con altos edificios o en túneles.

3.2.2 Telefonía móvil GSM

En este sistema no se necesitaría de un hardware adicional específico para calcular la posición del usuario sino que sería el propio teléfono móvil quien haría de localizador.

Para la localización por GSM los métodos más usados por ser los más sencillos de implementar son: 'identificación de celda' y *'time-of-arrival'*. El primero se basa en hallar la localización del usuario identificando la estación base a la que está conectada el móvil, con la posterior consulta a una base de datos que nos indicará la posición exacta de dicha estación base. Se observa que el resultado estimado tiene poca precisión porque no está indicando la posición del usuario, sino la antena a la que está conectada su dispositivo móvil. Este sistema es el que usa

la última aplicación de GoogleMobile, en el que sus primeras versiones sólo hacían uso del GPS del dispositivo para localizar al usuario [Web-6].

Figura 3.2: Captura de la aplicación GoogleMobile haciendo uso de la 'identificación de celda'.

El segundo método hace uso del 'tiempo de llegada' (*time-of-arrival*), en el que el dispositivo mide el tiempo de llegada de las señales de cada antena que se encuentran alrededor. Esta medición de tiempos es inherente al sistema GSM porque en las transmisiones se usan tramas que se envían sincronizadamente. Una vez estimada la distancia a cada antena y conociendo la posición exacta de cada una de ellas, se aplicaría multilateración.

Esto implica que en zonas urbanas, donde existen mayor densidad de células, se podría obtener una precisión mucho mayor (decenas de metros) que en las zonas rurales, en donde se depende de una sola estación base para un rango de varios kilómetros.

Una variante del *ToA* es el *ToA-Diferencial*, que compara las señales recibidas de dos en dos estaciones. Con este método se compensan errores de reflexión respecto al *ToA* anterior cuando el usuario se encontrara en zona urbana. Conociendo la posición de las estaciones el dispositivo crea unas hipérbolas, en vez de circunferencias centradas en cada estación base. Esta hipérbola se crea calculando la distancia hacia dos estaciones base (BTS) y haciendo constante la

suma de ambas cantidades (ver figura 3.3). La intersección de las hipérbolas indicará la posición estimada del usuario. [Fis98]

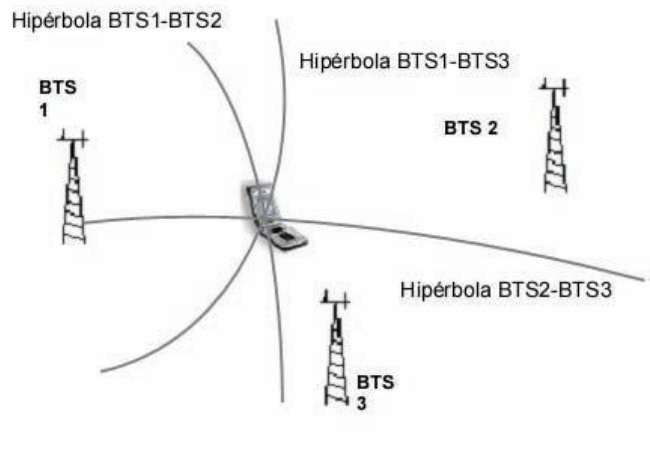


Figura 3.3: Ejemplo de hipérbolas formadas por dos estaciones base en el método ToA-Diferencial.

3.3 Localización en interiores

Como se verá más adelante, la problemática de la localización en interiores se debe a los fenómenos que sufren las ondas electromagnéticas interactuando con la estructura del edificio (reflexión, difracción y *multipath*). En estos últimos años se han investigado nuevas propuestas que hacen uso de diferentes tecnologías para alcanzar un éxito similar de los sistemas empleados en zonas de exteriores. Estas investigaciones plantean dos cuestiones importantes: una es el factor económico y la otra es el factor tecnológico. Económicas, porque las opciones propuestas necesitan de una importante infraestructura (sensores, estaciones base, puntos de control, etc). Y técnicas, porque plantean retos tecnológicos algo más avanzados que en los sistemas de localización habituales.

En primer lugar, y como método más primitivo, tendríamos los sistemas basados en sensores fijos colocados en puntos críticos del edificio. El objetivo de estos sistemas no era localizar al usuario, sino controlar los accesos a zonas particulares del edificio. En una primera fase, los sensores detectan que hay un usuario cercano a su posición. Posteriormente, identifican a ese usuario. Si se quisiera utilizar este sistema para localizar al usuario, su posición quedará limitada a las cercanías del sensor o punto de control [Maa97].

Por otro lado, se encuentran los sistemas que hacen uso de métodos matemáticos, como trilateración o triangulación, para obtener un punto

determinado. Para realizar estos cálculos matemáticos y hallar las distancias se utilizan métodos variados, tales como medir el retardo de propagación de las señales, o medir la potencia de señal que se recibe.

Un tercer sistema de localización en interiores se basa en el efecto físico que tiene la estructura del edificio sobre las ondas electromagnéticas aplicando la teoría electromagnética o de rayos [Tay09].

3.3.1 Aplicaciones

Las aplicaciones para un sistema de localización en interiores pueden ser las siguientes:

- Para la organización de almacenes e inventarios en el ámbito industrial. Desde un centro servidor se haría un seguimiento de los empleados, vehículos y de los productos de la empresa. Se consigue una supervisión y una producción más eficiente, se mejoraría la seguridad de los empleados y se reducirían los errores en las entregas.
- En el área doméstica se podrían realizar aplicaciones informáticas para domótica que se iniciarían automáticamente cuando el usuario entrase en un área determinada.
- En áreas subterráneas de las ciudades como las alcantarillas, ayudaría a encontrar rápidamente a los empleados de limpieza y salud pública que trabajen bajo tierra y tengan algún percance inesperado.
- En museos o ferias de exposiciones, en donde el usuario, además de orientarse conociendo su posición, recibiría información en su dispositivo de cada obra expuesta a la que se acerque.

3.3.2 Identificación por radiofrecuencia (RFID)

Su función principal es la de identificar al usuario para permitirle acceso a la siguiente estancia. A cada usuario se le entrega un *tag* (o identificador) que puede ser tanto pasivo, no poseen batería, como activo, con batería. Un tag pasivo se compone de una unidad de procesamiento y un transmisor de radiofrecuencia. Su función exclusiva es emitir una identificación numérica al acercarse a un lector. El tag se cargará por inducción electromagnética al acercarse al lector, en cuyo momento tendrá energía suficiente para emitir su identificación (ver figura 3.4). La

desventaja de estos tags pasivos es su poco alcance (0'5 metros) y su poca capacidad para emitir durante un tiempo más alargado.

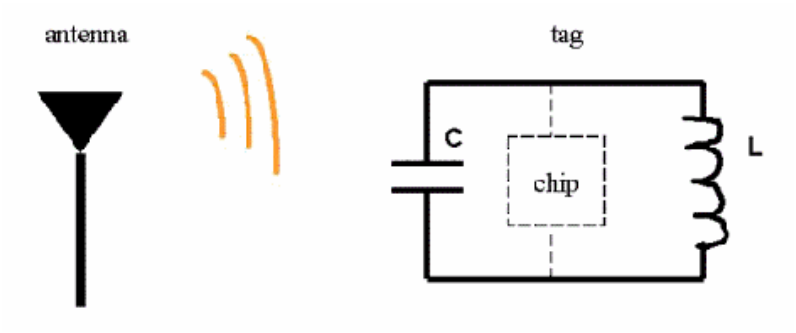


Figura 3.4: Tag pasivo. El circuito entra en resonancia y comienza a emitir su ID codificada dentro del chip.

Los tags activos poseen una batería con la que pueden radiar su información a mucha más distancia (entre 10 y 80 metros). Estos tags son los adecuados, por tanto, para los sistemas de localización. Con tags activos y múltiples sensores instalados se detectaría la presencia del usuario y se aplicarían técnicas de triangulación para obtener un punto bien definido de localización [Fin03].

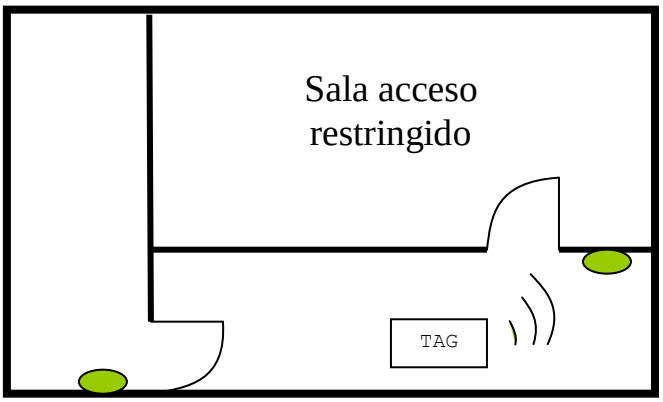


Figura 3.5: Tag que envía por radiofrecuencia la identificación del usuario para acceder a zonas determinadas.

3.3.3 Infrarrojos

Fue de las primeras tecnologías usadas para la implementación de sistemas de localización en interiores. Los tags que se utilizan en este modelo radian en modo isotrópico, y no punto-a-punto como es habitual en los sistemas de infrarrojos. La principal desventaja de esta tecnología es que las señales infrarrojas no atraviesan ningún obstáculo o pared, por lo que habría que instalar sensores en cada habitación. Aún emitiendo en 360º, el cuerpo del propio portador del tag hará de pantalla con lo que habria que instalar sensores en paredes opuestas para que siempre se pueda recibir la señal del portador [Wan92].

Además, los infrarrojos son señales que trabajan casi a la misma longitud de onda que la luz visible, con lo que la propia radiación solar podría enmascarar dichas señales y anular el sistema por completo.

Este servicio de localización calcula la posición del usuario desde un servidor conectado a los sensores infrarrojos, ya que este servidor es quien conoce la localización de cada sensor.

3.3.4 Ultrasonidos

Este sistema sigue usando los tags como elemento a detectar, pero en este caso, los tags responderán a mensajes que envíe la infraestructura del edificio. La empresa AT&T ha desarrollado un sistema comercial basado en ultrasonidos llamado *Bat*. La red se compone de sensores y estaciones base formando una red en malla. Una estación base emite periódicamente el código de identificación que quiere detectar. El tag aludido emitirá un pulso de muy corta duración que será recibido por los sensores desplegados. Estos, a su vez, habrán contabilizado el tiempo de retardo desde que emitió la estación base hasta que se recibió la señal del tag correspondiente, con lo que sabremos la distancias a los sensores. Si hay varios sensores que reciban la señal del tag se podrá conocer la posición exacta del tag con un error de hasta tres centímetros, como se publica en las especificaciones del sistema BAT [Web-7].

Dicho sistema fue ideado por los laboratorios de AT&T. Su principal desventaja es el alto presupuesto para la instalación del sistema, ya que para conseguir tal precisión, se colocaron 240 sensores en el techo en una superficie de 1000 m^2.

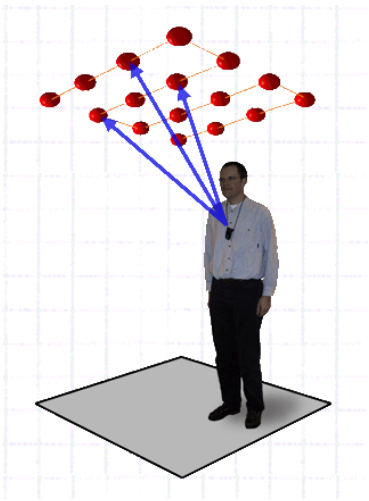


Figura 3.6: Sistema BAT. Usuario con un tag de ultrasonidos colgado al cuello y sensores colocados en el techo.

3.3.5 Visión artificial

Esta tecnología es un campo de la inteligencia artificial. Se encarga de recoger la información ofrecida por cámaras y analizarlas utilizando técnicas de procesado de la imagen, con el objeto de identificar cada objeto o individuo que aparezca en pantalla. Este modelo suele estar optimizado para entornos controlados y no es efectivo en entornos donde haya mucha actividad, con lo que haría falta que los usuarios lleven *tags*. En este caso serían 'tags visuales' que el ordenador sea capaz de reconocer fácilmente.

Un grupo de investigación de la Universidad de Alcalá ha desarrollado un sistema de navegación automovilística en el que al pasar por un túnel o zona montañosa que interfiera en la señal GPS se pondría en activo un sistema de dos cámaras que marcarán al usuario la trayectoria a seguir [Web-8].

3.3.6 Wireless LAN

Como se ha comprobado en las descripciones anteriores, para los sistemas de localización en interiores se debe hacer una instalación previa de sensores o estaciones base. Para la localización por Wifi esto no tiene por qué ocurrir, ya que en la mayoría de las situaciones se aprovechará la instalación de los puntos de acceso (*access points*) que ya se han desplegado por el edificio para la conexión a Internet.

Hay tres mecanismos distintos de localización basados en redes Wifi. El primero de ellos se basa en la medición de las potencias de señal que se recibe de cada punto de acceso, el segundo es el ToA (*Time of Arrival*) y el tercero AoA (*Angle of Arrival*).

Este PFC se centrará exclusivamente en el método de la lectura de las potencias de señales el cual se basa en realizar un estudio previo del edificio donde quiera instalarse el sistema. Este estudio previo consiste en realizar un mapeado de potencias de cada planta del edificio para ver qué nivel de potencia se recibe en cada punto, mientras el usuario se acerca y se aleja del punto de acceso.

Un ejemplo de mapeado de potencias podría ser el mostrado en la siguiente figura 3.7, donde se representa en color claro la potencia más alta y en color oscuro la más baja:

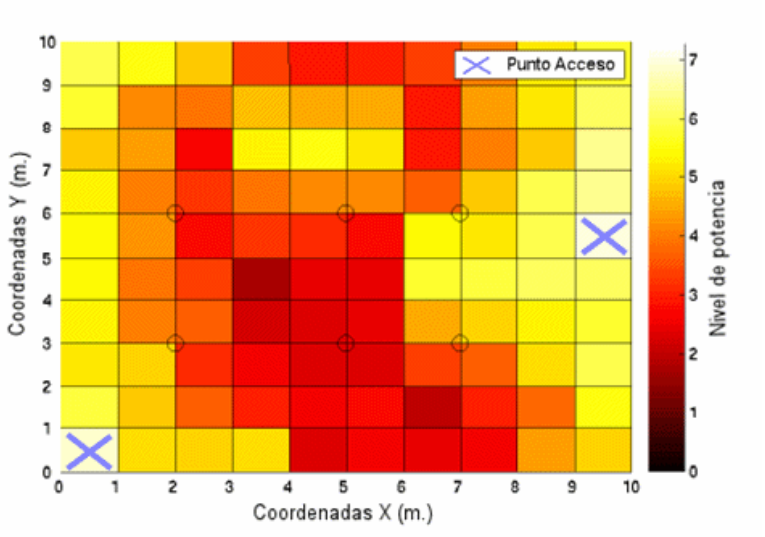


Figura 3.7: Mapeado de potencias que representa a color los distintos niveles de potencia para un punto de acceso.

El método ToA se basa en el envío de tramas patrón con un *timestamp* determinado en cada una de ellas y compararlo con el propio *time* del dispositivo. A través de las diferencias de tiempos se estimaría la distancia de cada punto (ver figura 3.8).

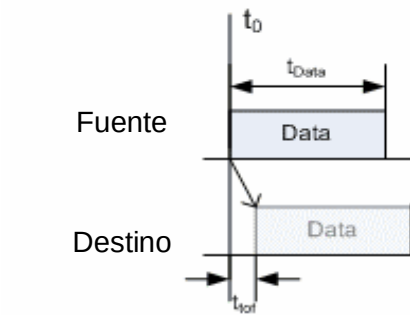


Figura 3.8: Diagrama que muestra la diferencia de tiempos desde que se envía un paquete hasta que se recibe.

En este segundo sistema todos los dispositivos deberían tener su horario bien sincronizado. La ventaja de ambos sistemas es que son compatibles entre sí y se podrán mejorar las prestaciones haciendo uso de los dos sistemas a la vez.

El tercer sistema, AoA, se basa en el cálculo del ángulo de llegada de las señales. En este caso no será el usuario quien calcule su posición sino los propios puntos de acceso. Esto se debe a que el propio usuario no podrá calcular el ángulo de llegada de las señales porque las antenas de los dispositivos móviles son omnidireccionales, de forma que reciben señal desde todos los ángulos posibles sin conocerse su procedencia.

Sin embargo, si se diseñan los puntos de acceso con antenas giratorias direccionales, las cuales poseen un diagrama de radiación de alta directividad (es decir, envían y reciben señales en una dirección determinada), el propio punto de acceso será capaz de conocer desde qué ángulo está recibiendo la mayor potencia de señal de algún dispositivo móvil:

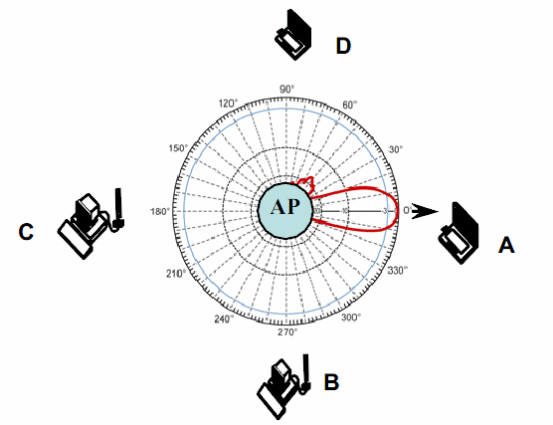


Figura 3.9: Diagrama de radiación de un punto de acceso con antena directiva que gira continuamente 360°.

Un servidor informático accederá a la información en cada punto de acceso de dichos ángulos y será el mismo servidor quien calcule la posición del usuario a través de triangulación. Por ejemplo, en el caso de la figura 3.9, mientras el punto de acceso realiza un barrido detectará mayor potencia de señal del dispositivo A a cero grados. Para el resto de dispositivos el servidor informático conocerá, igualmente, en qué ángulo se encuentra cada uno: el dispositivo D (a 90 grados), el dispositivo C (180 grados) y el dispositivo B (270 grados). El servidor realizará otra consulta al resto de puntos de acceso y obtendrá otra serie de valores. Basta que el servidor conozca dos ángulos de dos puntos de acceso, como se demuestra en el apartado 4.2 'Triangulación', para aplicar triangulación y averiguar la posición del usuario (ver figura 3.10).

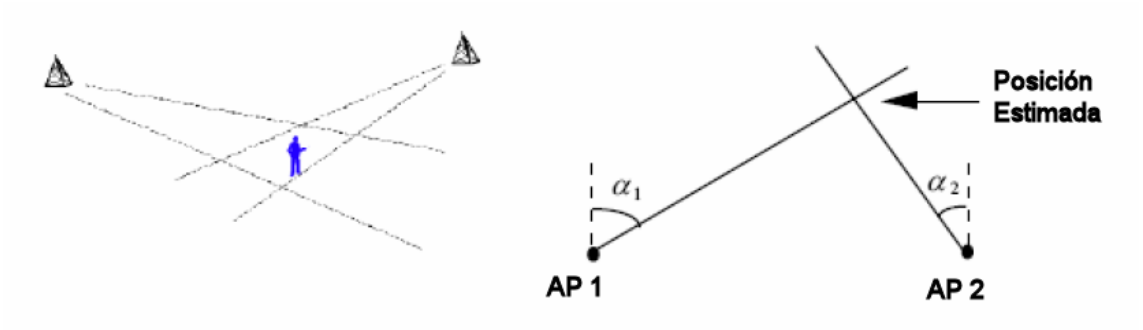


Figura 3.10: El AoA calcula la posición basándose en el ángulo de recepción desde los puntos de acceso (AP), no desde el usuario.

3.3.7 Ventajas e inconvenientes de un sistema basado en Wifi

La ventaja primordial del sistema de localización Wifi es el aprovechamiento de los puntos de acceso que ya existen desplegados en casi cualquier edificio público o de negocios, con lo que este proyecto se ahorraría el presupuesto para la instalación de los puntos de acceso Wifi del edificio. De esta forma, además, el despliegue de puntos Wifi cumpliría las dos funciones simultáneamente: conexión a Internet o a la red local y sistema de localización. Esto no ocurre en un sistema de infrarrojos o Bluetooth, en donde las comunicaciones no soportan redes de área local, sino comunicaciones punto a punto exclusivamente. Para dar cabida a una red local habría que diseñar e implementar un protocolo de comunicaciones específico para cada entorno, en donde existiera un servidor que gobierne todos los puntos de contol infrarrojos o Bluetooth de la zona, con el aumento considerable de presupuesto que conllevaría su diseño.

Destacar, también, que para un sistema de localización por Wifi el usuario no debe acceder a la red desplegada con ningún nombre de usuario ni contraseña, con lo que no importa si los puntos de acceso usan encriptación.

Como un nivel adicional en la infraestructura del sistema de localización Wifi, se podría usar un servidor central al que el dispositivo móvil envíe los datos necesarios para que el servidor sea quien calcule su posición. Con esto se conseguiría mayor facilidad para el usuario a la hora de usar su dispositivo porque no tendría que tener previamente instalado ningún mapa del edificio, sino que sería el servidor quien entregase al usuario el mapa y su localización. Además, con este servidor central, se tendría un control total sobre los usuarios existentes en la sede o empresa para una mayor seguridad.

En el caso de no disponer de red Wifi en el edificio, su instalación sería de muy bajo presupuesto porque sólo habría que adquirir puntos de acceso e instalarlos en diferentes puntos del edificio, sin tener que configurarlos para la conexión a Internet ni cablearlos entre ellos.

El factor más perjudicial a la hora de utilizar un sistema de localización por Wifi es el comportamiento de las señales electromagnéticas en zonas de interior, esto es, la variabilidad continua y aleatoria de las señales en el entorno que resulta muy difíciles de describir matemáticamente. En el siguiente apartado se explicará la propagación de dichas señales de radiofrecuencia y los diferentes modelos matemáticos que intentan representar su comportamiento empírico.

3.4 Propagación de Señales en Interiores

En un sistema de localización en interiores es fundamental conocer el comportamiento de las ondas electromagnéticas para poder realizar una estimación teórica lo más fiel posible del escenario donde se implantará el sistema. Las señales sufren de atenuación en el vacío, atenuación por absorción de la señal, reflexiones y otros fenómenos físicos que se estudiarán a continuación.

Actualmente hay dos modelos de propagación de ondas de radio: el empírico y el determinístico. El modelo empírico es el que se basa en estudiar experimentalmente la zona tomando medidas de señales en cada punto. Estas medidas se adaptan, posteriormente, a un determinado modelo matemático. El modelo empírico es el elegido en la realización de este PFC.

El modelo determinístico describe el comportamiento de las señales a través de la teoría óptica de rayos, en donde se suceden los fenómenos de reflexión, refracción y scattering. Este modelo realiza los cálculos a partir de la estructura del

edificio, los materiales usados en su construcción, características de la señal emisora, como la potencia de emisión, longitud de onda de trabajo, tipo de antena emisora, etc.

3.4.1 Modelo empírico

En el modelo empírico se estudian las señales a través de medidas experimentales de la potencia que se recibe en cada punto del entorno. La magnitud de potencia varía en un rango muy amplio con lo que se usan escalas logarítmicas para su estudio, como son el "dBm".

La unidad 'dBm' se define tal que:

$$dBm = 10 \times \log \frac{Potencia\,(mW)}{1mW}$$

Para modelar el comportamiento de las ondas electromagnéticas se comenzará con el estudio de la potencia de señal recibida a una distancia "d" del transmisor, cuando entre ellos hay línea de visión directa sin obstáculos:

$$P_{fs}(dB) = 20\log_{10}\frac{4\pi \cdot d}{\lambda}$$

Donde 'λ' es la longitud de la onda emitida. Esta ecuación también se conoce como fórmula de Friis.

En el caso de señales Wifi 802.11(b/g) que trabajan a 2'4 Ghz, se obtiene una longitud de onda tal que:

$$\lambda = \frac{c}{f} = \frac{3 \cdot 10^8\ (^m/_s)}{2'4 \cdot 10^9\ (Hz)} = 0'125\ m.$$

En el caso que concierne a este PFC (entornos en interiores) las señales, aunque viajan libremente por el aire, también interactúan con las paredes, mobiliario y personas que se encuentren en la trayectoria. Un modelo matemático simple que represente dicho comportamiento se describe con la siguiente fórmula:

$$Pérdidas(dB) = P_{fs}(d_0) + 10n\log_{10}\frac{d}{d_0}$$

Donde 'd' es la distancia a la que se encuentra el usuario respecto la fuente, y 'd_0' es una distancia de referencia que suele tomarse como 1 metro. El factor 'n' es un factor que indicará la facilidad que tienen las ondas en llegar al destino, y depende, en gran medida, del entorno y la estructura del edificio.

En el espacio libre el factor 'n' toma un valor de dos. Si al realizar el estudio en interiores se obtiene un valor mayor de dos, significa que las señales tienen mayores inconvenientes para viajar respecto el espacio libre. Mientras que si 'n' es menor que dos, significa que el entorno ayuda a las señales a viajar distancias más lejanas. El fenómeno del multitrayecto es uno de los más factores que influirá favorablemente para que 'n' disminuya respecto el espacio libre, mientras que la absorción de obstáculos y paredes será el factor que influya negativamente.

Un segundo modelo matemático que debe tenerse en cuenta es el que incluye en sus cálculos la trayectoria de las señales a través de muros y paredes. Este nuevo modelo añade, a la ecuación anterior, dos términos adicionales que tienen en cuenta esta atenuación que sufre la propagación de señales [Skl97].

$$Pérdidas(dB) = \underbrace{P_{fs}(d_0)}_{1} + \underbrace{10n\log_{10}\frac{d}{d_0}}_{2} + \underbrace{\sum Muros}_{3} + \underbrace{\sum AtenuacionesVarias}_{4}$$

1. Es la fórmula de Friis en espacio abierto para una distancia de referencia 'd_0' de 1 metro.
2. Pérdidas debidas a la estructura del edificio. Dependerá del factor 'n'.
3. Pérdidas debidas a paredes interponiéndose en el campo de visión.
4. Son variaciones aleatorias de potencia de señal debido a cambios en el medio físico, interferencias u obstrucción de personas en movimiento.

Uno de los fenómenos que afectará en mayor medida al sistema de localización será la variabilidad de las señales a lo largo de su recorrido. En la figura 3.11 se puede comprobar cómo se comportan las señales en interiores. Se comprueba que la potencia de las señales no decrecen uniformemente según nos alejamos del transmisor, sino que tiene máximos y mínimos locales que se producen en cada longitud de onda de la señal (el color rojo representa mayor potencia, el azul menor potencia).

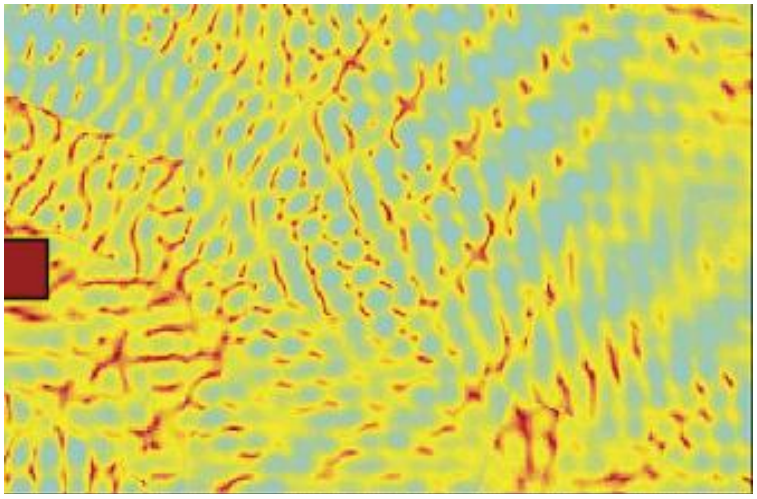


Figura 3.11: Espectrometría de señales Wifi en un recinto cerrado.

La atenuación debida a muros o paredes es fácil de calcular y así se establece en este PFC; pero la variabilidad del segundo término insta a utilizar medios estadísticos de variable aleatoria que no se han incluido en este proyecto.

3.4.2 Modelo determinista

El modelo determinista consiste en procesar cálculos de la teoría de rayos en donde intervienen fenómenos como la reflexión, difracción y dispersión. Este modelo (en inglés, *ray-tracing*) es el que se utiliza en el modelado de imágenes 3-D por ordenador, el cual produce efectos de luz y texturas realistas.

Se comienza realizando un modelado 3-D del entorno donde se va a implantar el sistema de localización, como se muestra en la figura 3.12.

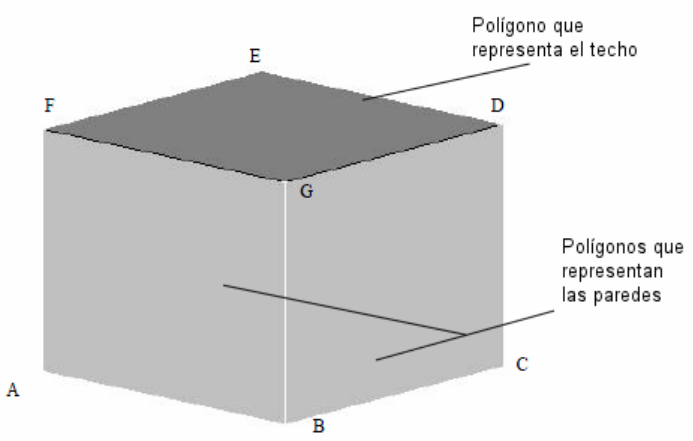


Figura 3.12: Modelado 3-D básico de un edificio.

Una vez finalizado el diseño hay que indicar de qué material están formadas las paredes, objetos y la rugosidad de cada uno. Estas características se usan para ver en qué grado afectan los fenómenos de reflexión, difracción y dispersión: unos materiales absorberán más energía y otros serán más reflectantes. [She00]

Una vez completado el modelado del entorno de trabajo se necesitan obtener las características del emisor, como son el diagrama de radiación de la antena, potencia de emisión y longitud de onda de trabajo. Si, además, se desea obtener un comportamiento muy detallado de las señales, se deben añadir factores como la meteorología: temperatura, humedad, etc. Con todos estos datos, se aplican las fórmulas del comportamiento de señales electromagnéticas de espacio-libre y se obtiene algo parecido a lo que muestra la figura 3.13.

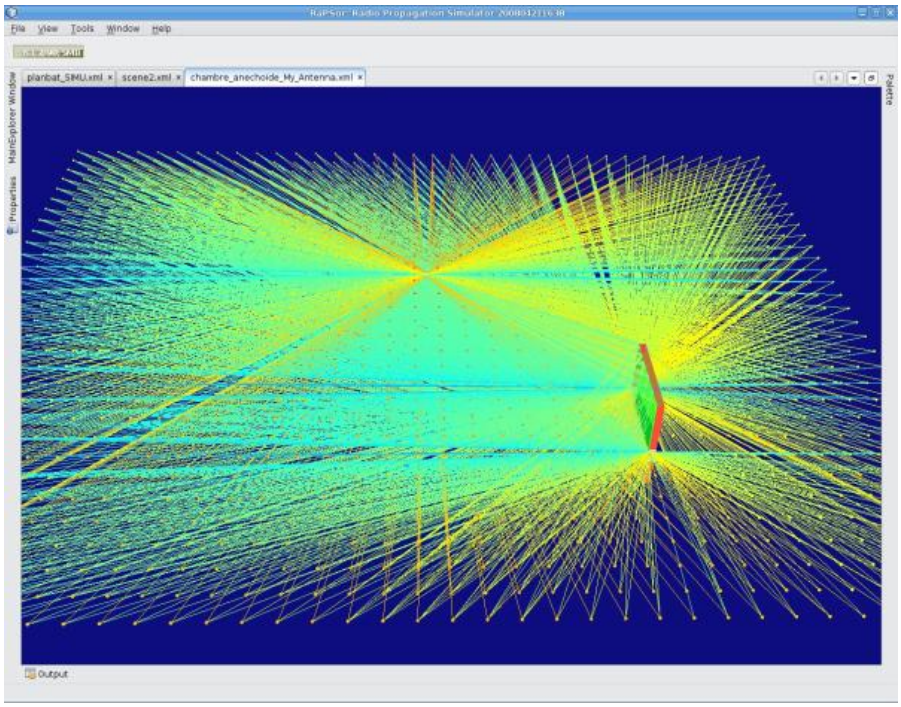


Figura 3.13: Ejemplo de ray-tracing que estudia el comportamiento de las señales con una antena emisora y un obstáculo.

Esta representación sí refleja con precisión el comportamiento de las ondas en interiores a costa de hacer un uso muy intensivo del procesador (en la industria del cine utilizan múltiples ordenadores para calcular un solo fotograma y tenerlo listo en varios días). Para sistemas en donde se necesiten resultados en tiempo real se simplifican algunas ecuaciones para no invertir en demasía en los sistemas computacionales [Nid06].

Los tres fenómenos más importantes que sufren las señales electromagnéticas y que son tenidos en cuenta por la teoría de rayos son: reflexión, difracción y scattering.

• **Reflexión**: ocurre cuando una onda choca con un objeto mayor que la longitud de onda de la propia señal. Aparte de reflejarse en otra dirección, también puede atravesar el propio obstáculo, aunque con una potencia menor.

