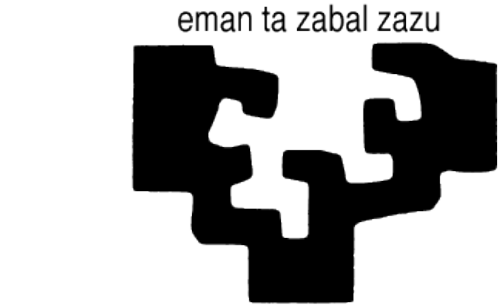


universidad del país vasco

euskal herriko unibertsitatea

Facultad de Informática / Informatika Fakultatea

Sistema de “Autofoco” en cámaras de Video/Foto mediante sensores inalámbricos

Alumno: Eduardo Sáenz Santiago

Director/a: D. Andoni Arruti Illarramendi

Dña. Amaya Ibarra Lasa

Proyecto Fin de Carrera, junio 2013

Resumen

El objetivo de este Proyecto de Fin de Carrera es el desarrollo de un sistema de medición de distancias entre objetos por radiofrecuencia, concretamente, a través de un indicador de potencia de señal recibida (RSSI). El proyecto está orientado a aplicar el sistema desarrollado a la grabación de video mediante cámaras fotográficas, de tal manera que el anillo de enfoque de las mismas se realice de modo automático; esto es, las distancias entre cámara y actores que intervienen en una escena serán los valores proporcionados por el sistema desarrollado, facilitando las distancias de enfoque de los diversos actores en movimiento de la escena. Con este fin, se ha dividido el proyecto en tres fases: primero se estudian las distintas opciones inalámbricas que permita el desarrollo del sistema, a continuación se estudia tanto la plataforma de desarrollo hardware (CC2430DK) como las herramientas software que se ponen a disposición del desarrollador; y por último se desarrolla e implementa el sistema de mediciones de distancias sobre la plataforma hardware y la red de comunicación inalámbrica seleccionada, así como la interfaz de usuario que monitoriza la cámara, los actores y sus respectivas distancias. Finalmente se analizarán los resultados obtenidos de una muestra de diez mediciones entre Cámara y Actor de los cuales se expondrán las conclusiones deducidas.

Summary

This project involves the development of a system of measuring distances between various objects via radio frequency, specifically through received signal power between them (RSSI). The project aims to implement the system developed to recording video using photo cameras, so that the focusing ring is automated, with the distances between camera and actors in a scene provided by the system developed, facilitating focusing distances of the various actors. To this end, the project has been divided into three phases: one in which we study the wireless options allow system development, a second phase in which we study the hardware development platform and software tools available to the developer, and another which develops and implements the distance measurement system and hardware platform on the selected wireless communication.

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO 1: INTRODUCCIÓN

El presente Proyecto de Fin de Carrera se centra en el desarrollo de un sistema de medición de distancias entre objetos o personas por radio frecuencia, cuya finalidad es la aplicación práctica del mismo en la grabación de video sobre cámaras fotográficas, quedando el anillo de enfoque de la cámara automatizado mediante los valores de distancias medidos por el sistema, de tal forma que el seguimiento del foco a actores u objetos en una escena quedaría automatizado, eliminando la figura del foquista, encargado de manipular manualmente el anillo de enfoque de la cámara. La motivación del proyecto viene generada a raíz de la necesidad de reducir tiempo a la hora de abordar sesiones de grabaciones en las que las distancias son tomadas manualmente sobre las posiciones finales e iniciales de los actores u objetos que se quieren mantener enfocados en la escena. Dichas motivaciones quedan patentes en el siguiente apartado en el cual se describe los antecedentes que motivaron el planteamiento del proyecto, así como en el apartado de objetivos del proyecto en al cual se defines los objetivos a alcanzar.

1.1: Análisis de Antecedentes

Uno de los factores más importantes a la hora de abordar una sesión de grabación, ya sea publicitaria, cinematográfica o televisiva, se da en la elección del soporte o sistema de grabación. Actualmente existen tres:

- **Cinematográfico**: Cuyo soporte físico es el celuloide (35mm, 70mm) o soporte digital (Tarjetas de memoria).
- **Video:** En cuya categoría incluimos todas las videocámaras, las cuales puedes grabar en soporte físico digital, tarjetas de memoria, o en soporte analógico: MiniDV, Betacam, DvCam, etc. La calidad es considerablemente inferior a la cinematográfica, siendo las ópticas cinematográficas una de las razones de su calidad superior.
- **Fotográfico**: Como tercer soporte, tenemos las cámaras de fotos réflex con la reciente incorporación de la funcionalidad de grabar video con calidad fotográfica. Este tipo de cámaras se encuentra en auge debido a su relación calidad precio, ya que presentan unas calidades muy superiores al video, acercándose a las cinematográficas.

Todos los sistemas de grabación, independientemente del soporte físico en el que se grabe (analógico o digital) disponen de parámetros básicos o controles habituales como: foco, diafragma, zoom (distancia focal variable), balance de blancos y velocidad de obturación. En cámaras cinematográficas y fotográficas, además, disponemos de la sensibilidad ISO de la película.

En la grabación de cualquier secuencia, y sin entrar en detalles, se tendrían que ajustar todos los parámetros con el fin de tener el encuadre correcto, con la profundidad de campo deseada, color y personajes a foco (enfocados). Uno de los hándicap más comunes y laboriosos, se da con el seguimiento de foco; es decir, mantener enfocado a un personaje mientras este o la cámara se mueven, variando así la distancia de foco y teniendo que ajustarla manualmente. Actualmente existen sistemas de enfoque automático (AF) los cuales varían la distancia de foco mediante evaluaciones de contraste. El autoenfoque basados en la medida del contraste parten del principio de que una imagen desenfocada posee menor contraste, mientras que una imagen enfocada tiene un mayor contraste, especialmente en los contornos o relieves de las figuras. Puesto que la cámara no conoce la distancia del objeto, no basta con una sola evaluación del contraste para realizar el enfoque. Una vez realizadas dos pruebas de contraste con distintos enfoques, no solo la máquina puede saber en qué dirección mover el enfoque, sino que por extrapolación podría hasta llegar a realizar el enfoque. Normalmente se realizan varias pruebas de contraste mientras se mueve el foco; cuando el contraste es máximo el objeto está enfocado. Este método suele fallar al enfocar superficies planas sin contraste o contornos (cielo despejado, pared, etc.), así como en escenas de escasa iluminación.

Todo esto hace que, en cualquier producción profesional o amateur, el enfoque se realiza de modo manual mediante un sistema de marcas; es decir, se toman manualmente las distintas distancias finales del objeto en movimiento a enfocar, se marcan dichas distancias en el anillo de foco y se manipula el anillo de enfoque manualmente conforme el objeto de mueve.

1.2: Objetivos del Proyecto

1.2.1Objetivo general del proyecto

Una vez planteado el problema del anillo de enfoque y dado el apogeo en el que se encuentran las cámaras fotográficas réflex (SLR), se plantea como objetivo de este proyecto, la realización de un sistema de autoenfoque por radiofrecuencia aplicada a cámaras fotográficas en la grabación de video, de tal forma que independientemente del movimiento del objeto a enfocar o el movimiento de la cámara, así como del cambio de foco a otro objeto, éste se realice de forma automática.

Cada personaje u objeto que en algún momento de la escena vaya a ser enfocado y seguido, debería portar un transpondedor (emisor y receptor por radiofrecuencia). De tal manera, que el usuario (operador de cámara) solo se debería encargar de seleccionar el transpondedor (Personaje u objeto) a seguir. Así el sistema de enfoque sería automatizado y se precisaría de la figura del Foquista (Encargado de mover el anillo de foco a las marcas previamente tomadas con las distancias de enfoque).

Por tanto, teniendo el enfoque automatizado y una vez ajustados el resto de parámetros: diafragma, zoom (distancia focal variable), balance de blancos, velocidad de obturación y sensibilidad ISO; una única persona podría hacerse cargo de la grabación, moviéndose libremente a través de la escena.

1.2.2 Objetivos específicos del proyecto

- Investigar el funcionamiento de las tecnologías de comunicación inalámbricas.
- Elegir la tecnología inalámbrica a utilizar y desarrollar el proyecto sobre dicha tecnología.
- Diseñar un programa que permita visualizar la ubicación de la cámara y los objetos u actores en movimiento, así como sus respectivas distancias de separación. Todo ello sobre la tecnología seleccionada en los objetivos anteriores.

CAPÍTULO 2: ESTRUCTURA DE LA MEMORIA

En este apartado mostramos de forma ordenada los pasos a seguir para la realización de este proyecto. En primer lugar, tras el análisis de antecedentes y marcar los objetivos del proyecto, explicar las opciones de desarrollo del proyecto y el funcionamiento básico de una red basada en el estándar ZigBee/802.15.4. Se expondrán los elementos que se han utilizado para conseguir dicho funcionamiento tanto software como hardware, se explicará la interfaz de usuario que interactúa con la red y se expondrán las conclusiones globales deducidas.

Más concretamente, la estructura de la memoria queda organizada de la siguiente manera:

- **Capítulo 3: Análisis de factibilidad:** En este capítulo se detallan los pros y contras de la posible aplicación comercial del proyecto.
- **Capítulo 4: Opciones de desarrollo:** Capítulo en el que analizaremos posibles sistemas inalámbricos que nos permitan el desarrollo del proyecto.
- **Capítulo 5: ZigBee:** Profundizaremos sobre la red ZigBee, sistema inalámbrico elegido para la realización del proyecto, su funcionamiento y sus características.
- **Capítulo 6: Perfil Localización:** Teniendo como red de comunicaciones, la red ZigBee, profundizaremos sobre el perfil de pila a usar en este proyecto (LocationProfile, desarrollado por Texas Instrument y orientado a localizar objetos en un entorno controlado). Haremos uso de la red ZigBee

en lo que se refiere a las comunicaciones e implementaremos el perfil de localización en las capas de aplicación de la pila ZigBee.

- **Capítulo 7: Herramientas de desarrollo empleadas:** En este capítulo detallaremos las herramientas Hardware y Software empleadas en el desarrollo del proyecto teniendo como red de comunicación una red ZigBee y como perfil de pila, LocationProfile.
- **Capítulo 8: Diseño e implementación:** Capitulo en el que se expondrán todos los diseños de los roles asumidos por cada una de las partes Software del proyecto, así como los programas realizados en los distintos módulos hardware para asumir sus respectivos roles.
- **Capítulo 9: Análisis e interpretación de resultados:** Capítulo en el que se realizará una prueba de diez muestras de posicionamiento y distancias entre dos nodos finales, uno de ellos simulando la Cámara y el otro al Actor.
- **Capítulo 10: Conclusiones y bibliografía:** Capítulo en el que tras el análisis y recopilación de resultados, se expondrán las conclusiones deducidas y se contrastarán con los objetivos iniciales.
- **Capítulo 11: Anexos:** Capítulo en el que se incluirá el código fuente desarrollado al completo, tanto en Z-Tool como en Matlab.

CAPITULO 3: ANÁLISIS DE REQUISITOS Y FACTIBILIDAD

Los requerimientos hardware necesarios para cada persona u objeto que intervenga en el sistema a desarrollar debe constar de: Un emisor y receptor de señal por radio frecuencia (Transpondedor), un micro controlador que gestione las emisiones y recepciones de tramas, así como la manipulación de las mismas y un indicador de fuerza de señal de recepción de señal (RSSI).

3.1 Inconvenientes

En el siguiente apartado se expondrán los inconvenientes del sistema de medición de distancias entre objetos por radiofrecuencia como aplicación comercial en cámaras de fotos aplicado a la automatización del anillo de enfoque.

- **Hardware**:

Habría que añadir un transpondedor (emisor y receptor de señal por radio frecuencia) a la cámara y a cada persona u objeto a seguir. La parte de gestión la podría realizar el propio micro de la cámara.

- **Software:**

Una de las maneras de medir distancias mediante radio frecuencia es a través del parámetro RSSI, *Receive Signal Strength Indicación*, (Indicador de fuerza de señal de recepción del transpondedor) el cual se verá afectado por obstáculos y otras frecuencias, atenuando la señal y dando lecturas falsas. Se tendrían que utilizar una serie

de balizas (objetos de referencia fijos cuyas posiciones X e Y son conocidas) con el fin de tener varias lecturas y desestimar posibles lecturas falsas.

3.2 Ventajas

Una única persona podría realizar la tarea de grabar y enfocar, eliminando al foquista y la tediosa tarea de medir distancias manualmente para cada desplazamiento del actor u objeto. Obteniendo un considerable ahorro en tiempo, lo que conlleva a un menor coste de las producciones.

Añadiría maniobrabilidad a los movimientos de cámara ya que no tendrían que estar dos personas manipulándola al mismo tiempo. El operador de cámara con los movimientos de cámara y el foquista con el anillo de enfoque entorpeciéndose el uno al otro, teniendo que coreografiar sus movimientos para realizar las tomas.

3.3 Viabilidad

Vistas las ventajas e inconvenientes que presenta el proyecto, la viabilidad del mismo recaerá sobre el margen de error que se consiga en las mediciones de distancia de enfoque.

CAPITULO 4: OPCIONES DE DESARROLLO

En un principio se estudió la posibilidad de realizar una comunicación mediante un sistema de etiquetas RFID (Identificación por radiofrecuencias).

4.1 Sistema RFID

Un sistema RFID (Radio Frequency IDentification) está compuesto básicamente por un lector de etiquetas y etiquetas (tags). RFID consiste en la transmisión de datos mediante ondas de radiofrecuencia entre un emisor y un receptor (véase Figura 1). Para ello, se emplean tags o "etiquetas" inteligentes que son adheridas a un elemento. La información transmitida es a su vez capturada por un lector RFID a través de una antena. Finalmente, estos datos son enviados a un sistema de procesamiento de datos llamado "Middleware", que se encarga de filtrar y trasladar la información captada por el lector hasta las aplicaciones de interés para que la información sea visible.

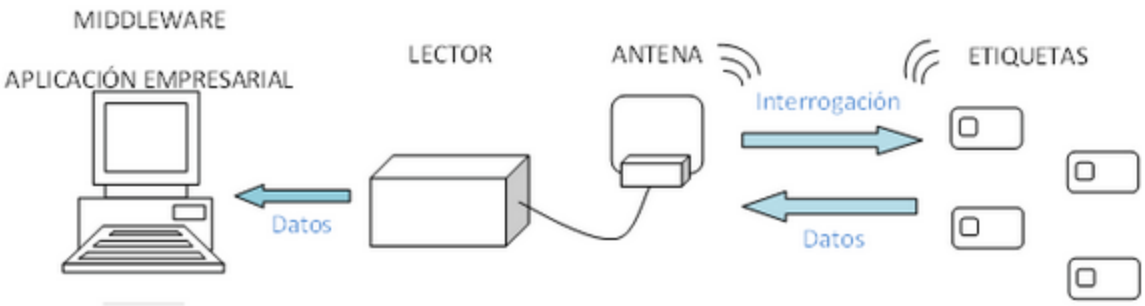


Figura 1 : Esquema general del funcionamiento de un sistema RFID

4.1.1 Tags Pasivos:

- **Ventajas:** Los tags pasivos no poseen alimentación eléctrica. La señal que les llega de los lectores induce una corriente eléctrica pequeña y suficiente para operar el circuito integrado CMOS del tag, de forma que puede generar y transmitir una respuesta.
- **Inconvenientes:** La distancia operativa práctica es muy pequeña para el proyecto en cuestión.

4.1.2 Tags Activos:

- **Ventajas:** A diferencia de los tags pasivos, los activos poseen su propia fuente autónoma de energía, que utilizan para dar corriente a sus circuitos integrados y propagar su señal al lector. Por lo que su alcance es mayor, de hasta 500 m.
- **Inconvenientes:** Habría que crear de cero una red de comunicación entre todos los nodos (Balizas, objetos a enfocar y gestor) para la realización del proyecto.

4.2 Redes inalámbricas

Los puntos a tener en cuenta a la hora de la creación de estándares siempre han sido el máximo alcance posible ofreciendo la máxima tasa de transferencia de datos.

No obstante, un gran sector de las telecomunicaciones se basa en el campo de sensores, donde la cantidad de información generada es bastante inferior en relación a otros dispositivos que se basan en otros protocolos como el 802.11 (Wi-Fi) o el 802.15.1 (Bluetooth).

Frente a esta necesidad, así como a la reducción de consumo, optaremos por la red ZigBee, ofreciendo comunicaciones entre nodos bajo el estándar 802.15.4 y permitiendo el modo sleep de los nodos finales y cuya topología, tipos de dispositivos de la red, pila de protocolos y sus características quedarán expuestas en el siguiente apartado.

CAPITULO 5: ZIGBEE /802.15.4

5.1 Introducción a la Red ZigBee/802.15.4

Las Redes ZigBee comenzaron a concebirse por el año 1998, cuando muchos ingenieros observaron que estos dos estándares Wi.Fi y Bluetooth, resultaban inadecuados para cierto tipo de aplicaciones. Por otro lado, hasta hace pocos años no existía un estándar inalámbrico dedicado a sensores y dispositivos de control. Dichos dispositivos no requieren de un gran ancho de banda pero si requieren de un bajo consumo para lograr una duración de baterías más larga y redes con un mayor número de dispositivos.

Hoy en día existen soluciones creadas por diferentes fabricantes para cumplir con estos requerimientos, pero debido a la falta de un estándar abierto, hay un problema de interoperabilidad entre ellas. Por este motivo, un gran número de empresas fabricantes han formado una alianza con el objetivo de promover un estándar global y abierto para redes de sensores y control denominadas ZigBee.