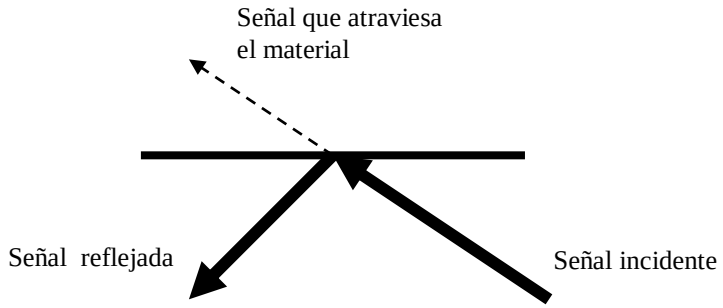


Figura 3.14: Fenómeno de reflexión, donde gran parte de la señal se refleja y otra atraviesa el obstáculo.

• **Difracción**: ocurre cuando una señal choca con una superficie con esquinas o ángulos agudos. Esto provoca que los ángulos hagan de fuente de radiación de la señal. Este efecto es útil en un entorno de interiores porque ayuda a que la señal llegue a puntos que no podría llegar en visión directa.

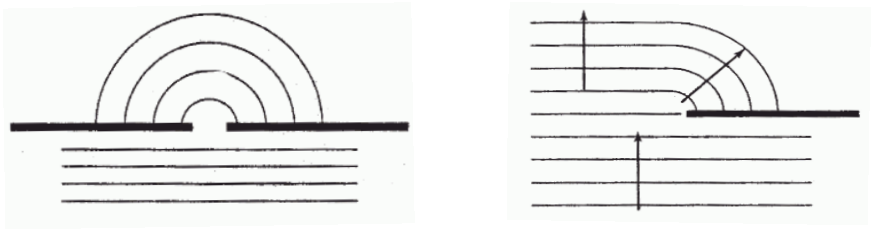


Figura 3.15: Creación de nuevos frentes de ondas debido al fenómeno de la refracción.

- **Dispersion (Scattering)**: ocurre cuando una señal choca con un objeto más pequeño que la longitud de onda de la señal. El scattering disipa levemente la señal y se produce un efecto parecido al de la reflexión.

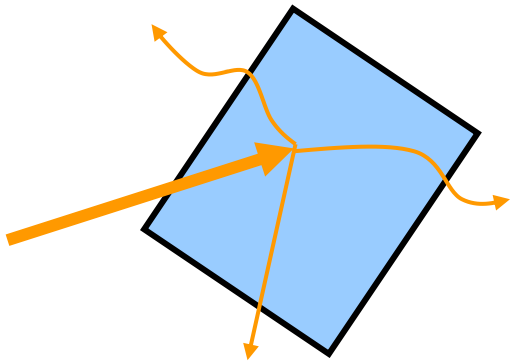


Figura 3.16: Efecto de scattering, donde la señal se dispersa al chocar con un obstáculo.

- **Multitrayecto (Multipath)**: El fenómeno del multitrayecto es consecuencia de la difracción de la reflexión y de la dispersión. Estos fenómenos físicos provocan que una señal llegue a un mismo punto desde varias trayectorias distintas aparte de la línea de visión directa (ver figura 3.17). Cuando estas trayectorias se cruzan en un punto pueden, tanto aumentar, como disminuir la calidad de la señal según cómo intercedan las ondas (diferencia de fase entre cada una de ellas). Por ejemplo, dos ondas iguales con una diferencia de fase de 180º se cancelarán la una a la otra.

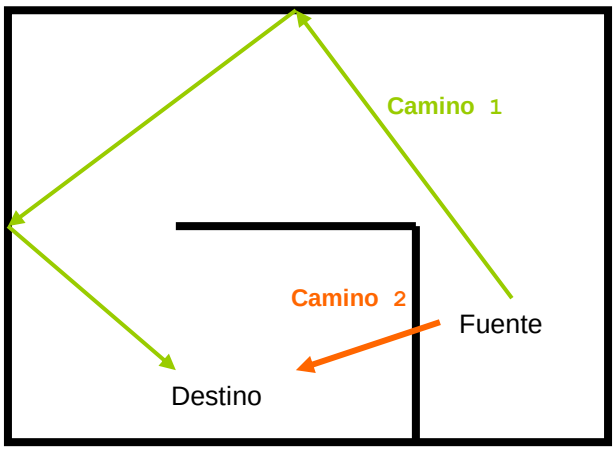


Figura 3.17: Ejemplo de propagación multi-trayecto.

CAPÍTULO 4

ALGORITMOS DE LOCALIZACIÓN

4.1 Introducción

Todo sistema de localización debe hacer uso de un algoritmo que será el encargado de procesar los datos obtenidos por el dispositivo del usuario y ofrecer una posición estimada. Dependiendo de si el sistema es al aire libre o en zonas de interior, algunos algoritmos no podrán implementarse. En el caso de este PFC, que se basa en la lectura de la potencia de señal, no se podrá usar, por ejemplo, el método de triangulación, que se basa en la lectura de ángulos de recepción.

En este capítulo se describe el funcionamiento de los algoritmos implementados para este PFC, como son: trilateración, multi-lateración, minimización del residuo y Min-Max. Además, se han añadido otros algoritmos usados en otros sistemas de localización, como son: triangulación o métodos estadísticos, para ofrecer un estado global de las implementaciones más usadas.

4.2 Triangulación

Este método se basa en calcular la posición a través de dos ángulos respecto dos puntos conocidos. Se suele utilizar en navegación marítima. Una vez conocidos dos ángulos respecto a la costa y la distancia entre ellos, se podrá obtener la distancia del barco a la costa.

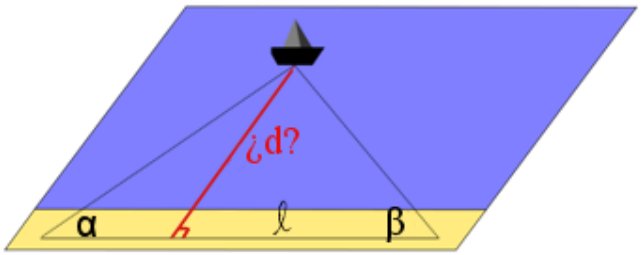


Figura 4.1: Variables involucradas en la triangulación.

El objetivo es conocer la distancia "d". Aplicando el teorema de Pitágoras en ambos triángulos se obtiene la ecuación:

$$l = \frac{d}{\tan\alpha} + \frac{d}{\tan\beta}$$

De aquí se despeja "d" y ya se conocería la localización del usuario:

$$d = l \cdot \frac{\tan\alpha \cdot \tan\beta}{\tan\alpha + \tan\beta}$$

Este método es el utilizado en el sistema *Angle-of-Arrival*, donde los puntos de acceso, diseñados específicamente con un antena unidireccional giratoria, son quienes calculan dichos ángulos.

En una zona de interiores este sistema suele apoyarse con otro sistema de localización alternativo, ya que las señales recibidas vendrán rebotadas desde distintos ángulos tras chocar con muros y otros obstáculos del edificio.

4.3 Trilateración

Se basa en conocer la posición a través de las distancias de tres puntos conocidos. Primeramente se calcula la distancia hacia cada punto de acceso y se dibuja en el plano una circunferencia centrada en el punto de acceso, y de radio, la distancia estimada por el dispositivo:

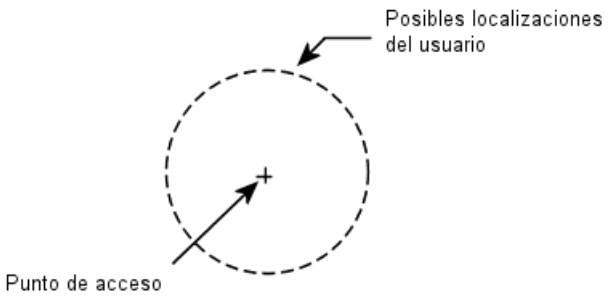


Figura 4.2: Circunferencia formada por un punto de acceso.

La circunferencia formada representa todos los puntos posibles en los que se podría encontrar el usuario.

A continuación, se representa en el mapa la circunferencia del segundo punto de acceso con la distancia estimada al usuario, con lo que aparecerán dos posibles localizaciones del usuario (figura 4.3):

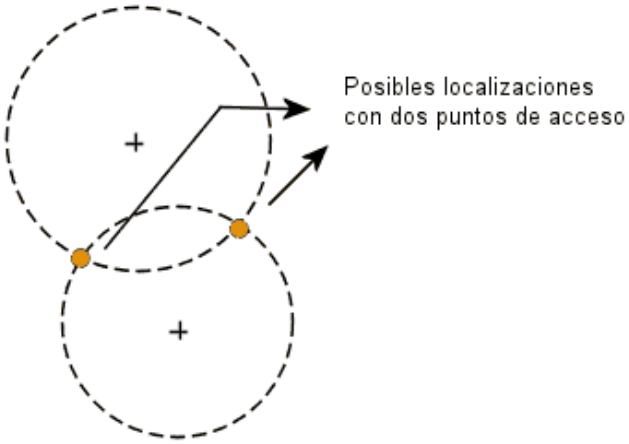
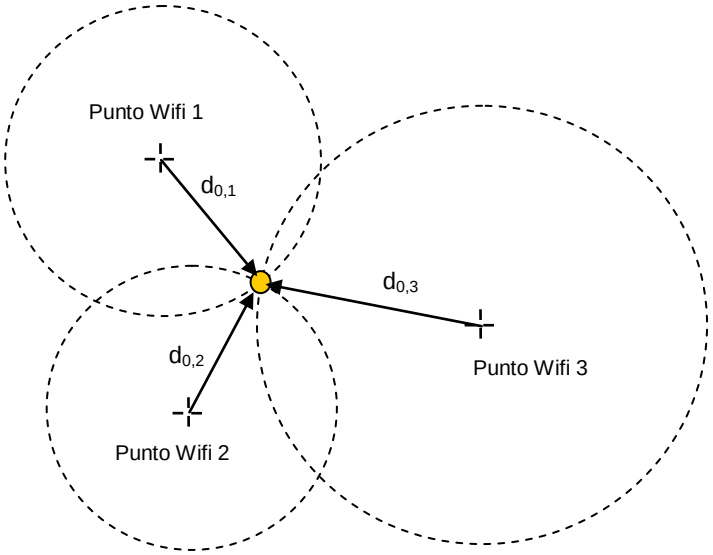


Figura 4.3: Intersección de dos circunferencias con dos puntos de acceso.

Finalmente, se añade la tercera circunferencia del tercer punto de acceso, con lo que se averiguará la posible localización del usuario, tal como se muestra en la figura 4.4.

Para el cálculo matemático de este algoritmo se comienza definiendo las tres ecuaciones de la circunferencia correspondientes a cada punto Wifi (con sus coordenadas x_i,y_i y radio de circunferencia '$d_{0,i}$'):



Ecuación del punto Wifi 1: $(x-x_1)^2+(y-y_1)^2=d_{0,1}$

Ecuación del punto Wifi 2: $(x-x_2)^2+(y-y_2)^2=d_{0,2}$

Ecuación del punto Wifi 3: $(x-x_3)^2+(y-y_3)^2=d_{0,3}$

Figura 4.4: Punto de intersección de tres circunferencias y sus correspondientes ecuaciones.

Las incógnitas son "x" e "y". Para que un ordenador pueda resolverlas hay varios métodos; en nuestro caso despejaremos manualmente las coordenadas "x" e "y". Para ello, se reduce el sistema anterior a un sistema lineal restando una ecuación de las otras dos:

$$d_{0,1}{}^2 - d_{0,3}{}^2 = 2(x_3 - x_1)x + x_1{}^2 - x_3{}^2 + 2(y_3 - y_1)y + y_1{}^2 - y_3{}^2$$
$$d_{0,2}{}^2 - d_{0,3}{}^2 = 2(x_3 - x_2)x + x_2{}^2 - x_3{}^2 + 2(y_3 - y_2)y + y_2{}^2 - y_3{}^2$$

Ahora despejamos la coordenada "x" e "y". Agrupando algunas operaciones en "A" y "B" para una presentación más agradable al lector obtenemos:

$$A = (y_1 - y_2) \cdot (x_1{}^2 - x_3{}^2 + y_1{}^2 - y_3{}^2 + d_{0,3}{}^2 - d_{0,1}{}^2)$$
$$B = (y_1 - y_3) \cdot (x_1{}^2 - x_2{}^2 + y_1{}^2 - y_2{}^2 + d_{0,2}{}^2 - d_{0,1}{}^2)$$

$$x = \frac{A \cdot B}{2 \cdot ((y_1 - y_2) \cdot (x_1 - x_3) - (y_1 - y_3) \cdot (x_1 - x_2))}$$

$$y = \frac{(x_1{}^2 - x_2{}^2) - 2 \cdot x \cdot (x_1 - x_2) + y_1{}^2 - y_2{}^2 + d_{0,2}{}^2 - d_{0,1}{}^2)}{2 \cdot (y_1 - y_2)}$$

La trilateración es sensible a las distancias estimadas de cada circunferencia, hasta el punto de obtener resultados extremadamente lejanos a donde supuestamente deberíamos estar. Esos puntos lejanos pueden caer, incluso, fuera de nuestro mapa. A estos puntos lejanos los denominaremos "puntos no válidos".

La aplicación de este PFC se ha programado de forma que cuando se obtiene un punto no válido, se modificarán las coordenadas del resultado para que la posición se encuentre dentro de los límites de la planta del edificio, cuyas dimensiones son, en el caso que estudia este PFC, de 61'5 por 11'5 metros.

4.3.1 Bilateración

Este escenario se produce cuando sólo existen dos puntos Wifi conocidos. La intersección de dos circunferencias ofrece tres tipos de resultados: o se intersectan en dos puntos, o se intersectan en un solo punto o no se intersectan. En el caso de que se intersecten en dos puntos (caso 1 en la siguiente figura) sólo habrá que resolver un sistema de dos ecuaciones con dos incógnitas.

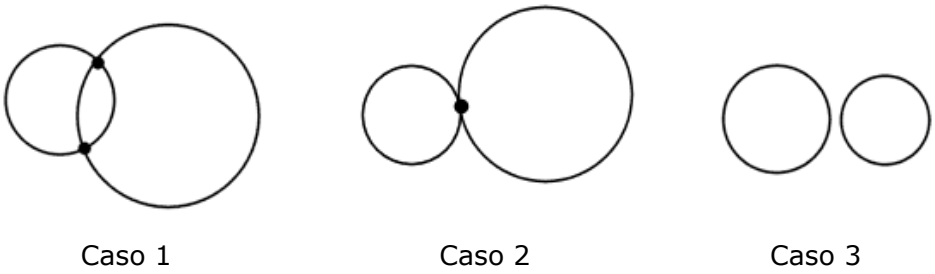


Figura 4.5: Posibles casos de bilateración.

La aplicación mostrará las dos posibles localizaciones en el mapa. Habrá veces que una de las soluciones se encuentre fuera del mapa, con lo que el otro resultado obtenido será el punto válido y ese será la única localización que se muestre en pantalla.

Habrá otras veces que ambos puntos estén fuera del mapa, con lo que la localización no será posible debido a una estimación errónea de las distancias y, al igual que en el apartado anterior, se modificará el resultado para adaptarlo a la planta del edificio.

En el caso 2 el dispositivo ha averiguado las distancias hacia cada punto de acceso con gran precisión, con lo que el sistema de ecuaciones obtendrá un único resultado.

En el caso 3 de la figura 4.5, las circunferencias no intersectan en ningún punto. En estos casos se aplicará un método distinto para resolver porque el sistema de ecuaciones no será resoluble. La metodología será hallar el punto medio entre las dos circunferencias, como aparece en la figura 4.6.

El resultado obtenido será una localización única y, además, siempre válida, porque el espacio entre las dos circunferencias siempre abarcará un espacio del interior del mapa.

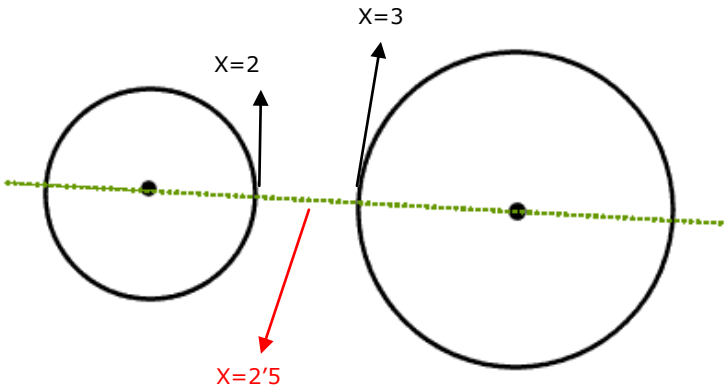


Figura 4.6: Cálculo de la coordenada X cuando las dos circunferencias no intersectan.

4.3.2 Una sola circunferencia

En el caso que sólo se detecte un punto Wifi a nuestro alrededor, no se obtendrá ningún punto de localización sino una zona en forma de circunferencia alrededor de ese único punto Wifi, como se muestra en la figura 4.7.

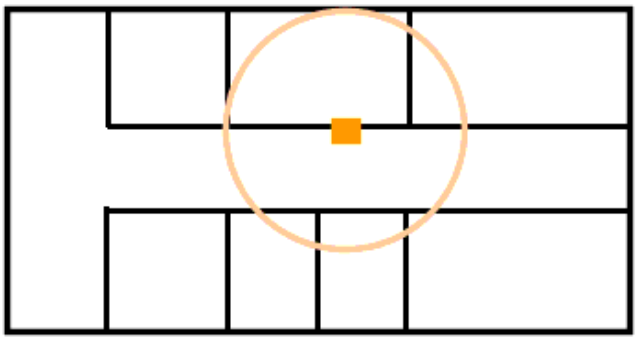


Figura 4.7: Representación de una circunferencia en el caso de que sólo exista un punto de acceso.

4.4 Multi-lateración

En el caso de existir más de tres puntos Wifi, a primera vista, se podría pensar que se obtendría una mayor precisión de la localización, pero no siempre ocurre así. La muti-lateración será efectiva cuando se estimen correctamente las distancias a cada punto Wifi, porque si se desvía levemente el valor de la distancia de una sola circunferencia, seobtendría un punto no válido (tal como ocurre en el

caso de la trilateración). Como en el caso de trilateración se definirán las ecuaciones de cada punto Wifi:

$$\begin{aligned}
&(x_1 - x)^2 + (y_1 - y)^2 = d_1^2, \\
&\vdots \\
&(x_n - x)^2 + (y_n - y)^2 = d_n^2,
\end{aligned}$$

Donde "n" es el número de puntos de acceso que nuestro dispositivo móvil está detectando; y "d_n" es la distancia a cada punto de acceso.

Para resolver un sistema de múltiples ecuaciones no-lineales se aplicaría el mismo paso que en la trilateración: restar la última ecuación de las *n-1* restantes.

$$\begin{aligned}
&x_1^2 - x_n^2 - 2(x_1 - x_n)x + y_1^2 - y_n^2 \\
&\quad - 2(y_1 - y_n)y = d_1^2 - d_n^2, \\
&\vdots \\
&x_{n-1}^2 - x_n^2 - 2(x_{n-1} - x_n)x + y_{n-1}^2 \\
&\quad - y_n^2 - 2(y_{n-1} - y_n)y = d_{n-1}^2 - d_n^2.
\end{aligned}$$

Ahora se reordenan los términos según la estructura matricial del tipo: $A \cdot \begin{pmatrix} x \\ y \end{pmatrix} = b$ Donde *(x,y)* es el punto a calcular, y 'A' y 'b' son:

$$A = \begin{bmatrix} 2(x_1 - x_n) & 2(y_1 - y_n) \\ \vdots & \vdots \\ 2(x_{n-1} - x_n) & 2(y_{n-1} - y_n) \end{bmatrix},$$

$$b = \begin{bmatrix} x_1^2 - x_n^2 + y_1^2 - y_n^2 + d_n^2 - d_1^2 \\ \vdots \\ x_{n-1}^2 - x_n^2 + y_{n-1}^2 - y_n^2 + d_n^2 - d_{n-1}^2 \end{bmatrix}$$

Al igual que en la trilateración, se está trabajando sobre un sistema *sobre-determinado*, es decir, el número de ecuaciones es mayor que el número de incógnitas. Sin embargo, en este caso se usará un método distinto de resolución que valdrá para cualquier "n". Con este método se ahorra el trabajo de despejar las

"x" e "y" para cada "n" particular –como se implementó en el caso de la trilateración–, además que la complejidad para hacerlo manualmente sería enorme mientras "n" se incrementa.

En este nuevo caso, para desarrollar el sistema de ecuaciones se aplicará la "aproximación de los mínimos cuadrados", que sigue una ecuación matricial del tipo:

$$\hat{x} = (A^T \cdot A)^{-1} \cdot A^T \cdot b$$

En donde la matriz $\hat{x} = \begin{pmatrix} x \\ y \end{pmatrix}$ son las dos coordenadas del punto deseado.

Se puede comprobar que cuando existen tres puntos de acceso (n=3) se obtendrán los mismos valores que para la trilateración, con la diferencia que la trilateración obtendrá los resultados más rápidamente ya que los valores "x" e "y" ya están despejados manualmente.

4.5 Minimización del residuo

Si no hay forma de resolver los sistemas anteriores porque se han obtenidos resultados fuera de los límites del edificio, o bien, ha ocurrido algún error en el cálculos matemáticos (hay ocasiones en que la matriz A^TA de la aproximación por mínimos cuadrados tiene algún coeficiente igual a cero, por lo que no se podría calcular su inversa), se puede usar un método alternativo que consiste en obtener el punto *(X,Y)* que minimice la fórmula del residuo. El residuo es el error del resultado obtenido, con cualquiera de los otros algoritmos, respecto a la posición de los puntos de acceso:

$$residuo = \frac{\sum_{1}^{n} \sqrt{(x_i - x)^2 + (y_i - y)^2} - d_i}{Num.PuntosAcceso}$$

Es decir, se calcula la distancia del punto final que se ha obtenido (x, y) hacia el centro de cada circunferencia (x_i, y_i). Finalmente, se divide entre el número de puntos de acceso.

En la siguiente figura 4.8 se muestra un ejemplo en donde existen tres puntos de acceso: el punto de color rojo es el resultado obtenido por algún otro

algoritmo. La fórmula del residuo calculará las distancias e1, e2 y e3. Mientras más cerca esté el punto rojo del punto naranja menor será el residuo; de forma que si se llegase al estado en donde e1, e2 y e3 sean iguales a d1, d2 y d3, el residuo obtenido será cero.

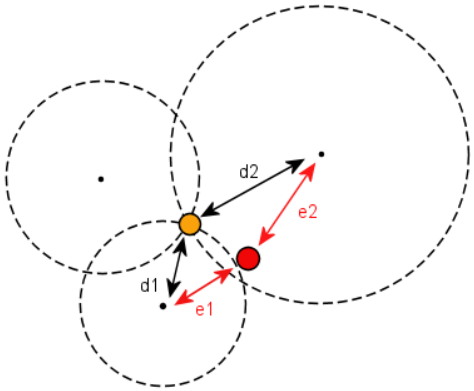


Figura 4.8: Error del resultado obtenido respecto al centro de tres circunferencias.

Para la minimización de la función del residuo se ha escogido el algoritmo de Nelder-Mead [Web-9]. El algoritmo recorrerá la función del residuo desde un punto inicial hacia la dirección de la pendiente. Para indicarle un punto inicial se ejecutará previamente un algoritmo que nos ofrezca un punto desde donde partir. Nelder-Mead aplicará ese punto en la función del residuo y comenzará a calcular el mínimo valor posible.

En el caso que la función posea varios mínimos locales, como se muestra en la figura 4.9, Nelder-Mead sólo detectará el primer mínimo que se encuentre, ya que, matemáticamente, dicho punto se encuentra en un valle y el algoritmo no puede seguir continuando por una pendiente superior.

Con este método conseguiríamos resolver gran parte de los sistemas de ecuaciones, a expensas de obtener un rango de error más grande, pero, al menos, se obtendría un punto válido dentro del mapa.

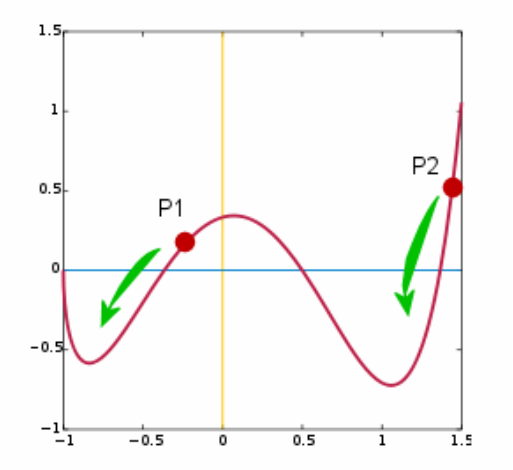


Figura 4.9: Nelder-Mead depende de los puntos iniciales para converger en uno u otro mínimo de la función.

4.6 Intersección de cuadrados (Min-Max)

Este algoritmo lo propuso Savvides en [Sav02]. En sus trabajos suele nombrar al algoritmo como "Min-Max", y así se denominará en este PFC. Se basa en calcular la intersección de cuadrados en lugar de circunferencias; con lo que la intersección no será un punto exacto sino una zona rectangular.

Para cada punto de acceso se dibujará un cuadrado en vez de una circunferencia:

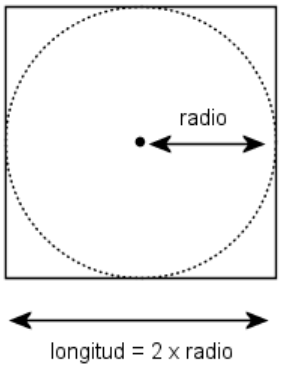


Figura 4.10: Dibujo del cuadrado centrado en cada punto de acceso.

Las dimensiones del cuadrado se hallan a partir del radio de la circunferencia, es decir, el cuadrado tendrá cada lado de longitud: dos veces el valor del radio.

A continuación, se representan sobre el mapa los cuadrados de todos los puntos de acceso y se halla su intersección:

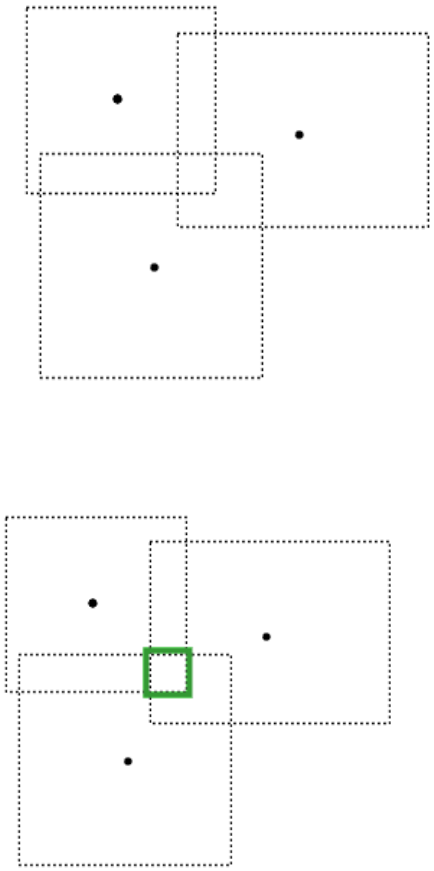


Figura 4.11: Intersección de cuadrados.

El término Min-Max hace referencia a la obtención de la zona de color verde, en donde intersectan los cuadrados. Para dicha intersección se calculará el vértice inferior izquierdo, calculando el máximo valor de los valores más pequeños de los cuadrados. Y el vértice superior derecho se obtiene como el mínimo de los valores máximos de los cuadrados, como se muestra en la figura 4.12:

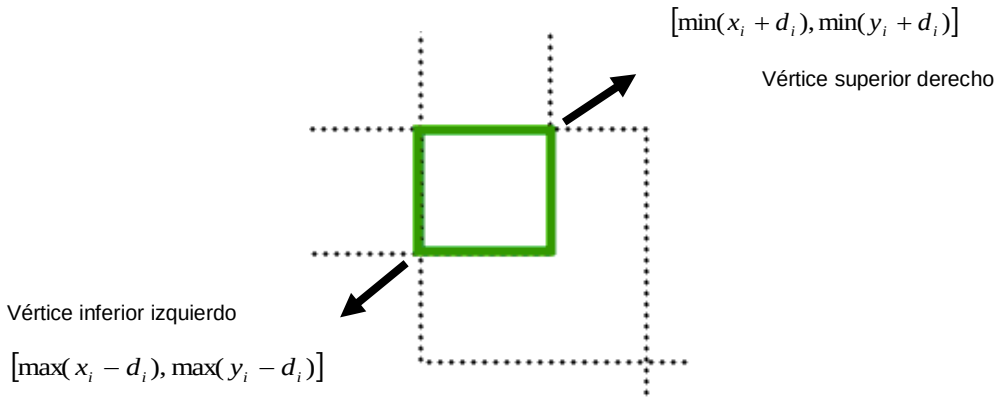


Figura 4.12: Obtención de los vértices para MinMax.

Con esos dos puntos calculados, nuestra localización la estimamos como el punto medio entre los dos vértices.

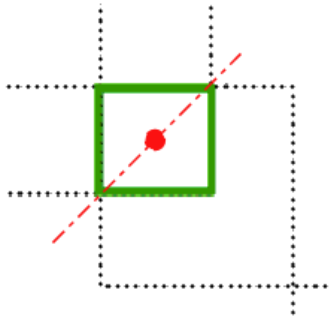


Figura 4.13: Punto medio de la intersección en el algoritmo Min-Max.

4.7 Algoritmos estadísticos

Se basan en el estudio previo de la zona, en donde se medirá la señal recibida por un punto de acceso a lo largo de todo el mapa (normalmente la medición se realizaría cada metro). Una vez recopilada dicha información se realizaría un nuevo barrido para cada punto de acceso. El algoritmo estadístico más común es el de Bayes, donde se comparan los datos obtenidos por el usuario con los que se tabularon previamente. A partir de ahí se obtendría una probabilidad de estar en una o en otra posición. Un algoritmo estadístico fue el que se usó en el primer método de localización por Wifi: 'RADAR' [Par00].

El mayor inconveniente es que el algoritmo debe leer todo el mapeado completo de principio a fin, con lo que si el sistema posee varios mapas de distintas zonas supondría un tiempo de cálculo mayor.

Para reducir este tiempo de cálculo se podría usar, previamente, un método determinístico para así reducir el rango de actuación del algoritmo estadístico. Otra ventaja del algoritmo estadístico es que siempre ofrecería una localización dentro de los límites del edificio.

Aunque este algoritmo no se ha implementado para este PFC, se ha comentado, brevemente, como información adicional al lector.

CAPÍTULO 5

ENTORNO DE DESARROLLO

5.1 Introducción

Tras estudiar detenidamente la tecnología Wifi, el comportamiento de sus señales en interiores y de los diferentes algoritmos de localización, en los próximos apartados se describirán las funciones y módulos elementales de las aplicaciones de este PFC.

En este capítulo se describirá el entorno de trabajo usado para su desarrollo, que se basa en tres grandes bloques: .NET Compact Framework 3.5, MFC 9.0 y Visual Studio 2008.

Posteriormente, se explicará la comunicación con la tarjeta de red inalámbrica del dispositivo a través del driver NDIS, provisto por Microsoft.

5.2 Tecnologías de la aplicación

El objetivo de este PFC es la creación de una aplicación para el sistema operativo Windows Mobile, así que el lenguaje de programación elegido es uno del tipo "Visual" que facilita la creación del software en sistemas Windows. Entre los lenguajes de tipo "Visual" se ha escogido C++ que ofrece un lenguaje orientado a objetos. Con esto se facilita la comunicación a bajo nivel entre los componentes hardware, a la vez que se permite la creación de clases independientes, con sus métodos y atributos privados, y se heredan propiedades que facilitan la creación de dichas clases.

El entorno de trabajo necesario para el desarrollo de una aplicación VisualC++ para Windows Mobile ha sido Visual Studio 2008, haciendo uso de las librerías MFC 9.0 y .NET Compact Framework 3.5, que se explicarán a continuación.

5.3 .NET Framework

Con .NET Compact los desarrolladores pueden usar cualquier lenguaje de programación .NET (C#, C++, Basic) de forma segura, a la vez que son capaces de usar las características innatas de los dispositivos móviles.

Antes de describir .NET Compact se comentará, previamente, la versión completa de .NET. Hay que tener en cuenta que la versión reducida de .NET Compact no es una versión a la que se hayan eliminado componentes, sino que ha sido desarrollada independientemente con el objetivo primordial de minimizar los recursos consumidos por las aplicaciones.

5.3.1 Common Language Runtime

En el corazón de .NET se encuentra el CLR (tiempo de ejecución en lenguaje común - *common language runtime*), que es el que permite a la aplicación ejecutarse en un entorno seguro y aislado.

El CLR es una máquina virtual similar a la JVM de Java, que provee servicios como:

- Priorización y planificación de threads.
- Gestión de la memoria (incluyendo el recolector de basura)
- Integración de lenguaje cruzado.
- Manejo de excepciones.
- Seguridad mejorada.
- Interacción y paso de mensajes entre procesos.
- Depuración y profiling.

Todo código en lenguaje de .NET no se ejecutará directamente en el sistema operativo, sino que será ejecutada por este CLR como intermediario.

El CLR contiene varios compiladores que compilan el código hacia un código intermedio llamado MSIL (Microsoft Intermediate Language). MSIL define las instrucciones pertinentes según la plataforma de CPU específica, es decir, desde aquí se crea el código en ensamblador, que puede ser tanto una librería .DLL, como un ejecutable '.EXE'.

5.3.2 Dominio de la aplicación (Application Domain)

El CLR ejecuta las aplicaciones .NET en un dominio particular controlado por el propio CLR. El CLR controlará la seguridad del dominio comprobando las relaciones entre *clases* y no dejando que las aplicaciones accedan a otros dominios a los que no pertenecen (para que unas aplicaciones se comuniquen con otras sólo se puede hacer a través de conexiones de red).