ZigBee es una alianza tecnológica, sin fines de lucro, formada por más de 100 empresas, la mayoría de ellas fabricantes de semiconductores, con el objeto de auspiciar el desarrollo e implementación de una tecnología inalámbrica de área personal (Wireless personal area network, WPAN) a bajo costo.

ZigBee Alliance recurrió al estándar IEEE 802.15.4 como base para desarrollar las niveles inferiores del protocolo, nivel físico (PHY) y nivel MAC(Control de Acceso

al Medio), permitiendo utilizar una topología de red tan variada como el número de aplicaciones, incluyendo además características de seguridad.

La especificación ZigBee completa este estándar añadiendo cuatro componentes principales:

- Nivel de red
- Nivel de aplicación
- Objetos de dispositivos ZigBee (ZDO, ZigBee Device Objects)
- Objetos de aplicación definidos por el fabricante

Además de añadir dos capas de alto nivel (nivel de red y de aplicación) a la pila de protocolos, el principal cambio es la adición de los ZDO (ZigBee Device Objects) ya que son responsables de llevar a cabo una serie de cometidos, entre los que se encuentran el mantenimiento de los roles de los dispositivos, la gestión de peticiones de unión a una red, el descubrimiento de otros dispositivos y la seguridad. Los tipos de dispositivos ZigBee (ZDO) que podemos encontrar son: routers, coordinador y dispositivos finales cuyas funcionalidades quedan expuestas en el siguiente apartado.

En la Figura 2 vemos de forma esquematizada los distintos niveles del estándar 802.15.4 y la especificación ZigBee.

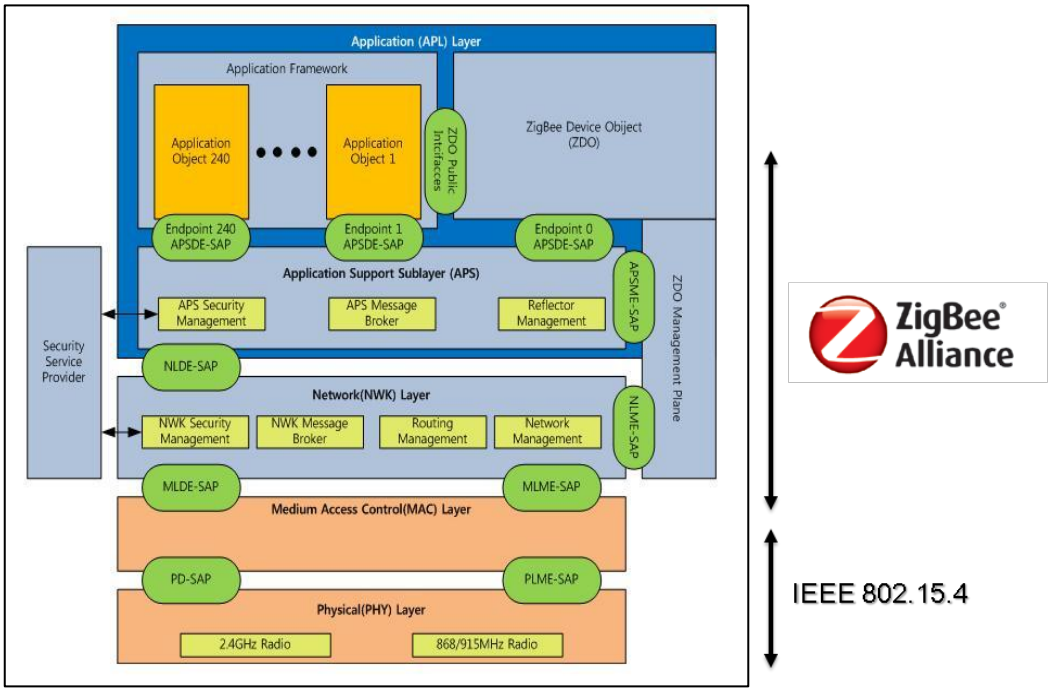


Figura 2. Pila de Protocolos ZigBee

5.2 Tipos de dispositivos ZigBee

ZigBee Alliance define tres tipos diferentes de dispositivos ZigBee según el rol que cumplen en la red:

5.2.1 Coordinador ZigBee

Es el tipo de dispositivo más complejo. Existe uno por red, independientemente de la topología utilizada. Inicia la forma de la red y selecciona la frecuencia de canal a ser usado. Es el responsable de la asociación y desasociación de dispositivos.

5.2.2 Router ZigBee

Interconecta dispositivos separados en la topología de la red. Se asocia con el coordinador de la red u otro Router ZigBee. Es capaz de enrutar mensajes entre dispositivos y no soporta estar en estado “sleep”.

5.2.3 Dispositivo final:

Posee la funcionalidad necesaria para comunicarse con su nodo padre (el coordinador o un Router), pero no puede transmitir información destinada a otros dispositivos. De esta forma, este tipo de nodo puede estar dormido la mayor parte del tiempo, aumentando la vida media de sus baterías. Los dispositivos finales están siempre localizados en los extremos de la red.

En Base a su funcionalidad puede plantearse una segunda clasificación:

6.3 Clasificación en base a la funcionalidad

5.3.1 Dispositivos de funcionalidad completa (FFD)

Los dispositivo de funcionalidad completa (full-function device, FFD) puede funcionar como coordinador, router o nodo final. Implementa un modelo general de comunicación que le permite establecer un intercambio con cualquier otro dispositivo.

5.3.2 Dispositivo de funcionalidad reducida (RFD)

Los dispositivos de funcionalidad reducida (reduced-function device, RFD) son dispositivos muy sencillos con recursos y necesidades de comunicación muy limitadas. Por ello, sólo pueden comunicarse con FFD's y nunca pueden ser coordinadores.

5.4 Topología

ZigBee permite tres topologías de red:

5.4.1 Topología en estrella

Esta topología de red consta de un dispositivo FFD funcionando como Coordinador y una serie de FFD y/o RFD configurados como dispositivos finales. Todos los dispositivos finales están directamente conectados al coordinador, que hace las funciones de inicialización, control y gestión de la red; por tanto, toda comunicación existente entre cualquiera de los nodos, tiene que pasar necesariamente por el coordinador. El principal inconveniente de esta configuración de red es que el alcance máximo de la red queda definido por el coordinador, por lo que no se pueden conseguir redes de más de10 metros alrededor de este.

.

5.3.3 Topología de malla

Es la topología característica de ZigBee/802.15.4. En este tipo de redes, el coordinador funciona como un *router* mas, con la salvedad de que és este quien inicializa y elige los principales parámetros de la red. Los *router* dan la posibilidad de ampliar la red tanto en número de nodos como en rango de alcance y el hecho de que no solo el coordinador sea el encargado de gestionar la red sino que los *routers* también tengan esta funcionalidad hace que la red sea mucho más fiable. Con esta configuración, se puede establecer comunicación entre cualquier par de nodos con la existencia, además, de varios caminos posibles. La elección de un determinado camino viene determinada por el nivel de red utilizando un protocolo de pregunta respuesta, para seleccionar el camino optimo.

5.4.2 Topología en árbol

En esta topología los dispositivos se organizan de manera jerárquica. El coordinador sería el primer nivel, al que pueden conectarse tanto dispositivos FFD como RFD (llamados nodos hijos). De cada FFD pueden seguir conectándose dispositivos estableciéndose los siguientes niveles de profundidad. En esta topología, al igual que pasaba con la configuración mallada, tanto los *routers* como el coordinador tienen capacidad de encaminamiento. Además, estos *routers* permiten expandir la red más allá del alcance máximo del coordinador, por lo que se pueden cubrir mayores aéreas que en una topología en estrella

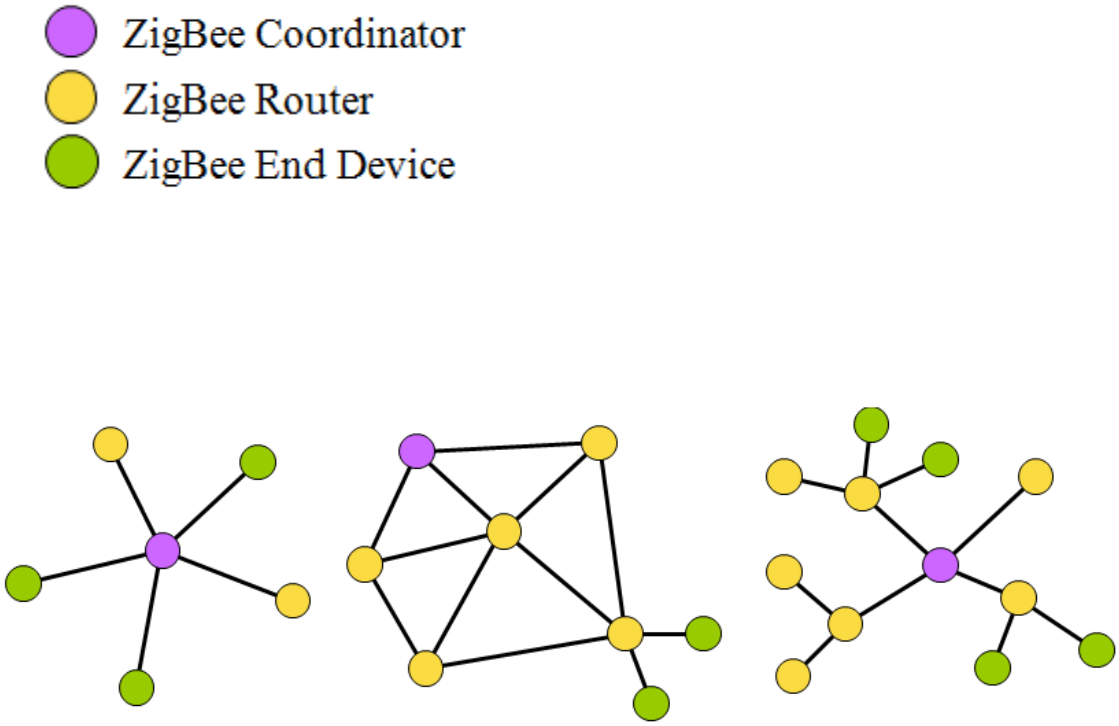


Figura 3. Topologías posibles en una red ZigBee /802.15.4

(a) Estrella. (b) Malla. (c) Árbol

5.4 Pila de Protocolos

Se define ZigBee/802.15.4 como una pila de protocolos que permite la comunicación de forma sencilla entre múltiples dispositivos. Especifica diversas capas adecuándose al modelo OSI (*Open Systems Interconnect*). Las capas básicas, capas física y MAC (*Medium Access Control)* están definidas por el estándar IEEE 802.15.4, LR-WPAN (*Low Rate-Wireless Personal Area 5etwork*) y se han diseñado pensando en la sencillez de la implementación y el bajo consumo sin perder potencia ni

posibilidades. Las capas superiores, capas de red y aplicación, están definidas por el estándar ZigBee, desarrollado por ZigBee Alliance, agrupación de más de 300 compañías que trabajan conjuntamente por convertir la tecnología ZigBee en un referente importante en el marco de aplicaciones ya expuesto en el capítulo anterior.

5.4.1 Capa Física (PHP)

La capa física es la encargada de proporcionar un medio por el que transmitir y recibir datos. El estándar 802.15.4 ofrece la posibilidad de trabajar en tres bandas de frecuencia distintas, todas utilizando DSSS (*Direct Sequence Spread Spectrum*) como técnica de ensanchado de espectro: 868 MHz, 915 MHz y 2.4 GHz. La banda de 868 MHz proporciona un único canal de comunicaciones entre las frecuencias 868 y 868.6 MHz consiguiendo una velocidad de transmisión de 20 Kbps. En la banda de 915 MHz existen 10 canales de comunicación, repartidos uniformemente entre las frecuencias 902 MHz y 928 MHz con una separación entre canales de 2 MHz y obteniendo velocidades de transmisión de 40 Kbps. Estas dos bandas de frecuencias tienen como principal ventaja respecto a la banda de 2.4 GHz su mayor alcance debido a menores perdidas de propagación, sin embargo, su mayor inconveniente es que no son bandas universales, la banda de 868 MHz es de uso exclusivo en Europa y la banda de 915 MHz en Estados Unidos y Australia, por lo que la movilidad y zonas de aplicación de dispositivos funcionando a estas frecuencias está altamente limitado. No es el caso de la banda de 2.4 GHz, cuyo uso está permitido en prácticamente todo el mundo. Proporciona 16 canales entre las frecuencias 2.405 GHz y 2.48 GHz, con una separación entre canales de 5 MHz y un ancho de banda de 2 Mbps, alcanzando

velocidades de transmisión de hasta 250 Kbps. En la siguiente figura se pueden observar los distintos canales radio existentes en cada una de las bandas de frecuencias:

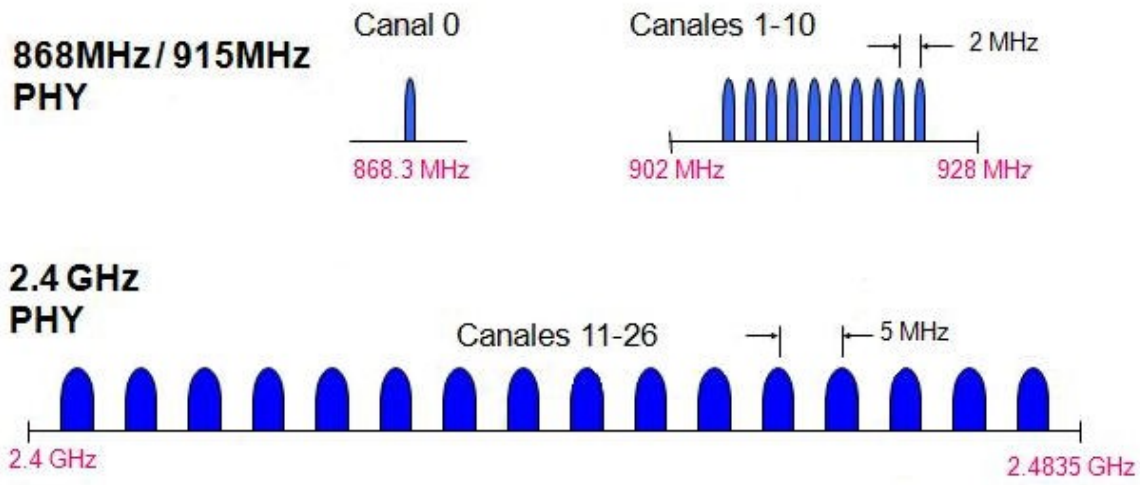


Figura 4. Canales radio en las tres bandas de frecuencia de trabajo

5.4.1.1Funciones de la capa física

Las funciones a realizar por la capa física son varias, aunque las más destacables son:

• **Activación y desactivación del transceptor radio**

El transceptor radio tiene tres posibles estados de funcionamiento: transmisión, recepción y modo dormido (*Sleeping*). Este último estado permitirá al dispositivo un importante ahorro de energía. Es la capa MAC la que decide el estado del transceptor en cada instante.

• **Detección de energía (ED)**

La medición de la energía obtenida por el detector se utiliza en la capa de red como parte del algoritmo de selección de canal, así la capa física se encarga de realizar esta medición en un determinado canal. No se realiza decodificación alguna de señal, solo se obtiene una estimación de la potencia recibida en dicho rango de frecuencias que se traduce en un valor de 8 bits, cuyo rango varía entre 0x00 (que indica que la potencia recibida es menor de 10 dB sobre la especificada por el receptor) y 0xFF.

• **Indicación de la calidad del enlace (LQI)**

La medida del LQI es una caracterización de la potencia y/o calidad del paquete recibido. Esta medida se puede implementar utilizando el detector de energía, un estimador de relación señal a ruido o mediante una combinación de ambos métodos. El valor obtenido se traduce en 8 bits (entre 0x00 y 0xFF) cuyos valores máximo y mínimo vienen determinados por la mayor y menor calidad de señal IEEE 802.15.4 detectable, siguiendo los demás valores de calidad una distribución uniforme entre ambos limites.

• **Selección de canal**

La capa MAC a través de un algoritmo decidirá el canal más adecuado para establecer una comunicación, pero es la capa PHY la que toma físicamente ese canal para realizar las transmisiones y/o recepciones de datos.

• **Evaluación de canal libre (CCA, *Clear Channel Assessment*)**

La evaluación de canal libre se desarrolla utilizando uno de los siguientes tres métodos:

- Energy above threshold: CCA avisará de la ocupación de un medio si detecta cualquier tipo de energía por encima de un umbral predeterminado.

- Carrier Sense Only: CCA notificará sobre la ocupación de un medio si detecta una señal con la modulación y características de propagación propias de una señal IEEE 802.15.4. Esta señal puede recibirse con independencia del umbral.

- Carrier Sense with energy above threshold: Unión de los dos métodos anteriores.

• **Transmisión y recepción de datos por el medio físico**

Esta capa proporciona además, dos servicios: servicio de datos (PHY *data service*) y servicio de gestión (PHY *management service*) ejerciendo de interfaz entre la capa MAC y el medio físico en cuestión, en este caso, los canales radio. El servicio de datos permite la transmisión y recepción de paquetes PPDU (Physical Protocol Data Unit).

5.4.2 Capa MAC

La capa MAC o capa de acceso al medio es la responsable de las siguientes tareas:

• Generación de balizas (beacons) si el dispositivo es un coordinador.

• Sincronización de los beacons de la red.

• Gestión de las asociaciones y desvinculaciones de los dispositivos a la red.

• Empleo del algoritmo CSMA-CA (*Carrier Sense Multiple Access-Collision Avoidance*) para acceso al canal.

• Gestión de la técnica GTS (*Guaranteed Time Slot).*

• Gestión de un enlace fiable entre las capas MAC de los nodos contiguos

6.4.2.1 Modos de funcionamiento

La capa MAC permite dos modos de funcionamiento, la elección de uno u otro modo se realiza al configurar los dispositivos, aunque es el coordinador el encargado de informar a la red qué modo se utilizara en la nueva red creada. Estos dos modos son los siguientes:

• **Modo balizado (beacon-enabled network)**

Modo de funcionamiento que consigue un importante ahorro energético. Está basado en la utilización por parte de los dispositivos FFD de balizas con las que marcan los tiempos en los que es posible la recepción y transmisión de información. Fuera de estos tiempos, todos

los dispositivos (incluido el coordinador) pueden estar en modo "dormido", modo en el que se minimiza en consumo. La baliza se genera periódicamente por el coordinador y se distribuye por toda la red a través de los *routers*. Estas balizas se encargan de sincronizar los dispositivos, de modo que todos los dispositivos se "despierten" en un determinado instante en el cual se realiza la comunicación entre nodos.

• **Modo no balizado (non beacon-enabled network)**

En este modo no existe sincronización entre dispositivos, por lo que los únicos nodos que pueden pasar al estado "dormido" son los dispositivos finales. Estos se despertaran de forma periódica para preguntar si existen datos destinados a ellos o bien para mandar información.

5.4.2.2 Algoritmo CSMA-CA

Al utilizar este algoritmo de acceso al medio, cada dispositivo anuncia su intención de realizar una transmisión antes de empezar con ella, de forma que solo empezará a mandar información si encuentra el canal libre. Si varios dispositivos encuentran el canal ocupado, cada nodo espera un tiempo aleatorio antes de volver a pedir el canal con el fin de evitar colisiones. Un dispositivo en posesión del canal no lo libera hasta recibir la confirmación de recepción de su

trama enviada. Por tanto, el proceso que sigue CSMA-CA se puede resumir, básicamente, en cuatro pasos:

1) El dispositivo con intención de enviar datos, escucha el canal para ver si este está libre.

2) En caso de estar libre, envía la información. Si no lo está, espera un tiempo aleatorio antes de intentarlo de nuevo.

3) Una vez enviada la información, espera la llegada de una confirmación de recepción con la que asegurarse que la transmisión se ha realizado con éxito.

4) Tras esa confirmación, da la transmisión por concluida.

Existen dos versiones de este algoritmo, según se esté operando en modo balizado o en modo no balizado. Para el modo balizado se utiliza el meto CSMA-CA ranurado mientras que para el modo no balizado se usa el método CSMA-CA no ranurado.

5.4.2.3 Inicio y mantenimiento de la red

La capa MAC es la encargada de inicializar una red así como de gestionar las conexiones y desconexiones de dispositivos en ésta. Para poder desarrollar estas tareas, se sirve de una serie de procedimientos:

• **Detección de energía de canal**

Con la finalidad de que el coordinador pueda elegir aquel canal que considere más limpio de interferencias, la capa MAC puede pedir a la capa física que realice una detección de energía de una serie de canales seleccionados por las capas superiores.

• **Escaneo activo de canal**

Cuando un *router* o dispositivo final se quiere asociar a una red utiliza este sistema, por el cual manda balizas periódicamente a través de cada uno de los canales en busca del coordinador. Si el coordinador se encuentra funcionando en modo no balizado, al recibir esta baliza, mandará una respuesta de baliza puntual, con la que el dispositivo en búsqueda detectará la presencia del coordinador y empezará el proceso de asociación. En el caso del modo balizado, el coordinador ignorará estas balizas y mandará periódicamente las suyas, el dispositivo en búsqueda detectará esta baliza y encontrará al coordinador.

• **Escaneo pasivo de canal**

En este caso, el nuevo dispositivo encendido, no manda balizas sino que se dedica a escuchar cada uno de los canales en busca de las

balizas periódicas del coordinador. Como es lógico, es un tipo de escaneo utilizado solo en el modo balizado.

• **Escaneo de orfandad**

Si las capas superiores reciben repetidas veces fallos de comunicación al solicitar datos, puede determinarse que el dispositivo ha quedado huérfano. En este caso, el dispositivo manda notificaciones de orfandad por aquel conjunto de canales que ordenen las capas superiores y desecha todos los datos entrantes a la capa física que no sean una respuesta por parte del coordinador a esta notificación durante un tiempo definido por la variable *aResponseWaitTime*. Si durante este tiempo el coordinador responde a esta notificación de orfandad, el dispositivo consigue volver a sincronizarse.

5.4.2.4 Formato de trama MAC

La capa MAC puede enviar hasta cuatro tipos de tramas distintas: trama de comandos MAC, trama de baliza, trama de datos y trama de confirmación de recepción o trama ACK.

5.4.3 Capa de Red

La capa de red tiene como misión procurar una correcta funcionalidad de las capas inferiores (física y MAC), además de servir como interfaz de servicio para la capa

de aplicación, tanto en datos, como en gestión. Así, sus principales tareas se pueden resumir en:

• **Configuración de nuevos dispositivos**

Para que estos operen de la forma requerida, ya sea inicializando un dispositivo como coordinador o uniéndose a una red existente.

• **Inicialización de red**

La capa de red tiene la habilidad de establecer una nueva red.

• **Asociación, reasociación o abandono de una red**

Solicitado tanto por el dispositivo afectado, como por parte del coordinador o algún *router*.

• **Direccionamiento**

Esta capa otorga la capacidad tanto al coordinador como a los *routers* existentes en la red de proporcionar direcciones a los nuevos dispositivos que estén asociándose a la red.

• **Descubrimiento de vecinos**

Tanto descubrimiento como almacenamiento de información y aviso sobre dispositivos vecinos cercanos.

• **Descubrimiento de ruta**

Descubrimiento y registro de caminos entre dispositivos de forma que siempre se utilice el camino más optimo entre un destino y un origen.

• **Control de recepción**

La capa de red permite controlar si un dispositivo tiene la recepción activada y por cuánto tiempo.

• **Enrutamiento**

Proporciona la habilidad de utilizar diferentes mecanismos de enrutamiento como son: *unicast*, *broadcast* o *multicast* consiguiendo un intercambio de datos eficiente entre dispositivos. A nivel de red existen dos tipos de direcciones, direcciones cortas (16 bits) y direcciones largas o direcciones IEEE (64 bits). Cada dispositivo posee una única dirección IEEE que es asignada al dispositivo a la hora de fabricarlo y en la red se le asigna de forma dinámica una dirección corta, única para dicha red, que se utilizará para todas las comunicaciones en las que forme parte el dispositivo.

5.4.4 Capa de Aplicación

5.4.4.1. Descripción general.

La capa de aplicación se subdivide en la subcapa APS (*Application Support*), la subcapa ZDO (*ZigBee Device Object)* y los objetos de aplicación definidos por cada uno de los fabricantes, denominada AF (*Application Framework*).

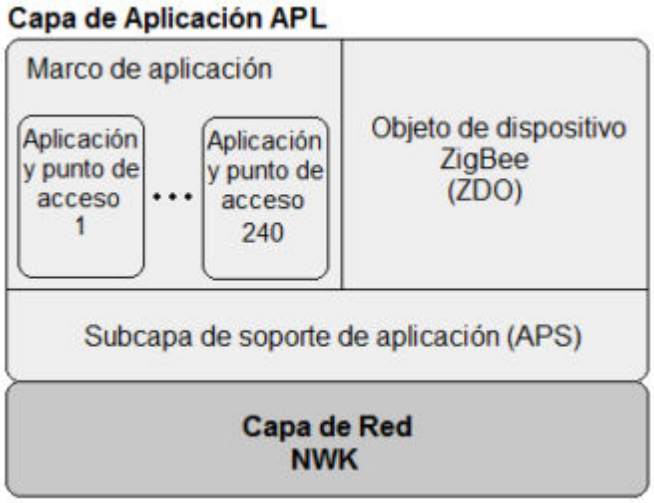


Figura 5. Capa de Aplicación

A continuación se describirá de forma detallada la funcionalidad de cada una de estas subcapas, pero antes se deben definir algunos términos necesarios para comprender en que se basa la funcionalidad de esta capa de aplicación:

• **Perfil de aplicación**

Describe los acuerdos sobre mensajes, formato de mensajes y acciones de procesado que permiten a los desarrolladores crear aplicaciones para un grupo determinado de dispositivos. Este perfil de aplicación permite a las aplicaciones mandar comandos, pedir datos y procesar la información recibida.

• ***Clusters***

Cada *cluster* hace alusión a una determinada aplicación, se identifica por la variable *ClusterID* y es la forma en la que los dispositivos indican a que aplicación en concreto se refieren los comandos o acciones que invocan.

5.4.4.2. Subcapas

Subcapa de soporte de aplicación (APS)

La subcapa de soporte de aplicación proporciona un interfaz entre la capa de red y la capa de aplicación a través de un conjunto de servicios para ser utilizados tanto por la subcapa ZDO como por AF. Estos servicios son dos: los servicios de datos y los de gestión. Algunas de las principales tareas que desarrolla esta capa son:

• **Generación de PDU:** a nivel de aplicación, denominada APDU.

• **Vinculación:** una vez que dos dispositivos están vinculados, la subcapa APS se encarga del intercambio de mensajes entre ambos dispositivos.

• **Filtrado de direcciones:** tiene la habilidad de realizar filtros sobre las posibles direcciones destino para crear subgrupos dentro de la red.

• **Fragmentación:** permite la segmentación y re ensamblado de mensajes de longitud mayor a la carga útil de un mensaje simple de la capa de red.

• **Evita duplicado:** rechaza aquellos mensajes que entran por duplicado a la capa de aplicación.

Subcapa marco de aplicación (AF)

Es el entorno en el cual se gestionan las distintas aplicaciones definidas. Se permiten hasta 240 aplicaciones distintas en un mismo dispositivo, asociadas a los puntos de acceso1 al 240. El punto de acceso 0 está reservado al nivel ZDO. Los puntos

de acceso 241 al 254 se reservan para futuros usos y el 255 se utiliza para comunicaciones de tipo broadcast. En esta subcapa también se definen los diferentes clusters que identificaran a cada una de las aplicaciones, denotados por el consiguiente ClusterID

Subcapa de Objeto de dispositivo ZigBee (ZDO)

Esta subcapa es responsable de las siguientes funciones:

• Inicialización de la subcapa APS, de la capa de red (NWK) y del proveedor de servicios de seguridad (SSP, *Security Service Provider*).

• Definición del tipo de dispositivo dentro de la red (coordinador, router o dispositivo final)

• Gestión de vínculos entre puntos de acceso.

• Asegurar conexiones seguras entre dispositivos.

CAPITULO 6: PERFIL DE LOCALIZACIÓN (Location Profile)

6.1 Introducción

Todos los dispositivos de la red ZigBee deben estar configurados sobre el mismo perfil; son desarrollados por cada uno de los fabricantes ZigBee, que en base a las necesidades que existen en el mercado, proporcionan soluciones tecnológicas específicas. Los perfiles por tanto tratan de unificar la tecnología con las necesidades del mercado.

Se usan para definir la capacidad de aplicación de un dispositivo y manejar los detalles de la aplicación. Un ejemplo de perfil sería el control de iluminación de una casa. Otro ejemplo de perfil seria el que utilizaremos en el desarrollo de este proyecto, el perfil de localización (Location Profile) Desarrollado por Texas Instrument, el cual define el tipo de dispositivos de dicho perfil (Descriptor de dispositivos) y sus aplicaciones (ClusterID).

6.2 Descriptor de dispositivo (Device Descriptor)

Es la delineación de la función de dispositivos dentro de un segmento profile. Por ejemplo, en el perfil de control de iluminación: Lámparas, Switches y otros dispositivos tienen cada uno su descriptor.

En el perfil de localización a usar en este proyecto, quedan definidos tres tipos de dispositivos:

6.2.1 Dongle

Definido en el perfil (Location Profile) como endpoint 203 y programado como dispositivo coordinador en la red ZigBee. Este nodo puede comunicarse con toda la red, puede realizar peticiones o configurar posiciones X e y de los nodos de referencia así como configurar los parámetros A, n de los nodos ciegos.

6.2.2 ReferenceNode (Nodo de referencia)

Definido en el perfil como endpoint 210 y programado como router de la red ZigBee. Este nodo debe estar configurado con los valores X e Y que corresponden a su posición física.

La principal tarea de este nodo es proporcionar las coordenadas X e Y, así como el promedio de valores RSSI obtenidos de las ráfagas del nodo ciego.

Ya que este nodo no utiliza el hardware de localización, no es necesario usar el circuito CC2431 para este dispositivo como se indicará más adelante.

6.2.3 BlindNode (Nodo ciego)

Definido en el perfil como endpoint 211 y programado como router en la red ZigBee. Un nodo ciego se comunica con los nodos de referencia más cercanos, recolectando X, Y, RSSI para cada uno de estos nodos, y calculará su

posición basado en los parámetros de entrada usando el hardware del motor de localización. Después, la posición calculada debería ser enviada a una estación de control, la cual puede estar conectada a un PC.

6.3 Sistema de Localización

7.3.1 Introducción

Una vez definidos los siguientes puntos:

- Red ZigBee, capas, topologías y tipos de dispositivos de la red: Router, coordinador y dispositivo final
- LocationProfile (Perfil de Localización). Perfil implementado por Texas Instrument, el cual define los tipos de dispositivos (Dongle, ReferenceNode y BlindNode) que intervienen junto con sus clusters de aplicación.

Pasaremos a exponer el Sistema de Localización implementado por Texas Instrument bajo el perfil LocationProfile sobre una red ZigBee.

6.3.2 Funcionamiento

El algoritmo de localización usado en el motor de localización del CC2431 está basado en valores del Indicador de Potencia de la señal Recibida RSSI. El valor RSSI disminuirá cuando aumenta la distancia.

La Figura siguiente muestra un sistema simplificado para la detección de la posición. El nodo de referencia es un nodo estático ubicado en una posición conocida. Por simplicidad éste nodo conoce su propia posición y puede decir a los otros nodos donde se encuentra cuando exista una petición.

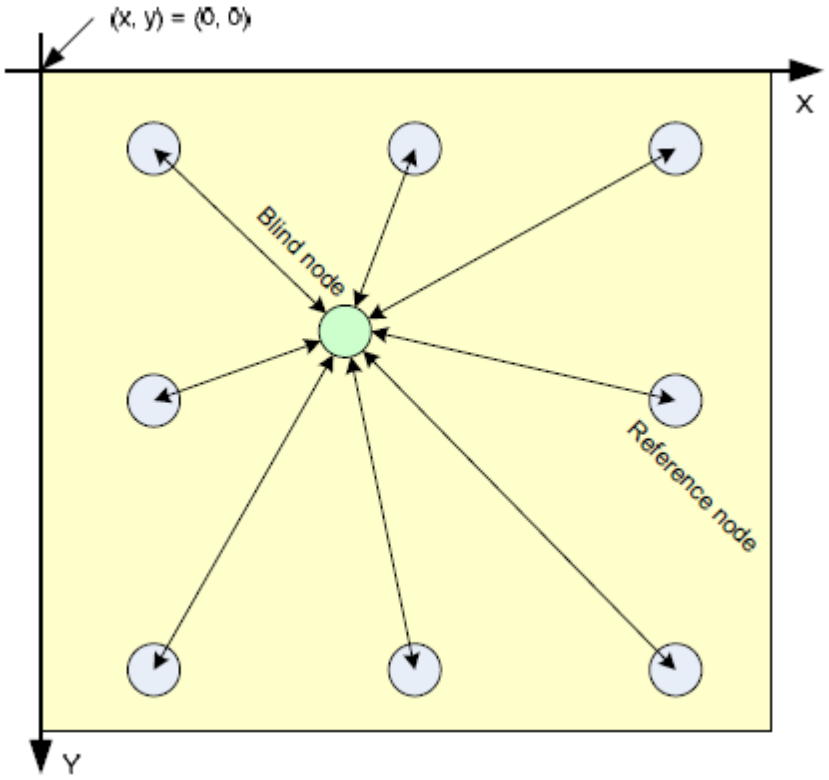


Figura 6. Estimación de la Localización

Un nodo de referencia no necesita implementar el hardware para la detección de la posición ya que no realiza ningún cálculo. El nodo ciego es el nodo implementado con el módulo CC2431EM (módulo que incorpora el hardware de localización). Este nodo recolectará las señales de todos los nodos de referencia que respondan a una petición, leerá los correspondientes valores de RSSI, ingresará dichos valores en el motor de localización, y después de esto leerá la posición calculada y enviará ésta información a una aplicación de control.

La cantidad de datos mínimos contenidos en un paquete enviado desde un nodo de referencia a un nodo ciego deberán ser las coordenadas X e Y seguido del valor RSSI.

La principal característica del motor de localización es que el cálculo de la posición puede ser efectuado en cada nodo ciego, por lo tanto el algoritmo de cálculo es descentralizado. Esta propiedad reduce la cantidad de transferencia de datos en la red, ya que solo la posición calculada es transferida.

Para mapear cada posición en un lugar distinto en un ambiente natural, se usa un sistema de referencia bidimensional, las direcciones serán denotadas como X e Y, siendo X el eje Horizontal e Y el eje Vertical. El circuito CC2431 solo puede manejar dos dimensiones.