Cada lenguaje compatible con .NET requiere de un compilador para producir código MSIL. Microsoft distribuye estos compiladores para los lenguajes C#, Visual Basic, J#, JScript y C++.

5.3.3 Librería de clases de .NET

.NET Framework contiene más de dos mil clases y 'tipos' organizados en *namespaces* ('espacios de nombres'). Estas clases permiten a los desarrolladores realizar muchas operaciones con el mínimo esfuerzo. Los espacios de nombres más utilizados son:

- El espacio SYSTEM, que contiene las clases básicas que implementan los strings, arrays, colecciones, funciones matemáticas, tipos de datos, conversiones de datos, manejadores de eventos ('*event handlers*'), etc.
- El espacio SYSTEM.NET que ofrece clases de acceso y conexión de redes remotas.
- Clases de seguridad en SYSTEM.SECURITY, como criptografía, permisos de usuario, políticas de acceso, etc.
- Clases para el entorno gráfico con SYSTEM.WINDOWS.FORMS
- Clases de E/S en SYSTEM.IO para trabajar con comunicaciones sincronas/asíncronas.
- Clases de multi-hilo en SYSTEM.THREADING. Threading para realizar programación multi-hilo, sincronización de procesos, monitorización, etc.

5.3.4 .NET Compact Framework

.NET Compact Framework es el entorno de .NET para dispositivos móviles. Aunque parezca que es una versión reducida extraída tal cual de la versión completa, no es así. Se ha creado y modificado en una instancia aparte de la versión completa, pensando, sobre todo, en el ahorro de consumo de recursos – tanto de memoria RAM, como de CPU– para un menor gasto de energía. Además, se han añadido clases nuevas que no existen en la versión completa (por ejemplo, la pantalla de un teléfono móvil es distinta que la de un PC de sobremesa; o, a la hora de introducir datos, se puede hacer por pantalla táctil, y no sólo por teclado).

Realizando una comparativa entre .NET y el .NET Compact obtenemos las siguientes características que no están disponibles en .NET Compact:

- Controles de impresión.
- Compatibilidad con bases de datos.
- GDI+ (tratamiento avanzado de gráficos)

- Registro de Windows
- Aplicación Web (no soporta cookies)
- Cifrado limitado.

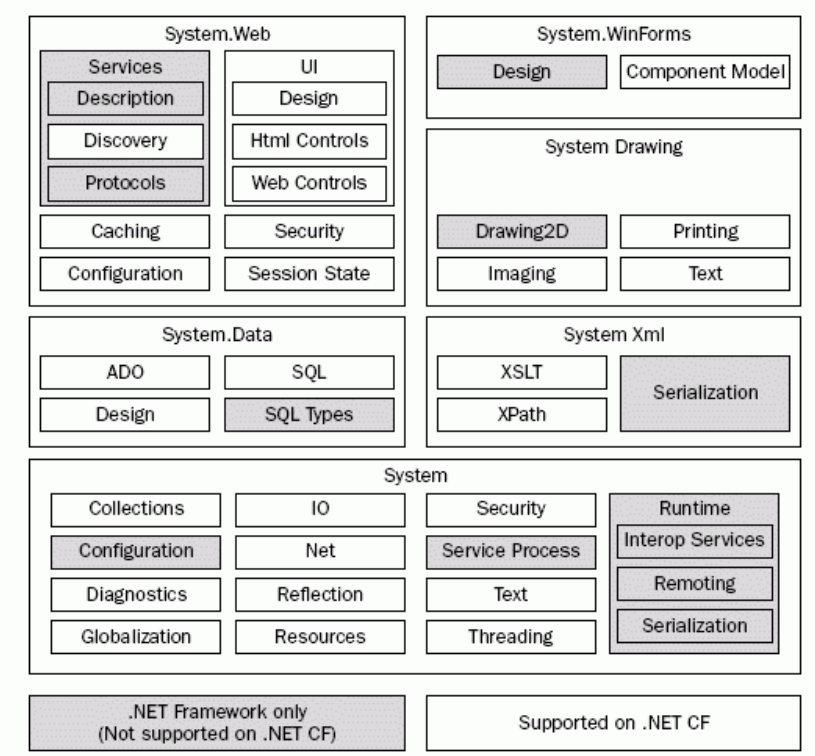


Figura 5.1: Los bloques sombreados no están incluídos en .NET Compact.

5.4 MFC 9.0

MFC es una librería de Microsoft que encapsula partes de la API de Windows en clases de C++. Las clases engloban a casi todos los manejadores de objetos que utiliza Windows internamente.

En el desarrollo del software de este proyecto se ha empleado la versión 9.0 de MFC, que está presente en Visual Studio 2008.

Al usar la librería MFC se estarán utilizando objetos en la aplicación que derivarán de alguna de las clases que muestra la siguiente figura 5.2:

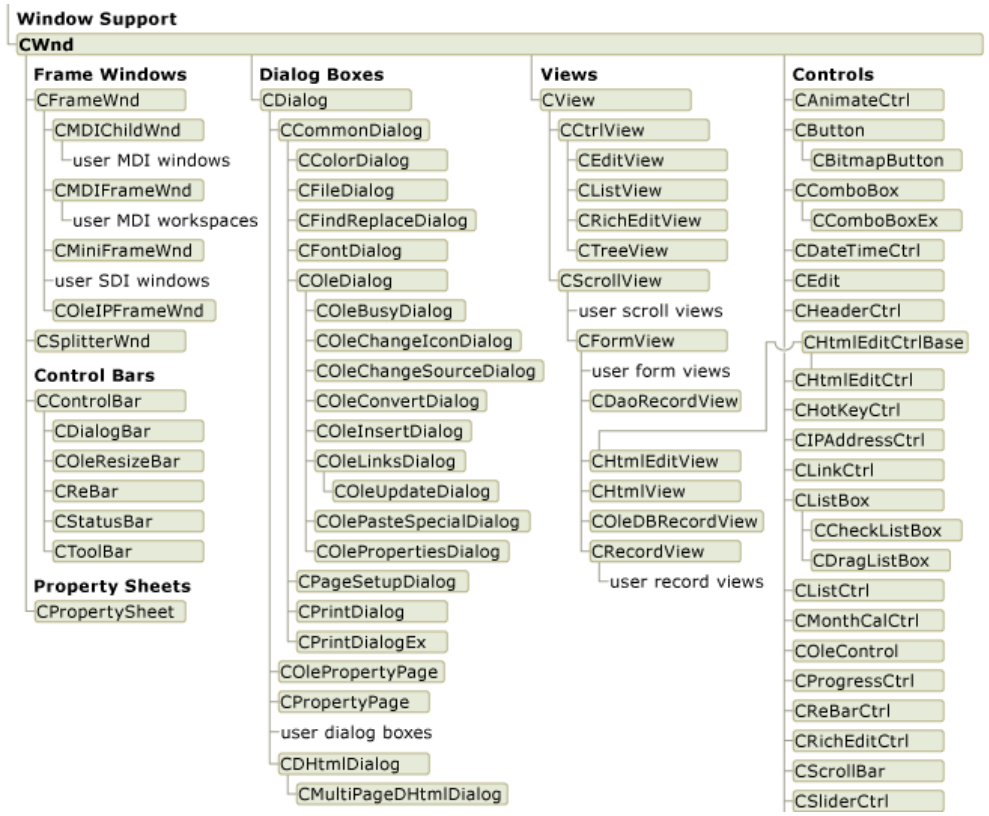


Figura 5.2: Pequeño extracto del diagrama de clases de MFC versión 9.0

La librería MFC está escrita para C++ y permite el desarrollo de aplicaciones en Microsoft Windows de forma más cómoda gracias a la gestión de ventanas, de cuadros de diálogo, al manejo de mensajes, la realización de operaciones básicas de entrada/salida, etc.

Con MFC se puede incluir código rápidamente gracias a la arquitectura de desarrollo *Documento/Vista*. Esto es, lo que el usuario puede ver será la "vista", un objeto perteneciente a la clase *CView*; y el "documento" serán los datos de usuario y/o configuración del propio programa, es decir, un objeto derivado de *CDocument*. Por último, estará el marco donde se ejecuta la aplicación, que es derivada de *CFrameWnd*. Las distintas clases se comunican entre sí mediante paso de mensajes.

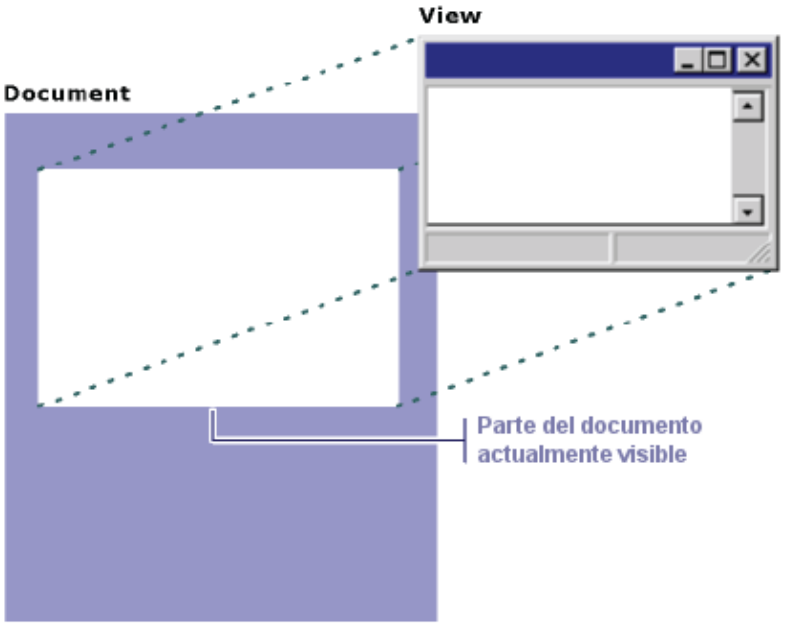


Figura 5.3: Diagrama que muestra la relación entre las clases CDocument y CView.

La mayor ventaja de la arquitectura *Documento/Vista* es la mayor facilidad para diseñar interfaces gráficas de usuario, pero otro uso muy extendido es el poder tener dos instancias de una misma aplicación con varios documentos abiertos a la vez, y cada uno de ellos puede incluir una o más vistas como, por ejemplo, abrir varios documentos .doc en MS-Word, o abrir varias fotografías en Adobe Photoshop. En la aplicación de este PFC no se hace uso de múltiples documentos ni múltiples vistas, sino de pestañas. Este modo se denomina "basado en diálogo" (*dialog based*) que se elige en el momento de crear el proyecto en Visual Studio.

5.5 Visual Studio 2008

Visual Studio es un entorno de desarrollo multi-lenguaje para aplicaciones Windows. Puede manejar tanto código 'administrado' como 'no-administrado'. El código 'administrado' se ejecuta en la capa superior de .NET Framework (por ejemplo, C++), y el codigo 'no-administrado', también conocido como 'código nativo', es el que se ejecuta en el sistema operativo a bajo nivel.

Si se usan las clases de .NET Framework se podrá trabajar en múltiples lenguajes distintos utilizando la misma convención sintáctica y de llamadas a métodos.

Referente a los dispositivos móviles, desde Visual Studio 2005 se ha mejorado su desarrollo de aplicaciones, ya que antes de Visual Studio 2005 los programadores tenían que usar Visual eMbedded C++ para crear código 'no-administrado' específico para dispositivos con Windows Mobile. Para el caso de este PFC, con VS2008, sólo se ha tenido que instalar el entorno de desarrollo específico para dispositivos móviles que se llama 'SDK WindowsMobile'.

La versión 2008 está enfocada a aplicaciones de Windows Vista, Office 2007 y aplicaciones Web. Como novedad visual se tiene un nuevo entorno de Windows Presentation Foundation (*WPF*) y un nuevo editor de HTML/CSS.

VS2008 necesita de .NET Framework 3.5, aunque a la hora de compilar el proyecto se puede decidir la versión que el desarrollador desee: desde la 2.0 a la 3.5, pasando por la versión Compact.

Como mejoras para desarrollos en Visual C++ se cuenta con la versión 9.0 de las librerías MFC (Microsoft Foundation Classes), que aportan nuevos estilos visuales para las ventanas y entornos gráficos, más acordes a los de Windows Vista.

También aprovecha los múltiples núcleos de la CPU (dual-core, quad-core, octo-core…) con lo que, además de compilar los proyectos más rápidamente haciendo uso de todos los núcleos, también brinda la posibilidad de depurar nuestro código y predecir cómo se ejecutarán los hilos (o threads) en una máquina con múltiples CPU's.

En el momento de escribir esta memoria ya se ha hecho pública la versión de Visual Studio 2010. En esta nueva versión se ha modificado la organización de las ventanas flotantes y opciones a las que más recurren los programadores. También ofrece capacidad de multi-monitor, para ver distintos códigos en distintas pantallas. Se ha modificado el código interno para aceptar más fácilmente plugins de desarrollo para nuevos lenguajes. Y como novedad más curiosa se encuentra el soporte de programación multi-paradigma, que ofrece al programador libertad para implementar sus ideas de la forma más optimizada posible: no hay un paradigma mejor o peor, sino que cada uno tiene sus ventajas respecto a los demás.

VS2010 vendrá con .NET Framework 4.0 y enfocado para aplicaciones de Windows 7 [Ran10].

5.6 Programación del adaptador Wifi: NDIS

Para la comunicación con los adaptadores de red en sistemas Windows (tanto si son PC's de escritorio como dispositivos móviles) se utiliza un driver provisto por Microsoft que se llama NDIS (Network Driver Interface Specification). NDIS es un API para facilitar la comunicación entre las aplicaciones de usuario y las tarjetas de red. Actualmente, también es posible usar NDIS en sistemas basados en Linux [Web-10].

La principal ventaja es que conseguimos estandarizar la comunicación de todas las tarjetas de red de cualquier fabricante que deseen que su tarjeta la usen en sistemas Windows.

Un inconveniente a la hora de trabajar con un interfaz universal es la disminución del rendimiento comparado a si se usaran órdenes directas al driver del fabricante. Se da el caso, por ejemplo, de algunos modelos de tarjetas que rechazan algunas instrucciones para no perder la conexión con el punto de acceso actual: en el caso de que la aplicación software quiera escanear la zona en busca de nuevas redes la tarjeta tendrá que sintonizar en toda la banda de frecuencia de la antena, y esto puede implicar la pérdida de conexión con el punto de acceso al que estemos vinculados en ese momento.

La versión más extendida de NDIS (la versión 5.1) tiene una restricción importante para el desarrollador de software en sistemas Wifi: las tramas de datos aparecen encapsuladas en tramas Ethernet tradicionales, en lugar de indicarlas como tramas del protocolo 802.11, con lo que no se podrá acceder a las tramas de control y de gestión del 802.11.

La interfaz NDIS se podría considerar parte del nivel de enlace y del nivel de red del sistema OSI (niveles 2 y 3).

5.6.1 Tipos de drivers en NDIS

Los miniport drivers son los drivers a más bajo nivel, programados a nivel del kernel del sistema operativo. Principalmente, manejan el trasvase de las tramas 802.11 desde, y hacia, el dispositivo; gestiona su encendido y su apagado, la sintonización de un canal determinado o la selección de un adaptador Wifi (en el caso de que hayan varios instalados en el sistema).

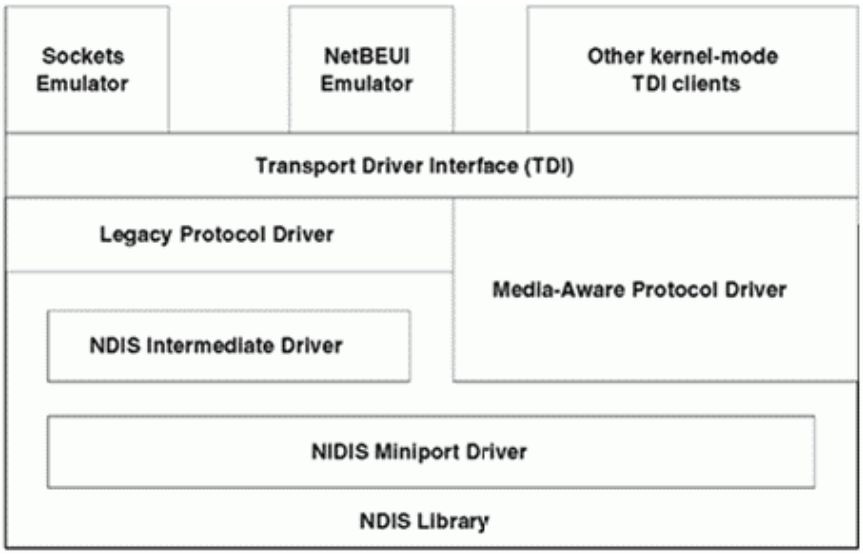


Figura 5.4: Clasificación y jerarquía de drivers NDIS.

En un nivel superior se encuentran los drivers intermedios (*Intermediate drivers*) que ejecutan acciones tanto del nivel de red como de transporte. Estos drivers no acceden directamente al medio físico, sino a un minipuerto virtual. Este minipuerto virtual permitiría implementar un driver que maneje dos conexiones físicas (en nuestro caso, dos tarjetas Wifi) para enviar la información a ambos adaptadores dando mayor velocidad de conexión usuario.

Los drivers de protocolo (*protocol driver*) hacen de intermediarios entre el nivel de transporte y los drivers intermedios. Las aplicaciones de los niveles superiores harán uso de él en algún momento determinado para realizar alguna consulta a bajo nivel. Suelen ser comandos del estilo: "¿Qué redes detectas?" o "¿Qué nivel de potencia recibes del punto de acceso actual?" o "Intenta conectarte a la red X ". Así, se podría implementar un driver que funcione tanto para 802.11 como para Bluetooth, con estos comandos similares en ambas tecnologías.

Por último, nos encontramos con los drivers de transporte. Estos sí implementan las acciones del nivel de transporte como tal (por ejemplo, TCP o UDP).

En nuestra aplicación trabajaremos con el driver de protocolo que nos permitirá saber los dos datos fundamentales para nuestro sistema de localización: la potencia de señal recibida de cada estación base y su dirección MAC.

5.6.2 Operaciones del driver de protocolo

Las OID's (*Object Identifiers*) son las operaciones de entrada-salida (E/S) que desea ejecutar el programador sobre el driver de protocolo.

Son comandos con un parámetro de entrada y otro de salida. Dependiendo de la OID, tanto pueden ejecutarse instantáneamente (de forma síncrona) o de forma retardada (asíncronamente).

Las operaciones que se usarán en este PFC para comunicarse con la tarjeta de red se detallan a continuación, junto con un diagrama de los pasos necesarios:

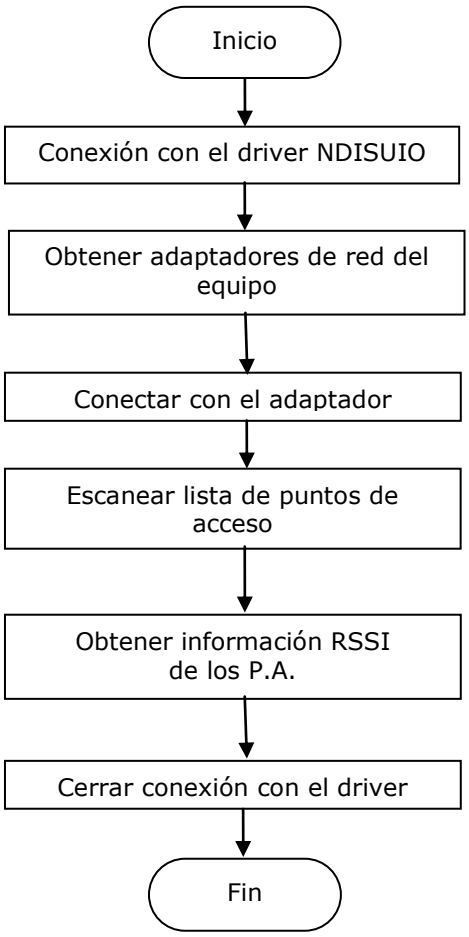


Figura 5.5: Diagrama que muestra el paso secuencial de mensajes para comunicarse con la tarjeta de red.

`IOCTL_NDISPROT_OPEN_DEVICE`

Se debe ejecutar esta OID en primer lugar para indicar al driver de protocolo que queremos iniciar una conexión con alguna tarjeta de red.

IOCTL_NDIS_GET_ADAPTER_NAMES

Esta OID se usa para conocer qué dispostivos de red tenemos en nuestra PDA/dipositivo móvil. Aquí se incluyen tarjetas Ethernet, Wifi, Bluetooth, GPRS, etc.

OID_GEN_PHYSICAL_MEDIUM

Tras consultar la IOCTL anterior, se necesita conocer qué tipo de adaptador posee el dispositivo. Esta OID sólo está implementada en los dispositivos inalámbricos (como son los infrarrojos, GPRS,...), por tanto, en el caso de no tener ningún interfaz inalámbrica se devolverá "OID no soportada" y no seguiremos ejecutando nuestra aplicación.

OID_BSSID_LIST_SCAN

Esta llamada no devuelve ningún dato. Es específica de las tarjetas Wi-fi y su función es realizar un escaneo para actualizar las redes detectadas a nuestro alrededor y guardar la información en la memoria del propio adaptador.

OID_BSSID_LIST

Esta llamada sí devuelve una lista de los puntos de acceso (BSSID) que hemos detectado con el último SCAN. Además, para cada estación se nos muestran datos particulares de cada una de ellas, como son: la dirección MAC, el nombre de la estación, la potencia de señal recibida y si usa autenticación segura o no.

OID_802_11_DISASSOCIATE

Con este comando se finaliza la conexión con el punto de acceso Wifi.

5.6.3 Drivers NDIS 6.0

Tras las restricciones para el programador en las versiones anteriores de NDIS, como no acceder directamente a las tramas 802.11, Microsoft desarrolla una nueva versión 6.0 que es la que está presente en Windows Vista y versiones posteriores [Web-11]. Las características mejoradas respecto NDIS 5.1 son:

- Mayor escalabilidad porque soporta multi-threading, por lo que más de una CPU podrá gestionar las operaciones de E/S.

- Mayor detalle en los mensajes de error provenientes del adaptador de red.

- Aparecen unos nuevos drivers intermedios llamados "*lightweight filters*" con más opciones y mejor rendimiento.

- Soporte nativo de dispositivos inalámbricos, con lo que ya no convertirá las tramas 802.11 a tramas Ethernet y podremos acceder directamente a ellas.

5.7 Aplicación de apoyo 'PeekPocket'

Uno de los bloques primordiales en el diseño de la aplicación es la comunicación con el adaptador Wifi del dispositivo móvil para poder obtener los distintos datos necesarios de cada punto Wifi. Tras estudiar los distintos programas que existen en el mercado que se comunican con las tarjetas Wifi en entornos WindowsMobile sólo se encontró uno que fuera de código abierto: PeekPocket [Web-13]. El resto de aplicaciones del mercado eran desarrolladas por los propios fabricantes del dispositivo móvil, con lo que eran de código cerrado y, además, usaban sus propios drivers específicos, por lo que si se usaban dichas aplicaciones en otros modelos distintos no se producía comunicación alguna con la tarjeta de red. La aplicación PeekPocket hace uso del driver de comunicaciones NDIS provisto por el propio sistema WindowsMobile, que permite conectar con cualquier tarjeta de red que sea compatible con WindowsMobile.

El objetivo de PeekPocket es mostrar en pantalla una lista de los puntos Wifi detectados en cada momento con los siguientes datos de cada uno: nombre de la estación, dirección MAC, potencia recibida, canal en el que emiten, si transmiten con encriptación o son de libre acceso y, finalmente, si la señal es de un punto de acceso o 'peer'. En la siguiente figura 5.6 se muestra una captura de la lista de puntos de acceso que ofrece PeekPocket:

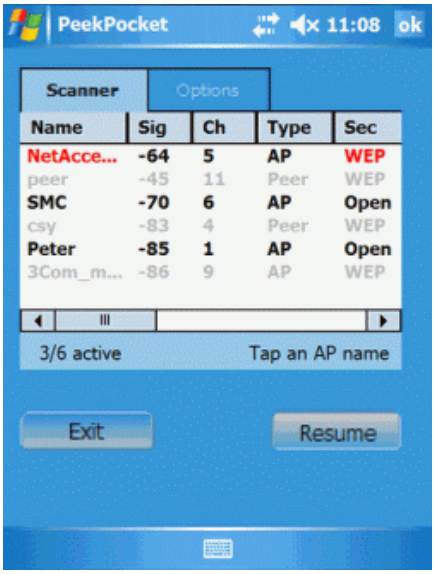


Figura 5.6: Pestaña "Scanner" de 'PeekPocket'.

En una segunda pestaña se ofrecen diferentes opciones para modificar la visualización de la lista de los puntos de acceso:

Figura 5.7: Pestaña "Opciones".

- 'Use adapter' ofrece la posibilidad de elegir una tarjeta Wifi en el caso que el dispositivo poseyera más de una.
- 'Scan speed' permite cambiar la velocidad de refresco a la que se leen los datos de cada punto de acceso. Varía desde 500ms hasta 2 segundos.

- 'List font size' permite aumentar o disminuir el tamaño de letra de la pestaña 'Scanner'.
- 'Hide secure' muestra exclusivamente los puntos de acceso libres.
- 'Hide peers' muestra exclusivamente los puntos de acceso sin tener en cuenta los usuarios móviles (peers).
- 'Sound off/on' deshabilita el sonido de la aplicación.
- 'Pause/Resume' detiene el escaneo de puntos de acceso manteniendo en pantalla los últimos datos recibidos.
- 'Exit' sale de la aplicación.

A continuación se presenta el diagrama de clases de PeekPocket:

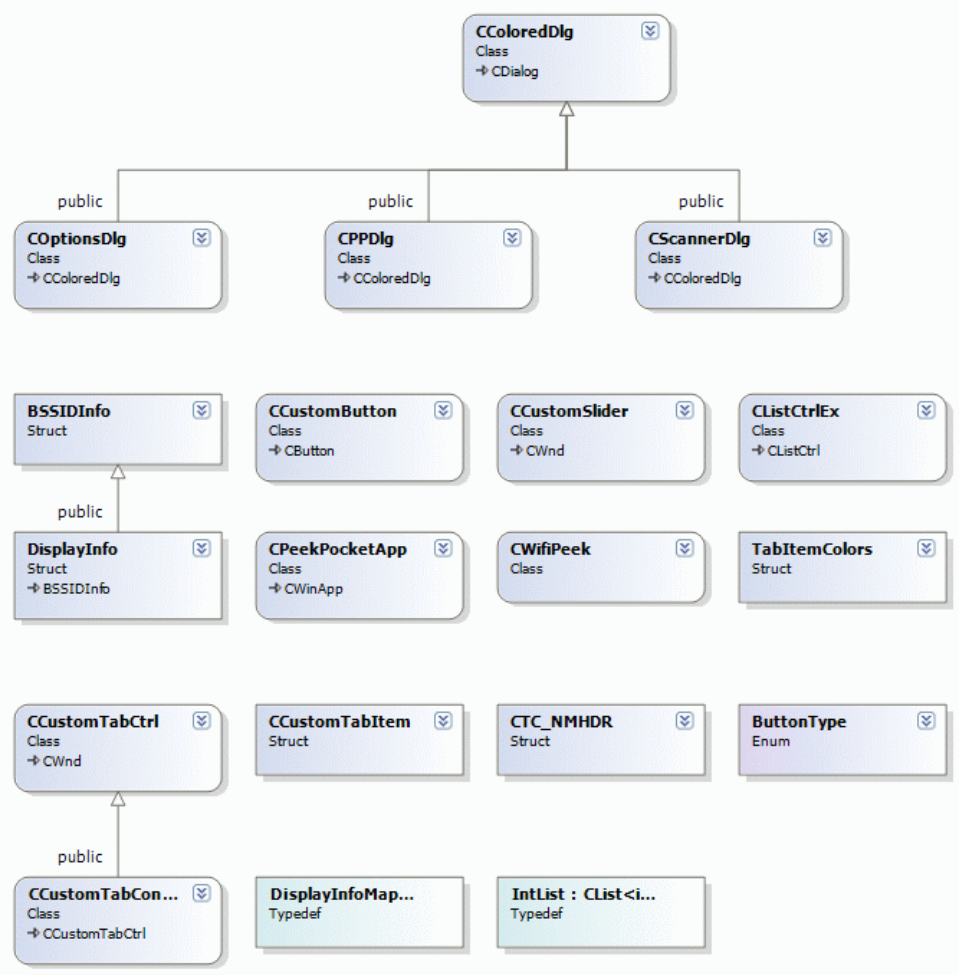


Figura 5.8: Diagrama de clases de PeekPocket.

5.7.1 Clase CWifiPeek

Una parte importante de la aplicación *PeekPocket* se encuentra en la clase *'CWifiPeek'*, que controla todas las comunicaciones con la tarjeta Wifi. Esta clase puede usarse en otras aplicaciones C++; incluso aquellas que no sean MFC porque esta clase no hace uso alguno de dichas librerías. Para añadir esta clase a cualquier otro proyecto simplemente habrá que añadir *CWifiPeek.h* y *CWifiPeek.cpp* al proyecto en particular y deshabilitar el uso de *"precompiled headers"*.

'CWifiPeek' hace uso del driver NDIS para el escaneo de los puntos Wifi como se ha explicado en el apartado 5.6. Siguiendo el mismo esquema 5.5 sobre el diagrama de flujo que hay que llevar a cabo para comunicarse con la tarjeta de red, *'CWifiPeek'* contiene las siguientes funciones:

- `bool` **`AbrirDriver()`** Esta función comunica al sistema operativo que se desea iniciar una comunicación con la tarjeta de red del dispositivo. No neecesita de ningún argumento de entrada, sólo pide un manejador (*handle)* al driver NDIS que se utilizará para identificar a la interfaz de red cuando la aplicación necesite comunicarse con él. Si NDIS devuelve un manejador válido la función devuelve TRUE y el manejador se almacena en una variable global de tipo *`handle`*. Esta función hace uso de la OID `'IOCTL_NDISPROT_OPEN_DEVICE'`.

- `bool` **`ListaAdaptadores`** `(LPWSTR pDest, DWORD &dwBufSizeBytes)`. Esta función pregunta por el número de adaptadores de red (Wifi y no Wifi) de nuestro dispositivo. Para llamar a la función hay que pasarle un array de WCHAR y su longitud. La función devolverá el nombre de cada adaptador en ese array separados por coma. Automáticamente filtrará los adaptadores "infrared", "GPRS" y "ActiveSync". La función hace uso del OID *IOCTL_NDIS_GET_ADAPTER_NAMES*. Hay que conocer los nombres de los adaptadores porque así lo requiere posteriormente las llamadas a NDIS.

- `bool` **`ActualizaBSSIDs`**`(LPWSTR pAdapter)`. Esta función realiza el escaneo de las redes que se encuetran a nuestro alrededor. Sólo hay que indicarle el nombre del adaptador como parámetro de entrada. Si no hubo errores en el escaneo, devolverá TRUE, con lo que significará que los nuevos datos están guardados en la memoria del propio adaptador

Wifi. A esta función se recurre muy a menudo para refrescar los datos de cada punto de acceso.

- `bool` **`TraspasaBSSIDs`**`(LPWSTR pAdapter, struct BSSIDInfo *pDest, DWORD &dwBufSizeBytes, DWORD &dwReturnedItems).` Esta función se utiliza para traspasar a memoria RAM las estaciones que previamente se han guardado en la memoria interna del adaptador. La función devuelve un array de estructuras y el número de estructuras recibidas. El formato de la estructura recibida es el siguiente:

```
struct BSSIDInfo
{
    BYTE BSSID[6];            //dirección MAC
    WCHAR SSID[32];           //nombre de la estación
    int RSSI;                 //potencia de señal
    int Canal;                //Canal en el que emite.
    int Infraestructura;      //Punto de acceso o "peer"
    int Autenticacion;        //Red abierta o segura.
};
```

- `bool` **`CerrarDriver().`** Esta función solicita el fin de la comunicación con el chip de red. La variable *handle* que se recibió en *`AbrirDriver()`* quedará liberada para otras operaciones. Si se desea volver a comunicar con el chip de red se debe realizar una nueva petición de *AbrirDriver()* y se obtendrá, posiblemente, un *handle* diferente al anterior en el caso que otros procesos hayan intentado iniciar nuevas comunicaciones. Esta función hace uso de la OID `'OID_802_11_DISASSOCIATE'.`

La estructura *DisplayInfo* gestiona los datos de los puntos Wifi que se están detectando y que se muestran en pantalla en la pestaña SCANNER. Hereda los atributos de la estructura *BSSIDInfo, y* añade, además, estos que se describen a continuación:

- *bActivo*: indica si el punto de acceso se ha detectado anteriormente o es nuevo. Este valor indicará a la aplicación el uso del color negro o gris con el que aparecerá el punto Wifi en pantalla, tal como se muestra en el ejemplo de la figura 5.6. En ella aparecen seis puntos de acceso que se han detectado desde el inicio de la aplicación, pero sólo tres se encuentran disponibles en la última actualización. El color rojo del punto de acceso viene marcado por la variable *Autenticacion* de la estructura

BSSIDInfo, que indica si el punto de acceso ofrece una comunicación abierta o encriptada.

- *dPrimeraVista*: indica la hora en la que se detectó el punto Wifi por primera vez.
- *dUltimaVista:* indica la hora en la que se detectó por última vez el punto Wifi.
- *ID*: indica la posición en el array del punto Wifi para ayudar a la aplicación a localizarlo.
- *ilSenalHistorial:* guarda el historial de potencias recibidas por el punto Wifi a lo largo del tiempo.
- *iPeorPot:* indica el peor nivel de señal que se ha obtenido del punto Wifi. Este factor indica si el nivel de señal es adecuado para la comunicación o no.
- *iMejorPot:* indica el nivel de señal más alto que se ha detectado del punto Wifi en cuestión.