6.3.3 Motor de Localización

El motor de localización es usado para estimar la posición de los nodos de una red inalámbrica ad-hoc. Los nodos de referencia existentes tienen coordenadas conocidas, típicamente porque éstos son parte de una infraestructura instalada. Otros son los nodos ciegos, cuyas coordenadas necesitan ser estimadas. Estos nodos generalmente son móviles y unidos a dispositivos que necesitan ser rastreados. En el caso del proyecto habría que hacer el seguimiento de la cámara y los actores para luego determinar las distancias que los separan, con el fin de llevar un foco automático.

El motor de localización implementa un algoritmo de cómputo distribuido que usa los valores de Indicador de Potencia de Señal Recibida (RSSI) de los nodos de referencia conocidos. Realizar los cálculos de localización a nivel de nodo reduce el tráfico de la red y retrasos de comunicación que de otra forma estarían presentes en un enfoque de computación centralizado.

El motor de localización tiene las siguientes características:

- De 3 a 16 nodos pueden ser usados para el algoritmo de estimación de localización.
- Resolución de 0,25 m de lectura de la posición estimada ya que utiliza dos bits para indicar el valor decimal de la estimación.
- El tiempo para estimar la posición de un nodo es de 50 a 13 ms.
- Rango de localización de 64 x 64 metros.
- Mínimo uso de la CPU para correr la estimación de la posición.

El error de la localización depende del entorno (atenuación de la señal), de los obstáculos (atenuación de la señal al atravesar objetos), de la forma de despliegue de los nodos de referencia y del número de estos en un área dada.

6.3.4 Operación del Motor de Localización

El motor de localización requiere un conjunto de 3 a 16 coordenadas de referencia que deben ser ingresadas con un conjunto de otros parámetros medidos. La salida del motor de localización consiste en un par de coordenadas de la posición estimada.

Antes de que cualquier dato sea ingresado, el motor de localización debe ser habilitado escribiendo un 1 en el bit de habilitación LOCENG.EN. Cuando el motor de localización no está en uso, escribir un cero en OCENG.EN reducirá el consumo de energía de CC2431 desconectando la señal de reloj del motor.

La siguiente figura muestra operaciones básicas del motor de localización:

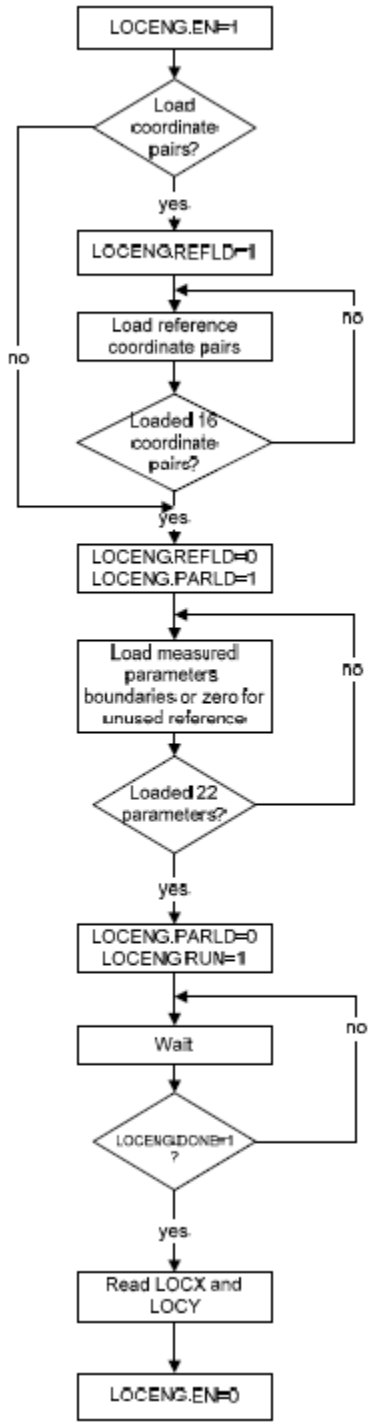


Figura 7. Diagrama de Flujo del motor de localización

6.3.5 Coordenadas de referencia

En motor de localización requiere un conjunto de entre 3 y 16 pares de coordenadas de referencia [x0,y0,x1,y1,…,x15,y115] sean ingresadas. Las coordenadas de referenciad expresan cada una de las posiciones de los nodos de referencia en metros como valores sin signo en el intervalo entre [0, 63.75] metros. La mejor resolución posible de lectura es de 0,25 metros. El formato usado es de datos de de punto fijo con 2 bits menos significativos (LSB) representando la parte fraccional y los restantes 6 bits representando la parte entera, así por ejemplo 63,75 es representado por 0xFF.

Las coordenadas de referencia son cargadas en el registro RF REFCOORD. Antes de escribir en el registro REDCOOD, un 1 debe ser escrito en el bit de registro LOCENG.REFLD para indicar que un conjunto de coordenadas de referencia están siendo escritas. Una vez que comienza el proceso de carga de las coordenadas, 16 pares de coordenadas siempre deben ser escritas, Sin embargo es posible para el motor de localización usar menos de 16 coordenadas de referencia, marcando ciertas coordenadas como no usadas.

6.3.6 Parámetros medidos

Después de haber sido escritos los datos de las coordenadas de referencia, un conjunto de parámetros medidos deben ser ingresados al motor de localización. Estos consisten en los parámetros de radio *A* y *n*, cuatro parámetros de límites de búsqueda y 16 valores RSSI. Los parámetros de radio son los valores *A* y *n*, que son usados en el algoritmo del motor para calcular la posición estimada.

Los parámetros A y n pueden ser usados para describir el entorno de propagación en la cual una red de dispositivos operará.

6.3.6.1 Parámetro *A*

El parámetro *A* es definido como el valor absoluto de la potencia promedio en dBm recibida desde la distancia de referencia de 1 m del transmisor, asumiendo un patrón de radiación omnidireccional. Por ejemplo, si la potencia recibida a 1m es -40dBm, el patrón *A* es especificado como 40.

El motor espera que el parámetro *A* se encuentre en el rango de [30.0, 50.0] con una precisión del 0.5. El parámetro *A* está dado como un valor de punto fijo sin signo donde el bit menos significativo (LSB) es la parte fraccional y los bit restantes son la parte entera. Un valor típico de *A* es 40.0.

6.3.6.1 Parámetro *n* (Coeficiente de propagación de la señal)

El parámetro de radio *n* es definido como el exponente de perdida que describe la señal al decaer la potencia de la señal al incrementar la distancia desde el transmisor. Esta atenuación es proporcional a d^{-n}, donde d es la distancia entre transmisor y receptor.

El valor actual del parámetro *n* escrito en el motor de localización es un valor índice entero seleccionado de una tabla de consulta mostrada a continuación. Por ejemplo, en el caso de que *n*=2.98, encontrado por mediciones, el valor disponible más cercano de *n* en la tabla es de 3.00, correspondiente al índice 13, Por lo tanto, el valor entero 13 es usado para el parámetro *n* escrito en el motor de localización.

n index	n	n index	n
0	1.000	16	3.375
1	1.250	17	3.500
2	1.500	18	3.625
3	1.750	19	3.750
4	1.875	20	3.875
5	2.000	21	4.000
6	2.125	22	4.125
7	2.250	23	4.250
8	2.375	24	4.375
9	2.500	25	4.500
10	2.625	26	4.625
11	2.750	27	5.000
12	2.875	28	5.500
13	3.000	29	6.000
14	3.125	30	7.000
15	3.250	31	8.000

Tabla 1. Valores posibles del parámetro *n*

CAPITULO 7: HERRAMIENTAS DE DESARROLLO EMPLEADAS

En este capítulo se expondrán todas aquellas herramientas que han sido imprescindibles para el correcto desarrollo de este proyecto. Por un lado se detallarán las características del sistema hardware que se ha utilizado, el kit de desarrollo CC2430DK de TI (*Texas Instruments)* y por otro lado se expondrán las herramientas software utilizadas en la comunicación con el hardware.

7.1 Herramientas Hardware

7.1.1 Introducción

Texas Instruments ofrece tres alternativas de arquitecturas hardware con las que crear redes ZigBee/802.15.4:

• Arquitectura basada en un transceptor 802.15.4 y un microprocesador. El transceptor se ocupa de la capa física y el microprocesador aloja la capa MAC, de red y aplicación. Esta estructura otorga una gran flexibilidad en el diseño de red ya que el desarrollador tiene el control de prácticamente toda la pila de protocolos y por tanto puede modificar la mayoría de los parámetros de configuración existentes.

• Arquitectura basada en un procesador que alberga las capas física, MAC y de red y un microprocesador que controla la capa de aplicación. En este caso, el desarrollador solo tiene acceso a la capa de aplicación. Por lo tanto se pierde

flexibilidad a favor de ganar simplicidad y de liberar al microprocesador de las tareas relacionadas con la red.

• Arquitectura SoC (*System on Chip*) en la que tanto el transceptor como el microprocesador están integrados en un único chip.

El kit de desarrollo utilizado en este proyecto pertenece al tercer grupo; está compuesto por módulos CC2430 y CC2431 SoC (*System on Chip*) denominados CC2430EM (Evaluation Module) y CC2431EM. Para los dispositivos que actúen como BlindNodes, en este proyecto en concreto, deberán utilizarse los módulos CC2431, ya que están provistos del hardware de localización necesario para su localización en base a los valores RSSI de los nodos de referencia

El coordinador y los nodos de referencia bien pueden estar implementados en CC2430 ó CC2431.

7.1.2 CC2431EM / CC2430EM

El CC2431EM es un sistema integrado en un chip (*System-On-Chip*) adaptado específicamente para poder ser usado con el estándar IEEE 802.15.4 ZigBee y sus aplicaciones. Permite la construcción de nodos a un precio muy razonable gracias a su bajo coste de fabricación.

Este chip en concreto combina características de dos de sus antecesores, el CC2420 y el CC2430. Respecto al CC2420 se ha mejorado con un microcontrolador 8051 MCU (*Micro Controller Unit*), una memoria flash de 32/64/128KB (en función

del dispositivo) y una memoria RAM de 8KB. Respecto al CC2430, es idéntico excepto por el hecho de añadir un sistema de localización que calcula la posición en función del resto de nodos de la red.

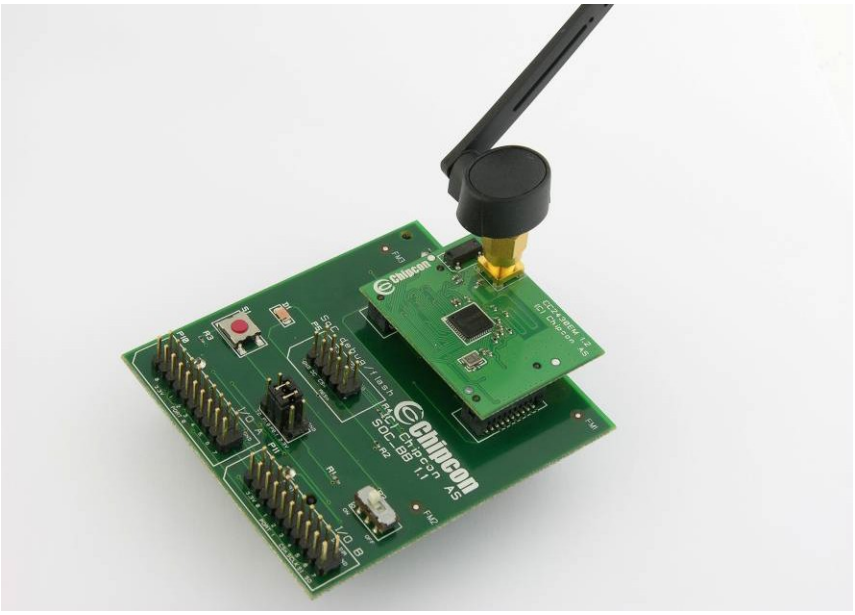


Figura 8. CC2430 acoplado SOC_BB

7.1.3 SmartRF04EB

La placa SmartRF04EB es el puente de comunicación entre el ordenador y los módulos CC2430EM y CC2431EM. Gracias a él podemos comprobar el buen funcionamiento del SoC, depurando el código paso a paso y usando los diversos periféricos que se encuentran conectados.

El modo de conexión es simple: se acopla el SoC (CC2430EM) al SmartRF04EB y se conecta éste último mediante USB al ordenador. El puerto USB proporciona al circuito 3.3V tras pasar por el regulador de tensión.

Figura 9. CC2430 acoplado a SmartRF04EB

Los periféricos que pueden ser utilizados son:

- Conector jack de CC.
- Pantalla LCD.
- Conector RS232.
- Conectores SMA.
- Potenciómetro.
- Salida de audio.
- Entrada de audio.
- Control de volumen.

- Conectores de I/O.
- 4 LED’s.
- Pulsadores.
- 1 Joystick con 5 posiciones: arriba, abajo, izquierda, derecha, centro.

Sin embargo, no todos estos periféricos pueden ser implementados por el CC2431. Por ejemplo, de los 4 LEDs sólo son funcionales el primero y el tercero. También cabe destacar que muchas de las conexiones de los periféricos están conectadas con algunos de los 40 pins de I/O. En la siguiente tabla se muestra la relación de estas conexiones:

Pin	Function
1	N/C
2	N/C
3	P0_0/MIC_IN
4	VDD
5	VDD
6	N/C
7	P0_1/BUTTON_PUSH
8	N/C
9	P0_2/UART_RD
10	N/C
11	P0_3/UART_TD
12	N/C
13	P0_4/RTS
14	N/C
15	P0_5/JOY_PUSH
16	N/C
17	P0_6/JOY
18	N/C
19	P0_7/POT
20	GND

Pin	Function
1	N/C
2	N/C
3	VDD
4	P2_0/LED4
5	P1_0/LED1
6	P2_1/DD
7	P1_1/PWM_OUTPUT
8	P2_2/DC
9	P1_2/LED2
10	P2_3/SDA
11	P1_3/LED3
12	P2_4/SCL
13	P1_4/CSn
14	N/C
15	P1_5/SCLK
16	RESET_N
17	P1_6/MOSI
18	Debug Data Direction(DD_DIR)
19	P1_7/MISO
20	GND

Tabla 2. Conexiones entre los pins del SoC (CC2431EM) y la placa Smart04EB

- **Drivers:** Instalar drivers USB CEBAL (Setup_SmartRF_Drivers-1.2.0) e ir al administrador de dispositivos y seleccionar los más antiguos, si no, tanto IAR como Z-Tool no reconocerán la placa SmartRF04EB.

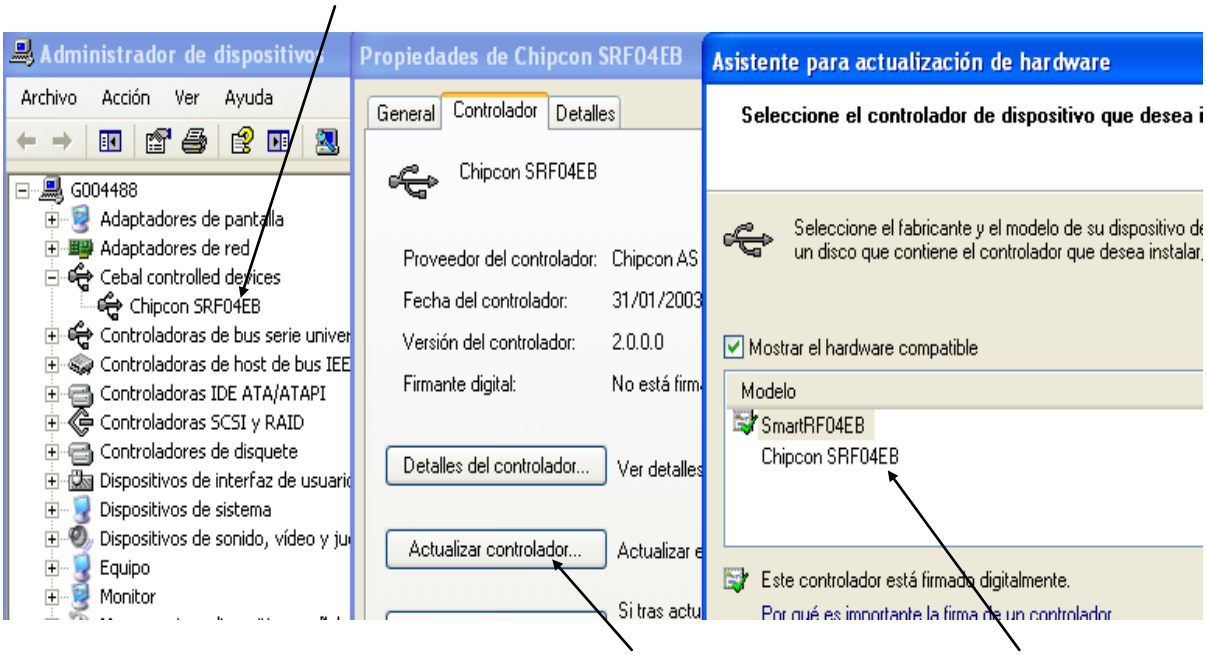


Figura 10. Actualizar CEBAL

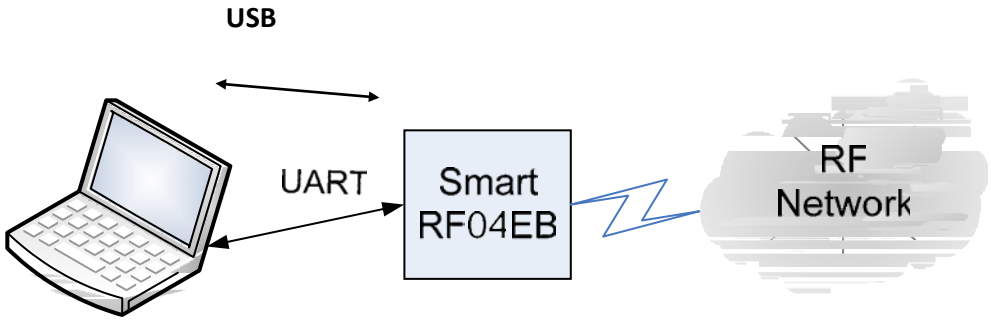


Figura 11. Esquema comunicación PC y red ZigBee

7.1.4 SOC_BB

La funcionalidad del SOC_BB (*Battery Board*) es la de alimentar el SOC para que funcione sin necesidad de estar conectado a ningún ordenador. Tiene 2 pilas AA de 1.5V que lo alimentan, y sus funcionalidades son reducidas respecto al SmartRF04EB, ya que solo dispone de un botón y un LED, además del interruptor de encendido/apagado

Figura 12. SOC_BB con CC2431EM acoplado

7.1.5 SOC_DEM debug plug in board

Modulo que permite programar los dispositivos y realizar la depuración en la propia placa (ICE In Circuit Emulator). Se requiere tener instalado IAR Embedded Workbench, desde el cual ejecutaremos la operación, programar los módulos CC2430EM y CC2431EM acoplados a SOC_BB mediante el software IAR Embedded Workbench comunicado a través del puerto serie y USB a la placa SmartRF04EB.