5.7.2 Clase CColoredDlg

En un segundo plano se encuentra la clase *CColoredDlg* que gobierna la apariencia de la aplicación. Esta clase sustituye a la clase inicial *CDialog* que es la clase base en todo proyecto de Visual C++. *CColoredDlg* sustituye a la anterior porque ofrece la posibilidad de usar colores personalizados a cada control de Windows: cada vez que se muestra un cuadro de diálogo en pantalla *CColoredDlg* controlará el proceso.

Se puede usar esta clase en cualquier otro proyecto sin mayores problemas: se añaden los dos ficheros *ColoredDlg.h* y *.cpp* al proyecto en particular. Y en los ficheros .h y .cpp del nuevo proyecto hay que sustituir cada *"CDialog"* por *"CColoredDlg"*.

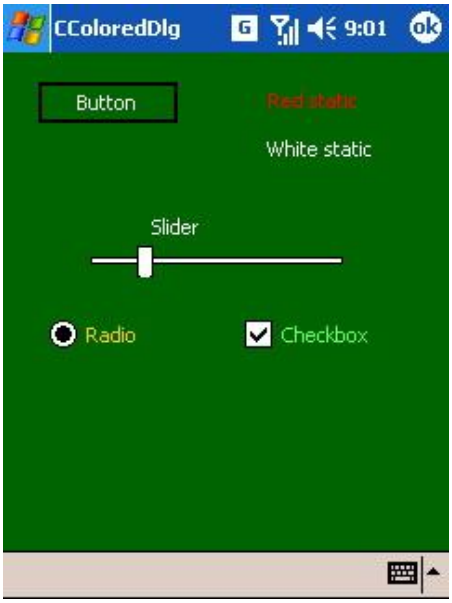


Figura 5.9: Ejemplo que muestra varias funciones de coloreado simultáneas con "CColoredDlg".

CAPÍTULO 6

ANÁLISIS PREVIO Y FUNCIONAL

6.1 Definición del problema

El objetivo principal de este Proyecto Fin de Carrera es la realización de un sistema de localización en interiores para dispositivos móviles que posean un interfaz de red Wifi. Deberá ser capaz de indicar la posición del usuario utilizando para ello los niveles de potencia de la señal recibida de los puntos de acceso a la red Wifi que existan a su alrededor. Por tanto se necesitará de un algoritmo de localización que realice los cálculos matemáticos necesarios para ofrecer la posición. También deberá realizarse un análisis y comparativa de diferentes algoritmos para conocer el rendimiento de cada uno de ellos, basándose en criterios como el consumo de CPU, la precisión y la robustez frente a errores en la lectura de las señales Wifi.

6.2 Estudios previos necesarios

Para resolver el problema es necesario conocer las condiciones de propagación de las ondas electromagnéticas, y en particular, su propagación en el interior de edificios (ver capítulo 3). Como conclusión de dicho estudio, diremos que se necesita el factor de atenuación que describe la variación de potencia de las señales a lo largo de sus trayectorias, influyendo en este parámetro tanto las condiciones de emisión como las de recepción de la señal.

Con dicho factor junto a la potencia de señal recibida en un instante y la potencia que se recibe a un metro de distancia del punto de acceso Wifi se estimará la distancia a la que se encuentra el usuario del punto de acceso.

Es necesario estudiar también los diferentes algoritmos de localización, que pueden ser de aplicación, y su posterior implementación en un lenguaje de programación, en el caso de este PFC será C++. Estos algoritmos de localización se describen en el capítulo 4.

Para obtener la potencia de señal recibida habrá que comunicarse con el interfaz de red del dispositivo móvil. En el caso que nos ocupa, se trabaja sobre una aplicación ejecutada sobre sistemas operativo Windows Mobile, siendo necesario el acceso a los drivers NDIS mediante el uso del lenguaje de programación VisualC++, tal y como se comenta en el capítulo 5.

Finalmente, con estos estudios previos se ha obtenido una metodología de aplicación para solucionar el problema planteado. Dado que es necesario aplicar esta método en diferentes etapas, describimos en el siguiente apartado cada una de ellas.

6.3 Solución del problema

El sistema de localización necesita conocer una serie de datos que se deben obtener en un estudio previo del entorno. Por tanto, se puede decir que el sistema de localización posee dos fases de implementación: fase 'offline' y fase 'online'.

En la fase 'offline' se deben obtener los datos referidos a los puntos de acceso existentes en la zona y al comportamiento de las señales electromagnéticas. Los datos necesarios de cada punto de acceso son: su localización, el factor de atenuación de las señales y la potencia recibida a un metro de distancia de cada punto Wifi.

Para poner en funcionamiento el sistema de localización hay que estudiar la distribución y dimensiones del entorno donde se vaya a implementar. Una vez obtenido un mapa del entorno donde se aplicará el sistema de localización se dividirá dicho mapa en coordenadas de posición de un metro por un metro (aunque se deja a elección del usuario dichas dimensiones). Posteriormente, se definirá un eje de coordenadas (x,y) para orientar y localizar tanto los puntos Wifi existentes como la localización del usuario.

Mapeado de potencias

La toma de datos se realizará en las zonas de visión directa con los puntos Wifi. Normalmente será en los pasillos donde se instalen dichos puntos de acceso, ya que es allí donde las señales electromagnéticas alcanzan la mayor distancia por estar libre de obstáculos.

En cada coordenada se realizará una captura de potencias de cada punto Wifi. En una primera toma de datos se comprobó que la potencia variaba según la orientación del usuario (ver figura 6.1):

Figura 6.1: Muestras de potencia recibidas en una coordenada para distintas orientaciones.

Por tanto se decidió realizar una captura de datos por cada orientación (norte, sur, este y oeste) como aparece en la figura 6.2. Con todas las potencias obtenidas, se aplica la media aritmética y se obtiene el resultado final para cada coordenada.

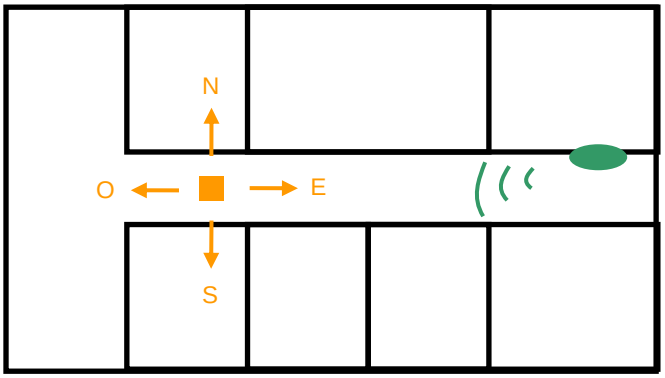


Figura 6.2: Toma de datos de los puntos Wifi desde los cuatro puntos cardinales.

Una vez recopilada toda la información se debe obtener la potencia que se recibe a un metro para poder calcular, finalmente, el factor de atenuación (factor 'n') a través de las potencias que se han medido por toda la zona de visión directa.

Cálculo de los índices de atenuación

Una vez obtenidas las potencias de cada punto Wifi en cada coordenada habrá que relacionarlas con la distancia real a la que se encuentra el usuario. Para ello se usará la ecuación del modelo de propagación de señales con visión directa que se vio en el apartado 3.4.1:

$$P_{TX} - P_{RX}\ (dB) = P_{fs} + 10 \cdot n \cdot log_{10}(d)$$

Donde 'P_{fs}' es la potencia recibida a un metro; 'd' es la distancia a la que se encuentra el usuario respecto el punto de acceso; 'P_{TX}' es la potencia transmitida por el punto de acceso; y 'P_{RX}' es la potencia recibida en ese momento. Por tanto, se obtendrá un factor "n" (factor de pérdidas) para cada coordenada estudiada:

$$n = \frac{P_{TX} - P_{RX} - P_{fs}}{10 \cdot log(d)}$$

Con estos datos de factor 'n' , 'P_{fs}' y las coordenadas de localización de cada punto Wifi ya se puede hacer uso del sistema de localización. Dicha aplicación medirá la potencia que recibe de todos los puntos de acceso y seleccionará exclusivamente los puntos Wifi que concuerden con su base de datos, es decir, los puntos Wifi estudiados en la fase 'offline'. Con dichas potencias recibidas, con el factor 'n' y con la potencia recibida a un metro, calculará la distancia hacia cada punto Wifi conocido.

Análisis de los algoritmos de localización

Una vez conocidas las distancias hacia cada punto de acceso se ejecutará un algoritmo de localización elegido por el usuario de una lista que se presenta en pantalla. La aplicación calculará la posición del usuario y la mostrará sobre el mapa del entorno que se ha estudiado.

En esta última fase se evaluarán los diferentes algoritmos implementados para conocer cuál ofrecerá mejor rendimiento según estos criterios: precisión, robustez frente a variaciones de señal y consumo de CPU. Dicho consumo de CPU está ligado al consumo de batería del dispositivo móvil, un factor importante porque dichas baterías tienen una duración bastante limitada comparada con ordenadores portátiles u ordenadores conectados a la red eléctrica.

6.4 Problemática de los datos

A continuación se describen los problemas que afectaron a las mediciones de potencias de señal: la *granularidad*, que depende del hardware de la interfaz Wifi, y la variación de la potencia de las señales electromagnéticas a lo largo del tiempo.

6.4.1 Granularidad

Un inconveniente que, dependiendo del fabricante hardware, se convierte en un factor relevante para el sistema de localización es la *granularidad* de las interfaces Wifi de los dispositivos móviles, esto es, la precisión en las medidas de las señales que recibe la antena del dispositivo.

El estándar 802.11 define el mecanismo por el que el adaptador de red mide la potencia de señal que recibe de las estaciones Wifi. Este factor de potencia se denomina RSSI (*Received Signal Strength Indication*) que se define en el estándar como un número entero de 8 bits (esto es, su rango de valores puede ir desde 0 a 255). Sin embargo, el estándar no define en absoluto los valores que puede tomar

la RSSI: será el propio fabricante quien decida el uso específico. Según el estándar: *"La RSSI se usará como una medida relativa. La precisión en la medida del RSSI no se especificará."* Que se sepa hasta ahora, ningún fabricante ha hecho uso de los 255 valores posibles de la RSSI; CISCO es el que más valores utiliza con un rango de 0 a 100. [Wil02].

El valor de RSSI lo obtiene la propia circuitería del chip de red, y se utiliza, por ejemplo, cuando el dispositivo quiere emitir información. El adaptador de red debe detectar si el canal está disponible o no, para ello comprobará el nivel de señal a través del RSSI. Si está por debajo de un nivel umbral el chipset sabrá que el canal está libre.

Como no se especifica el uso del valor del RSSI no se puede relacionar de ninguna manera la RSSI con la potencia de señal real que se está recibiendo. Algunos fabricantes sí establecen una relación entre RSSI y dBm, como por ejemplo, CISCO (ver la tabla 6.1), pero, incluso así, posee contradicciones en sus valores. Por ejemplo, sus adaptadores suelen tener una sensibilidad de -96 dBm, pero en la tabla que se presenta aparecen 17 valores de RSSI que corresponden a valores menores de -96 dBm, es decir, son valores imposibles de detectar por la propia tarjeta inalámbrica. En otros casos, para distintos valores de RSSI se repite los mismos valores en dBm (por ejemplo, desde 94 a 100).

RSSI	0 = -113 dBm	86 = -19 dBm
	1 = -112	87 = -18
	2 = -111	88 = -17
	3 = -110	89 = -16
	4 = -109	90 = -15
	5 = -108	91 = -14
	6 = -107	92 = -13
	7 = -106	93 = -12
	8 = -105	94 = -10
	9 = -104	95 = -10
	10 = -103	96 = -10
	11 = -102	97 = -10
	…	98 = -10
	…	99 = -10
	…	100 = -10

Tabla 6.1: Relación de la RSSI respecto dBm en adaptadores CISCO

CISCO, al ofrecer 100 valores de RSSI, está ofreciendo una alta *granularidad,* es decir, muchos niveles intermedios en el valor de RSSI. El concepto de *granularidad* es importante, porque, por ejemplo, el fabricante SYMBOL, que utiliza un valor de RSSI entre 0 y 60, ofrece sólo 30 niveles de granularidad, es decir, el valor RSSI no avanza de uno en uno como debiera esperarse, sino de dos en dos [Wil02].

Por tanto, el factor de granularidad afectará al sistema de localización cuando no permita discernir con cierta precisión el valor de potencia de señal recibida de cada punto de acceso. Este nivel de potencia recibida es el factor más importante para el correcto funcionamiento de la aplicación, ya que es el valor que se usará para estimar las distancias del usuario hacia cada punto de acceso.

Al realizar el primer estudio del entorno para este PFC se trabajó con una PDA IPAQ 5550. Tras analizar todos los datos se comprobó que la potencia recibida no cambiaba en tramos de 15-20 metros, con lo que la estimación de la distancia hacia el punto de acceso se convertía en una tarea inviable: la aplicación obtendría una misma distancia a lo largo de todo ese tramo. Por tanto, se adquirió un nuevo dispositivo para el desarrollo de este PFC: un teléfono móvil HTC Diamond, que sí posee una alta granularidad. A partir de este dispositivo se han realizado nuevos barridos del entorno y se han obtenido nuevos valores, que son los que se publican en este PFC.

6.4.2 Variaciones aleatorias de la potencia

Al ejecutar el sistema de localización se obtuvieron unos resultados muy alejados de la posición a la que realmente se encontraba el usuario. Se comprobó que las potencias recibidas por el dispositivo variaban algunos dBm en medidas consecutivas para una misma posición. Estas variaciones de potencia implicaban cometer un error en las distancias estimadas de tres metros por cada dB de diferencia. Así que se decidió implementar una mejora al sistema de localización: la aplicación no estimaría las distancias a partir de la última potencia recibida, sino que guardaría un histórico de las últimas potencias y obtendría un valor promedio que será el que se utilice para el cálculo de las distancias.

Además, se descubrió un fenómeno que no se había tenido en cuenta en la fase 'offline', esto es, la potencia recibida variaba enormemente cuando el usuario se giraba 180 grados estando en una misma coordenada. Tal como se muestra en la figura 6.1, hay una diferencia de unos 13 dBm entre las orientaciones este y oeste. Este fenómeno se debe a que el propio usuario forma un obstáculo importante entre el router Wifi y el dispositivo móvil: mientras el dispositivo se encuentre en visión directa con el router Wifi la potencia recibida es mucho mayor que cuando el usuario intermedia entre ambos. Por tanto, la aplicación estimaría las distancias con un error de unos 39 metros cuando el usuario se encontrara orientado en una u otra dirección.

Para solventar dicho *hándicap*, se optó por realizar las pruebas del sistema con el usuario realizando un giro de 360 grados en cada coordenada para que la

aplicación midiera las potencias en las cuatro orientaciones (norte, sur, este y oeste) y obtener un valor promedio, tal y como se hizo en el estudio previo del entorno.

6.5 Descripción de la aplicación

Para este PFC se han desarrollado dos aplicaciones con distintos objetivos. La aplicación principal 'WifiLocaliza' se encargará de recibir las señales Wifi que se encuentren a su alrededor e identificará qué estaciones Wifi se encuentran en su base de datos. A partir de las potencias recibidas aplicará una serie de ecuaciones para obtener una estimación de las distancias hacia cada punto de acceso. Finalmente, con dichas distancias, calculará la posición del usuario en base a uno de los algoritmos de localización implementados.

La segunda aplicación, 'WifiReader' ayudará al usuario a realizar el estudio previo del entorno (fase 'offline'). El objetivo de esta aplicación es guardar datos en un fichero de texto de los puntos Wifi que se reciben a lo largo de toda la planta del edificio.

Finalmente, se modificó el código de cada algoritmo, escrito en VisualC++, hacia un C++ puro sin ningún elemento referente a MFC ni .NET Compact. Dicho código C++ se utilizó en una máquina Linux para realizar la fase de simulaciones donde se analizó el comportamiento de cada algoritmo de localización según criterios de consumo de CPU, robustez y precisión.

6.6 Diseño funcional

6.6.1 Aplicación 'WifiLocaliza'

La aplicación principal 'WifiLocaliza' engloba las siguientes funciones (ver figura 6.3):

- **Lectura RSSI.** La aplicación se comunicará con el adaptador de red para pedir los datos relevantes, como son la potencia recibida, nombre de la estación, dirección MAC, etc., de los puntos Wifi existentes alrededor del dispositivo móvil.

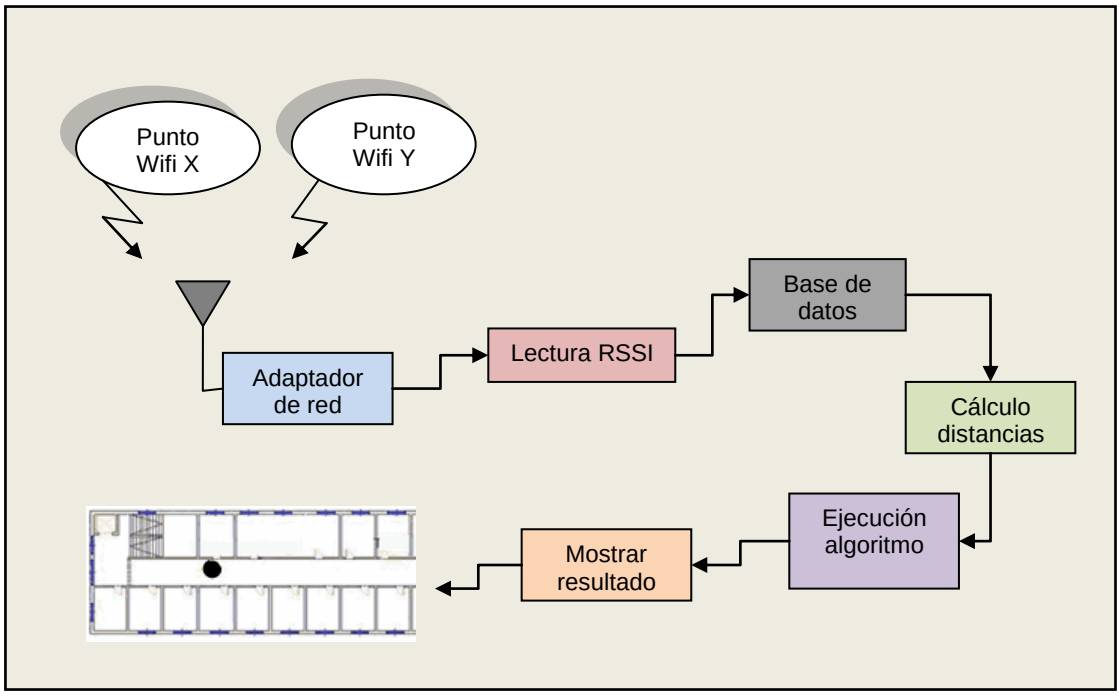


Figura 6.3: Esquema funcional de 'WifiLocaliza'.

Para ello se usará la clase *CWifiPeek* que permite la comunicación con el adaptador de red. Este bloque extraerá los valores de dirección MAC y potencia recibida (RSSI) de todos los puntos Wifi. La dirección MAC es el campo necesario para que el siguiente bloque ('Base de datos') seleccione los puntos Wifi conocidos en el estudio previo del entorno. De estos puntos Wifi conocidos se extraerá, finalmente, la lectura RSSI que será necesaria para el bloque 'Cálculo de distancias'.

- **Consulta a la base de datos.** Tras la detección de todos los puntos Wifi encontrados por el dispositivo la aplicación deberá desechar la información de aquellos que no se encuentren en la base de datos. Dicha base de datos está formada por un fichero de texto con las direcciones MAC de los puntos Wifi que se analizaron en la fase 'offline' del proyecto. Este bloque hace uso de la clase *EntradaFichero* con el método `LeerDelTxt()` para ir leyendo los datos del fichero.

  Aunque en este bloque sólo se haga una búsqueda de las direcciones MAC en la base de datos, el formato del fichero .TXT contiene también otros datos necesarios para los bloques subsiguientes, como son: coordenadas de localización (X,Y) de cada punto Wifi, factores de atenuación (factor 'n') y potencias de referencia recibidas a un metro.

- **Cálculo de la distancia hacia cada punto Wifi.** Tras el estudio previo del entorno realizado por el usuario, se habrá calculado el factor de atenuación. Con este factor 'n' y el valor de potencia recibida en el apartado anterior, se estimará la distancia hacia cada punto Wifi conocido. Esta distancia se halla con la fórmula:

$$d = log^{-1}\left(\frac{P_{TX} - P_{RX} - P_{fs}}{10 \cdot n}\right)$$

- **Ejecución algoritmo.** El usuario escogerá un algoritmo de localización y el dispositivo ejecutará dicho algoritmo hasta obtener una posición en el mapa. Dicho algoritmo necesita de dos datos: distancias hacia cada punto Wifi y localización de cada punto Wifi. Dichas localizaciones se encuentran en la base de datos y las distancias son los valores calculados por el bloque anterior.

  Los algoritmos desarrollados para este PFC son: trilateración, multi-lateración, Min-max y Nelder-Mead. En el caso de que sólo existan dos puntos Wifi disponibles la aplicación obviará la elección del usuario porque dichos algoritmos necesitan de tres o más puntos de acceso para ejecutarse. En este caso se aplicará la bilateración.

  En la siguiente captura se muestra la pestaña 'Opciones' que permite seleccionar el algoritmo:

Figura 6.4: Pestaña 'Opciones'

El resto de opciones ofrecen la misma funcionalidad que en la aplicación original 'PeekPocket' descrita en el apartado 5.7.

- **Mostrar resultado.** Finalmente, se representa en pantalla el mapa de la planta del edificio y la posición calculada en el apartado anterior mediante un punto negro. En el caso de la bilateración, en la que se suelen obtener dos posibles puntos de localización, se mostrarán ambos puntos en el mapa. Y en el caso que sólo se detecte un punto Wifi se representará en pantalla una circunferencia que rodee a dicho punto de acceso.

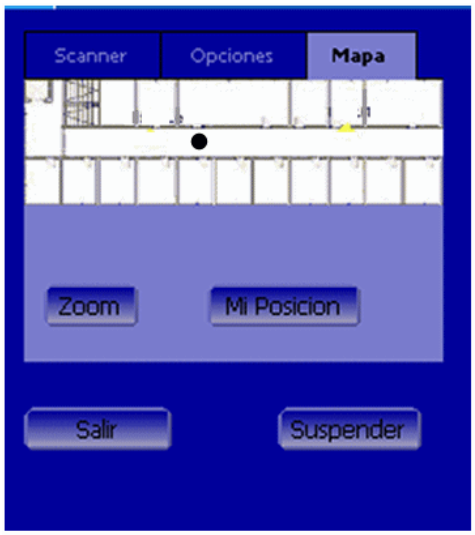


Figura 6.5: Pestaña "Mapa".

Para conocer la posición se deberá pulsar "MI POSICION". La opción "ZOOM" permitirá al usuario agrandar el mapa una determinada escala para poder apreciar algún detalle si el usuario aún no se encuentra orientado. El sistema estará ofreciendo la posición del usuario continuamente hasta que se vuelva a pulsar "MI POSICION".

El mapa se ha incluído como 'recurso' (*resource*) del proyecto en VisualC++, por lo que para elegir otro mapa debe incluirse en el código fuente antes de compilar. El mapa actual es un BMP de 16 colores con dimensiones: 630 por 109 pixels.

6.6.2 Aplicación 'WifiReader'

La segunda aplicación desarrollada para este PFC tiene como objetivo hacer una lectura de los valores de potencia de señal que se reciben de cada punto Wifi y guardarlos en un fichero de texto. Esta aplicación reducirá considerablemente el tiempo empleado para realizar el estudio previo del entorno (fase 'offline') ya que,

automáticamente, guarda múltiples muestras de la potencia de señal sin que el usuario tenga que escribirlos a mano. Este estudio previo es necesario para conocer el factor 'n', que describirá cómo se comportan las señales de radiofrecuencia en el entorno de trabajo.

Esta segunda aplicación también se basa en el código de 'PeekPocket' para la lectura de los niveles de potencia de cada estación base. El esquema funcional de esta aplicación es el siguiente:

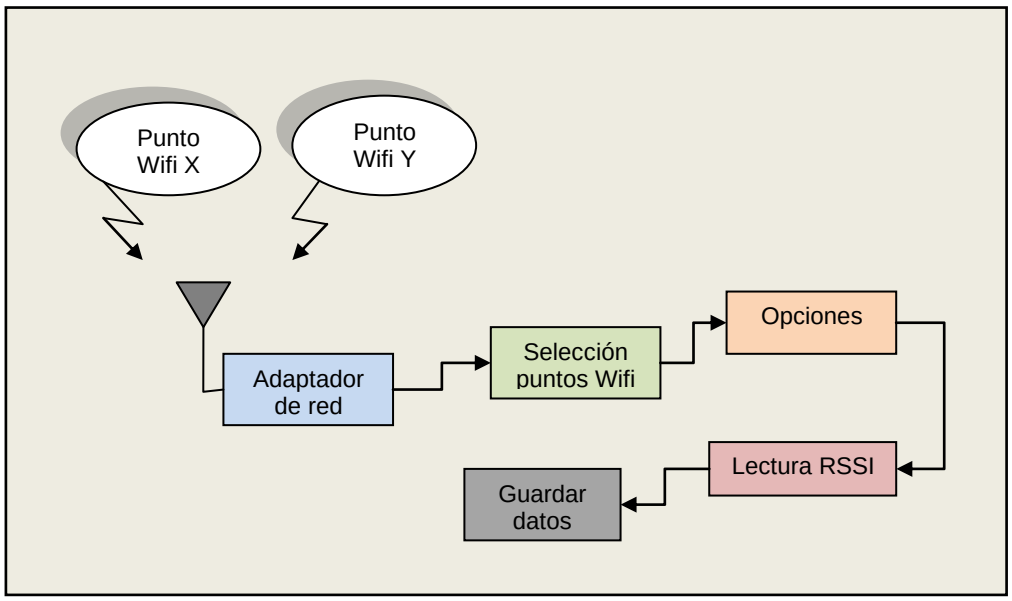


Figura 6.6: Esquema funcional de 'WifiReader'.

- **Selección puntos Wifi.** En este apartado el usuario deberá seleccionar los puntos Wifi de los que conozca su localización. La aplicación guardará los datos pertinentes únicamente de estos puntos Wifi seleccionados.
  La selección se realizará haciendo doble-click en los puntos Wifi deseados que aparecen en pantalla, como se muestra en la figura 6.7.

Figura 6.7: Pestaña 'Scanner' de la aplicación.

- **Opciones.** En este apartado el usuario debe indicar en qué coordenada se encuentra en cada momento y el número de muestras que desea obtener de cada punto Wifi. Opcionalmente, se ha añadido un apartado en el que el usuario puede indicar la temperatura ambiente actual con el fin de comparar resultados entre diferentes climatologías (ver figura 6.8).

Figura 6.8: Opciones para la captura de datos.

Una vez que el usuario pulse "INICIAR" la aplicación mostrará una ventana de aviso en el momento que se hayan guardado el número de muestras seleccionado y se detendrá la grabación de nuevos datos.

- **Lectura RSSI.** Este bloque funcional es exactamente el mismo que en la aplicación anterior, en donde el sistema realiza una lectura de las potencias de señales recibidas según los datos recibidos por el adaptador de red.

- **Guardar datos.** Finalmente, la aplicación guardará en un fichero de texto la información de cada punto Wifi seleccionado, el número de muestras de potencia indicadas en el apartado correspondiente y la orientación en la que se encontraba el usuario. Para cada coordenada elegida por el usuario se creará un nuevo fichero de texto.

6.7 Ficheros de datos utilizados

La aplicación 'WifiLocaliza' necesita los datos de los puntos de acceso instalados en la zona, como son: su localización, dirección MAC para identificar a cada uno de los puntos de acceso, factor 'n' de pérdidas de señales electromagnéticas y la potencia de referencia medida a un metro de distancia.

Todos estos datos deben ser guardados en un fichero .TXT llamado "data.txt" para facilitar el uso al usuario, ya que así sólo tendrá que manejar un simple editor de textos como el "Bloc de notas" en sistemas Windows. El formato es el siguiente:

```
DirecciónMAC-1 | Coord-X1 | Coord-Y1 | Factor 'n1' | Pot.a.1m |
DirecciónMAC-2 | Coord-X2 | Coord-Y2 | Factor 'n2' | Pot.a.1m |
DirecciónMAC-3 | Coord-X3 | Coord-Y3 | Factor 'n3' | Pot.a.1m |
     …             …            …            …            …
     …             …            …            …            …
```

Tabla 6.2: Formato del fichero "data.txt"

Dicho fichero "data.txt" debe estar guardado en la carpeta "*MyDocuments*\" del dispositivo móvil.

Se usará el delimitador '|' para cada campo y se finalizará cada línea con el mismo delimitador. No puede haber ningún 'espacio'. Un ejemplo de línea puede ser:

```
01—A2-45-CB-F1-43|23.5|5.0|1.73|-50|
```

Puede ocurrir que los puntos Wifi no encajen perfectamente en la plantilla de cuadrículas que se creó a la hora de realizar el estudio previo del entorno, por tanto, sus posiciones tendrán valores decimales en caso de que se encuentren entre dos cuadrículas.

Al ejecutar la aplicación 'WifiReader' se pedirán varios datos al usuario: las coordenadas en las que se encuentra el usuario en ese momento (X,Y), la orientación del mismo y la temperatura.

Esta aplicación recoge la potencia de las señales Wifi y guarda dichos datos en un fichero de texto (.TXT) en la misma carpeta donde se encuentra la aplicación, con el siguiente formato:

MAC_1 Canal1	MAC_2 Canal2	MAC_3 Canal3	Fecha	Orientación	Temperatura
-60	-48	-48			
-62	-49	-50			
-62	-46	-51			
-60	-48	-48			
-62	-49	-50			
-62	-46	-51			
…	…	…			

Tabla 6.3: Formato del fichero TXT durante la toma de datos.

Para cada coordenada elegida por el usuario se creará un nuevo fichero de texto. En el caso que el usuario haya escaneado las coordenadas (5,5) y (6,5) obtendría dos ficheros con los nombres: "C_5_5.txt" y "C_6_5.txt".

Cada vez que se capturen datos en las mismas coordenadas con distinta orientación, los nuevos datos se irán añadiendo al final del mismo fichero, obteniendo algo similar al siguiente ejemplo:

MAC_1 Canal1	MAC_2 Canal2	MAC_3 Canal3	09/06/10 11:48:08	Oeste	22°C
-60	-48	-48			
-62	-49	-50			
-62	-46	-51			
-60	-48	-48			
-62	-49	-50			
-62	-46	-51			
MAC_1 Canal1	MAC_2 Canal2	MAC_3 Canal3	09/06/10 11:49:00	Este	22°C
-48	-60	-60			
-42	-60	-57			
-50	-57	-61			
-40	-56	-68			
-45	-60	-60			
-44	-61	-61			
MAC_1 Canal1	MAC_2 Canal2	MAC_3 Canal3	09/06/10 11:50:45	Norte	22°C
-60	-50	-48			
-62	-49	-55			
-62	-46	-52			
-60	-48	-49			
-62	-53	-50			
-62	-46	-57			

Tabla 6.4: Ejemplo de una captura de datos en la misma coordenada con las distintas orientaciones.

CAPÍTULO 7

ANÁLISIS ORGÁNICO

7.1 Introducción

En este capítulo se describirán las clases que han sido implementadas para llevar a cabo las funciones más relevantes comentadas en el análisis funcional, como son las que facilitan la comunicación con la tarjeta inalámbrica, los algoritmos de localización que calcularán la posición del usuario, y las clases que definirán la apariencia de la aplicación.

7.2 Diagrama de clases

En la figura 7.1 se representa el diagrama con las clases más relevantes de este proyecto y sus relaciones con el resto. En la parte superior se encuentra *CColoredDlg* que gobierna la apariencia de toda la aplicación. De ella dependen las tres pestañas principales del programa: "Scanner", "Opciones" y "Mapa" (implementadas en las clases *CScannerDLg, COptionsDlg* y *CDisplayDlg*). Aparte se encuentra "PPDlg" que es el código principal o "*main()".*

En el centro del diagrama se encuentra la clase *CWifiPeek* que gestiona la comunicación con el driver NDIS y hace uso de tres estructuras: *BSSIDInfo, AP_PDAInfo* y *DisplayInfo*.

A la derecha de *CWifiPeek* se encuentran los algoritmos de localización que son independientes del resto de la aplicación, con lo que pueden importarse fácilmente a cualquier otro proyecto C++. Las clases son: CBilateracion, CTrilateracion, CMultilatera y CMinMax.

Finalmente, se encuentra la estructura *EntradaFichero*, que gestiona la lectura de datos desde un fichero de texto con el método *"LeerDelTxt()"*. Al crear esta estructura con un método privado el desarrollador se ahorra volver a escribir código cada vez que quiera leer de un fichero de texto.

7.3 Clase CDisplayDlg

La clase *CDisplayDlg* es la que ejecuta el sistema de localización desarrollado para este PFC. Está definida como una nueva pestaña en la aplicación 'PeekPocket' con el nombre 'Mapa'.