Los pasos a seguir en la programación y depurado de los módulos se detallarán en la sección perteneciente al herramientas software.

Figura 13. Tarjeta SOC_DEM

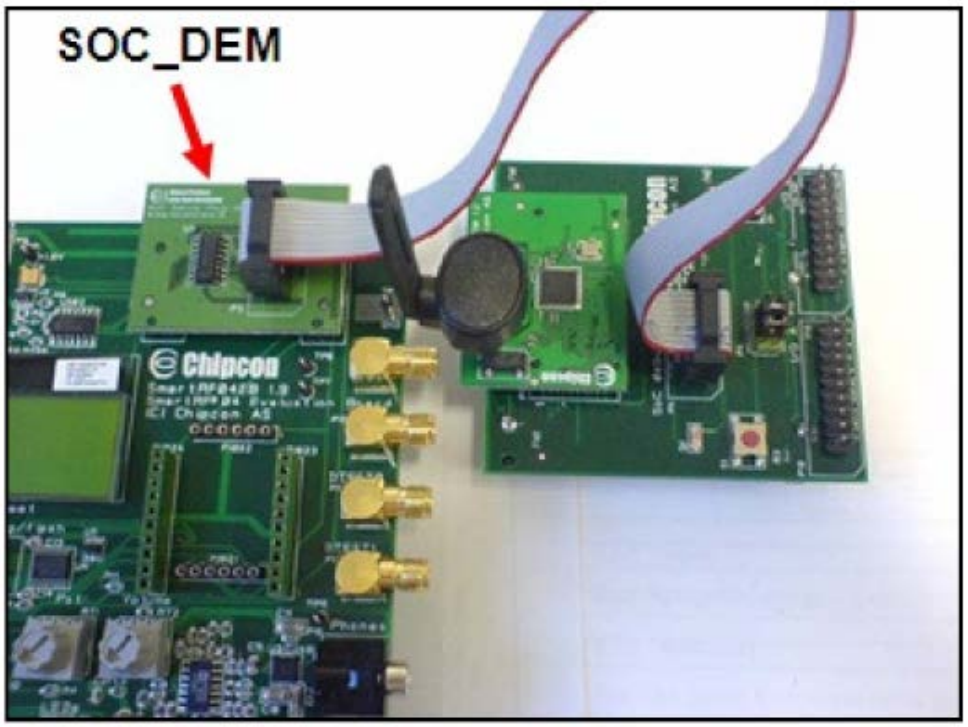


Figura 14. Conexión SOC_DEM

7.2 Herramientas Software

7.2.1 IAR Embedded Workbench 7.30b

IAR es un entorno desarrollado para trabajar sobre microprocesadores MSP430 y en nuestro caso con micros 8051. Esta herramienta permite programar dichos microprocesadores en un lenguaje de alto nivel (C, C++), mucho más cómodo para el desarrollador, traduciéndolo a un lenguaje ensamblador entendible por los microprocesadores. Además ofrece una serie de funcionalidades como la simulación de

código, inserción de puntos de ruptura o seguimiento de variables que lo convierten en una herramienta muy útil para obtener un funcionamiento optimo del sistema a crear.

Para la realización de este proyecto se ha utilizado la versión 7.30b de la herramienta, La versión de pila utilizada con el proyecto Location (ZStack-1.4.3-1.2.0) y su correspondiente perfil (LocationProfile) solo funciona con esta versión, así que asegurarse de tenerla instalada. El interfaz con el usuario tiene el siguiente aspecto:

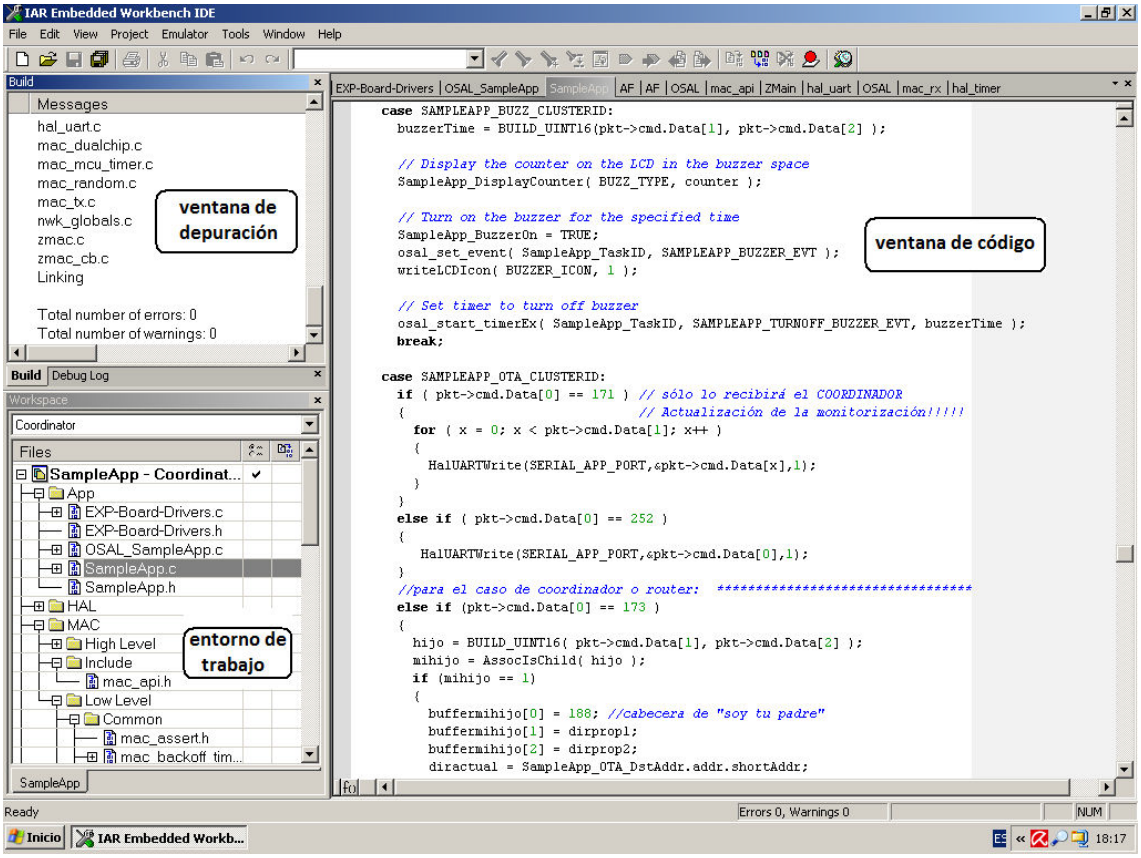


Figura 15. Pantalla principal de IAR Embedded Workbench

Como se puede observar en la figura 14, el interfaz con el usuario consta principalmente de 3 ventanas:

- Ventana de entorno de trabajo, donde se muestran las carpetas y los ficheros de código que componen el total del sistema.
- Ventana de código, en la que se visualiza e introduce por parte del desarrollador el código necesario para conseguir el funcionamiento deseado.
- Ventana de depuración, donde se detallan los posibles errores o avisos detectados al compilar el código y fallos en el ensamblaje del conjunto de códigos que completan el sistema.

Una vez que el código ha sido correctamente cargado en el microprocesador, aparece un segundo interfaz en tiempo de ejecución, cuyas ventanas y funciones más características se exponen y comentan a continuación:

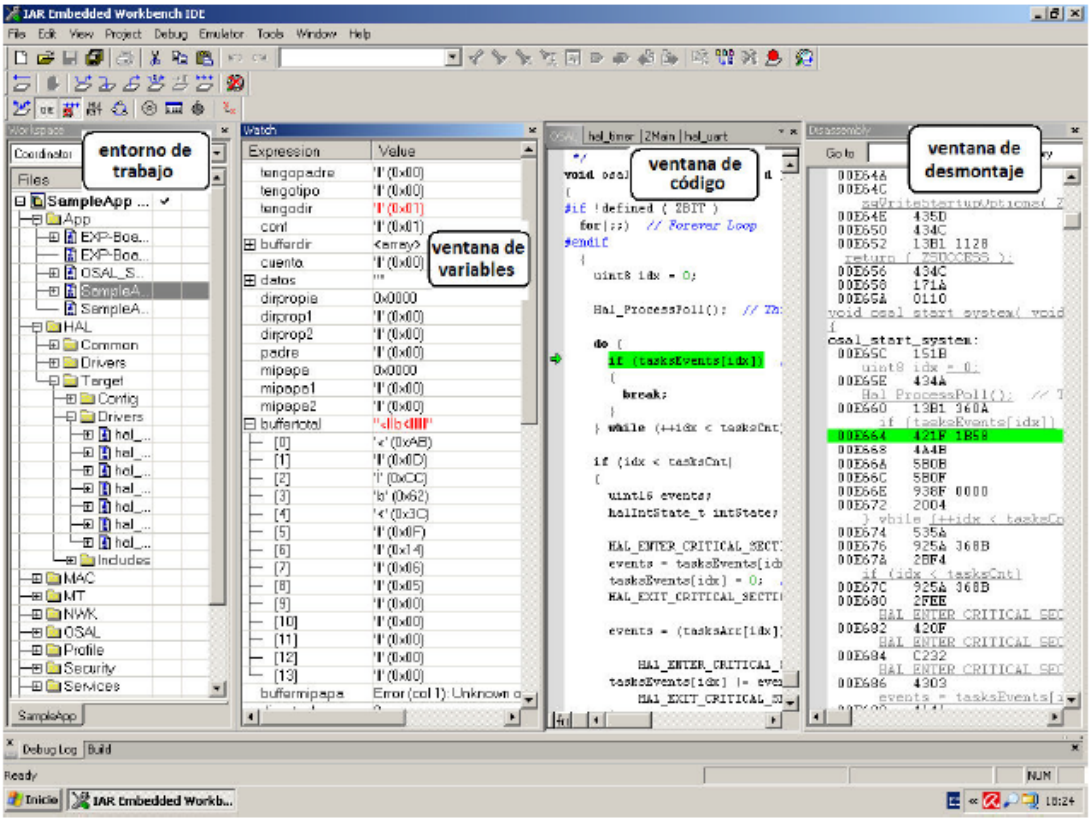


Figura 16. Vista del depurador del IAR Embedded Workbench

Desde esta vista se puede ejecutar normalmente el código, o bien, en el caso de ser necesario, ejecutar el programa instrucción a instrucción para monitorizar el valor de las variables deseadas. Además, permite introducir hasta ocho puntos de ruptura con los que hacer seguimiento de los pasos que realiza el código en ejecución y detectar posibles anomalías.

Las principales ventanas visibles en este interfaz son:

- Ventana de entorno de trabajo.
- Ventana de código, en el que se visualiza instrucción a instrucción la evolución del código.

- Ventana de desmontaje, donde es posible ver la traducción de código de alto nivel a código ensamblador.
- Ventana de variables. Aquí se pueden visualizar aquellas variables globales cuyo valor en un instante determinado sea de interés.

7.2.2 Z-Tool 2.0

Antes de comenzar con la descripción del software, Z-Tool 2.0 trabaja con la plataforma .NET 2.0, por lo que debemos asegurarnos de que las versiones posteriores estén desinstaladas. Para ello ir al panel de control de Windows y abrir "Agregar o quitar programas". En caso de que hubiese versiones posteriores, eliminarlas y en caso de no tener instalado .NET, instalar la versión .NET 2.0.

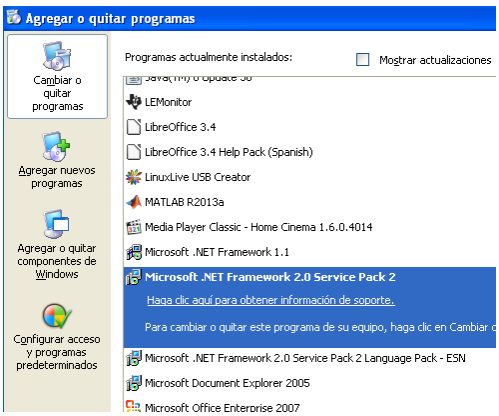


Figura 17. Plataforma .NET 2.0

En general, Z-Tool está pensado para trabajar con cualquier dispositivo que implemente la pila de protocolos Z-Stack de Texas Instruments. En la imagen vemos la aplicación Z-Tool en ejecución. Sus distintas partes son:

- Monitorización. En la subventana derecha se muestran todos los comandos, que recibe el MSP430 por parte del CC2430EM. Cuando el usuario hace que el MSP430 envíe un comando al CC2430 éste también aparece por pantalla.
- Selección de comando. Cuando queremos enviar un comando determinado al CC2430EM acoplado a la SmartRF04EB que actúa como Dongle, debemos previamente seleccionarlo en la subventana de selección de comando. En dicha ventana podemos ver todos los comandos agrupados según el tipo de interfaz: SYS, AF o ZDO. Pantalla situada en la parte superior izquierda.
- Configuración de comando. Algunos comandos deben ser configurados antes de ser enviados, se describirán más detalladamente los comandos utilizados en el Jscrip desarrollado junto con sus respectivos atributos a configurar. Pantalla situada en la parte inferior izquierda.

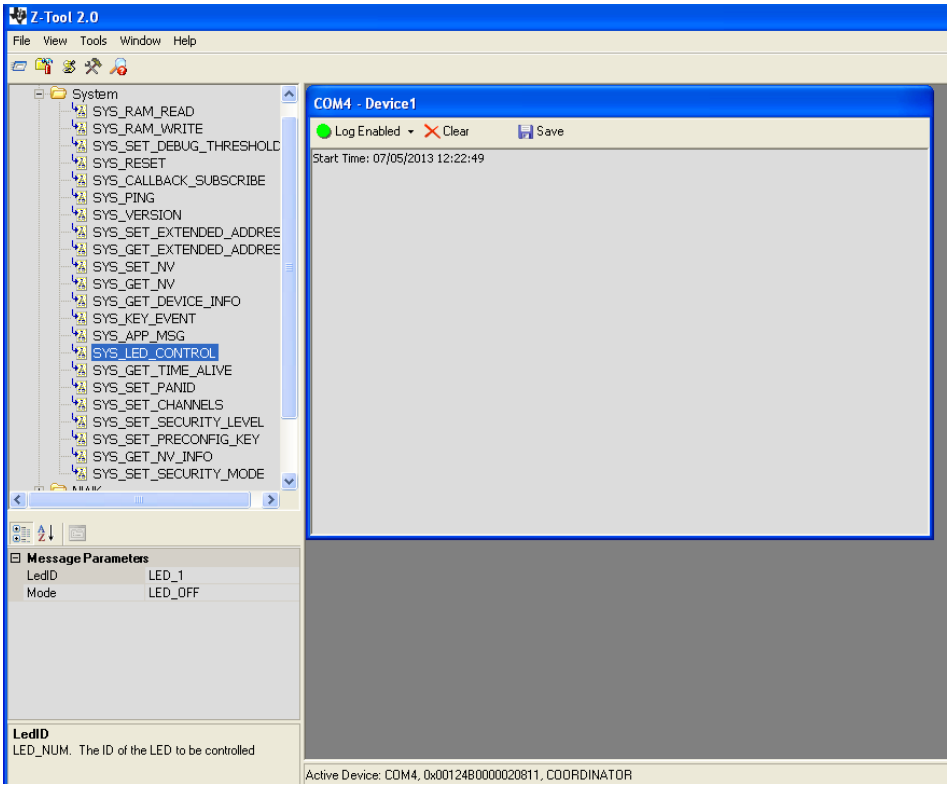


Figura 18. Pantalla Principal Z-Tool

Una vez abierta la aplicación Z-Tool, esta comenzará a intentar localizar la plaza SmartRF04EB, en el caso de no localizarla deberemos configurar el puerto serie, ya que este es el puente de comunicación entre la red ZigBee y el PC. Para ello debemos dirigirnos al menú *tolls* y seleccionar *Settings*. Aparecerá una nueva ventana de configuración.