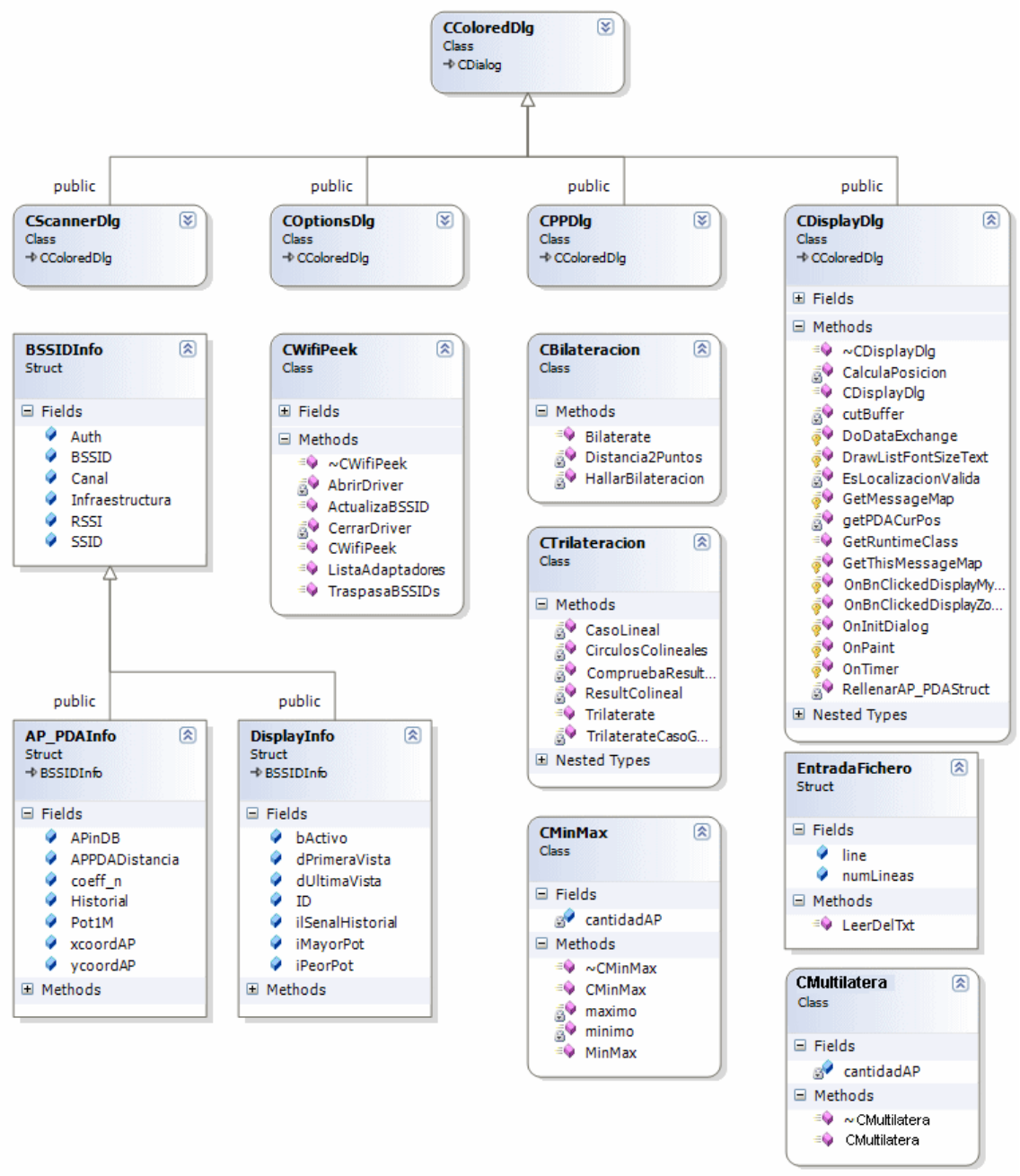


Figura 7.1: Diagrama de clases.

Para añadir una nueva pestaña a la aplicación 'PeekPocket' se ha estudiado la clase *'CCustomTabCtrl'* de MFC, que gestiona la aparición de pestañas personalizadas. Para añadir pestañas adicionales se debe llamar a *'AddDialog'* dentro de la función *'OnInitDialog'* (función que se ejecuta antes que cualquier otra en un proyecto VisualC++ y que define el comportamiento de esa ventana). El código es el que se muestra a continuación:

```
BOOL CPPDlg::OnInitDialog()
{
   …
   …

   // Se crea el "dialog" Display
   m_pDisplayDlg=new CDisplayDlg();
   m_pDisplayDlg->Create(CDisplayDlg::IDD, this);
   //..se eligen los colores
   m_pDisplayDlg->SetBkgColor(RGB_AVERAGE(GetSysColor(COLOR_ACTIVECAPTION),
                                          RGB(255, 255, 255)));
   m_pDisplayDlg->SetFrgColor(RGB(0, 0, 0));
   //..se añade la pestaña al control
   m_tab.AddDialog(2, _T("Mapa"), m_pDisplayDlg);
}
```

- `void` **`OnPaint()`**: Una vez ejecutado el código de inicialización (*OnInitDialog*) se representará en pantalla el mapa de localización y dos botones: "ZOOM" y "MI POSICION". El mapa está incluído en el código fuente como 'recurso' (*resource*) con formato BMP y 16 colores de paleta y para mostrarlo en pantalla se utiliza el siguiente código:

```
void CDisplayDlg::OnPaint()
{
    CColoredDlg::OnPaint(); // Antes de nada, llamar a 'OnPaint'
                            // de CColoredDlg

    //Cargamos en HBMPL1 el mapa.
    HBITMAP hBmpL1=::LoadBitmap(AfxGetResourceHandle(),
                                MAKEINTRESOURCE(IDB_DISPLAY_BITMAP));

    // Crear contexto de dibujo (Device context)
    CRect rc;
    this->GetClientRect(&rc);
    HDC hdc1 = ::GetDC(this->m_hWnd);
    HDC hdcmem = ::CreateCompatibleDC(hdc1);
    ::SelectObject(hdcmem,hBmpL1);

    // 'StrecthBlt' mostrará HDCMEM en el contexto HDC1
    ::StretchBlt(hdc1, 0,0, 205, 37, hdcmem,0,0, 620, 109,SRCCOPY);

    // Liberamos el device context y HDCMEM
    ::SelectObject(hdc1,(HBITMAP)::SelectObject(hdc1, hBmpL1));
    ::DeleteObject(hBmpL1);
}
```

Cuando el 'zoom' esté activado habrá que ampliar el mapa. La escala escogida es 1'8:1. La aplicación tendrá en cuenta la posición del usuario de forma que dirigirá la ampliación hacia la zona izquierda del mapa o hacia la zona derecha.

Esto se consigue variando los parámetros de la función *'StretchBlt'* como se muestra a continuación:

```
// Si 'zoom' está activado
if (zoom)
{
  if (xcoord1<=31.0)  // Si el usuario está en la zona izquierda.
  {
      ::StretchBlt(hdc1, 0,0, 364, 64, hdcmem,0,0, 620, 109,SRCCOPY);
  }
  else
  {
       ::StretchBlt(hdc1, 0,0, 364, 64, hdcmem,270,0, 620, 109,SRCCOPY);
  }
}
else  // No hay zoom y se muestra el mapa completo.
{
   ::StretchBlt(hdc1, 0,0, 205, 37, hdcmem,0,0, 620, 109,SRCCOPY);
}
```

La aplicación actual tiene definidos por defecto las dimensiones del entorno que se ha estudiado en este PFC, por tanto, si se incluyera un mapa con distintas dimensiones habrá que variar dichos parámetros que definan el entorno. En el caso que ocupa este PFC es una planta rectangular; si se desea introducir otro mapa con otras dimensiones habrá que modificar el código para que tenga en cuenta hacia dónde hacer el zoom.

- `void` **`OnBnClickedDisplayMyPosBtn()`** Esta función controla la pulsación del botón "MI POSICION". Su función es activar un *timer* que se encargará de mostrar la posición en pantalla cada dos segundos. Cuando el usuario vuelva a pulsar el botón, la localización se detendrá. El código se muestra a continuación:

```
void CDisplayDlg::OnBnClickedDisplayMyPosBtn()
{
   //Si flagtimer=TRUE desconectamos el timer.
  if (flagtimer)
  {
      KillTimer(PROMEDIO_ID);
      AfxMessageBox(_T("Localización detenida."), MB_ICONINFORMATION);
      flagtimer=false;
  }
    //Si flagtimer=FALSE lo activamos.
  else
  {
      AfxMessageBox(_T("Comienza localización."), MB_ICONINFORMATION);
      SetTimer(PROMEDIO_ID,2000,0);
      flagtimer=true;
  }
}
```

El código del *timer* es el siguiente:

```
void CDisplayDlg::OnTimer(UINT nIDEvent)
{
   switch(nIDEvent)
   {
       case PROMEDIO_ID:
       {
           KillTimer(PROMEDIO_ID);
           CalculaPosicion();
           SetTimer(PROMEDIO_ID,2000,0);
       }
   }
}
```

- `void` **`CalculaPosicion()`** Esta función determina la posición del usuario a través de los niveles de potencia de señal que esté recibiendo de los puntos Wifi que coincidan en su base de datos. Realiza varias llamadas a otras funciones que se describen a continuación.

- `bool` **`LeerDelTxt()`** Esta función está incluída dentro de la estructura *EntradaFichero*. C++ permite este tipo de programación en el que se pueden desarrollar métodos dentro de estructuras. Dicha estructura está definida como:

```
struct EntradaFichero
{
   char linea[128][100];
   int numLineas;
   bool LeerDelTxt();
};
```

La función LeerDelTxt() recorre un fichero de texto (cuyo nombre está definido en la constante _DATAFILE) y guarda su contenido línea a línea en la variable '*linea*'. Devuelve *true* si no ha habido ningún problema en la lectura. El método '*LeerDelTxt()*' se ha definido así:

```
bool EntradaFichero::LeerDelTxt()
{
        char buf[100];
        char* eofDetected;

        FILE *fp=fopen(_DATAFILE,"r");
        if(fp==NULL)
        {
           return false;
        }

        numLineas=0;
        while(1)
```

```
        {
                memset(buf,' ',100);
                eofDetected = fgets(buf,100,fp);
                if(!eofDetected || feof(fp))
                {
                        break;
                }
                strcpy(linea[numLineas++],buf);
        }
        fclose(fp);
        return true;
}
```

Una vez que se haya recorrido el fichero, se accederá a cada línea con el comando: *EntradaFichero.linea[i]*

- bool **RellenarAP_PDAStruct(AP_PDAInfo* apPDAInfo, int APsDetectados, int& numAPEncontrados)** Esta función determina qué puntos Wifi de los que se están detectando pertenecen al entorno que se ha estudiado previamente. Para llamar a esta función hay que pasarle un array con los datos de todos los puntos Wifi que se están recibiendo en ese momento y el número de puntos Wifi. La función devolverá el número de puntos de acceso que se han encontrado en la base de datos. En el caso que encuentre alguna coincidencia, la función traspasará los datos del .TXT pertenecientes a dichos puntos Wifi hacia la misma estructura de entrada (*apPDAInfo*). Además, calculará la distancia estimada hacia dicho punto de acceso.
  La estructura *AP_PDAInfo* gestiona los datos de cada punto Wifi necesarios para ejecutar los algoritmos de localización. Es una estructura que hereda de *BSSIDInfo* sus atributos y añade, además, los siguientes:

  - *APinDB*: indica si el punto de acceso se encuentra en la base de datos, es decir, si es un punto conocido de la planta del edificio del que se ha hecho un estudio previo de potencias. Si no estuviera dicho punto Wifi en la base de datos se descartaría para los cálculos de localización.
  - *APPDA_Distancia:* es la distancia estimada del usuario hacia el punto Wifi.
  - *Coeff_n*: es el factor 'n' que representa el índice de atenuación de las señales en el entorno estudiado. Este dato se recoge de la base de datos.
  - *Pot1M:* es la potencia recibida a un metro que se detectó en el estudio previo del entorno. Este dato se recoge de la base de datos.
  - *XcoordAP:* coordenada X conocida del punto Wifi.
  - *YcoordAP:* coordenada Y conocida del punto Wifi. Ambas coordenadas se recogen de la base de datos.

- *Historial:* guarda las diez últimas muestras de potencias recibidas. Dicho valor se puede modificar con la constante '*PROM*' en tiempo de compilación.

El valor de salida *numAPEncontrados* ofrecerá el número de puntos de acceso conocidos en la base de datos, con este valor se establecerá si se puede ejecutar el algoritmo seleccionado por el usuario (por ser algoritmos que necesitan de tres puntos de acceso o más), o si sólo se puede ejecutar bilateración, o, simplemente, mostrar una circunferencia porque sólo existe un punto de acceso conocido. El código de la función es como sigue:

```
bool CDisplayDlg::RellenarAP_PDAStruct(AP_PDAInfo*  apPDAInfo,
                              int APsDetectados, int& numAPEncontrados)
{
     char temp[100];
     BYTE tempMac[6];
     double tempX;
     double tempY;
     double tempN;
     double temp1M;
     int retPos = -1;
     int sumRetPos=0;
    char temp2[8];
     numAPEncontrados=0;
     for(int i=1;i<m_EntradaFichero.numLineas;i++)
     {
            memset(temp, ' ', 100);
            sumRetPos=0;
            //Leemos primer campo del TXT (direccion MAC)
            retPos = cutBuffer('|',100,m_EntradaFichero.line[i]+sumRetPos,
                                                                  temp);
            sumRetPos +=retPos;
            if(retPos==-1)
                  break;
            int j=0;
            int iter=0;
            while(temp[j])
            {
                  int k=0;
                  memset(temp2,' ',8);
                  while((temp[j])&&(temp2[k++]=temp[j++])!='-');
                  temp2[k]=NULL;
                  tempMac[iter++] = (BYTE) strtol(temp2,0,16);

            } //direccion MAC guardada en TEMPMAC

            //Leemos segundo campo del TXT (coordenada X)
            memset(temp, ' ', 100);
            retPos = cutBuffer('|',100,m_EntradaFichero.line[i]+sumRetPos,
                                                                  temp);
            sumRetPos +=retPos;
            if(retPos==-1)
                  break;
            tempX = atof(temp);
```

```
            //Leemos coordenada Y
            memset(temp, ' ', 100);
            retPos = cutBuffer('|',100,m_EntradaFichero.line[i]+sumRetPos,
                                                                  temp);
            sumRetPos +=retPos;
            if(retPos==-1)
                  break;
            tempY = atof(temp);

            //Leemos Factor N
            memset(temp, ' ', 100);
            retPos = cutBuffer('|',100,m_EntradaFichero.line[i]+sumRetPos,
                                                                  temp);
            sumRetPos +=retPos;
            if(retPos==-1)
                  break;
            tempN = atof(temp);

            //Leemos Potencia recibida a 1 metro
            memset(temp, ' ', 100);
            retPos = cutBuffer('|',100,m_EntradaFichero.line[i]+sumRetPos,
                                                                  temp);
            sumRetPos +=retPos;
            if(retPos==-1)
                  break;
            temp1M = atof(temp);

            //Tenemos todos los valores en memoria y los pasamos
           // a la struct 'apPDAInfo'
            for(int m=0;m<APsDetectados;m++)
            {
                  if(tempMac[0]==apPDAInfo[m].BSSID[0]
                  && tempMac[1]==apPDAInfo[m].BSSID[1]
                  && tempMac[2]==apPDAInfo[m].BSSID[2]
                  && tempMac[3]==apPDAInfo[m].BSSID[3]
                  && tempMac[4]==apPDAInfo[m].BSSID[4]
                  && tempMac[5]==apPDAInfo[m].BSSID[5])
                  {
                        apPDAInfo[m].APinDB=true;
                        apPDAInfo[m].coeff_n=tempN;
                        apPDAInfo[m].xcoordAP=tempX;
                        apPDAInfo[m].ycoordAP=tempY;
                        apPDAInfo[m].Pot1M=temp1M;
                        double promedio=0.0;
                        for (int z=0;z<PROM;z=z+1)
                        {
                              promedio+=apPDAInfo[m].Historial[z];
                        }
                        promedio=promedio/PROM;

                       // Calculamos distancia hacia el punto de acceso
                       apPDAInfo[m].APPDADistancia=pow(10,(15-promedio+
                                                  temp1M) / (10.0*tempN));
                  }
                  numAPEncontrados=numAPEncontrados+1;
                  break;
            }
      }
return true; }
```

La distancia no se calcula a partir de la última muestra de potencia recibida, sino a partir de un array de potencias que hará la función de 'historial de potencias'. La constante PROM representará la longitud de dicho array (por defecto son 10 muestras). La velocidad con la que se guardan las muestras en el array está definida por la opción *"Velocidad de escaneo"* en la pestaña 'Opciones'.

Cada vez que se vaya a calcular la posición del usuario se aplicará el promedio a dicho historial de potencias, y el valor obtenido será el utilizado para calcular la distancia.

Una vez recopilados todos los datos de los puntos Wifi conocidos y las distancias hacia cada uno de ellos se podrán pasar dichos datos a los algoritmos de localización, que se describirán en los siguientes apartados.

Una vez que se reciba el resultado del algoritmo seleccionado, existe una función que comprobará si el resultado está dentro de los límites del mapa o, si por el contrario, se halla fuera. Está expresamente diseñada para los límites del entorno que se estudió en este PFC por lo que para otras localizaciones se deben variar los límites de largo por ancho del mapa en tiempo de compilación. La función devuelve *true* si la posición es válida, o *false* si se encuentra fuera de los límites del edificio, tal como se describe en el siguiente código:

```
// Comprobamos si el punto esta dentro del mapa
bool CDisplayDlg::EsLocalizacionValida(const CPuntos &punto)
{
    if ( (punto.x>62.0) || (punto.x<0.0) || (punto.y>10.0) || (punto.y<0.0))
        return false;
    else
        return true;
}
```

Si devuelve *false,* la aplicación modificará la posición del usuario hacia un punto en el interior del mapa de la siguiente forma:

- Si la coordenada X del resultado es menor que cero, se modifica dicho valor a 1 metro (se localiza al usuario en el extremo izquierdo del edificio).
- Si la coordenada X del resultado es mayor que 61, se modifica dicho valor a 60 metros (localiza al usuario en el extremo derecho del edificio).
- Si la coordenada Y del resultado es menor que cero, se modifica dicho valor a 1 metro (se localiza al usuario en uno de los despachos de la fila superior).

- Si la coordenada Y del resultado es mayor que 10, se modifica dicho valor a 9 metros (se localiza al usuario en uno de los despachos de la fila inferior).

Una vez ejecutado el algoritmo con la llamada a la función *'CalculaPosicion'*, la clase 'CDisplayDlg' mostrará sobre el mapa la localización del usuario. En una aplicación software .NET no habría problema en dibujar el punto de localización (un círculo pequeño) con el comando adecuado *'CDC::Ellipse ()'*. Sin embargo, la aplicación está desarrollada con .NET Compact, y como se comprobó en la figura 5.1, .NET Compact no ofrece ninguna ayuda al desarrollador para dibujar en 2D. Para ello el desarrollador debe crear sus propias funciones a bajo nivel para comunicarse con la pantalla del dispositivo. Como dicha programación no se encuentra en los objetivos del proyecto, se ha decidido adoptar una solución alternativa, esta es, crear dicho punto en un editor de gráficos e incluirlo como recurso BMP en el proyecto de VisualC++ (al igual que se hizo con el mapa del edificio) tal como aparece en la figura 7.2:

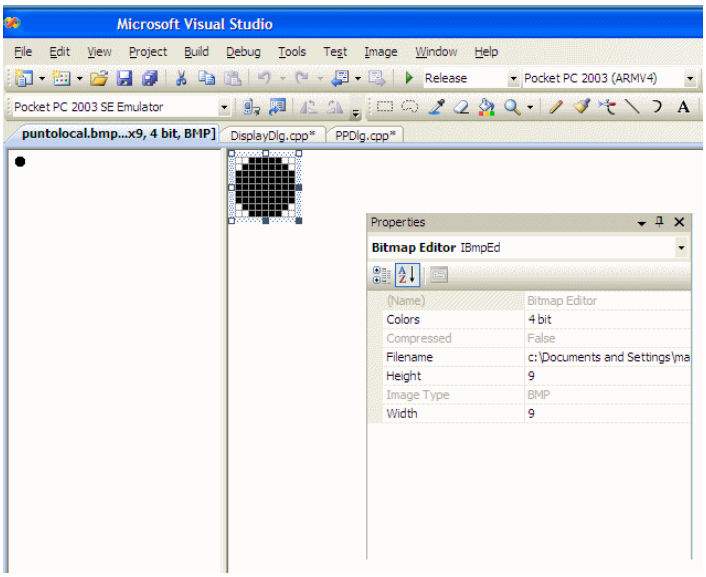


Figura 7.2: Punto de localización añadido como recurso 'bitmap' porque .NET Compact no ofrece soporte para dibujar objetos 2D.

Por tanto, cuando 'CDisplayDlg' quiera mostrar la localización del usuario cargará el *bitmap* y lo mostrará en pantalla de la misma forma que hizo con el mapa. En vez de usar *'StrecthBlt'* se usará *'BitBlt'* que consume menos recursos a costa de que no permite modificar las dimensiones del gráfico como sí hace

'StrecthBlt' para seleccionar una parte del mapa u otra cuando está el zoom activado. A continuación se muestra el código para mostrar la localización del usuario. Dependiendo de si el zoom está activado o no, se modifican los parámetros de posicionamiento en pantalla para *'BitBlt'*:

```
// Cargamos en HPUNTO el recurso bitmap
HBITMAP hPunto=::LoadBitmap(AfxGetResourceHandle(),
                                        MAKEINTRESOURCE(IDB_PUNTO));
::SelectObject(hdcmem,hPunto);

if (zoom)
{
   escala=5.65; // Factor para equiparar (X,Y) con las
                // dimensiones  de la pantalla y el zoom.

   if (xcoord1>=31.0)
   {
       ::BitBlt(hdc1, (int)((xcoord1-27.0)*escala),
              (int)((ycoord1)*escala), 10, 10, hdcmem,0,0,SRCCOPY);

       ::SelectObject(hdcmem,hBmpL1);
   }
   else
   {
   ::BitBlt(hdc1, (int)((xcoord1)*escala), (int)((ycoord1)*escala),
                                        10, 10, hdcmem,0,0,SRCCOPY);
   ::SelectObject(hdcmem,hBmpL1);
   }
}
else
{
    escala=3.24;
   ::BitBlt(hdc1, (int)((xcoord1)*escala), (int)((ycoord1)*escala),
                                        10, 10, hdcmem,0,0,SRCCOPY);
    ::SelectObject(hdcmem,hBmpL1);
}
```

Se observa en este código que se ha utilizado un nuevo parámetro (*escala*) que es específico del entorno estudiado en este PFC. Su función es ajustar las coordenadas de la posición del usuario (cuya medida son 'metros') a las dimensiones de la pantalla del dispositivo.

7.4 Algoritmos de localización

La parte más importante de la aplicación de este PFC es el desarrollo de los diferentes algoritmos de localización. Cada algoritmo se presenta como una clase individual de C++ con sus métodos y variables privadas. De esta forma se permite una sencilla re-utilización del código en otras aplicaciones alternativas que desee el usuario.

Cada algoritmo presenta un método público al que debe pasarse los siguientes datos: las coordenadas (X,Y) de los puntos de acceso y la distancia estimada del usuario hacia cada uno de los puntos de acceso.

Cada localización se define por las coordenadas X e Y englobadas en un tipo llamado '*CPuntos*', tal que:

```
class CPuntos
{
       public:
          double x;
          double y;
};
```

Se ha añadido tal nivel de abstracción para conseguir que cada par de coordenadas sea siempre una estructura compacta e indivisible, y no dos coordenadas numéricas independientes entre sí.

Los resultados que ofrecen los algoritmos al finalizar su ejecución también son del tipo *CPuntos*, a los que se les ha añadido algunos campos más:

- **Número de puntos**. Hay que saber cuántas localizaciones ha obtenido el algoritmo, porque en el caso de la bilateración se suelen obtener dos posibles puntos, y el resto de la aplicación deberá saber manejar este caso para representar ambos puntos en pantalla y no sólo uno.
- **Código de retorno.** En algunos casos, el resultado es un punto no válido, o bien, el algoritmo ha encontrado algún tipo de error de computación. Con este código el programa conocerá del error pertinente.

Todos estos campos se han reunido en la clase *CResultado* tal que:

```
class CResultado
{
public:
       int numDePuntos;
       int retCode;
       CPuntos punto1;
       CPuntos punto2;
};
```

Por tanto, para acceder a los puntos que un algoritmo ha calculado habría que teclear, por ejemplo: `resultado.punto1.x` para obtener la coordenada X del primer punto obtenido. O bien, para conocer el número de puntos obtenidos en el resultado habría que usar: `resultado.numDePuntos`

Un dato importante a tener en cuenta es que para este PFC se ha decidido elegir un eje de coordenadas tal como se muestra en la figura 8.2 porque es el mismo que utiliza WindowsMobile a la hora de representar gráficos en pantalla. Sin embargo, ese eje no es correcto para las operaciones matemáticas, donde se espera que el eje Y sea de signo contrario al de la figura 8.2. Por tanto, a la hora de pasar las coordenadas de los puntos Wifi al algoritmo se debe cambiar el signo de las coordenadas de dicho eje. Por ejemplo, si un punto de acceso tiene de coordenadas (26,5) se deben modificar a (26,-5) para que el algoritmo opere correctamente con ellas.

7.4.1 Asignación dinámica de variables

Un factor importante a la hora de escribir el código de este PFC es el consumo de recursos del dispositivo móvil. Un dispositivo móvil tiene una memoria RAM reducida y una duración de batería igualmente reducida en comparación con los ordenadores portátiles; más aún, cuando el sistema operativo en el que se ejecuta esta aplicación (WindowsMobile) es un sistema que permite ejecutar distintas aplicaciones simultáneamente. Por tanto, a la hora de crear el código hay que intentar optimizar cuanto se pueda el uso de memoria y de CPU.

Una de las optimizaciones que se ha hecho es a la hora de pasar datos hacia el algoritmo de localización seleccionado. Los datos que necesita el algoritmo son: coordenadas "x" e "y" de cada punto Wifi y la distancia estimada hacia cada uno de ellos. Por tanto habrá que inicializar un array de coordenadas y otro de distancias.

Como no se puede saber de antemano cuántos puntos Wifi se detectarán en el futuro, se tendría que elegir un array de longitud larga para cubrir todas las posibilidades. Pero si se escoge un número alto y al ejecutar la aplicación se detectan, únicamente, dos o tres puntos de acceso, se estarían desperdiciando posiciones de memoria RAM. Por el contrario, si se inicializa la longitud a un valor bajo, se estarían perdiendo los puntos Wifi extra que se detectarán en algún momento.

Por tanto, en vez de inicializar los arrays a un número fijo determinado se realizará una *asignación dinámica* en memoria, es decir, la aplicación creará las posiciones en memoria necesarias para cada situación. Y, además, una vez acabado el cálculo de la localización se liberarán dichas posiciones de memoria para que otra aplicación pueda hacer uso de ellas [Sal00].

7.4.2 Bilateración

El fichero *CBilateracion.h* muestra los métodos incluídos en esta clase:

```
class CBilateracion
{
public:
	CResultado Bilaterate(CPuntos &p1, double d1,
                                                CPuntos &p2, double d2);
private:
	double Distancia2Puntos(const CPuntos &p1,const CPuntos &p2);

};
```

Se comprueba que para llamar a este algoritmo hay que pasarle las dos localizaciones de los puntos Wifi y las dos distancias respectivas. El método *Distancia2Puntos* mide la distancia que hay entre dos coordenadas; se utilizará para comprobar si hay intersección o no entre las circunferencias y aplicar, por tanto, el segundo método de resolución, tal como se explicó en el apartado 4.3.1.

```
if ((p1.x+d1)>(p2.x+d2))    // Variante del CASO 3
                            // (1 circunf. dentro de otra)
	{
		BilatPunto1.x=(p2.x+d2+p1.x+d1)/2; // Valor medio.
		BilatPunto1.y=(p2.y+p1.y)/2;
		ret.numDePuntos=1;
		ret.punto1=BilatPunto1;
	}

	else if (Distancia2Puntos(p1,p2) > (d1+d2)) // CASO 3

	{
		if (p2.x>p1.x)
		{
			BilatPunto1.x = (p1.x+d1+p2.x-d2)/2;
		}
		else
		{
			BilatPunto1.x = (p2.x+d2+p1.x-d1)/2;
		}

		BilatPunto1.y = (p1.y+p2.y)/2;
		ret.punto1=BilatPunto1;
		ret.numDePuntos=1;

	}
	else
	{
		// Interseccion general de dos circunferencias C((a,b),r)
		// con C((c,d),s)

		double a,b,c,d,e, f;
		double g, h, r, s;
```

```
        a=p1.x; b=p1.y; r = d1;
        c=p2.x; d=p2.y; s = d2;
        e = c - a;
        f = d - b;
        g = sqrt(e*e + f*f);
        h = (g*g + r*r - s*s) / (2*g);

        BilatPunto1.x = a + e*h/g + (f/g) * sqrt(r*r - h*h);
        BilatPunto1.y = b + f*h/g - (e/g) * sqrt(r*r - h*h);

        BilatPunto2.x = a + e*h/g - (f/g) * sqrt(r*r - h*h);
        BilatPunto2.y = b + f*h/g + (e/g) * sqrt(r*r - h*h);

        ret.numDePuntos=2;
        ret.retCode=1;
        ret.punto1=BilatPunto1;
        ret.punto2=BilatPunto2;

}
return ret;
```

7.4.3 Trilateración

Para calcular la trilateración deben existir exclusivamente tres puntos de acceso a nuestro alrededor. La llamada es similar a la bilateración, donde en este caso se añade una localización y una distancia extra, que corresponde al tercer punto Wifi:

```
CResultado Trilaterate(const CPuntos &p1, double d1, const CPuntos &p2,
                       double d2, const CPuntos &p3, double d3)
```

El código desarrollado para este algoritmo, basado en el apartado 4.3, ha sido el siguiente:

```
CResultado retPoint;

// Cálculo coordenada X
retPoint.punto1.x = ((p1.y - p2.y)* (p1.x*p1.x - p3.x*p3.x  +
      p1.y*p1.y - p3.y*p3.y   + d3*d3 - d1*d1)  -
      (p1.y - p3.y)*(p1.x*p1.x - p2.x*p2.x + p1.y*p1.y -
      p2.y*p2.y   + d2*d2 - d1*d1))/
      ( 2* ( ((p1.y - p2.y) *(p1.x - p3.x)) - ((p1.y -
      p3.y)*(p1.x - p2.x))));

// Cálculo coordenada Y
retPoint.punto1.y = ((p1.x*p1.x)-(p2.x*p2.x)-
      (2*retPoint.punto1.x*(p1.x-p2.x))+p1.y*p1.y-
      p2.y*p2.y+d2*d2-d1*d1)/ (2*(p1.y-p2.y));

retPoint.numDePuntos=1;
retPoint.retCode=1;
```

```
        return retPoint;
```

En el caso que los tres puntos de acceso tengan una localización colineal, es decir, los centros de las tres circunferencias se encuentran sobre la recta formada por $y = mx + n$, donde "m" es la pendiente de dicha recta, la trilateración no se puede resolver ya que uno de los divisores anteriores es igual a cero. En la figura 7.3 se muestran tres circunferencias colineales.

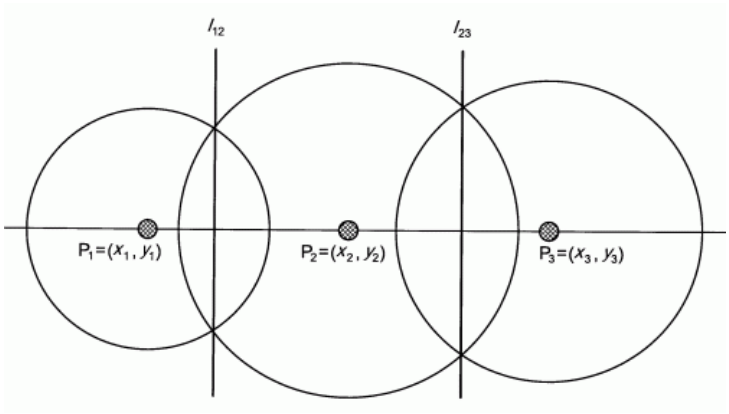


Figura 7.3: Ejemplo de tres circunferencias con centros colineales.

La condición que marca si los tres centros son colineales es la siguiente:

$$x_1(y_2 - y_3) + x_2(y_3 - y_1) + x_3(y_1 - y_2) = 0$$

En el caso que los tres puntos de acceso tengan localizaciones colineales se hallarán todos los puntos en los que intersectan entre cada par de circunferencias, es decir, se resolverán consecutivamente tres bilateraciones (tres sistemas de dos ecuaciones con dos incógnitas).

```
        CPuntos Cir1Cir2Point1;
        CPuntos Cir1Cir2Point2;
        CPuntos Cir1Cir3Point1;
        CPuntos Cir1Cir3Point2;
       double a2, b2, c2, a3, b3, c3, div2, div3;

        //caso 1: coordenadas "Y" coinciden.
        if((p1.y == p2.y) && (p1.y == p3.y))
        {
              Cir1Cir2Point1.x = (p1.x*p1.x - p2.x*p2.x - d1*d1 +
d2*d2)/(2*(p1.x - p2.x));
         Cir1Cir2Point2.x = (p1.x*p1.x - p2.x*p2.x - d1*d1 +
d2*d2)/(2*(p1.x - p2.x));
```

```
            Cir1Cir3Point1.x = (p1.x*p1.x - p3.x*p3.x - d1*d1 +
d3*d3)/(2*(p1.x - p3.x));
        Cir1Cir3Point2.x = (p1.x*p1.x - p3.x*p3.x - d1*d1 +
d3*d3)/(2*(p1.x - p3.x));

            a2 = 4*(p1.x-p2.x)*(p1.x-p2.x);
            b2 = -8*p1.y*(p1.x-p2.x)*(p1.x-p2.x);
            c2 = (p1.x*p1.x - p2.x*p2.x - d1*d1 + d2*d2 - 2*p1.x*(p1.x-
p2.x))*
                  (p1.x*p1.x - p2.x*p2.x - d1*d1 + d2*d2 -
2*p1.x*(p1.x-p2.x)) -
                  4*((p1.x-p2.x)*d1)*((p1.x-p2.x)*d1) +
                  4*((p1.x-p2.x)*p1.y)*((p1.x-p2.x)*p1.y);

            a3 = 4*(p1.x-p3.x)*(p1.x-p3.x);
            b3 = -8*p1.y*(p1.x-p3.x)*(p1.x-p3.x);
            c3 = (p1.x*p1.x - p3.x*p3.x - d1*d1 + d3*d3 - 2*p1.x*(p1.x-
p3.x))*
                  (p1.x*p1.x - p3.x*p3.x - d1*d1 + d3*d3 -
2*p1.x*(p1.x-p3.x)) -
                  4*((p1.x-p3.x)*d1)*((p1.x-p3.x)*d1) +
                  4*((p1.x-p3.x)*p1.y)*((p1.x-p3.x)*p1.y);

            div2 = b2*b2 -4*a2*c2;
            div3 = b3*b3 -4*a3*c3;

            if(div2 >=0.0 && div3 >=0.0)
            {
                  div2 = sqrt(div2);
                  div3 = sqrt(div3);

                  Cir1Cir2Point1.y = (-b2 + div2)/(2*a2);
                  Cir1Cir2Point2.y = (-b2 - div2)/(2*a2);

                  Cir1Cir3Point1.y = (-b3 + div3)/(2*a3);
                  Cir1Cir3Point2.y = (-b3 - div3)/(2*a3);;
            }

      }
      //caso 2 + 3: coord "X" coinciden y
      //caso general de centros colineales
      else
      {
            double cons2_1 = (p1.x*p1.x)-(p2.x*p2.x)+(p1.y*p1.y)-
(p2.y*p2.y) + d2*d2 - d1*d1 - 2*p1.y*(p1.y-p2.y);
            double cons2_2=  -2*(p1.x-p2.x);
            double b2 = 2*cons2_1*cons2_2 - 8*(p1.y-p2.y)*(p1.y-
p2.y)*p1.x;
            double a2 = 4*(p1.y-p2.y)*(p1.y-p2.y) + cons2_2*cons2_2;
            double c2 = 4*(p1.y-p2.y)*(p1.y-p2.y)*p1.x*p1.x
+cons2_1*cons2_1 - 4*(p1.y-p2.y)*(p1.y-p2.y)*d1*d1;
            double div2 =  b2*b2-4*a2*c2;

            double cons3_1 = (p1.x*p1.x)-(p3.x*p3.x)+(p1.y*p1.y)-
(p3.y*p3.y) + d3*d3 - d1*d1 - 2*p1.y*(p1.y-p3.y);;
            double cons3_2 = -2*(p1.x-p3.x);
            double b3 = 2*cons3_1*cons3_2 - 8*(p1.y-p3.y)*(p1.y-
p3.y)*p1.x;
```

```
            double a3 = 4*(p1.y-p3.y)*(p1.y-p3.y) + cons3_2*cons3_2;
            double c3 = 4*(p1.y-p3.y)*(p1.y-p3.y)*p1.x*p1.x
+cons3_1*cons3_1 - 4*(p1.y-p3.y)*(p1.y-p3.y)*d1*d1;
            double div3 =  b3*b3-4*a3*c3;


            if(div2 >=0.0 && div3 >=0.0)
            {
               div2 = sqrt (div2);
               div3 = sqrt (div3);

               Cir1Cir2Point1.x = (-b2+div2)/(2*a2);
               Cir1Cir2Point1.y = ((p1.x)*(p1.x) - (p2.x)*(p2.x) -
2*Cir1Cir2Point1.x*(p1.x-p2.x)
                     +(p1.y)*(p1.y)-(p2.y)*(p2.y) +d2*d2 - d1*d1)
/(2*(p1.y-p2.y));
               Cir1Cir2Point2.x = (-b2-div2)/(2*a2);
               Cir1Cir2Point2.y = ((p1.x)*(p1.x) - (p2.x)*(p2.x) -
2*Cir1Cir2Point2.x*(p1.x-p2.x)
                     +(p1.y)*(p1.y)-(p2.y)*(p2.y) +d2*d2 - d1*d1)
/(2*(p1.y-p2.y));
               Cir1Cir3Point1.x = (-b3+div3)/(2*a3);
               Cir1Cir3Point1.y = ((p1.x)*(p1.x) - (p3.x)*(p3.x) -
2*Cir1Cir3Point1.x*(p1.x-p3.x)
                     +(p1.y)*(p1.y)-(p3.y)*(p3.y) +d3*d3 - d1*d1)
/(2*(p1.y-p3.y));


               Cir1Cir3Point2.x = (-b3-div3)/(2*a3);
               Cir1Cir3Point2.y = ((p1.x)*(p1.x) - (p3.x)*(p3.x) -
2*Cir1Cir3Point2.x*(p1.x-p3.x)
                     +(p1.y)*(p1.y)-(p3.y)*(p3.y) +d3*d3 - d1*d1)
/(2*(p1.y-p3.y));
            }
   }
   return  ResultColineal(Cir1Cir2Point1, Cir1Cir2Point2,
                                    Cir1Cir3Point1, Cir1Cir3Point2);
```

7.4.4 Multilateración

Para el algoritmo de la multilateración se trabajará con matrices para desarrollar la aproximación por mínimos cuadrados, tal como se describió en el apartado 4.4, y cuya ecuación es:

$$\hat{x} = (A^T \cdot A)^{-1} \cdot A^T \cdot b$$

Para este PFC se desarrollaron funciones básicas de operaciones con matrices para agilizar y evitar la re-escritura de código. Las funciones diseñadas fueron: la multiplicación de dos matrices (función '*Mult*'), obtención de una matriz

traspuesta (función *'Trasp'*) y el cálculo de la matriz inversa (función *'MatInversa'*). A continuación se detalla el código de todas ellas:

```
// MULTIPLICACION   Mult = A * B
// Si las matrices son cuadradas M=N=K
// Mult (salida,A,B,filas de A,columnas de B, columnas de A)
void Mult(double **mult, double **A, double **B,int M,int N, int K)
{
int i,j,k;
for (i=1; i<=M; i++)       // matriz A = "M" x n
{
   for (j=1; j<=N; j++)  // matrix B = m x "N"
   {
       mult[i][j]=0;
       for (k=1; k<=K; k++)  // matriz A = m x "N"
       {
              mult[i][j] += A[i][k]*B[k][j];
       }
   }
}
}


// TRASPUESTA DE A
void Trasp(double **trasp, double **A)
{
double temp;
int i,j;
for (i=1; i<=M; i++)
{
   for (j=1; j<=M; j++)
   {
     temp=A[i][j];
     trasp[i][j]=trasp[j][i];
     trasp[j][i]=temp;
   }
}
}

// MATRIZ INVERSA
void MatInversa(double **x,double **a)
{
int i,j,k;
double Sum,m;
int N=2;            // Porque la matriz será siempre (2x2)
double **b, **q;
b=new double *[N+1];
q=new double *[N+1];
for (i=0; i<=N; i++)
{
       b[i]=new double [N+1];
       q[i]=new double [N+1];
}

for (i=1;i<=N;i++)
for (j=1;j<=N;j++)
{
  b[i][j]=0;
```

```
        q[i][j]=a[i][j];
        if (i==j)
             b[i][j]=1;
      }
      // Operaciones de filas
      for (k=1;k<=N-1;k++)
      for (i=k+1;i<=N;i++)
      {
        m=q[i][k]/q[k][k];
        for (j=1;j<=N;j++)
        {
             q[i][j]-=m*q[k][j];
             b[i][j]-=m*b[k][j];
        }
      }

      // Sustitucion
      for (i=N;i>=1;i--)
      for (j=1;j<=N;j++)
      {
        Sum=0;
        x[i][j]=0;
        for (k=i+1;k<=N;k++)
             Sum += q[i][k]*x[k][j];
        x[i][j]=(b[i][j]-Sum)/q[i][i];
        }
        for (i=0;i<=N;i++)
        {
             delete b[i], q[i];
        }
        delete b, q;
}
```

A continuación, se muestra el código del algoritmo de multi-lateración desarrollado para este PFC:

```
CResultado ret;
int i;
double **A,**b;
double **At,**aux,**aux2;

M=cantidadAP-1;
xx=new double [cantidadAP+1];
yy=new double [cantidadAP+1];
dd=new double [cantidadAP+1];

A=new double *[M+1];
b=new double *[M+1];
At=new double *[3];
aux=new double *[M+1];
aux2=new double *[M+1];

for (i=0;i<=M;i++)
{
       A[i]=new double [3];
       b[i]=new double [2];
```

```
        At[i]=new double [M+1];
        aux[i]=new double [M+1];
        aux2[i]=new double [M+1];
}

// Obtengo las matrices necesarias: 'A' y 'b'
// que aparecen en la ecuación: ( A^T * A )-1  * A^T * b
for (i=1;i<=M;i++)
{
        A[i][1]=2*(x[i]-x[M+1]);
        A[i][2]=2*(y[i]-y[M+1]);
}

for (i=1;i<=M;i++)
{
        b[i][1]=x[i]*x[i]-x[M+1]*x[M+1]+y[i]*y[i]-
                        y[M+1]*y[M+1]+d[M+1]*d[M+1]-d[i]*d[i];
}

// Obtengo TRASPUESTA At
Trasp(At,A);

//MULTIPLICACION (A^t * A)
Mult(aux,At,A,2,2,cantidadAP-1);

//Inversa (A^t * A)^-1
MatInversa(aux2,aux);

// MULTIPLICACION ( Inversa * A^t )
Mult(aux,aux2,At,2,cantidadAP-1,2);

// MULTIPLICACION Inversa * (A^t * b) >> la salida será AUX2
Mult(aux2,aux,b,2,1,cantidadAP-1);

//  X,Y  =  aux2[1][1] , aux2[2][1]
ret.punto1.x=aux2[1][1];
ret.punto1.y=aux2[2][1];
ret.retCode=1;
ret.numDePuntos=1;

// Liberamos memoria.
for (i=0;i<=M;i++)
{
   delete A[i],b[i],At[i],aux[i],aux2[i];
}
delete A,b,At,aux,aux2;

return ret;

}
```

7.4.5 Min-Max

El algoritmo MinMax está definido en *Minmax.h* de la siguiente forma:

```
public:
	CResultado MinMax(double* x, double* y, double* d,
                                                    int numeroAP);
private:
	double minimo(double*);
	double maximo(double*);
```

Se comprueba que existen dos funciones privadas llamadas '*minimo*' y '*maximo*'. Estas funciones realizan las operaciones pertinentes para calcular el mínimo valor (o el máximo valor) de un array de números. Este algoritmo puede ejcutarse con cualquier número de puntos de acceso (*numeroAP*) mayor de dos.

```
CResultado ret;
double xmin,ymin,xmax,ymax,xfinal,yfinal;
double *aux;
aux=new double [numeroAP*2];

for (i=0;i<numeroAP;i=i+1)  // Cálculos de (Xi-di)
{
	aux[i]=x[i]-d[i];
}

xmin=maximo(&aux[0]);

for (i=0;i<numeroAP;i=i+1)  // Cálculos de (Yi-di)
{
	aux[i]=y[i]-d[i];
}

ymin=maximo(&aux[0]);


for (i=0;i<numeroAP;i=i+1)  // Cálculos de (Xi+di)
{
	aux[i]=x[i]+d[i];
}
xmax=minimo(&aux[0]);

for (i=0;i<numeroAP;i=i+1)  // Cálculos de (Yi+di)
{
	aux[i]=y[i]+d[i];
}
ymax=minimo(&aux[0]);

// Coordenadas X,Y del centro del cuadrado.
xfinal=(xmin+xmax)/2;
yfinal=(ymin+ymax)/2;

ret.punto1.x=xfinal;
ret.punto1.y=yfinal;
ret.retCode=1;
ret.numDePuntos=1;

// Liberamos memoria.
```

```
	delete [] aux;
	return ret;
	}

// Función que calcula el mínimo de un array de valores.
double CMinMax::minimo(double* x)
{
	int i;
	double aux;
	aux=x[0];

	for (i=1;i<numeroAP;i=i+1)
	{
		if (aux>x[i])
		{
			aux=x[i];
		}
	}
	return aux;

}

// Función que calcula el máximo de un array de valores.
double CMinMax::maximo(double* x)
{
	int i;
	double aux;
	aux=x[0];

	for (i=1;i<numeroAP;i=i+1)
	{
		if (aux<x[i])
		{
			aux=x[i];
		}
	}
	return aux;

}
```

7.4.6 Nelder-Mead

El algoritmo de minimización está basado en el código publicado en [One71], y portado a C por John Burkart en 2008. Para ejecutar el algoritmo hay que utilizar la función *'nelmin'* definida en el fichero *asa047.h*:

```
	void nelmin ( double fn ( double x[] ), int n, double start[],
			double xmin[], double *ynewlo, double reqmin,
			double step[], int konvge, int kcount, int *icount,
			int *numres, int *ifault );
```

El primer parámetro, '*fn*', es la función a minimizar, que corresponde a la fórmula del residuo tal como aparece en el apartado 4.5. Se ha definido de esta forma:

```
// Funcion a Minimizar (función del residuo)
double fn (double *z)
{
  double residuo=0.0;
  int i;
  for (i=1;i<=M;i++)  // M = cantidadAP
  {
    residuo+=sqrt( (xx[i]-z[0])*(xx[i]-z[0]) + (yy[i]-z[1])*(yy[i]-z[1])
                                                              )-dd[i];
  }
  return (residuo/M);
}
```

El resto de parámetros son los valores por defecto que el desarrollador del código 'asa047' estima oportunos y que en este PFC se definieron tal que:

```
int icount;
int ifault;
int kcount;              // Número máximo de iteraciones permitido.
int konvge;
int n=2;                 // Número de variables en estudio: (X,Y)
int numres;
double reqmin;
double *start;
double *step;
double *xmin;
double ynewlo;
start = new double[n];
step = new double[n];
xmin = new double[n];    // Resultado que ofrece NELMIN.
start[0] = aux2[1][1];   // Coordenada X de inicio.
start[1] = aux2[2][1];   // Coordenada Y de inicio.
konvge=10;
reqmin=1.0E-06;
kcount=500;              // Núnero máximo de iteraciones permitido.
step[0] = 1.0;
step[1] = 1.0;
ynewlo = f ( start );

// Llamada a NelderMead.
nelmin ( *f, n, start, xmin, &ynewlo, reqmin, step,
            konvge, kcount, &icount, &numres, &ifault );

ret.punto1.x=xmin[0]; // Coordenada X estimada por NelderMead.
ret.punto1.y=xmin[1]; // Coordenada Y estimada por NelderMead.
ret.retCode=1;
ret.numDePuntos=1;

delete [] start;
delete [] step;
delete [] xmin;
```

Se observa que existen algunas variables que se definieron pero no se hace uso alguno de ellas. Estas variables son de salida de la función de Nelder-Mead ('nelmin') y son datos que describen algunas estadísticas de cómo se ejecutó el algoritmo, como son: el número de iteraciones que se han ejecutado hasta alcanzar el valor estimado ('icount'), el número de reinicios del algoritmo ('numres') y un código de error ('ifault') que toma valores 0, 1 y 2, de forma que:

0 : No hubo errores.
1 : REQMIN, N o KONVGE tienen valores no permitidos.
2 : Se detuvo el proceso porque se superó el número de iteraciones marcado por KCOUNT.

Este PFC no hace uso de ninguno de estos valores de salida que ofrece Nelder-Mead. Sin embargo, durante la fase de pruebas se probaron múltiples valores para todas las variables de entrada para comprobar en cuánto afectaba a la salida y no se llegó a ninguna conclusión en la que se pudiera mejorar el resultado obtenido.

Este algoritmo no se ha desarrollado como clase de C++ porque el código C no permitía dicha posibilidad.

CAPÍTULO 8

CASO DE ESTUDIO

8.1 Introducción

Las pruebas experimentales se han desarrollado en el Pabellón C del Edificio de Electrónica y Telecomunicaciones. Concretamente, en la segunda planta, que corresponde al Departamento de Ingeniería Telemática (DIT) de la U.L.P.G.C.

El sistema de localización desarrollado en este PFC utiliza el nivel de potencia recibido de cada punto de acceso instalado a lo largo de la planta del edificio para calcular la posición del dispositivo móvil.

A partir de esta potencia se estimará la distancia respecto a cada punto de acceso. Una vez estimadas las distancias en metros, se aplicará un algoritmo de localización que determinará la posición del usuario.

8.2 Ajustes previos del sistema de localización

Para poner en funcionamiento el sistema de localización hay que realizar un estudio previo del entorno, ya que el comportamiento de las señales electromagnéticas dependen enormemente de las características físicas del mismo.

8.2.1 Definición del entorno.

La planta posee unas dimensiones de 61'5 metros de largo por 11'5 metros de ancho. Tiene dispuestos dos puntos Wifi para cubrir toda su extensión.

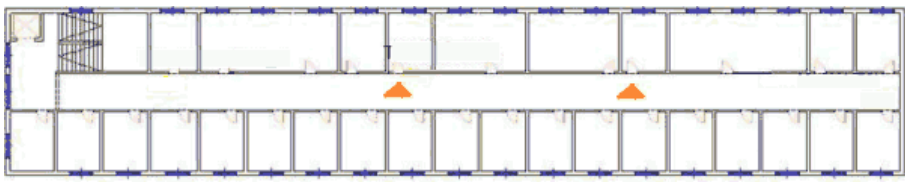


Figura 8.1: Segunda planta del edificio del DIT

Se ha dividido el plano en casillas que representan las coordenadas de posición metro a metro. El origen del eje de coordenadas se ha definido en la esquina superior izquierda, y tanto la coordenada 'X' como la coordenada 'Y' irán incrementándose positivamente a medida que avanzamos hacia la derecha o hacia abajo, respectivamente.

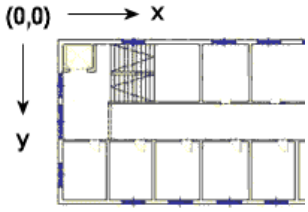


Figura 8.2: Origen y dirección del eje de coordenadas.

Según dicho mapa de coordenadas los dos puntos Wifi existentes se sitúan como se muestra en la tabla 8.1:

Nombre de la estación	Dirección MAC	Coordenadas (x,y)
EDUROAM	00-12-01-b5-62-d1	(26,5'5)
EDUROAM	00-12-01-b5-5b-81	(43,5'5)

Tabla 8.1: Coordenadas (X,Y) de los puntos de acceso.

Los muros del edificio son de hormigón, y su distribución forma espacios de 3 metros de largo por 4'5 metros de ancho a lo largo de toda la longitud de la planta. Estos espacios son los despachos del profesorado.

Por otro lado, también existen unas salas de dimensiones superiores dedicadas a laboratorios y salas de reuniones.

También existe una disposición de objetos de inmobiliario como pueden ser las mesas, armarios, buzones metálicos o puertas de madera, que también modifican el comportamiento de las señales Wifi.

Figura 8.3: Pasillo del departamento de Ingeniería Telemática.

8.2.2 Mapeado de potencias

Este paso consiste en recorrer toda la planta del edificio e ir guardando, en cada coordenada existente, la potencia que se recibe de los dos puntos Wifi. En primer lugar, se recorren las coordenadas que tengan visión directa hacia los puntos de acceso, esto es, el pasillo principal, tal y como se muestra en la figura 8.4:

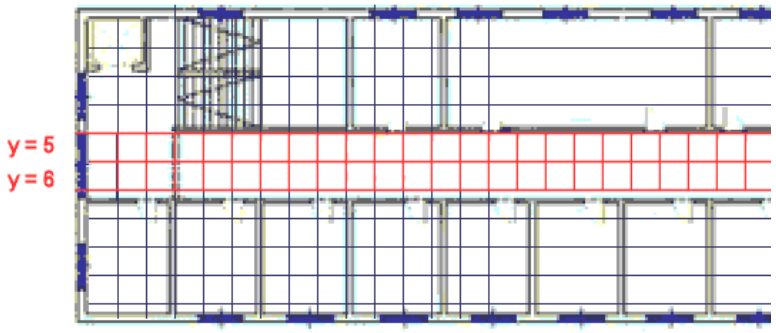


Figura 8.4: Coordenadas con visión directa hacia los puntos Wifi.

En cada casilla-coordenada se han obtenido unas cien muestras de niveles de potencia de las señales Wifi y se han guardado los datos en un fichero de texto gracias a la segunda aplicación diseñada para este PFC, que automatiza el proceso

tal y como se describe en el análisis funcional del capítulo 6. En cada coordenada se realizan entre 25 y 30 medidas por cada orientación (norte, sur, este y oeste), con lo que para cada coordenada se obtendrán unas cien muestras aproximadamente.

Con todas las potencias obtenidas, se aplica la media aritmética de todas las muestras y se obtiene el resultado final para cada coordenada.

8.2.3 Cálculo de los índices de pérdidas

Una vez obtenidas las potencias en cada coordenada habrá que relacionarlas con la distancia real a la que se encuentra el usuario. Para ello se representa en la siguiente gráfica 8.5 las potencias obtenidas en todo el pasillo para el punto de acceso EDUROAM cuya dirección MAC acaba en '81' (la línea vertical negra indica la posición del router Wifi):

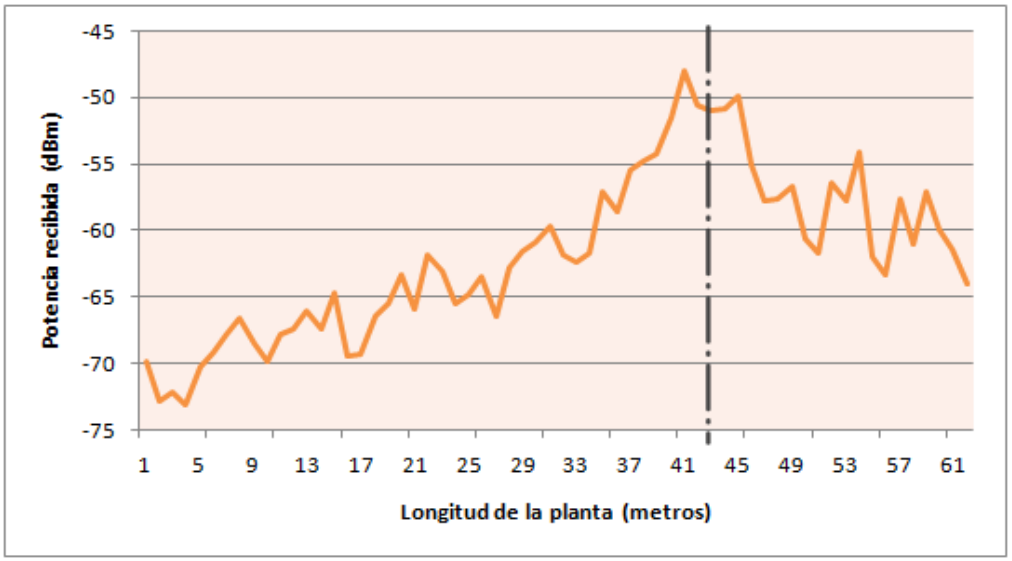


Figura 8.5: Relación entre potencia recibida y distancia del punto de acceso EDUROAM (81) a lo largo de todo el pasillo.

Para cada coordenada se obtiene un factor de atenuación, que se representa a continuación:

$$n = \frac{P_{TX} - P_{RX} - P_{fs}}{10 \cdot log(d)}$$

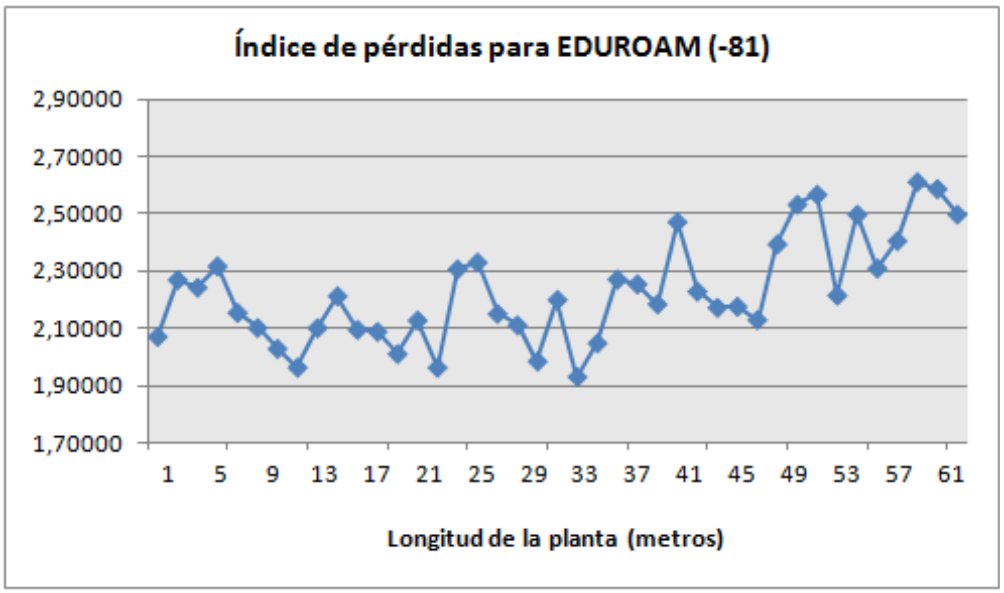


Figura 8.6: Valores del factor de atenuación del punto de acceso EDUROAM (81)

Se observa que el coeficiente 'n' varía sensiblemente debido a las fluctuaciones de la señal Wifi que se recibe del entorno.

Para el modelado matemático que representa el comportamiento de las señales de radiofrecuencia en zonas de interior se debe escoger un valor fijo para 'n' que se adecúe lo máximo posible a las lecturas de potencia realizadas a lo largo de la planta del edificio. Si se comienza con un valor de n=2'28, escogido por ser el valor promedio de la gráfica 8.6, la relación de potencias que se obtendría diferiría respecto a la experimental como se muestra en la figura 8.7.

El valor promedio 2'28 ofrece una gráfica que, a mayor distancia del punto de acceso ofrece unos valores por debajo de las medidas reales, mientras que en las cercanías del punto de acceso ofrece unos valores por encima de las medidas reales. Como el objetivo del modelado matemático es representar los valores empíricos con la mayor fidelidad, habrá que ajustar levemente dicho factor 'n' para igualar los datos lo máximo posible. En este caso, como se muestra en la figura 8.7, se ha decidido por un factor n=2'15, cuya gráfica sigue ofreciendo cierto error en las cercanías del punto de acceso, pero que a medida que se aleja el usuario se obtiene valores que compensan el error tanto positivo como negativo.

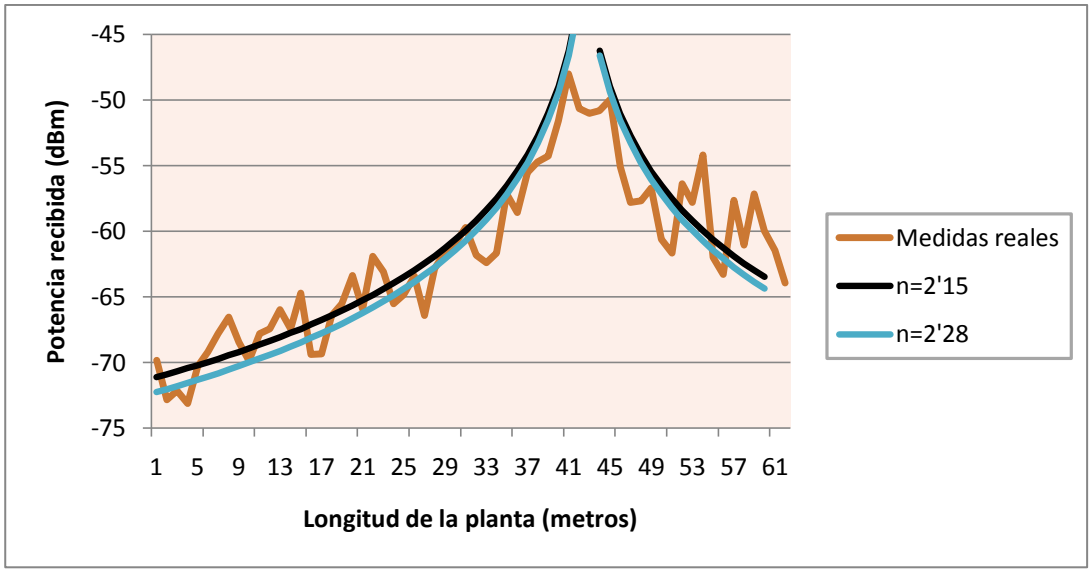


Figura 8.7: Estimación de la potencia según los valores del factor 'n' para el punto EDUROAM (81)

Otro factor a tener en cuenta a la hora de ajustar el factor 'n' es comprobar si se modela de igual forma el entorno hacia la izquierda del punto de acceso como hacia la derecha. Para ser exactos habría que obtener dos factores 'n' por cada punto de acceso, pero, obviamente, si la aplicación del PFC intenta localizar la posición del usuario, de ninguna manera podrá prever si el usuario se encuentra a uno u otro lado antes de ejecutar el algoritmo. Por tanto, habrá que estimar un solo factor 'n' que represente adecuadamente ambos lados del punto de acceso.

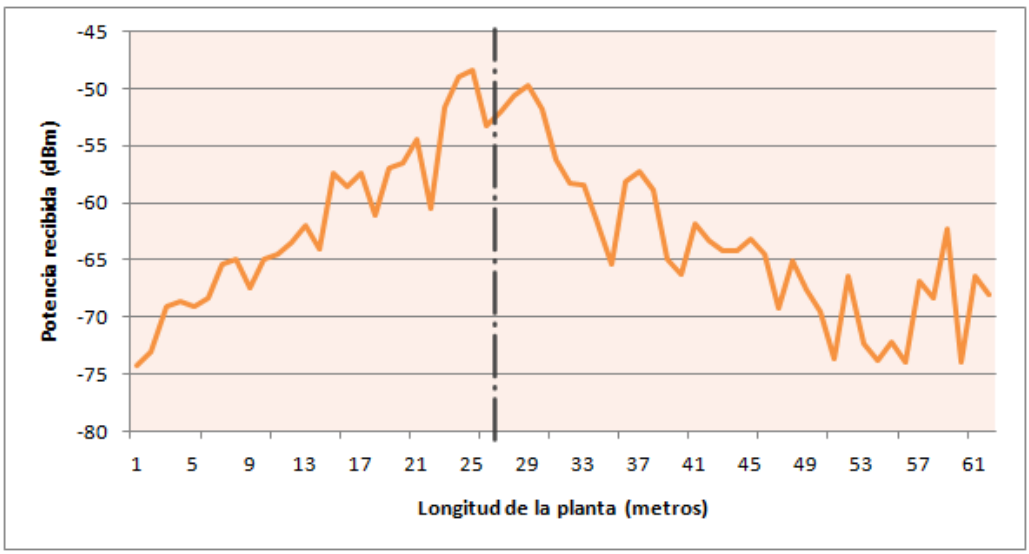


Figura 8.8: Relación entre potencia recibida y distancia del punto de acceso EDUROAM (D1) a lo largo de todo el pasillo.

Los datos para el segundo punto de acceso, cuya localización la representa la línea negra vertical, son los que se muestran en la figura 8.8. Y sus valores del factor de atenuación son:

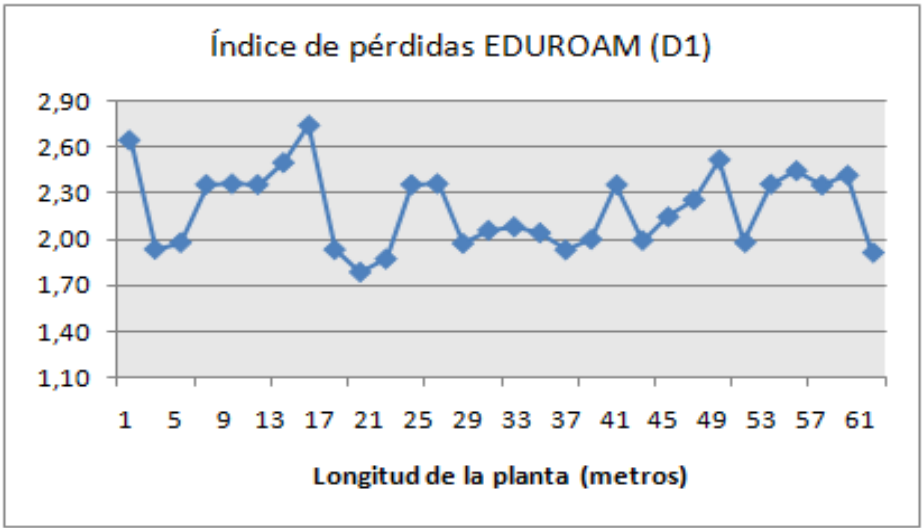


Figura 8.9: Valores del factor de atenuación del punto de acceso EDUROAM (D1)

Como las variaciones del factor de atenuación dependen de las fluctuaciones de la señal Wifi, el valor promedio no suele ajustarse a la situación real. Por tanto, habrá que ajustar manualmente dicho factor de atenuación hasta que se consiga la máxima fidelidad en los resultados obtenidos experimentalmente (figura 8.8). Partiendo del valor promedio de la gráfica 8.9, n=2’2, se ha ajustado manualmente hasta obtener una gráfica que compense todos los errores posibles para obtener máxima fidelidad de los resultados, factor n=2’35 (ver figura 8.10).