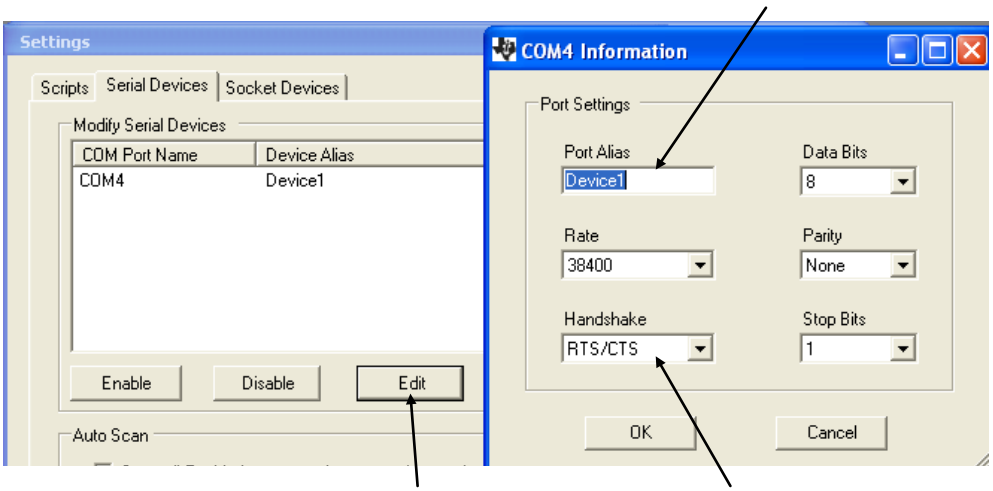


Figura 19. Configuración Serie

Editamos el puerto serie seleccionado la velocidad a 38400 y Handshake en RTS. El nombre introducido en Port Alias será el que se usará en el Jscrip para la selección de puerto a la hora enviar comandos al Dongle.

7.2.3 MATLAB

En un principio se opto por seguir con la plataforma .NET y usar VisualSudio, programando la monitorización mediante Visual Basic ya que proporciona un entorno de trabajo mediante eventos muy sencillo de implementar.

Ya que Z-Tool no admitía plataformas .NET posteriores a 2.0, se decidió instalar VisualStudio 2005. Finalmente seguía habiendo problemas de incompatibilidades entre

Z-Tool y VisualStudio por lo que se escogió realizar la parte de monitorización del proyecto en MATLAB.

Matlab es una herramienta de software matemático que ofrece un entorno de desarrollo integrado (IDE) con un lenguaje de programación propio. En el actual proyecto, ha sido utilizado para crear el interfaz grafico de comunicación con una base de datos Access, en la cual se van guardando los valores que el motor de comunicación creado en Jscript y ejecutado en Z-Tool va almacenando.

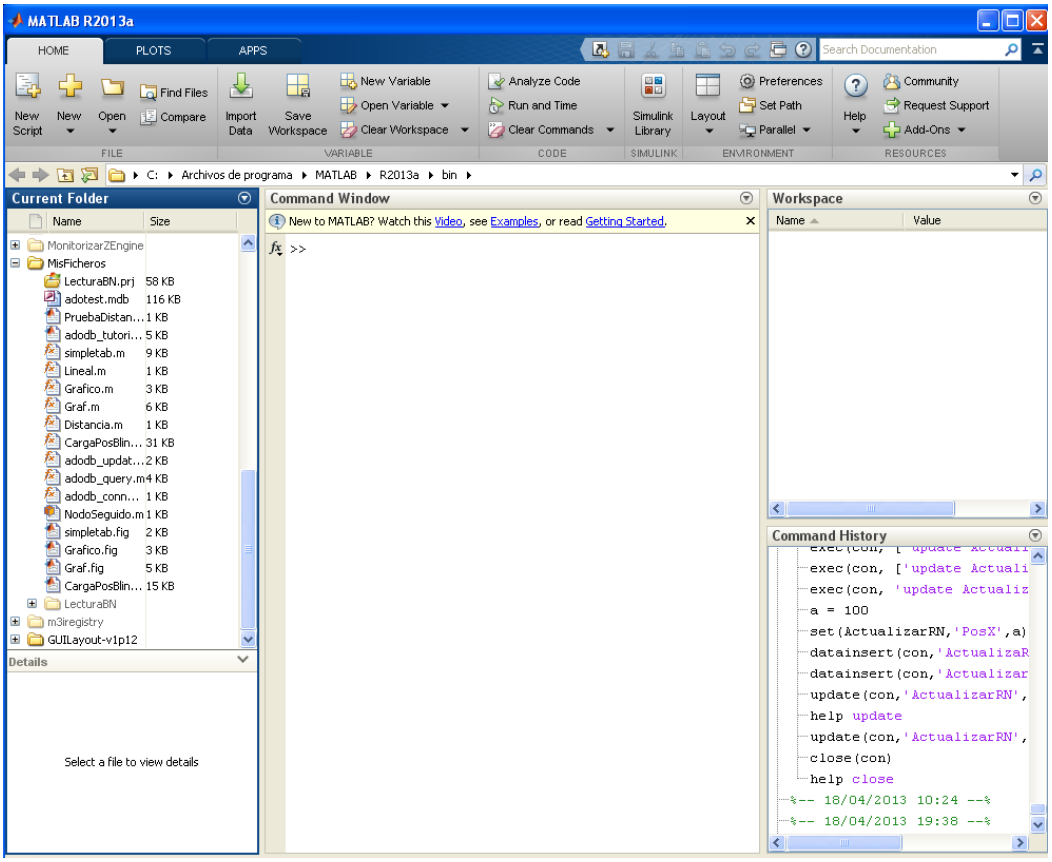


Figura 20. Pantalla principal MATLAB

La versión usada es Matlab R2013a, aunque se puede utilizar una versión anterior sin ningún tipo de problemas.

7.2.4 MS Access 2007

A pesar de usar Access 2007, a la hora de seleccionar el formato de la base de datos hemos optado por Access 2003 con la extensión mdb ya que en el Jscrip ejecutado en Z-Tool así lo requería.

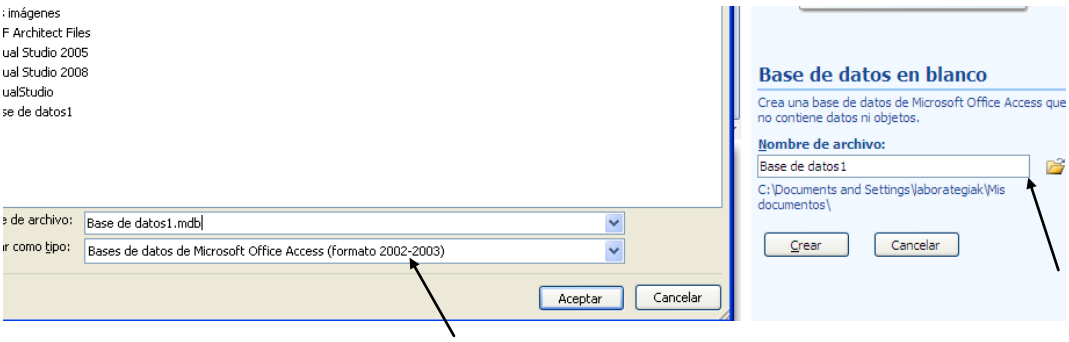


Figura 21. Formato ".mdb" en Access

7.2.5 ZStack-1.4.3-1.2.0

Versión de de protocolos de pila y proyectos realizados sobre esta que instala Z-tool 2.0. En este proyecto se trabajará con el proyecto Location, el cual tiene un perfil definido como LocationProfile y cuyos dispositivos se describieron en apartados anteriores. Dispositivos Dongle, ReferenceNode y BlindNode sobre una red ZigBee, también expuesta en apartados anteriores.

La ruta del espacio de trabajo (IAR Embedded Worbench) del proyecto Location se encuentra en:

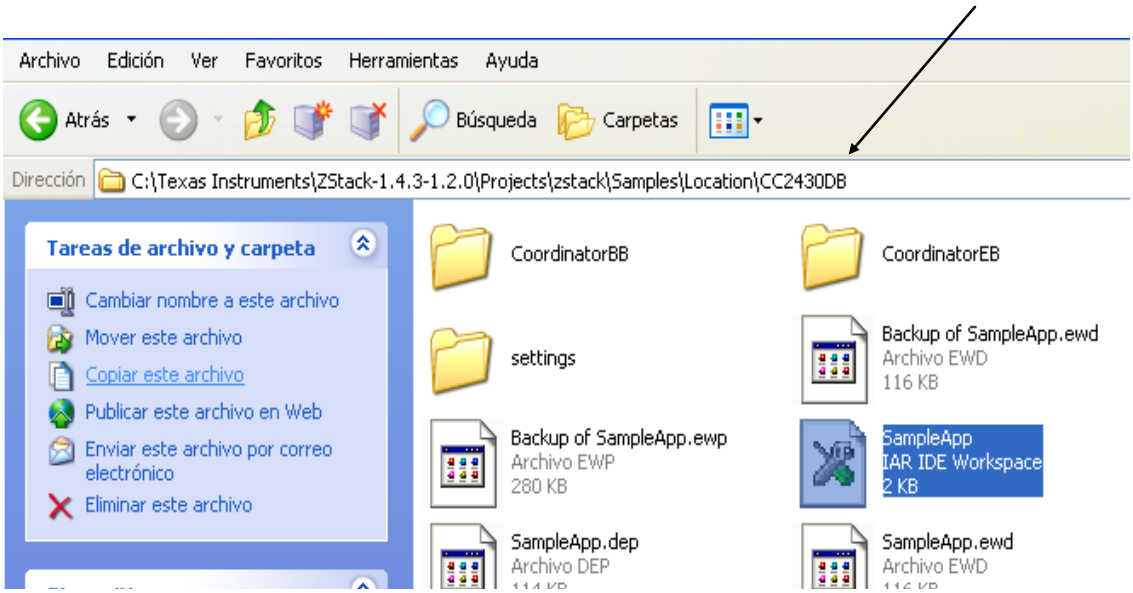


Figura 22. Ruta del espacio de trabajo

Una vez hagamos doble click, se nos abrirá el espacio de trabajo con el proyecto en IAR Embedded Worbench.

CAPITULO 8: DISEÑO E IMPLEMENTACIÓN

8.1 Programación Hardware -Módulos CC2430EM / CC2431EM

En el siguiente apartado se explicará como se programan los Módulos CC2431EM y CC2431EM de tal modo que ejerzan el Rol de dispositivo ZigBee que les corresponda (Router, EndNode, Coordinador) y cumplan con la función de dispositivo dentro del perfil LocationProfile oportuno (ReferenceNode, BlindNode, Dongle).

8.1.1 Programación CC2430 como Dongle

8.1.1.1 Conexión módulos

El nodo Dongle está implementado como coordinador de la red ZigBee ejerciendo el Rol de dispositivo Dongle dentro del perfil LocationProfile, usaremos un módulo CC2430EM acoplado a SOC_BB, el cual se conectará a la placa Smart04EB mediante SOC_DEM, tal y como se indica en la figura 13. De este modo podremos comenzar la programación y depuración mediante IAR Embedded Workbench.

8.1.1.2 Programar Dongle

- Abrir IAR Embedded Workbench y seleccionar dentro la de la carpeta Location el proyecto SampleApp, el cual contiene el programa en C++

para programar los distintos dispositivos del sistema Location. La versión de pila utilizada es ZStack-1.4.3-1.2.0

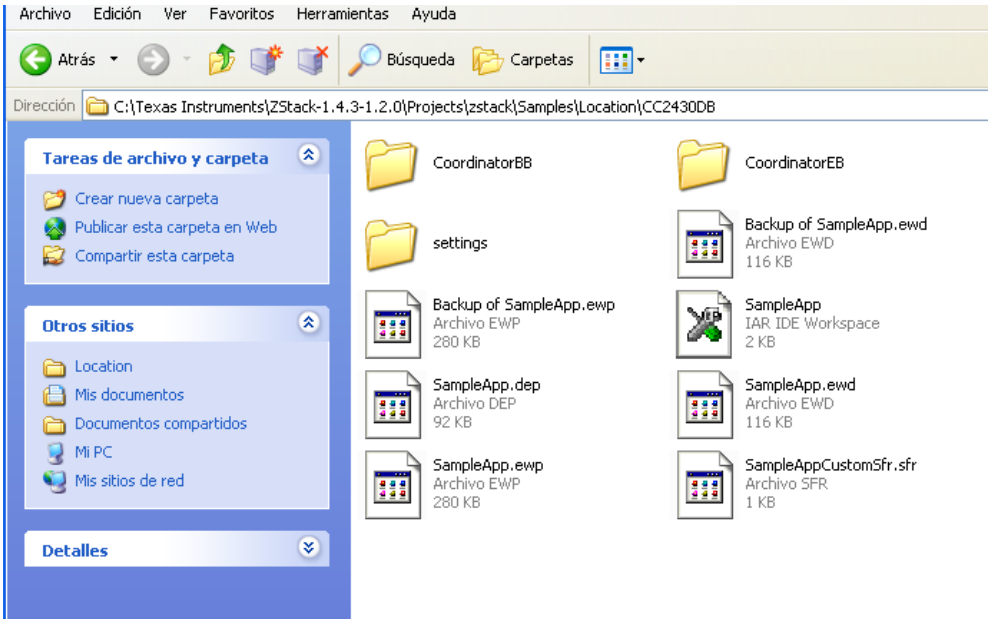


Figura 23. Ubicación proyecto a abrir en IAR

- Seleccionar el tipo de dispositivo a crear dentro de la red ZigBee, en este caso corresponde al coordinador.

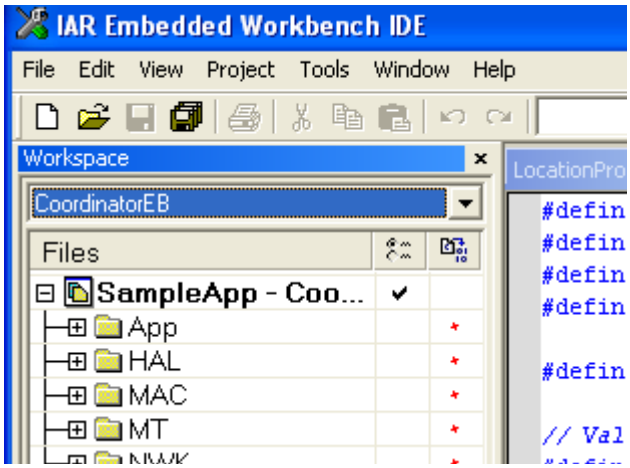


Figura 24. Selección ZDO (ZigBee Device Objec)

- Seleccionar el tipo de dispositivo a crear del perfil LocationProfile, en este cado es un dispositivo Dongle.

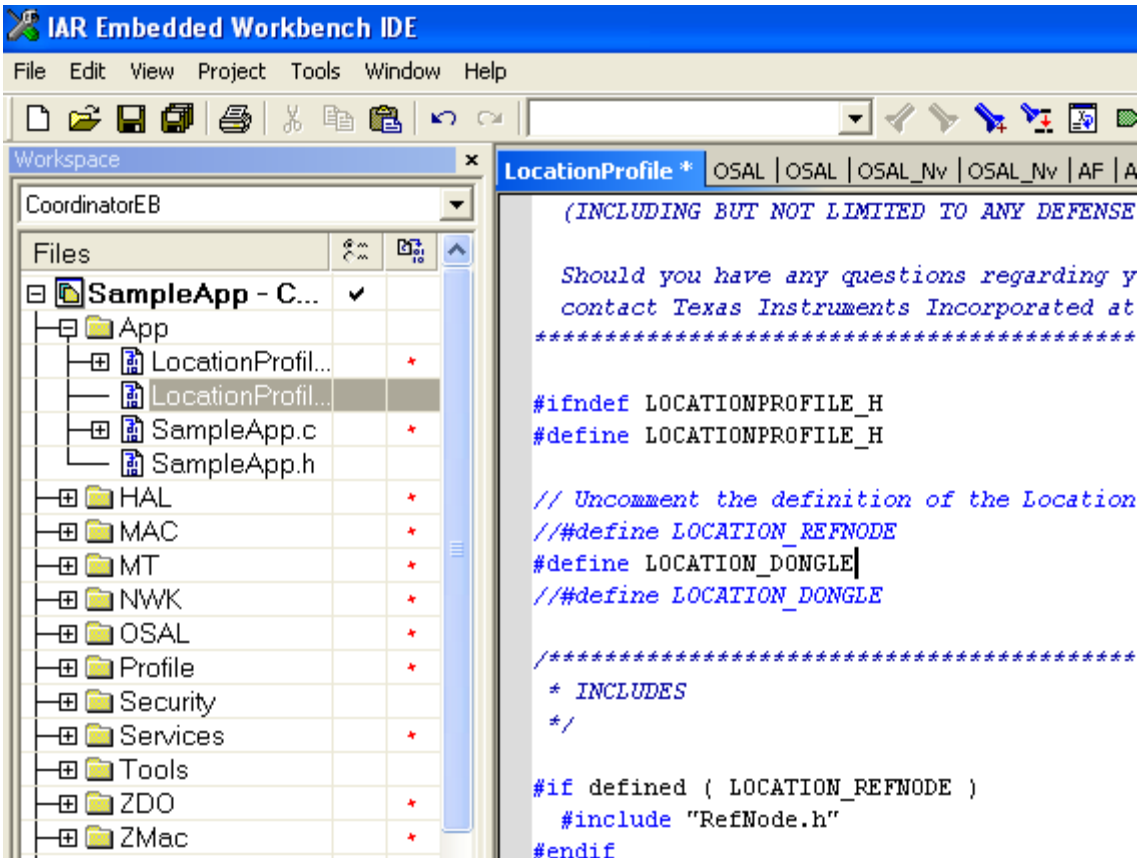


Figura 25. Selección del Rol del Dispositivo dentro del perfil LocationProfile

- Aumentamos el tamaño de memoria de programa para poder incorporar funcionalidades de las capas y posteriormente poder usarlas a nivel aplicación en Z-Tool. Para ello abriremos el archivo f8w2430.xcl dentro de la carpeta Tools y modificaremos la sección D_CODE_END=2x28FF, incrementando en 0x400 bytes el tamaño de memoria de código de programa. De tal manera que quedaría en D_CODE_END=0x2CFF

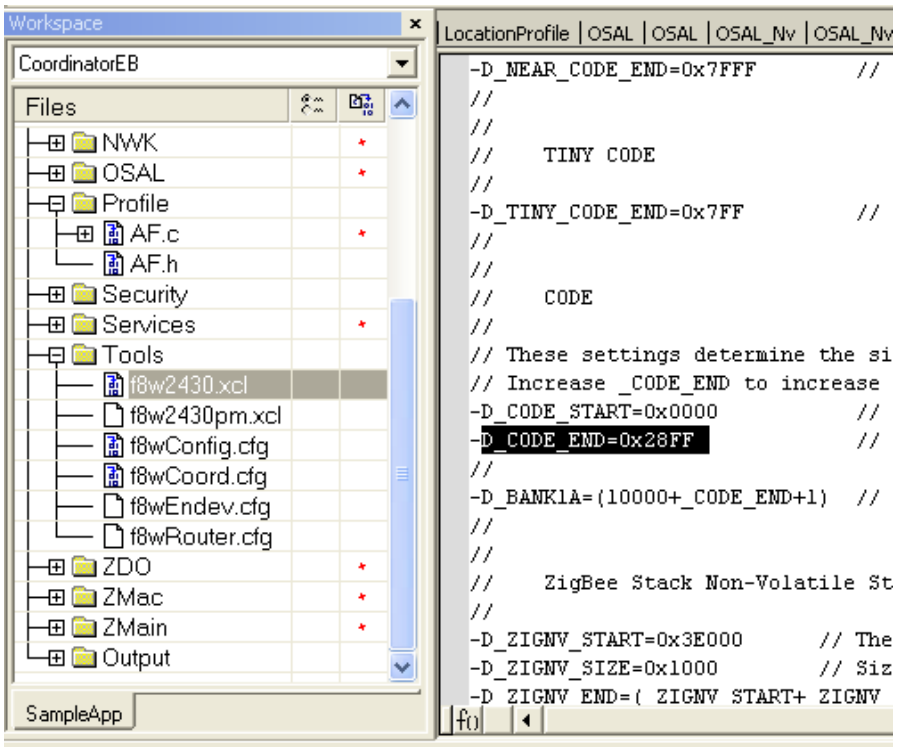


Figura 26. Tamaño de memoria de código de programa.

- Incorporamos las funciones de capa que luego podrán ser usadas en Z-Tool. Para ello vamos al menú Projec y seleccionamos options. Nos aparecerá una nueva ventana de la que seleccionamos C/C++ Compiler y nos dirigimos a la pestaña preprocessor.

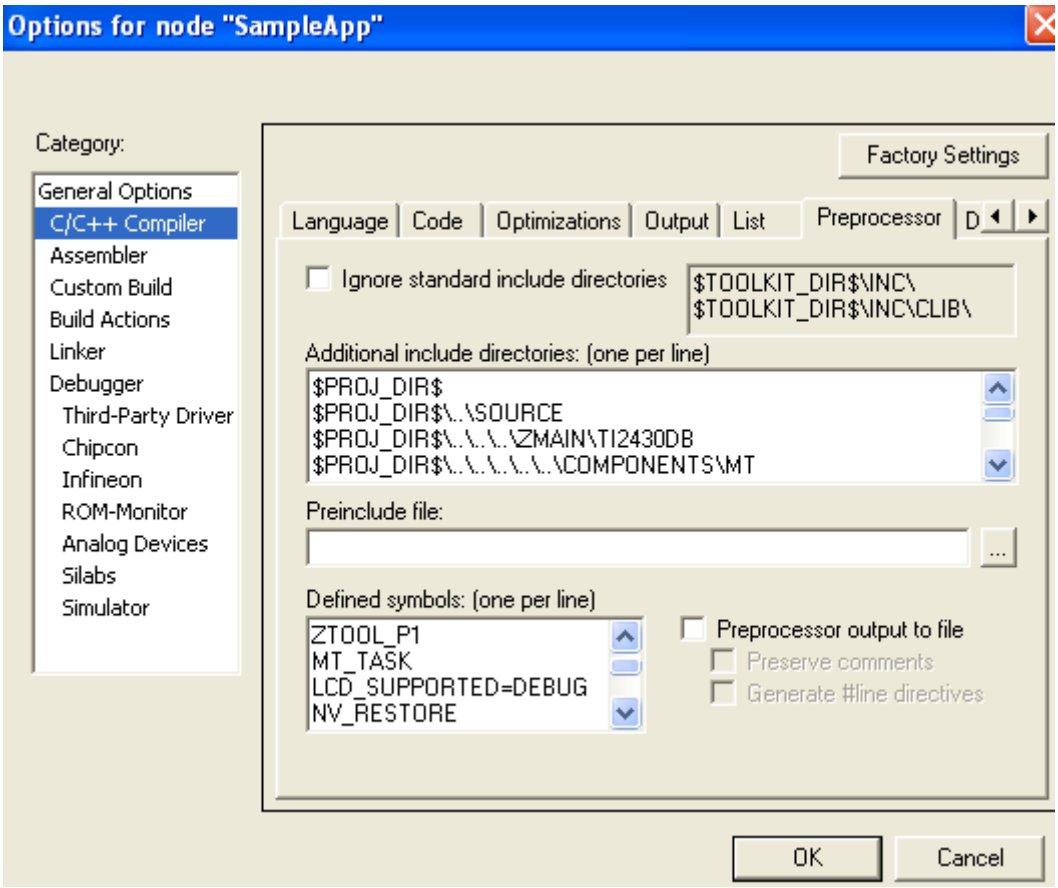


Figura 27. Añadir funcionalidades al Dongle

Dentro del cuadro “Defined symbols”, habría que incorporar todas las siguientes líneas de la tabla2, excepto las de la capa Mac.

MT_TASK	Enable Monitor-Test task and debug functionality. This must be set for using Z-Tool
MT_AF_FUNC	Enable Monitor-Test processing of AF commands issued from Z-Tool.
MT_AF_CB_FUNC	Enable Monitor-Test processing of AF callbacks issued from Z-Tool.
MT_MAC_FUNC	Enable Monitor-Test processing of MAC commands issued from Z-Tool
MT_MAC_CB_FUNC	Enable Monitor-Test processing of MAC callbacks issued from Z-Tool
MT_NWK_FUNC	Enable Monitor-Test processing of NWK commands issued from Z-Tool
MT_NWK_CB_FUNC	Enable Monitor-Test processing of NWK callbacks issued from Z-Tool
MT_ZDO_FUNC	Enable Monitor-Test processing of ZDO commands issued from Z-Tool
MT_ZDO_MGMT	Enable Monitor-Test processing of ZDO MGMT commands issued from Z-Tool
MT_USER_TEST_FUNC	Enable Monitor-Test processing of User commands issued from Z-Tool
MT_SEQ	Enable Monitor-Test command sequences to be used.

Tabla 3. Funcionalidades Z-Tool

- Una vez realizadas las operaciones anteriores, el sistema está listo para ser programado y hacer la depuración. Para ello abrimos el menú Projec y seleccionamos la opción Debug. A continuación se comenzará a programar el módulo CC2430EM y al finalizar se ejecutará la depuración del programa.

El dispositivo coordinador de la red ZigBee programado como Dongle del perfil de pila ya está listo.

8.1.1.2 Dongle en ejecución

Una vez programado el módulo que actuará como Dongle, podremos ponerlo en funcionamiento como se indica a continuación.

El nodo Dongle esta implementado como coordinador de la red ZigBee ejerciendo el Rol de dispositivo Dongle dentro del perfil LocationProfile. Usaremos el módulo ya programado CC2430EM acoplado directamente a la placa Smart04E como se muestra en la figura 8. En este proyecto en concreto el módulo CC2430EM hará de coordinador de la red ZigBee, el cual se comunicará con el PC a través del puerto serie que ofrece la placa SmartRF04EB.

8.1.2 Programación CC2430 como ReferenceNode

8.1.2.1 Conexión módulos

El nodo ReferenceNode está implementado como Router de la red ZigBee ejerciendo el Rol del dispositivo ReferenceNode dentro del perfil LocationProfile. Usaremos un módulo CC2430EM acoplado a SOC_BB, el cual se conectará a la placa Smart04EB mediante SOC_DEM. Tal y como se indica en la figura 9. De este modo podremos comenzar la programación y depuración mediante IAR Embedded Workbench.

8.1.2.2 Programar ReferenceNode

- Abrir IAR Embedded Workbench y seleccionar dentro la de la carpeta Location el proyecto SampleApp, el cual contiene el programa en C++ para programar los distintos dispositivos del sistema Location. La versión de pila utilizada es ZStack-1.4.3-1.2.0

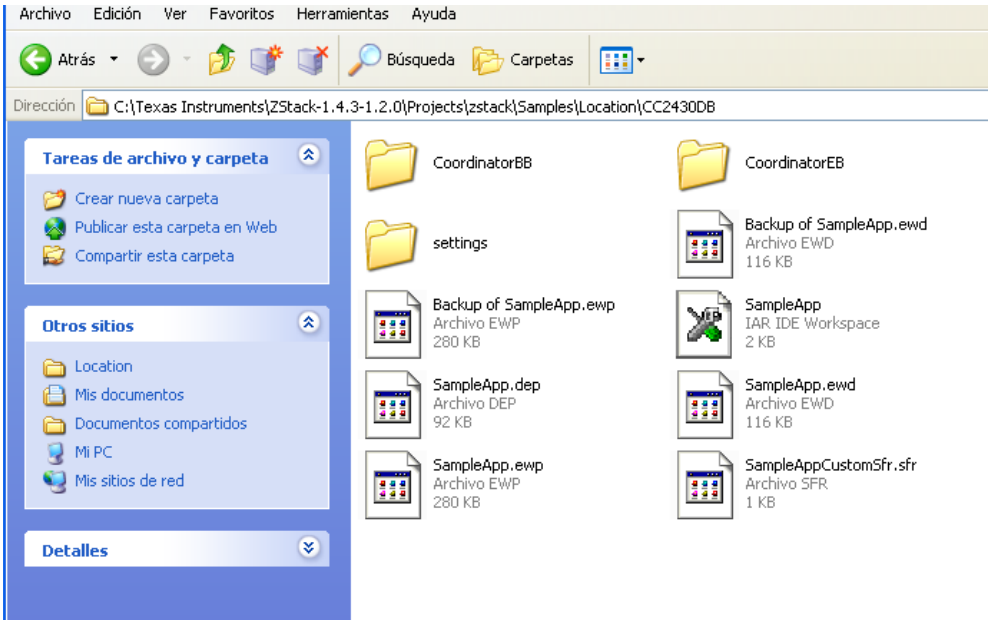


Figura 28. Ubicación proyecto a abrir en IAR

- Seleccionar el tipo de dispositivo a crear dentro de la red ZigBee, en este caso corresponde al Router.

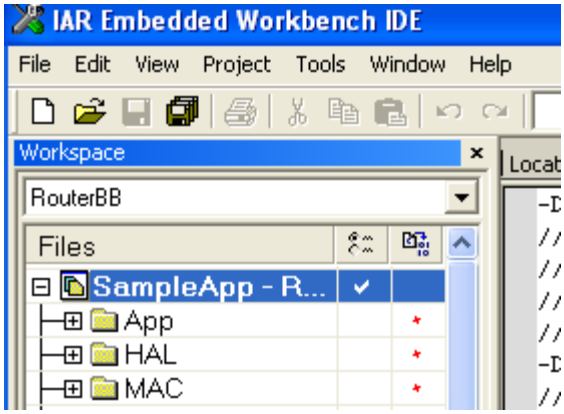


Figura 29. Selección ZDO

- Seleccionar el tipo de dispositivo a crear del perfil LocationProfile, en este cado es un dispositivo ReferenceNode.

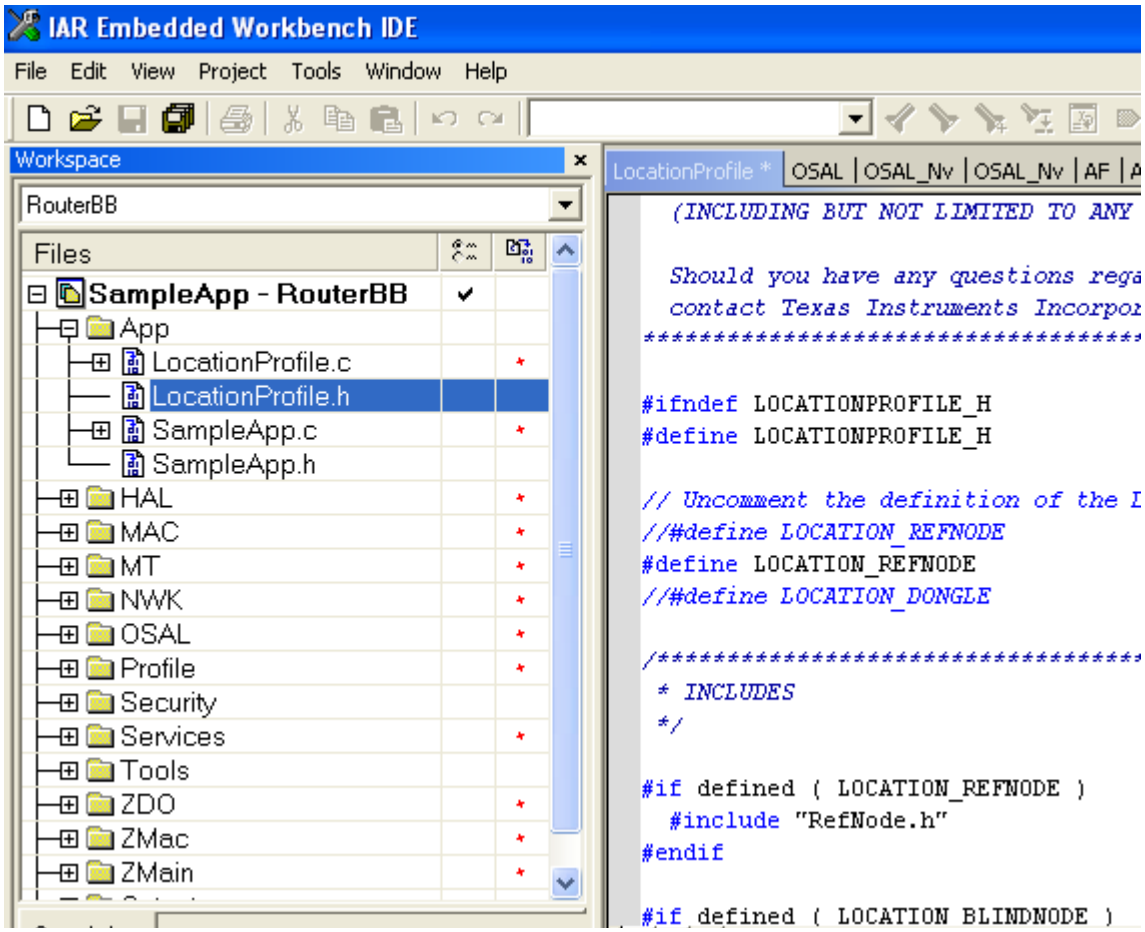


Figura 30. Selección dispositivo dentro del LocationProfile

- Una vez realizadas las operaciones anteriores, el sistema está listo para

programar y hacer la depuración. Para ello nos dirigiremos abrimos el menu Projec y seleccionaremos la opción Debug. A continuación se comenzará a programar el módulo CC2430EM y al finalizar se activará la depuración del programa.

El dispositivo Router de la red ZigBee programado como ReferenceNode del perfil de pila ya está listo.

8.1.2.3 ReferenceNode en ejecución

Una vez programado el módulo que actuará como ReferenceNode, podremos ponerlo en funcionamiento como se indica a continuación.

Para los nodos que cumplan funciones de Router en la red ZigBee y actúen como nodos de referencia dentro del perfil LocationProfile, usaremos módulo CC2430EM sobre la Placa SOC_BB (Batery Board). Esta Placa permite distribuir los módulos alimentados por baterías.

8.1.3 Programación CC2431 como BlindNode

8.1.3.1 Conexión módulos

El nodo BlindNode esta implementado como EndDevice de la red ZigBee ejerciendo el Rol de dispositivo BlindNode dentro del perfil LocationProfile. Usaremos un módulo CC2431EM acoplado a SOC_BB, el cual se conectará a la placa Smart04EB mediante SOC_DEM, tal y como se indica en la figura 13. De este modo

podremos comenzar la programación y depuración mediante IAR Embedded Workbench.