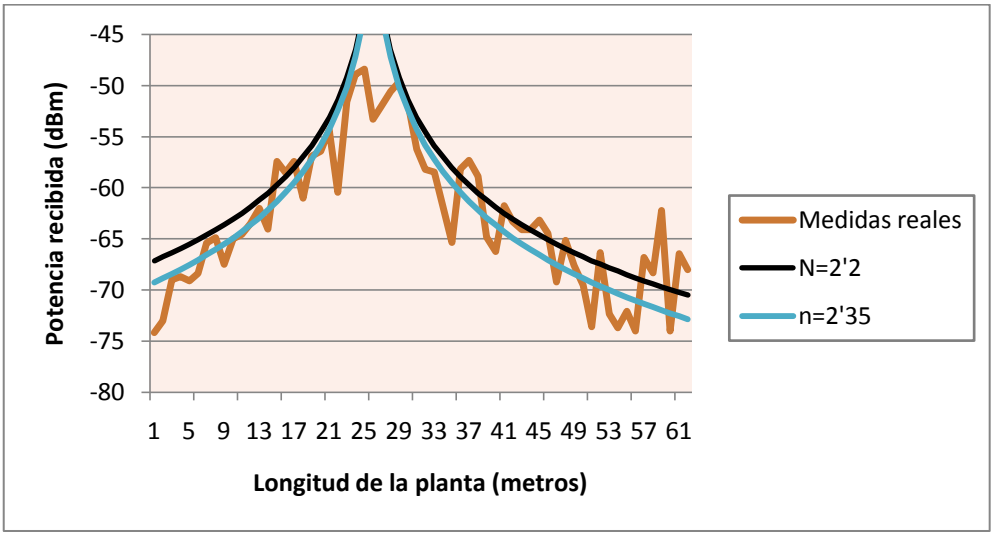


Figura 8.10: Estimación de la potencia según los valores del factor ‘n’ para el punto EDUROAM (D1)

8.2.4 Cálculo de las distancias

Cuando se ha obtenido el valor ideal del factor de atenuación ya se podrá estimar la distancia en metros conociendo un nivel de potencia determinado:

$$d = log^{-1}\left(\frac{P_{TX} - P_{RX} - P_{fs}}{10 \cdot n}\right)$$

Para estimar si el usuario se encuentra dentro de un despacho o en visión directa, la aplicación comprobará un valor umbral configurado por el propio usuario. Si la aplicación recibe menor potencia que dicho valor umbral indicará que el usuario está tras una, o varias, paredes. A partir de ese valor umbral, la aplicación calculará la distancia aplicando el segundo modelo de propagación de señales, que corresponde a la misma fórmula anterior a la que se le añade un término aditivo que corresponde a las pérdidas por absorción de las paredes:

$$P_{TX} - P_{RX}\ (dB) = P_{fs} + 10 \cdot log_{10}(d) + \sum AtenuaciónMuros$$

El valor umbral decidido para este entorno es -70 dBm, ya que es la potencia que se recibe en los límites del pasillo según la gráfica representada por los factores 'n' respectivos. A partir de este valor siginificará que el usuario se encuentra en alguno de los despachos o laboratorios.

El factor de atenuación de muros que aparece en la fórmula anterior se obtiene midiendo la potencia recibida detrás de una pared y la potencia recibida en visión directa a la misma distancia. Se restan ambos valores y el resultado será la atenuación debida a muros.

Puntos de acceso	Atenuación asociada a la pared (dB)
EDUROAM (81)	13
EDUROAM (D1)	13

Tabla 8.2: Atenuación sufrida por las señales de cada router Wifi debida a la pared.

Por tanto la ecuación para hallar la distancia quedaría así:

$$d = log^{-1}\left(\frac{P_{TX} - P_{RX} - P_{fs} - \mathbf{13}}{10 \cdot n}\right)$$

Con todos estos valores de configuración ya se podrá hacer uso de la aplicación desarrollada para este PFC.

8.3 Entorno real

Para estudiar el comportamiento y la efectividad del sistema de localización que presenta este PFC se ha recorrido el pasillo y se ha calculado la posición en ciertas coordenadas elegidas al azar. Como el entorno de trabajo sólo posee dos puntos de acceso sólo se podrá ejecutar el algoritmo de bilateración.

En una primera fase de pruebas, tras realizar un barrido por todo el pasillo, el 60% de los resultados obtenidos estaban localizados fuera del mapa. En este momento la aplicación no corregía las posiciones estimadas si se encontraban fuera de los límites del mapa; con lo que se modificó el algoritmo para ajustar dichos resultados y trasladarlos a una zona en el interior, como se describió al final del apartado 7.5.

Una vez ajustado el algoritmo se vuelve a probar la aplicación en el entorno de trabajo, como se muestra en la figura 8.11. El punto de color rojo indica la posición real. Los triángulos naranjas indican la posición de los dos routers Wifi.

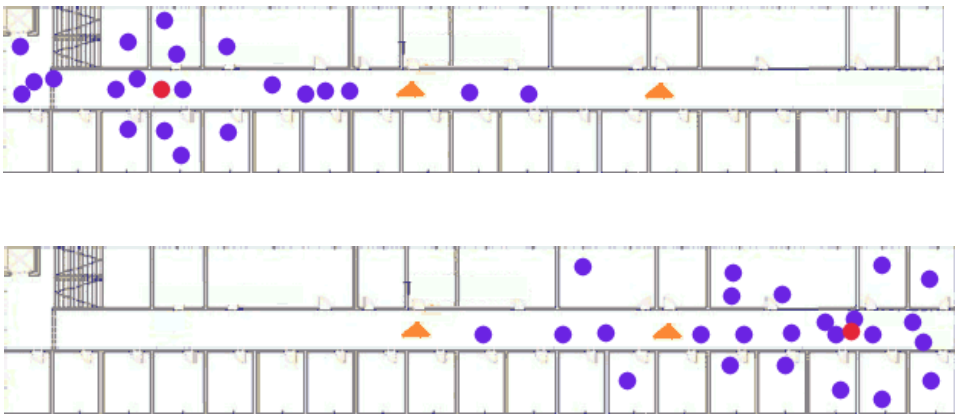


Figura 8.11: Dos ejemplos de localización tras ajustar los resultados a puntos en el interior del mapa.

Viendo que el error cometido llega a ser hasta la mitad de la longitud del mapa hubo que confirmar que el código del algoritmo estuviera bien escrito. Para ello se introdujeron manualmente valores de potencias de señal y se comprobó que los resultados obtenidos por el algoritmo eran los esperados.

Por tanto, sólo faltaba comprobar *in-situ* la lectura de potencias de señal que el dispositivo estaba recibiendo para ver si concordaba con los datos obtenidos en el estudio previo del edificio. Se descubrió un fenómeno que no se había tenido en cuenta: el efecto 'pantalla' del usuario sobre la antena Wifi del dispositivo. En una misma coordenada existía una diferencia en los valores de potencia de hasta 15 dB entre las distintas orientaciones en las que se encontraba el usuario. Esta gran diferencia implica que, al sustituir en la fórmula de estimación de la distancia un valor de -50 dBm como potencia recibida y, posteriormente, un valor de -60 dBm, la diferencia de ambas distancias era de 30 metros. Esto significa que por el solo hecho de que el usuario se girase, el sistema de localización lo situaría 30 metros alejados del punto real. Por tanto, para obtener una estimación aceptable de la distancia hacia el punto de acceso, se estimó que el usuario debería girar 360 grados y, posteriormente, calcular el valor promedio de las muestras recibidas durante ese giro.

De esta manera, para la siguiente fase de pruebas se modificó el algoritmo para añadir una memoria temporal (un array de 10 *integers)* que guardase las últimas 10 muestras recibidas por el dispositivo. De esta forma, la aplicación software calcularía el valor promedio de las potencias recibidas y, con dicho valor, estimaría la distancia hacia cada punto de acceso.

En esta última fase de pruebas se ha recorrido todo el pasillo y se ha calculado la posición del usuario cada cinco metros. En cada posición el usuario ha dado varias vueltas sobre sí mismo hasta obtener en pantalla un mínimo de diez resultados de posicionamiento. En las figuras 8.12 y 8.13 se muestran algunos resultados gráficos de esta última fase. Se ha añadido una plantilla en donde cada cuadrícula representa un metro por un metro para una mejor lectura del error cometido.

Figura 8.12: Resultados de la localización para la coordenada (10,5'5) usando el valor promedio de las potencias recibidas.

Distancia al punto exacto	Porcentaje de puntos
x < 2 (3’25% error)	30%
2 < x < 3 (4’9 % error)	6%
3 < x < 4 (6’5 % error)	16%
x > 4 metros	48%

Tabla 8.3: Dispersión de los resultados obtenidos para la coordenada (10,5’5)

El porcentaje de error que aparece en las tablas de dispersión se obtiene a partir de la longitud total de la planta (61’5 metros) respecto los errores de 2, 3 y 4 metros.

Figura 8.13: Resultados de la localización para la coordenada (55,5’5) usando el valor promedio de las potencias recibidas.

Distancia al punto exacto	Porcentaje de puntos
x < 2 (3’25% error)	14%
2 < x < 3 (4’9 % error)	26%
3 < x < 4 (6’5 % error)	30%
x > 4 metros	30%

Tabla 8.4: Dispersión de los resultados obtenidos para la coordenada (55,5’5)

Tras recopilar todas las localizaciones obtenidas a lo largo de todo el pasillo y el error cometido en cada una, en la gráfica 8.14 se muestran dichos resultados:

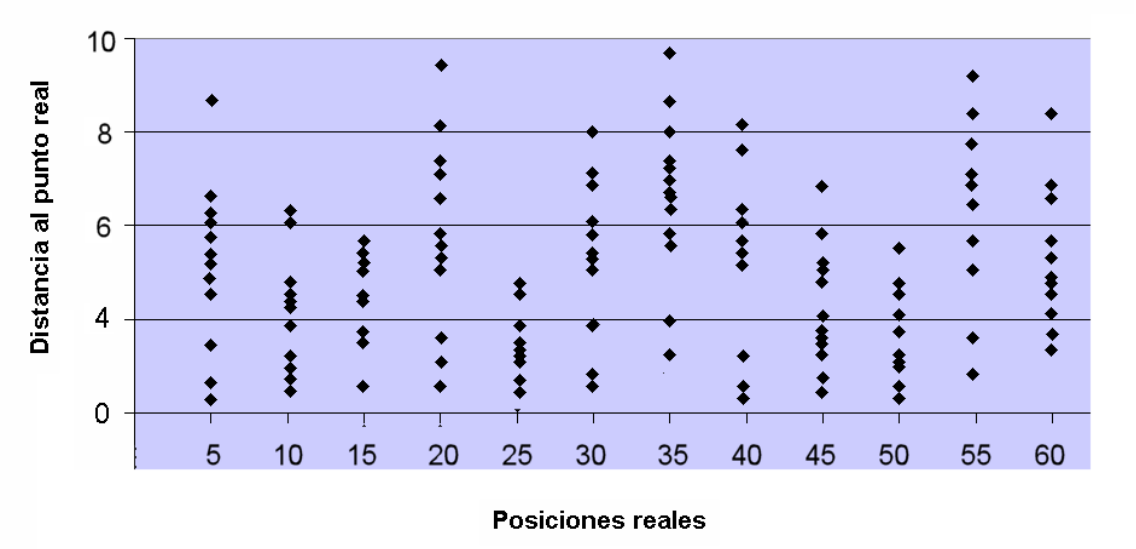


Figura 8.14: Dispersión de los datos a lo largo de la planta del edificio.

En la tabla 8.5 se han clasificado dichos resultados según el error cometido:

Distancia al punto exacto	Porcentaje de puntos
x < 2 (3'25% error)	10%
2 < x < 3 (4'9 % error)	12%
3 < x < 4 (6'5 % error)	24%
x > 4 metros	56%

Tabla 8.5: Dispersión de los puntos de localización tras realizar un recorrido por toda la planta.

Se comprueba que sólo un 22% de los puntos obtenidos se encuentran en un rango inferior a tres metros. Además, el usuario deberá estar continuamente girando sobre sí mismo para obtener dichos resultados y que se asemejen lo más posible al estudio previo del entorno porque, desde el momento en que se detenga el usuario, el dispositivo estará recibiendo, o bien, una potencia muy superior por estar en visión directa con el router Wifi, o bien, una potencia muy inferior porque el usuario está intermediando entre el router Wifi y el dispositivo móvil.

Estudios en los despachos

Para el estudio de la localización en los despachos se ha ejecutado la aplicación en varios despachos seleccionados al azar siguiendo la misma

metodología del apartado anterior. En la figura 8.15 se muestran dos ejemplos gráficos de localización:

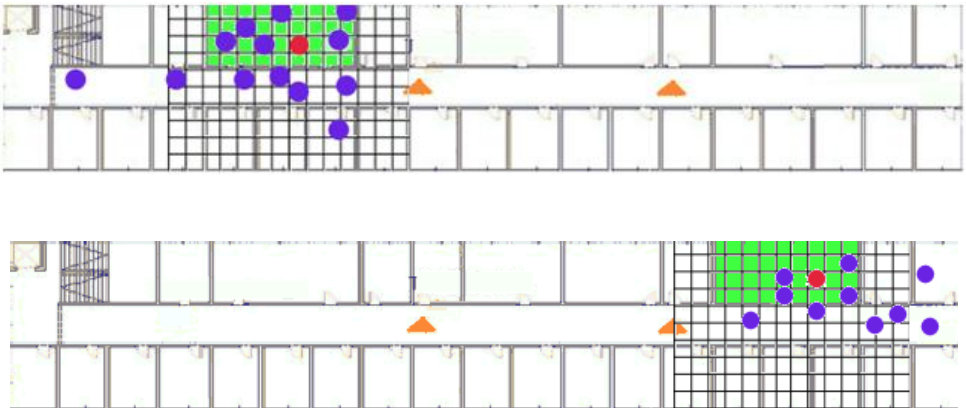


Figura 8.15: Resultados de localización en dos de los laboratorios del departamento.

Distancia al punto exacto	Porcentaje de puntos
x < 2 (3'25% error)	5%
2 < x < 3 (4'9 % error)	12%
3 < x < 4 (6'5 % error)	20%
x > 4 metros	63%

Tabla 8.6: Dispersión de los puntos de localización tras realizar un recorrido por cuatro despachos.

Se comprueba que la precisión es peor que en los casos de visión directa, sin embargo, en dos laboratorios cuyas dimensiones eran de diez metros de largo, como se muestra en la figura 8.15, casi el 50% de los puntos mostraban al usuario dentro del laboratorio.

Conclusiones

Se puede observar claramente que los resultados obtenidos no son aceptables en este proyecto de localización en interiores debido a la variabilidad en la recepción de señales. Sin embargo, dichos resultados son los obtenidos solamente para este entorno en particular, donde sólo existen dos puntos de acceso de referencia, por lo que sólo se estaría evaluando el algoritmo de la bilateración.

En el siguiente apartado, por tanto, se intentará realizar un análisis del resto de los algoritmos desarrollados para este PFC que necesiten de, como mínimo, tres puntos de acceso. Para ello se realizarán simulaciones en un ordenador en donde se le introducirán nuevos puntos de acceso al entorno.

8.4 Comparativa de los algoritmos de localización

En este apartado se ofrecerán resultados de simulaciones realizadas en un ordenador en el que se han introducido los datos manualmente: no han sido ejecutadas en un escenario real. De esta forma se puede, por ejemplo, introducir el error en la señal que el usuario desee, o añadir múltiples puntos Wifi en la simulación y comprobar si mejora la precisión del sistema, cuando en el entorno real estudiado sólo existen dos puntos de acceso.

Para el análisis de los distintos algoritmos de localización y comprobar cuál ofrece mejores prestaciones se estudiarán los siguientes criterios: consumo de CPU, precisión y robustez.

8.4.1 Consumo de CPU

El criterio más importante a la hora de realizar una comparativa de aplicaciones para dispositivos móviles se basa en el consumo de batería. Si se disponen de varias aplicaciones que cumplan correctamente los objetivos de un mismo proyecto, un factor importante a la hora de decantarse por uno u otro programa será el consumo de batería de cada uno.

Para calcular dicho consumo de CPU, en primera instancia se pensó usar una metodología basada en el número de instrucciones de máquina que utilizaba cada algoritmo. El inconveniente de esta metodología es que el comando de C++ que mide las instrucciones ("rdtsc") contabiliza todas las instrucciones que se ejecuten sin discernir a qué procesos pertenece cada una. Por tanto, en un sistema operativo multitarea como el Windows Mobile, donde se ejecutan tanto instrucciones de nuestro proceso como de otros procesos en segundo plano y servicios internos del sistema operativo, el contabilizar las instrucciones de nuestro proceso sería una tarea ardua.

La metodología elegida, por tanto, es la medición del tiempo que tarda el algoritmo en ejecutarse. Por un lado tenemos la función de C++ "`GetTickCount ()`" que nos devuelve los milisegundos transcurridos desde que Windows se inicia. En

esta cuenta, sin embargo, siguen contabilizándose todos los procesos simultáneos que se encuentren en ejecución.

Por tanto, se ha decidido utilizar el comando "time" de Linux. La salida de "time" ofrece tres valores en pantalla: real, sistema y usuario.

```
$ time script.sh trilateracion
real        0m4.103s
user        0m1.720s
sys         0m1.734s
$
```

"user": es el tiempo transcurrido en ejecutar nuestro código en modo-usuario.

"sys": es el tiempo usado por la CPU en ejecutar el código en modo-*kernel*, como las llamadas de sistema o accesos al disco duro.

"real": contabiliza el tiempo que ha pasado desde que se inició el comando hasta su finalización (aquí se incluyen los dos valores anteriores más las esperas de las instrucciones de E/S, que dejan a la CPU en *idle*).

Por tanto, el tiempo en el que se basará este PFC para realizar el *benchmarking* entre algoritmos será el "user", ya que es el tiempo total de CPU dedicado exclusivamente a nuestro proceso.

Como el procesamiento de los algoritmos era menor al milisegundo, el comando 'time' no podía ofrecer resultados concluyentes. Como no existe ninguna otra herramienta que pueda medir el tiempo de proceso menor de un milisegundo, la solución adoptada ha sido repetir el mismo proceso múltiples veces y contabilizar el tiempo total de cómputo. En este PFC se repetirá cinco mil veces cada algoritmo y se calculará el tiempo usando el comando "time" de Linux. La sintaxis del comando "time" es: `time script.sh [fichero_ejecutable]` Donde "script.sh" es el script que marca el bucle de las cinco mil repeticiones:

```
#!/bin/sh
for ((i=0;i<5000;i++)) do
      echo `./$1`
done
```

La instalación de Linux se ha hecho a partir de una máquina virtual creada por VMWare Workstation 6.0 y la distribución Debian 5.0. Las pruebas se han

realizado ejecutando la máquina virtual sobre un equipo Core-i7 860 con sistema operativo Windows 7.

La clasificación se hará según el número de puntos de acceso involucrados en la simulación, con el fin de comprobar si el uso de CPU de los algoritmos mantiene una evolución constante a mayor número de puntos de acceso, o si, por el contrario, algún algoritmo aumenta considerablemente su cómputo.

Se comenzará con tres puntos de acceso (P.A.) en donde participarán todos los algoritmos. A partir de cuatro o más, se dejará de lado la trilateración, ya que es un algoritmo que se ejecuta, exclusivamente, con tres puntos de acceso. En la tabla 8.7 se muestran los resultados:

	3 P.A.	**5 P.A.**	**10 P.A.**	**Promedio**	**Rapidez respecto MinMax**
Trilateración	1'57 s.	--	--		
Min-Max	1'68 s.	1'75 s.	1'80 s.	1'74 s.	0%
Multi-lateración	1'86 s.	1'99 s.	2'10 s.	1'98 s.	15% + lento
Nelder-Mead	2'08 s.	2'26 s.	2'39 s.	2'24 s.	29% + lento

Tabla 8.7: Tiempo de procesamiento de los algoritmos para 5000 repeticiones.

En el caso de que existan tres puntos de acceso el algoritmo de trilateración es el más rápido, siendo un 7% más rápido que MinMax y un 30% más rápido que Nelder-Mead.

Para el resto de casos con más P.A. el algoritmo MinMax es el más rápido, siendo un 15% más rápido que la multilateración y un 29% más rápido que Nelder-Mead. Además, el tiempo consumido por cada algoritmo aumenta linealmente a mayor cantidad de puntos de acceso sin que en ningún momento exista un incremento exagerado de CPU.

Estos datos corroboran la cantidad de operaciones algebraicas necesarias para cada algoritmo: MinMax se basa en sumas y restas, Multilateración se basa en productos matriciales y NelderMead se basa en múltiples iteraciones para encontrar el mínimo de una función.

8.4.2 Precisión

Esta prueba consiste en conocer la precisión de los cálculos matemáticos del algoritmo sin depender de fluctuaciones de las señales Wifi. Esta prueba no ofrece resultados muy relevantes, ya que simplemente sirve para encontrar los límites de precisión de un algoritmo. Por ejemplo, si en esta prueba un algoritmo obtiene un error de 5 metros, en una situación real, donde las medidas de señales varían continuamente, no se puede esperar que su precisión sea mejor de 5 metros.

La metodología usada para analizar la precisión consiste en introducir manualmente una coordenada del mapa, calcular la distancia hacia cada punto Wifi y ejecutar los algoritmos de localización. La función *'Distancia'* medirá la distancia exacta entre las coordenadas introducidas manualmente y cada punto de acceso:

```
double Distancia(double x1, double y1, double x2, double y2)
{
   return (sqrt((x2-x1)*(x2-x1) + (y2-y1)*(y2-y1)));
}
```

Se ha realizado esta prueba añadiendo nuevos puntos de acceso aleatoriamente, hasta un máximo de diez, para comprobar cómo mejora la precisión del sistema. Las coordenadas de todos los puntos Wifi y sus distancias hacia el punto introducido por el usuario se han definido con el siguiente código:

```
// Se pregunta al usuario por las coordenadas (xx,yy)
cout << "\n Coordenada X: ";cin >> xx;
cout << "\n Coordenada Y: ";cin >> yy;

// Localizacion de los puntos Wifi y distancias exactas
// calculadas según las coordenadas introducidas.
x1=10.8; y1=-4.5; d1=Distancia(x1,y1,xx,yy);
x2=26.0; y2=-5.5; d2=Distancia(x2,y2,xx,yy);  // EDUROAM (D1)
x3=60.6; y3=-6.8; d3=Distancia(x3,y3,xx,yy);
x4=2.0; y4=-2.1; d4=Distancia(x4,y4,xx,yy);
x5=35.6; y5=-4.0; d5=Distancia(x5,y5,xx,yy);
x6=17.5; y6=-5.2; d6=Distancia(x6,y6,xx,yy);
x7=23.2; y7=-8.5; d7=Distancia(x7,y7,xx,yy);
x8=43.0; y8=-5.5; d8=Distancia(x8,y8,xx,yy);  // EDUROAM (81)
x9=52.9; y9=-3.9; d9=Distancia(x9,y9,xx,yy);
x10=42.2; y10=-7.0; d10=Distancia(x10,y10,xx,yy);
```

Si se trasladan dichos puntos al mapa la distribución de puntos Wifi queda tal que así:

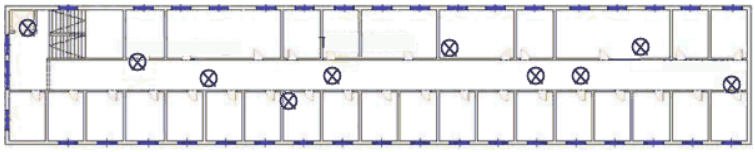


Figura 8.16: Distribución de puntos Wifi añadidos aleatoriamente para la simulación.

Se ha mostrado la distribución de los puntos Wifi en el entorno estudiado en este PFC, pero la simulación no depende del entorno donde se aplique ni de los muros que intermedien porque la simulación parte de una distancia exacta hacia cada punto Wifi. Los muros sólo afectan a la hora de estimar la distancia según la potencia recibida.

Se ha decidido realizar el estudio para unas coordenadas elegidas aleatoriamente, como son X=17 e Y=5. En la figura 8.17 se muestran los resultados. Se aprecia que los algoritmos basados en la *lateración* (como son bilateración, trilateración y multi-lateración) ofrecen un error cero en todos los casos.

En el caso de MinMax nos encontramos con un leve porcentaje de error que, además, se mantiene muy constante. Esto es un comportamiento importante que indica que a mayor cantidad de puntos Wifi no mejoraremos sustancialmente la precisión, teniendo en cuenta que estamos simulando un entorno donde las señales recibidas no varían en absoluto: se recibe una potencia fija de cada punto de acceso con lo que algoritmo está trabajando con distancias exactas hacia cada uno de ellos.

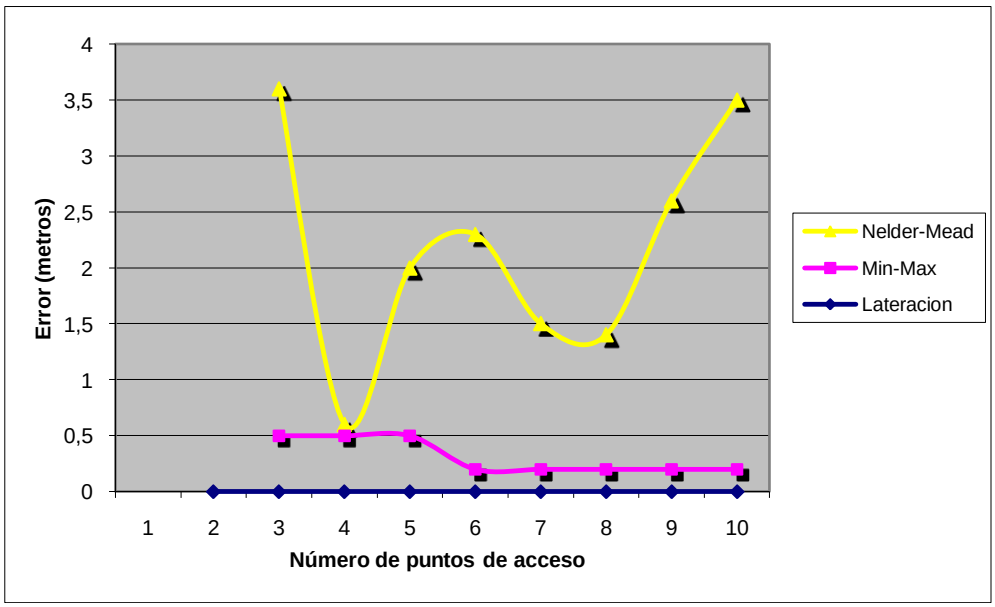


Figura 8.17: Error de precisión en un punto seleccionado al azar.

Por último, se encuentra el algoritmo de Nelder-Mead que no ofrece una buena precisión, con un error que varía entre 0'5 y 3'5 metros de forma arbitraria, sin depender de la cantidad de puntos de acceso.

8.4.3 Robustez

Como el entorno de estudio de este PFC sólo posee dos puntos de acceso para realizar la localización, no ha sido posible probar todos los algoritmos, ya que estos necesitan de tres puntos de acceso como mínimo para ser ejecutados. Por tanto, para dichos algoritmos se han realizado pruebas simuladas para medir la precisión que ofrecerán en un entorno cuasi-real, esto es, en donde se estiman las distancias hacia cada punto de acceso de forma variable. La metodología de este apartado ha sido la siguiente:

- De cada punto de acceso se calcula la distancia exacta a la que se encuentra el usuario, como en el apartado anterior.
- A dicha distancia obtenida se le introduce un error de un metro. Dicho desvío puede ser tanto positivo (se estima que el usuario se encuentra más lejos del punto de acceso), como negativo (el usuario está más cerca del punto de acceso). En cada prueba ejecutada se define de forma arbitraria dicha elección, de forma que una de las veces el error será negativo y otras veces, el error será positivo, con lo que el algoritmo ofrecerá resultados diferentes cada vez que se ejecute.
- Se ha ejecutado cada algoritmo cincuenta veces y se ha hallado el valor promedio del resultado.
- Se ha repetido la operación con más puntos de acceso para averiguar si mejora la precisión.
- El proceso anterior se ha repetido aumentando el error de las distancias con los siguientes valores: 2, 3, 4 y 5 metros.

Un ejemplo de código que muestra el error introducido en cada punto de acceso es el que sigue:

```
cout <<  "Introduce error de RSSI (metros): "; cin >> error;

// Modificamos semilla para obtener números realmente
// aleatorios en cada iteración.
srand ( time(NULL) );

a=rand() % 2;  // Integer "a" valdrá 0 ó 1.
x1=10.8; y1=-4.5; d1=Distancia(x1,y1,xx,yy)+error-(2*error*a);
```

```
a=rand() % 2; // Nuevo valor de "a" para el siguiente P.A.
x2=26.0; y2=-5.5; d2=Distancia(x2,y2,xx,yy)+error-(2*error*a);
```

Es decir, el valor de cada distancia calculada es:

`Distancia+error` ó `Distancia-error`

Las pruebas se han realizado en un sistema Linux porque las compilaciones del lenguaje C/C++ son mucho más fáciles y rápidas de realizar al utilizar una simple línea de comandos como es: `g++ -o [salida] [fichero.cpp]`

Una vez recopilados los datos se detallan a continuación las gráficas obtenidas para cada algoritmo. La figura 8.18 muestra los resultados para la multilateración (en donde E=x significa que a la estimación de la distancia se le han añadido 'x' metros de error):

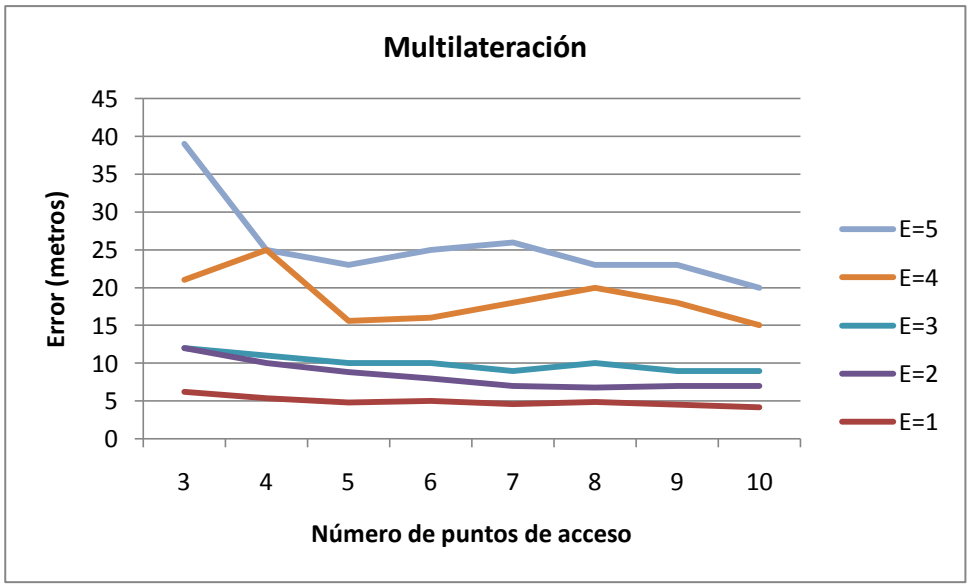


Figura 8.18: Precisión de la Multi-lateración frente a variaciones de señal.

La multilateración es la mayor afectada por la variación de potencias, esto es, el algoritmo es muy sensible a errores. Introduciendo un error de un solo metro (E=1) en la estimación de la distancia, se obtiene una localización que se aleja cinco metros de la posición real del usuario.

Si se intenta mejorar la precisión añadiendo nuevos puntos de acceso se comprueba que la mejora es inapreciable; salvo en el caso que el error introducido sea cada vez mayor, en donde con E=5 metros se consigue mejorar la precisión desde 40 hasta 25 metros añadiendo un solo punto de acceso. Sin embargo, un sistema de localización en interiores con una precisión de 25 metros no es viable.

Prácticamente todos los resultados obtenidos han sido puntos fuera del mapa del edificio, por tanto, cabría la posibilidad de mejorar el resultado sustancialmente si se modificara dicho punto obtenido y se desplazara hacia una zona dentro del mapa.

En el caso de este PFC se ha ajustado de la siguiente forma:

- Si la coordenada X del resultado es menor que cero, se modifica dicho valor a 1 metro (se localiza al usuario en el extremo izquierdo del edificio).
- Si la coordenada X del resultado es mayor que 61, se modifica dicho valor a 60 metros (localiza al usuario en el extremo derecho del edificio).
- Si la coordenada Y del resultado es menor que cero, se modifica dicho valor a 1 metro (se localiza al usuario en uno de los despachos de la fila superior).
- Si la coordenada Y del resultado es mayor que 10, se modifica dicho valor a 9 metros (se localiza al usuario en uno de los despachos de la fila inferior).

Realizando de nuevo la misma simulación se obtienen los siguientes resultados en la figura 8.19:

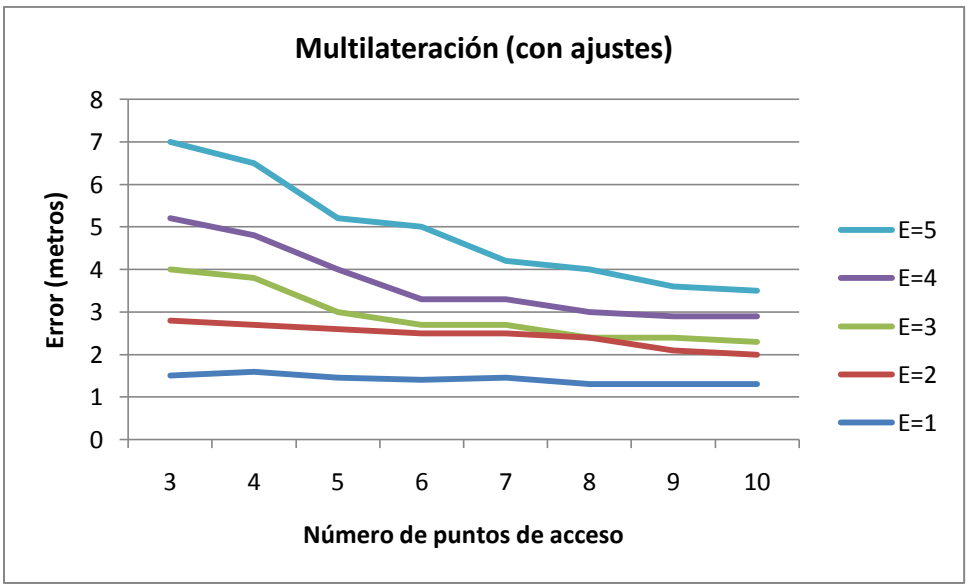


Figura 8.19: Precisión de la Multi-lateración en donde se han ajustado sus resultados a puntos en el interior del mapa.

En este caso se ha reducido el error en más de un 80% en el caso donde se comete un error de cinco metros en la estimación de distancias hacia cada punto de acceso. Se comprueba que la bilateración en el entorno real se ha comportado de forma similar; esto es porque los tres métodos (bi-, tri- y multi-) se basan en la *lateración* (intersección de circunferencias).

En el caso de Min-Max se comprueba un comportamiento similar respecto la simulación en donde no se ha introducido ningún error en la estimación de distancias: se obtiene una precisión relativamente alta e independiente del número de puntos de acceso que existan alrededor (ver figura 8.20). Con lo que se puede denominar a este algoritmo como preciso y muy robusto frente a variaciones de señal.

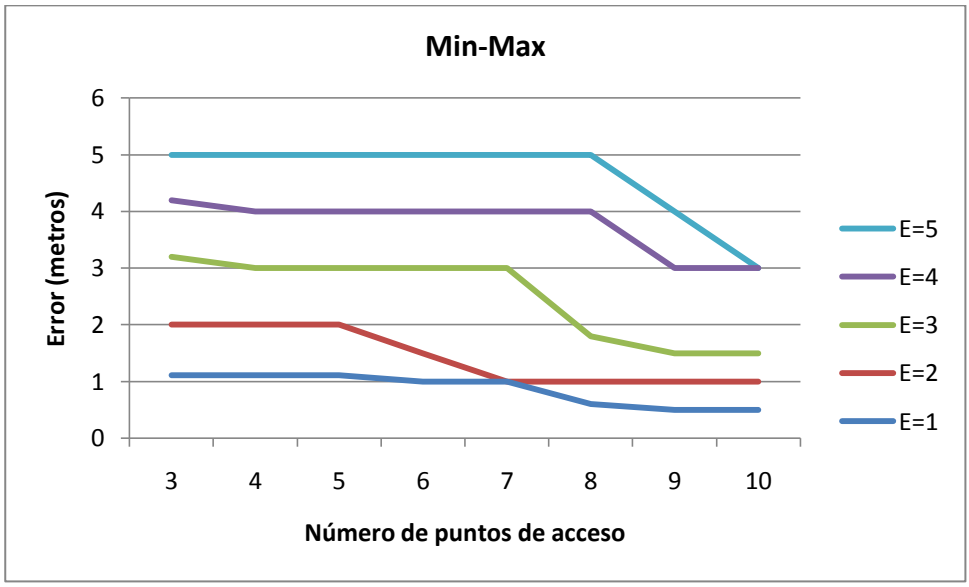


Figura 8.20: Precisión de Min-Max frente a variaciones de señal.

Si se desea aumentar la precisión del sistema instalando nuevos puntos Wifi la inversión necesaria no sería nada aconsejable, ya que se necesitaría un mínimo de nueve puntos Wifi para conseguir mejorar la precisión un solo metro.

Para el sistema de Nelder-Mead (ver figura 8.21) se confirman las mismas observaciones que en el caso de no introducir errores en la señal: se obtiene una precisión aleatoria, y no sigue esquema alguno cuando se introducen nuevos puntos de acceso que mejoren su precisión.

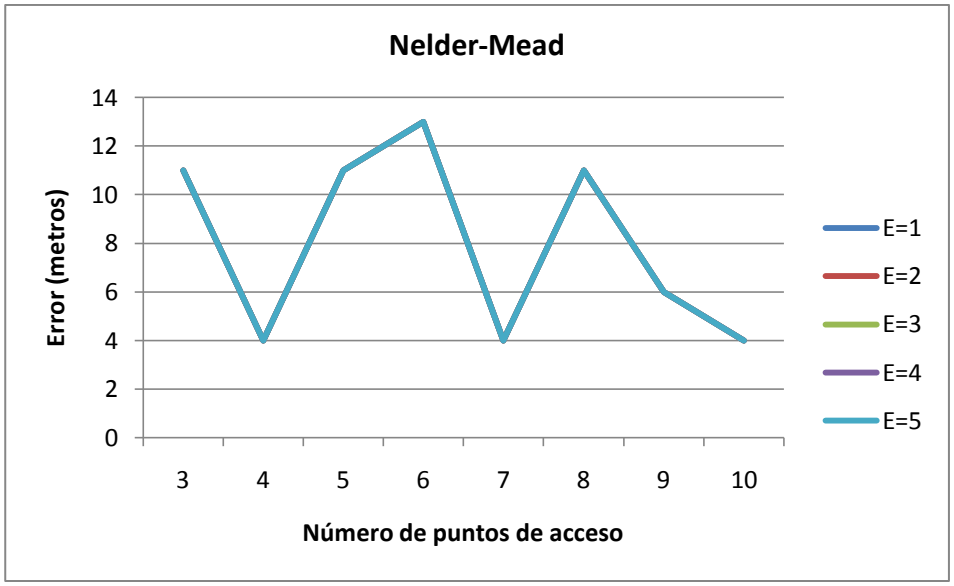


Figura 8.21: Precisión de Nelder-Mead frente a variaciones de señal.

Sin embargo, observando la gráfica se comprueba que se obtiene la misma precisión sea cual sea el error de señal introducido: todas las gráficas están superpuestas en una sola.

Este fenómeno se debe a que Nelder-Mead no trabaja sobre las distancias que recibe el usuario de cada punto de acceso, sino que trabaja directamente sobre la fórmula del residuo, cuyos datos no son variables que dependan de una estimación de distancias, sino que son datos fijos y conocidos: la localización de cada punto de acceso y un punto inicial desde donde parte le ejecución del algoritmo. Si siempre se le introduce el mismo valor inicial siempre llegará al mismo *valle* en la fórmula del residuo.

Por tanto, se puede confirmar que el algoritmo desarrollado por Savvides, llamado Min-Max (o *bounding-box*) es el que mejores prestaciones ofrece en cualquier situación simulada, y además, con el apoyo de sólo tres puntos de acceso instalados en la zona. Tampoco necesita de ajustar sus resultados a un entorno determinado como sí ocurre con los algoritmos basados en la lateración.

En la siguiente tabla 8.8 se resumen los datos de precisión de cada algoritmo en el caso que se cometa un error de +/- 5 metros al estimar las distancias hacia cada punto de acceso.

	3 P.A.	5 P.A.	10 P.A.	Promedio (m.)
Lateración	7	5'2	3'5	5'2
MinMax	5	5	3	4'3
NelderMead	11	11	4	8'7

Tabla 8.8: Precisión de los algoritmos cuando se comete un error de 5 metros en la estimación de la distancia.

CAPÍTULO 9

CONCLUSIONES Y FUTURAS LÍNEAS DE TRABAJO

9.1 Conclusiones

Una de las fases más importantes a la hora de implementar un proyecto de localización es decidir qué algoritmo ofrece los mejores resultados para el entorno de trabajo. En este sentido habría que definir el criterio primordial en el que se basa el sistema a implementar: si se desea un sistema que ofrezca una alta precisión, o bien, un sistema robusto frente a variaciones de la señal, o bien, si se desea un equilibrio entre el consumo de CPU y el resto de factores. Puede ocurrir que un algoritmo muy preciso no fuera útil al usuario si fuera muy sensible a las variaciones de las señales que se reciben. O, también, un algoritmo preciso y que además tenga la ventaja que sea robusto, si consumiera los recursos del dispositivo durante quince minutos podría no cumplir los objetivos del proyecto a implementar.

Este PFC ha intentado ofrecer una visión de algunos de esos algoritmos de localización y, además, se ha desarrollado un sistema que los ponga en práctica. Los algoritmos desarrollados fueron: MinMax, minimización del residuo y aquellos basados en la *lateración,* como son la bilateración, trilateración y multi-lateración.

Se han desarrollado dos aplicaciones software: 'WifiLocaliza' y 'WifiReader'. La primera realiza la función de sistema de localización; la segunda es una aplicación de apoyo al usuario para reducir el tiempo empleado en realizar el estudio previo del entorno (fase 'offline'). Ambas están basadas en la aplicación de código abierto *PeekPocket* que utiliza el driver NDIS para comunicarse con el interfaz de red. Es una aplicación en lenguaje VisualC++, que hubo que examinar detenidamente para conocer sus métodos, clases y estructuras implementadas.

Durante la fase 'offline' del proyecto hubo que enfrentarse a problemas en la lectura de potencia de las señales (unas intrínsecas al hardware y otras inherentes al comportamiento de las señales electromagnéticas) y se implementaron soluciones en el sistema que minimizaran dichos errores.

Tras analizar los algoritmos desarrollados para este PFC, los cuales se basan en la lectura de potencia de señal para estimar las distancias, se confirma claramente que el algoritmo propuesto por Savvides (MinMax) es el que mejores prestaciones ofrece al usuario en cualquier ámbito (precisión, consumo de recursos y robustez frente al ruido). Aunque dichos resultados se simularon en un ordenador, los errores obtenidos por la multi-lateración (entre 1 y 7 metros para tres puntos de acceso) se ajustan correctamente a los obtenidos en el entorno real por la bilateración (entre 1 y 10 metros), con lo que se cumpliría la evolución de mejor precisión a mayor cantidad de puntos de acceso.

Los algoritmos basados en la lateración adolecen de un error considerable cuando la recepción de señales varía sensiblemente, así que habría que descartar el

uso de estos algoritmos en dispositivos móviles y probarlos en dispositivos mayores como los de un ordenador portátil, en donde el efecto *pantalla* del usuario podría reducirse considerablemente.

Sin embargo, se ha de observar que un sistema de localización basado en la lectura del RSSI no es tan fiable para un entorno que necesite de una efectividad continua a lo largo del tiempo: las señales de radiofrecuencia, con sus condiciones anisotrópicas, varían continuamente. Además, también dependen del mobiliario y las personas que circulen alrededor de la zona, con lo que si algún día se organiza algún evento que reciba decenas de personas en la zona de cobertura del sistema los resultados pueden que no sean satisfactorios.

9.2 Futuras líneas de trabajo

En cuando a la mejora del sistema de este PFC se podría implementar un sistema en donde el dispositivo móvil del usuario no sea quien detecte la potencia de señales recibidas, sino que sean los routers Wifi instalados en el entorno de trabajo. Con este sistema no se dependería del fabricante de la tarjeta inalámbrica del dispositivo móvil, con lo que no habría que realizar un estudio previo del edificio para cada modelo que existiese en el mercado [Web-14]. En este caso existiría un servidor adicional que leería la potencia de señal que reciben todos los puntos de acceso y sería el propio servidor quien realizaría los cálculos necesarios para la localización del usuario. Una vez obtenido el resultado se lo enviaría al software del dispositivo móvil para que lo mostrase en pantalla.

Una mejora adicional al software de este PFC sería la posibilidad de usar un algoritmo estadístico que funcionara en conjunción con uno de los algoritmos ya existentes como se comentó en el punto 4.7 ("Algoritmos estadísticos").

Otra mejora para el software de este proyecto sería poder abarcar todas las plantas del edificio y, más aún, todos los edificios del campus. El software tendría guardados los mapas de todos los edificios, y una vez recibiera las señales Wifi a su alrededor, detectaría en qué planta se encontraría el usuario teniendo en cuenta qué puntos Wifi recibe con más fuerza.

Otra mejora del software sería la posibilidad de ofrecer el resultado de tipo simbólico, es decir, no ofrecer exclusivamente una localización en un mapa, sino explicar al usuario en qué ámbito se encuentra. Por ejemplo, un resultado simbólico sería: *"Está usted junto al despacho XX del departamento YY del edificio de Telecomunicaciones".*

CAPÍTULO 10

BIBLIOGRAFÍA

10.1 Referencias Bibliográficas

[Fin03] K. Finkelzeller, "The RFID Handbook" Ed. John Wiley & Sons, 2003.

[Fis98] S. Fischer, H. Grubeck, A. Kangas, H. Koorapaty, E. Larsson, P. Lundqvist, "Time of Arrival Estimation of Narrowband TDMA Signals for Mobile Positioning" Personal Indoor and Mobile Radio Communications, 1998.

[Gor02] Gordon Goth, "Read it and WEP". News & Trends, IEEE Intenet Computing, vol. 6, nº 1, Enero/Febrero 2002.

[Har94] A. Harter and A. Hopper, "A Distributed Location System for the Active Office," IEEE Network, Enero 1994

[Hig01] Jeffrey Hightower y Gaetano Borriello, *"A Survey and Taxonomy of Location Systems for Ubiquitous Computing"*. Computer Science and Engineering, Universidad de Washington. Agosto, 2001.

[Kap02] S. Kapp. "802.11a: More Bandwidth without the Wires". IEEE Intenet Computing, vol. 6, nº 4, julio/agosto 2002.

[Maa97] H. Maass, "Location-Aware Mobile Applications based onDirectory Services," MobiCom '97, pp. 23-33, Septiembre 1997

[Nid06] Michael Nidd, Stephen Mann and Jay Black, "Using Ray Tracing for Site-Specific Indoor Radio Signal Strength Analysis", Computer Science Department, University of Waterloo, Canada, 2007.

[One71] R. ONeill, "Applied Statistics", Volumen 20, Número 3, 1971

[Par00] Paramvir Bahl y Venkata N. Padmanabhan, "RADAR: An In-Building RFbased User Location and Tracking System", Proc. IEEE Infocom, Marzo 2000.

[Ran10] Nick Randolph, David Gardner, Chris Anderson, Michael Minutillo, "Professional Visual Studio 2010". Ed. WROX, 2010.

[Sal00] Shaharuddin Salleh, Albert Y. Zomaya, Sakhinah Abu Bakar , "Computing for numerical methods using Visual C++". Ed. John Wiley and Sons, 2000.

[San07] Raúl Sánchez Vítores, "Sistemas de Localización en Interiores", revista Tendencias, Diciembre, 2007.

[Sav02] A. Savvides, H. Park, M. Srivastava, "The bits and flops of the N-hop multilateration primitive for node localization problems", First ACM International Workshop on Wireless Sensor Networks and Application. Año 2002

[She00] Sheethalnath, Praveen T., "Novel Site-Specific Techniques for Predicting Radio Wave Propagation", Dept of Electrical and Computer Engineering, Insitituto Politécnico de Virginia. 2000.

[Skl97] B. Sklar, "Rayleigh Fading Channels in Mobile Digital Communications Systems: I. Characterization" IEEE Communications Magazine, vol. 35, no. 7, pp. 90-100, Julio 1997.

[Tay09] A. Tayebi, J. Gomez, F. Saez de Adana, and O. Gutierrez, "The application of ray-tracing to mobile localization using the direction of arrival and received signal strength in multipath indoor environments," Progress In Electromagnetics Research, 2009.

[Tek98] S. Tekinay, "Wireless Geolocation Systems and Services," Special Issue of the IEEE Communications Magazine, Abril 1998

[Wan92] Roy Want, Andy Hopper, Veronica Falcao y Jon Gibbons. *"The active badge location system"*. ACM Transactions on Information Systems, Enero, 1992.

[Wil02] "Converting Signal Strength Percentage to dBm Values", WildPackets, 2002

[Zan06] Giovanni Zanca, Francesco Zorzi, Andrea Zanella and Michele Zorzi, "Experimental comparison of RSSI-based localization algorithms for indoor wireless sensor networks", Department of Information Engineering, Universidad de Padova, Italia. 2006.

10.2 Referencias de Internet

A 1 de septiembre del año 2010 los siguientes enlaces permanecían activos.

[Web-1] http://www.wi-fiplanet.com/news/article.php/3674591
Tema: Origen del nombre "Wifi".

[Web-2] http://www.wi-fi.org/news_articles.php?f=media_news&news_id=545
Tema: Descripción estándar 802.11n

[Web-3] http://www.wireless-nets.com/resources/tutorials/migrate_80211n.html
Tema: Características estándar 802.11n

[Web-4] http://www.bluetooth.com/Bluetooth/Technology/Works/
Tema: Website del estándar Bluetooth

[Web-5] http://www.amperordirect.com/pc/r-electronic-resource/z-reference-bluetooth-class1-myth.html
Tema: Características estándar Bluetooth

[Web-6] http://www.google.com/mobile/gmm/mylocation/
Tema: Software para móviles "GoogleMaps"

[Web-7] http://www.cl.cam.ac.uk/research/dtg/attarchive/bat/
Tema: Sistema de ultrasonidos BAT.

[Web-8] http://www.robesafe.com/inicio/index.php
Tema: Visión artificial.

[Web-9] http://net.cs.uni-tuebingen.de/en/ambisense/wlan-trilateration/how-does-it-work/
Tema: Minimización del residuo con Nelder-Mead.

[Web-10] http://sourceforge.net/projects/ndiswrapper/
Tema: Software NDIS para linux.

[Web-11] http://msdn.microsoft.com/en-us/library/ff571081.aspx
Tema: Características NDIS.

[Web-12]
http://people.sc.fsu.edu/~jburkardt/cpp_src/asa047/asa047.html
Código fuente en C++ del algoritmo Nelder-Mead (asa047.cpp)

[Web-13]
http://www.codeproject.com/KB/windows/PeekPocket.aspx
Tema: Software "PeekPocket"

[Web-14]
http://www.cisco.com/en/US/docs/solutions/Enterprise/Mobility/wifich2.html
Tema: Sistema de localización por Wifi

PLIEGO DE CONDICIONES

1. Introducción

En el pliego de condiciones se definen los materiales y equipos que constituyen y toman parte en la elaboración del proyecto, así como el procedimiento de configuración de los distintos equipos para integrarlos dentro del PFC.

Se requiere previamente configurar adecuadamente todos los dispositivos para posibilitar la ejecución de forma apropiada de las aplicaciones diseñadas. Los dispositivos que hay que tener en cuenta a la hora de configurar son los dispositivos móviles con WindowsMobile y tarjeta inalámbrica incorporada.

2. Terminales inalámbricos

En este PFC se ha empleado como dispositivo inalámbrico un teléfono móvil HTC Diamond, cuyas características se exponen en la siguiente tabla. Aunque el único requerimiento es que el dispositivo sea compatible Wifi y ejecute el sistema operativo Windows Mobile.

Conectividad	Otras características
Banda: 850/900/1800/1900 Mhz **GPRS:** Sí. **Bluetooth:** Sí. **USB:** Sí. **WIFI:** Sí. **GPS / A-GPS:** Sí.	**CPU:** Qualcomm MSM7201A (528MHz) **Sistema Operativo:** Windows Mobile 6.1 **Dimensiones:** 102x51x11,5 mm **Peso:** 110 gr **Bateria:** Li-Ion 900 mAh **Memoria interna:** 4 Gb

Tabla 10.1: Características del teléfono HTC DIAMOND

3. Manual de instalación de las aplicaciones

Una vez cargado el código fuente en VS 2008 con el *plugin* "Smartphone SDK" se debe elegir como plataforma destino *PocketPC2003* y en 'solutions configuration' se debe elegir *'Release'* para obtener un ejecutable pequeño sin código de *debug* alguno. El fichero ejecutable (.exe) se copiará a cualquier carpeta del dispositivo móvil.

En el caso de la aplicación 'WifiLocaliza' el fichero con los datos de los puntos de acceso debe llamarse *'data.txt'* y debe guardarse exclusivamente en la carpeta

'My Documents' del dispositivo. Hay que tener en cuenta que en la HTC hay dos carpetas *'My Documents'*: una se encuentra en la memoria interna y otra en la tarjeta de expansión de memoria. Cuando el usuario copia archivos hacia la HTC dichos archivos se guardarán en la tarjeta de memoria. Sin embargo, la aplicación de este PFC no leerá el fichero *'data.txt'* de la tarjeta de memoria, sino de la memoria flash interna, con lo que el usuario deberá mover el archivo a la carpeta *'My Documents'* correcta.

Se puede elegir otro nombre y otra carpeta para *'data.txt'* antes de compilar el código fuente, buscando el texto *_DATAFILE*:

```
#define _DATAFILE "\\My Documents\\data.txt"
```

En el caso de 'WifiReader' no hará falta copiar ningún fichero adicional, será la propia aplicación quien cree los ficheros .TXT pertinentes en la misma carpeta donde se encuentre el fichero .EXE.

4. Manual de usuario de las aplicaciones

Antes de ejecutar la aplicación, primeramente, hay que encender el adaptador Wifi del dispositivo.

Una vez ejecutada la aplicación, si no aparece ningún punto de acceso en pantalla debe ir a la segunda pestaña "Opciones" y comprobar que nuestro adaptador Wifi aparezca como activo.

Figura 10.1: En 'Adaptador' debe aparecer el modelo de tarjeta Wifi del dispositivo.

Si no está operativo, se intentará lo siguiente:

- Deshabilitar la conexión Wifi y volver a habilitarla.
- Si la PDA está conectada al PC a través de ActiveSync ese puede ser el problema; se sugiere desconectar la PDA del PC.
- Si tiene configurado en su dispositivo "conectar automáticamente a una red", deshabilite esa opción.

Una vez que aparezcan los puntos de acceso en la pestaña 'Scanner' ya se podrá hacer uso de la aplicación.

4.1 WifiReader

De la lista de puntos de acceso que aparecen en pantalla se seleccionarán los puntos Wifi que se encuentren en el entorno donde se desea implementar el sistema de localización. Para seleccionar debe hacer doble-click en cada uno de ellos. Si hay varios puntos de acceso con el mismo nombre de estación puede seleccionarlo a través de la dirección MAC, dato que aparece en pantalla si se desplaza la lista hacia la derecha.

Figura 10.2: Pestaña 'Scanner' de 'WifiReader'

En la pestaña 'Opciones' (figura 10.1) se podrá seleccionar la velocidad a la que se tomarán los datos con la barra de opciones "Vel. Escaneo". La velocidad de actualización queda definida por:

- 'Más rápida': cada 500 milisegundos.
- 'Rápida': cada 800 milisegundos.
- 'Normal': cada segundo.
- 'Lenta': cada 1'5 segundos.
- 'Más lenta': cada 2 segundos.

"Algoritmo": El usuario podrá elegir uno de los siguientes algoritmos: Trilateración, Multi-lateración, Nelder-mead y MinMax. En el caso que sólo se detecten dos puntos Wifi se obviará dicha elección y se ejecutará la bilateración. 'WifiReader' no hace uso de esta opción.

"Con/Sin WEP": Si los puntos de acceso que se desean analizar utilizan encriptación deberá pulsar sobre esta opción hasta que aparezca en pantalla "Con WEP". Esta opción mostrará (o en su defecto, eliminará) los puntos de acceso encriptados que se encuentren al alcance del dispositivo móvil. En el caso de que active dicha opción los puntos de acceso encriptados aparecerán en color rojo. En la figura 10.2 se comprueba que está activada la opción de mostrar puntos de acceso encriptados porque aparece uno en color rojo.

"Con/Sin Peers": mostrará en pantalla los usuarios móviles (peers) que se encuentren al alcance del dispositivo móvil. Aparecerán definidos en la lista con la marca "peer". Los puntos de acceso aparecerán como "AP".

"Con/Sin Sonido": activará unos *beeps* cada vez que se actualicen datos en pantalla.

"Suspender": pausará la aplicación para que no se actualicen más los datos. Si se vuelve a pulsar, la aplicación continuará su funcionamiento.

"Salir": cierra la aplicación.

En la pestaña 'Record' (figura 10.3) aparecerán las opciones necesarias para el desarrollo de la fase 'offline' del proyecto. Sus opciones son:

(X,Y): coordenadas X e Y de donde se encuentre el usuario en ese momento tomando medidas. Desde que se inicie la captura hasta que finalice no debe desplazarse de posición ni de orientación.

"Número de muestras": es el número de muestras de cada punto Wifi que se guardarán en el fichero .TXT

"Orientación": indica la orientación norte, sur, este u oeste en la que esté el usuario en ese momento.

"Temperatura": indica la temperatura ambiente del entorno en donde se encuentre el usuario.

Figura 10.3: Pestaña 'Record' de 'WifiReader'

Una vez introducidos los datos anteriores y seleccionados los puntos Wifi que se desean analizar se debe pulsar INICIAR para comenzar la captura de datos. Una vez que la aplicación capture el número de muestras seleccionado anteriormente avisará con un mensaje de "Captura finalizada".

El fichero de texto de salida se encontrará en la misma carpeta en donde se encuentre el fichero 'WifiReader'. Se creará un fichero de texto nuevo por cada coordenada seleccionada, siguiente este modelo: "C_coordenadaX_coordenadaY.txt". En el caso que el usuario haya escaneado las coordenadas (14,7) y (29,6) obtendría dos ficheros con los nombres: "C_14_7.txt" y "C_29_6.txt".

4.2 WifiLocaliza

Para ejecutar esta aplicación se necesita haber realizado la fase 'offline' del proyecto y haber obtenido los siguientes datos: localización de los puntos Wifi del entorno estudiado, sus direcciones MAC, índice de pérdidas de cada uno y la potencia recibida a un metro. Se deben introducir dichos valores en un fichero de texto según el siguiente formato:

DirecciónMAC-1	Coord-X1	Coord-Y1	Factor 'n1'	Pot.a.1m
DirecciónMAC-2	Coord-X2	Coord-Y2	Factor 'n2'	Pot.a.1m
DirecciónMAC-3	Coord-X3	Coord-Y3	Factor 'n3'	Pot.a.1m
…	…	…	…	…
…	…	…	…	…

Tabla 10.2: Formato del fichero de texto para 'WifiLocaliza'

Este fichero de texto debe llamarse "data.txt" y debe estar guardado en la carpeta "*MyDocuments*\" del dispositivo móvil. En el caso de poseer una HTC debe tenerse en cuenta la existencia por duplicado de la carpeta "*MyDocuments*\", como se indicó en el apartado 3 de este capítulo.

La pestaña 'Scanner' y 'Opciones' es idéntica a la de WifiReader, que se ha descrito anteriormente.

La pestaña 'Mapa' es la que ofrece el sistema de localización.

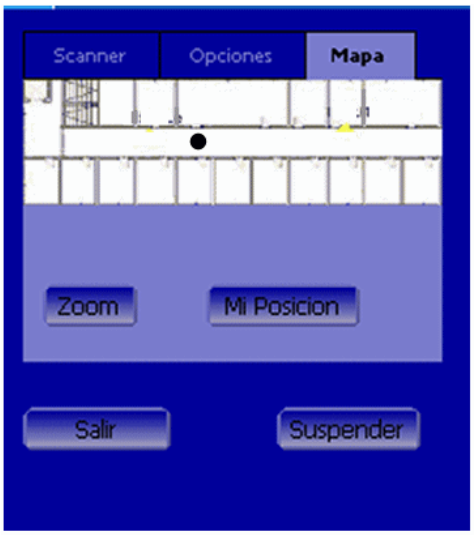


Figura 10.4: Pestaña "Mapa".

El botón "MI POSICION" muestra en pantalla la posición del usuario cada dos segundos. Si el usuario se desplaza a una zona en donde no se detecten los puntos Wifi analizados en la fase 'offline' la aplicación lo indicará con el siguiente mensajes: *"Ningún punto Wifi coincide con la base de datos"*. Si se vuelve a pulsar "MI POSICION", la aplicación dejará de mostrar la localización del usuario.

Si se desea ampliar el mapa para obtener más detalle se debe pulsar "ZOOM". Esta opción aumentará la imagen en un factor 1'8:1. En la figura 10.4 se

muestra el mapa con el zoom activado. Si vuelve a pulsarse "ZOOM" la aplicación mostrará el mapa con las dimensiones anteriores.

5. Puntos de acceso

Los puntos de acceso que se estudiaron en este PFC se encuentran instalados en la segunda planta del edificio de Telecomunicaciones del DIT del campus de la U.L.P.G.C.

Los datos técnicos de dichos puntos de acceso son:

Descripción	Características
Interfaz Ethernet	100 Base-T
R-F Banda de Frecuencias	2,4 GHz -2.4835 GHz
Potencia de Salida	15 dBm (32mW) ± 2dB
Seguridad	- Cifrado WEP de 64 y 128 bits. - WPA - MAC address Access Control
Técnicas de Modulación	OFDM, CCK, DQPSK y DBPSK

Tabla 10.3: Características del AP CISCO 1230-AG.

PRESUPUESTO

1. Introducción

Este capítulo está dedicado a efectuar la valoración económica del proyecto, realizando para ello, un presupuesto con los costes producidos durante la elaboración del mismo.

Hasta el año 2008 la cuantificación económica se hacía en base a las líneas trazadas por el COITT (Colegio Oficial de Ingenieros Técnicos de Telecomunicación) en su documento: *"Baremos de Honorarios Orientativos para Trabajos Profesionales"*, pero debido a una directiva europea, publicada en el BOE nº 308 de fecha de 23 de diciembre de 2009, se prohíbe cualquier recomendación de los honorarios. Se desprende, por tanto, que los honorarios son libres y responderán a un acuerdo entre el profesional y el cliente.

Los distintos conceptos que se tienen en cuenta para el cálculo del presupuesto son los siguientes:

- Gasto de administración y de redacción del proyecto
- Amortización del Hardware
- Amortización del Software
- Trabajo tarifado por tiempo empleado

2. Gastos de administración y de redacción del proyecto

Los gastos referentes a la redacción del proyecto incluyen el coste de los materiales necesarios para su redacción. Estos gastos se pueden clasificar en gastos de encuadernación y gastos fungibles, de consumo inmediato.

Concepto	Total (€)
Encuadernación	18
Fungibles	18
Papelería	40
Subtotal 1	**76**

Tabla 11.1: Gastos de redacción del proyecto

3. Amortización del material

Se ha supuesto un período de amortización de 3 años durante aproximadamente 275 días al año y trabajando 6 horas diarias. Por tanto, el número de horas totales ascienden a 4950 horas de utilización.

Para calcular el gasto del material, se parte del coste de cada recurso calculando la cantidad amortizada en una hora de trabajo (dividiendo el coste total por 4950). Por último, multiplicamos el coste por hora de trabajo por el número de horas trabajadas. El número de horas empleadas en la realización del proyecto se estima en 1440 horas.

Para su cálculo se ha estimado una media de 24 días al mes y trabajando 6 horas diarias durante 10 meses.

Tiempo empleado= 24 x 10 x 6 =1440 horas totales

La amortización del material se divide en amortización de Software y de Hardware.

3.1 Amortización del Software

Descripción	Unidades	Coste (€)/Unidad	Coste (€)/hora	Amortización total (€)
S. O. Windows XP Professional	1	166’68	0’034	48’96
Visual Studio 2008	1	800’00	0’162	233’30
Microsoft Office 2007	1	229’00	0’046	66’24
Subtotal 2				**348’50**

Tabla 11.2: Gastos de amortización del material software

3.2 Amortización del hardware

Descripción	Unidades	Coste (€)/Unidad	Coste (€)/hora	Amortización total (€)
Teléfono HTC Diamond	1	480’00	0’096	139’63
Punto de acceso Cisco	2	217’00	0’044	63’36
Subtotal 3				**202’99**

Tabla 11.3: Gastos de amortización del material hardware

4. Trabajo tarifado por tiempo empleado

Se contabilizan los gastos de mano de obra según el salario de la hora de trabajo de un Ingeniero Técnico de Telecomunicación, haciendo para ello una estimación de 1440 horas de trabajo en los diez meses de duración del proyecto, de las cuales 20 horas se desarrollaron en horario festivo.

Se ha establecido una hora de trabajo en 65 euros para aquellas labores realizadas dentro de la jornada laboral y, en 78 euros para los trabajos realizados en horarios fuera de la jornada laboral. Se aplica un coeficiente reductor en función del número total de horas trabajadas, estimado en 0’4 para periodos que exceden de 1080 horas trabajadas, según Baremo del COITT. Con lo que el total asciende a:

Concepto	Horas	Coste (€)/hora	Total (€)
Trabajo especializado horario laboral	1420	65	92300
Trabajo especializado horario festivo	20	78	1560
Total			**93860**
Subtotal 4		Total x 0’4	**37544**

Tabla 11.4: Tarifado por tiempo empleado

5. Presupuesto total

Para obtener el presupuesto total sumamos todas las cantidades parciales obtenidas en los anteriores apartados.

Concepto	Total (€)
Redacción del Proyecto	76’0
Amortización del Software	348’50
Amortización del Hardware	202’99
Trabajo Tarifado	37544’0
Total	**38171’49**

Tabla 11.5: Presupuesto total

Al importe del presupuesto se le debe aplicar el IGIC (Impuesto General Indirecto Canario) que para esta actividad grava un 5%, con lo que el presupuesto total del Proyecto Fin de Carrera asciende a:

Cuarenta mil ochenta euros con seis céntimos.

Coste total	40080’06 €

Fdo. José Julio Báez Redondo