8.1.3.2 Programar BlindNode

- Abrir IAR Embedded Workbench y seleccionar dentro de la carpeta Location el proyecto SampleApp, el cual contiene el programa en C++ para programar los distintos dispositivos del sistema Location. La versión de pila utilizada es ZStack-1.4.3-1.2.0

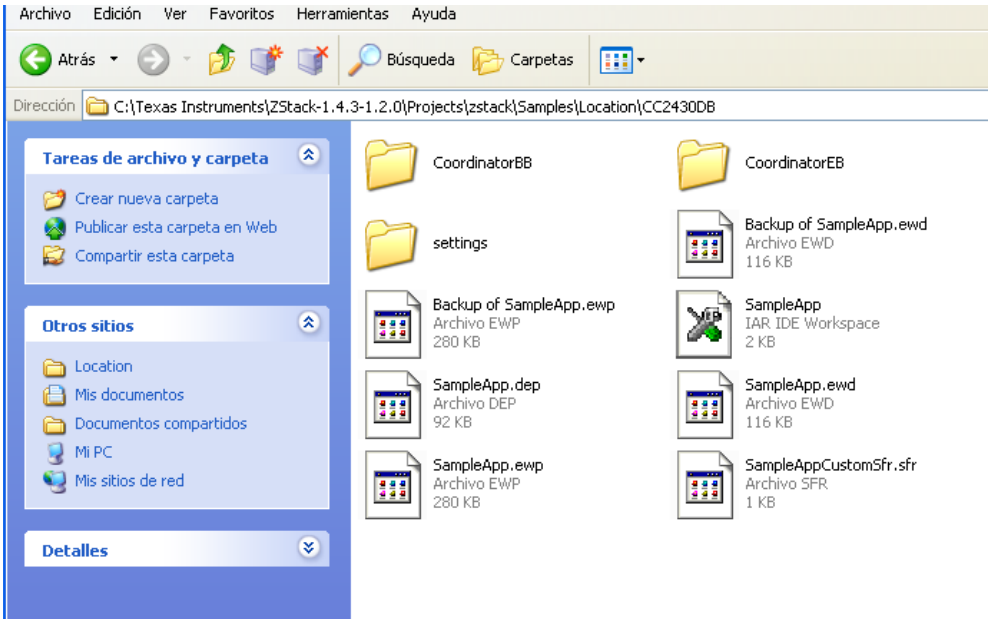


Figura 31. Ubicación proyecto a abrir en IAR

- Seleccionar el tipo de dispositivo a crear dentro de la red ZigBee, en este caso corresponde a EndDevice.

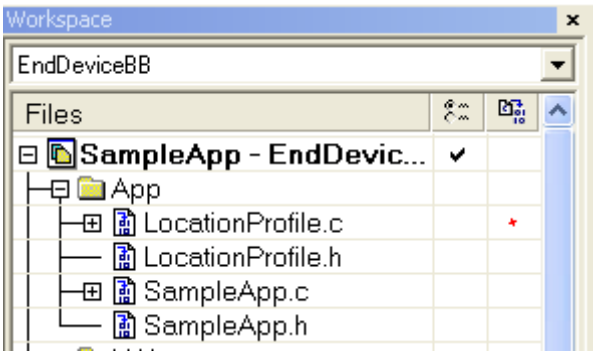


Figura 32. Selección ZDO

- Seleccionar el tipo de dispositivo a crear del perfil LocationProfile, en este cado es un dispositivo BlindNode.

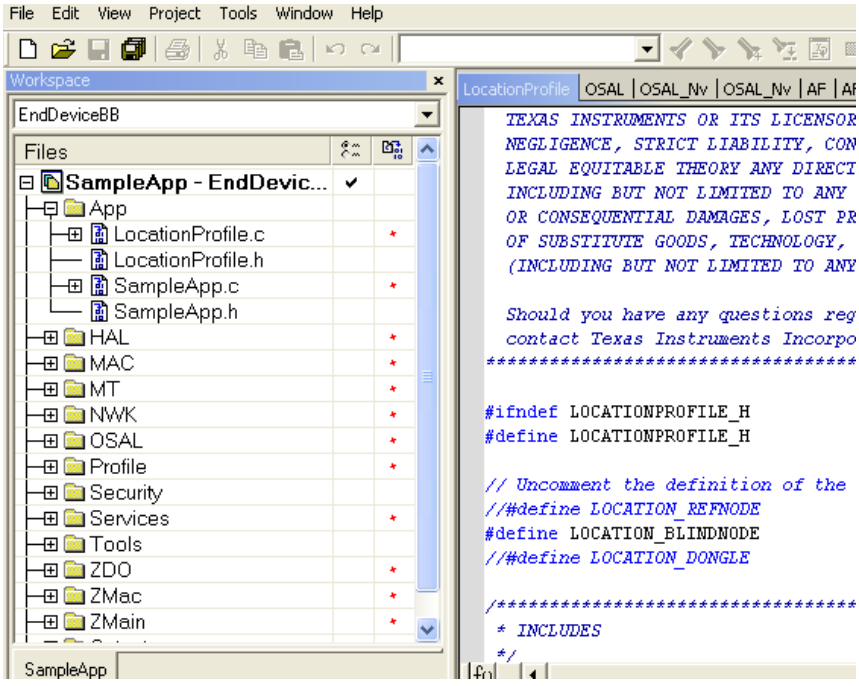


Figura 33. Selección dispositivo dentro del LocationProfile

- Una vez realizadas las operaciones anteriores, el sistema está listo para programar y hacer la depuración. Para ello abrimos el menú Project y seleccionaremos la opción Debug. A continuación se comenzará a programar el módulo CC2431EM y al finalizar saltara la depuración del programa.

El dispositivo EndDevice de la red ZigBee programado como BlindNode del perfil de pila ya está listo.

8.1.3.3 BlindNode en ejecución

Para los nodos que cumplan funciones de EndDevice en la red ZigBee y actúen como BlindNode dentro del perfil LocationProfile, usaremos módulo CC2431EM sobre la Placa SOC_BB (Batery Board), ya que están provistos del hardware de localización necesario para su localización en base a los valores RSSI de los nodos de referencia Esta Placa permite distribuir los módulos alimentados por baterías.

8.2 Diseño Software

El siguiente esquema muestra los roles asumidos por cada una de las parte Software del proyecto.

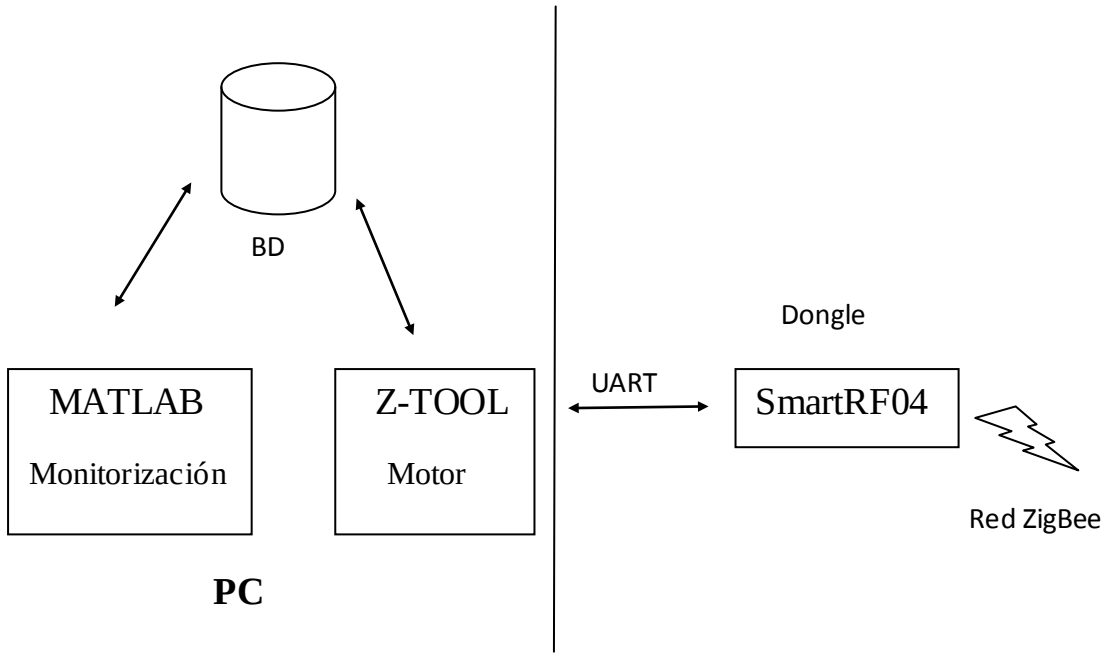


Figura 34. Comunicación general del sistema

8.2.1 Diseño Base de Datos Access

8.2.1.1 Tabla Tareas

En el interfaz gráfico creado en Matlab es posible configurar tanto los valores de *A*, *n* de los BlindNodes del sistema, así como las posiciones de todos los ReferenceNodes, estas dos tareas son controladas mediante las siguientes tablas.

- **Tabla Tareas**: Se activará la primera o segunda tarea en función de la tarea a realizar por el usuario en la pantalla de monitorización creada en Matlab.

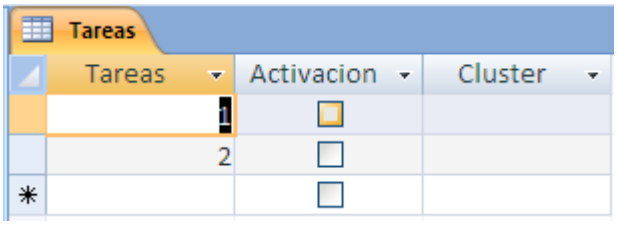
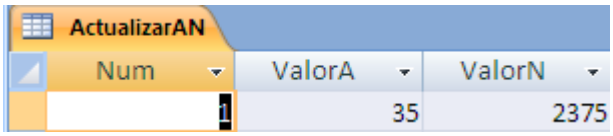


Tabla 4. Tabla Tareas

Tarea1: Actualizar posición X, Y de uno de los RefNode

Tarea2: Actualizar valores *A*, *n* del BlindNode.

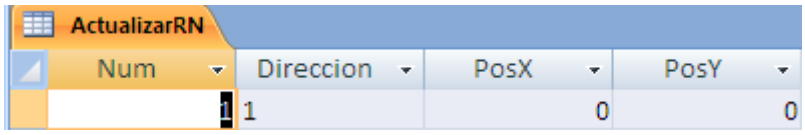
- **Tabla ActualizarAN**: En ella se introducirán los valores de *A* y *n* antes de activarse la Tarea2 de la tabla de Tareas.



ActualizarAN		
Num	ValorA	ValorN
1	35	2375

Tabla 5. Tabla ActualizarAN

- **Tabla ActualizarRN**: En ella se introducirán los valores X e Y del ReferenceNode que se quiera configurar antes de activarse la Tarea1 de la tabla Tareas..



ActualizarRN			
Num	Direccion	PosX	PosY
1	1	0	0

Tabla 6. Tabla ActualizarRN

8.2.1.2 Tabla ReferenceNodes

Esta tabla contiene todos los ReferenceNodes del sistema junto con sus posiciones.

ReferenceNodes

Direccion	PosX	PosY
1	0	0
143E	0	400
287B	400	0
3CB8	400	400

Tabla 7. Tabla ReferenceNodes

8.2.1.2 Tabla BlindNodes

Esta tabla contiene todos los BlindNodes del sistema junto con las coordenadas que va calculando el motor de Localización del modulo CC2431EM.

BlindNodes

Direccion	PosX	PosY
3520	300	100
796F	200	125
7970	125	200

Tabla 8. Tabla BlindNodes.

8.2.3 Diseño MATLAB

La monitorización de distancias entre el BlindNode que ejerce de Cámara y BlindNodes que actúan de actores se ha realizado en Matlab creando un interfaz gráfico intuitivo y de fácil manejo que permite interactuar con la red ZigBee configurando los valores de los Nodos de Referencia y los valores *A*, *n* de los BlindNodes. La siguiente figura muestra el interfaz grafico de usuario.

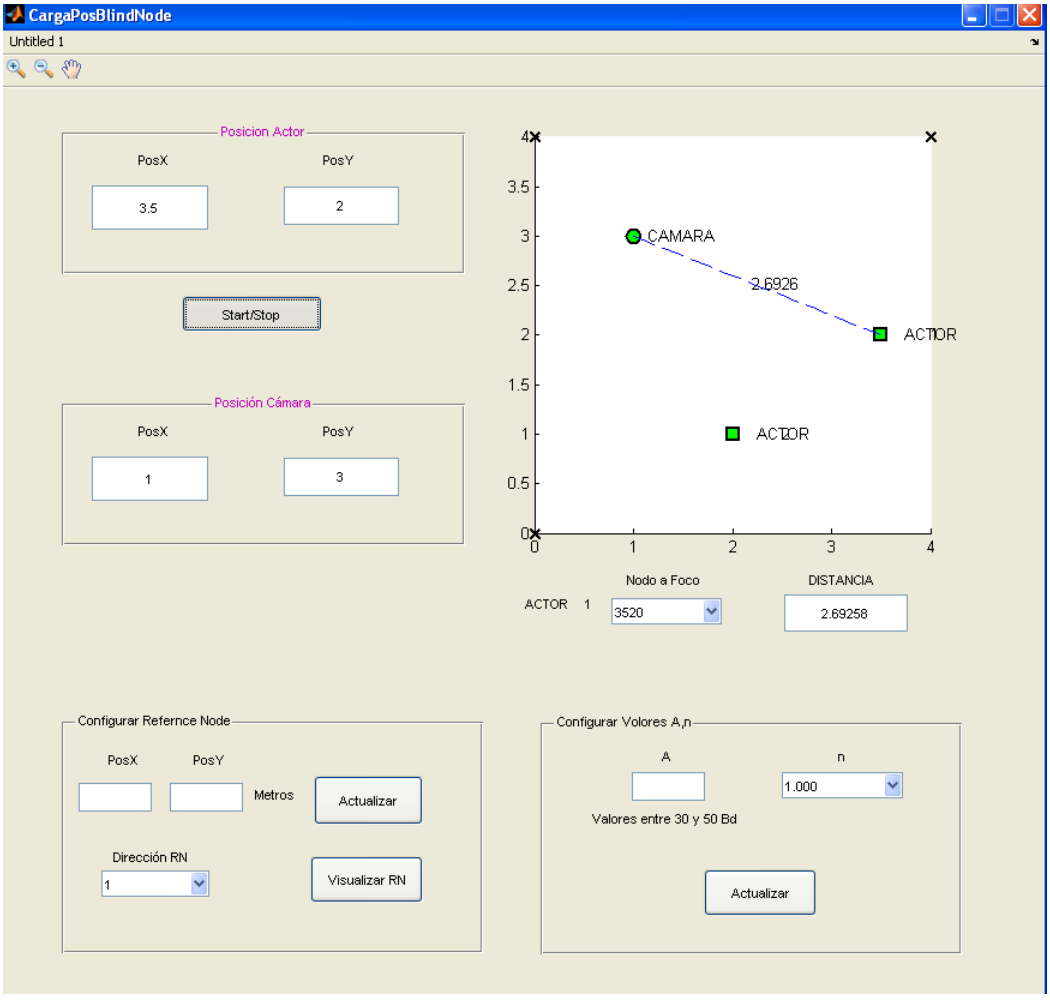


Figura 35. GUI usuario Matlab

Podemos observar como la pantalla dispone de tres ventanas:

- Ventana gráfica con distancias: Albergando toda la zona superior de la pantalla y mostrándonos una monitorización de la cámara y actores con sus posiciones y las respectivas distancias entre cámara y actor.
- Configuración de Nodos de Referencia: En esta zona situada en la parte inferior izquierda podremos configurar todas las posiciones de los Nodos de Referencia.
- Configuración de valores *A*, *n*: Situada en la parte inferior derecha, esta zona queda reservada para la configuración de los valores *A*, *n*, medidos en función del entorno.

8.2.2 Diseño Z-Tool

El diseño de aplicación desarrollada en Java Script y ejecutado en Z-Tool está compuesto por un Bucle principal de testeo de tareas pendientes y una función de atención a mensajes recibidos a través del Dongle.

8.2.2.1 Rutina de atención a mensaje recibido.

Por un lado, la aplicación Java Script ejecutada en Z-Tool está suscrita a interrumpirse por evento de llegada de mensajes al Dongle (Coordinador de la red ZigBee), integrando en el código de programa la función que activa las interrupciones por llegada de mensaje.

ZEngine.add_OnMessageZPI1(this.MessageHandler);

Así, cada vez que llegue un mensaje del Dongle, se ejecutará una función de atención a eventos de mensajes recibidos, en la cual miraremos si pertenece a alguno de los BlindNodes con sus posiciones calculadas. Los campos del mensaje son los siguientes:

- **SYS_APP_MSG_RESPONSE**

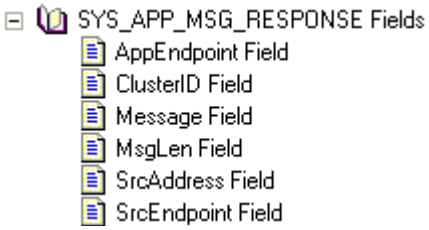


Figura 36. Campos del mensaje entrante

El perfil "Location Profile", además de definir los dispositivos de los que se compone (Dongle, ReferenceNode, BlindNode) junto con sus endpoints, tal y como se explico en apartados anteriores, tiene definidos una serie de clusters que definen la aplicación a la que hacen referencia, todo ello a nivel de aplicación dentro de la subcapa AF(Aplication Framework). Por ello en la función de atención a evento de llegada de mensaje solo prestaremos atención al ClusterID 0x0014.

Si el mensaje es de tipo ClusterID 0x0014 (Mensaje respuesta posición de BlindNode), entonces leeremos los datos del la ubicación calculada para dicho

BlindNode y pasaremos a introducir los datos en la tabla BlindNodes de la Base de datos. Los campos del CluserID 0x0014 son los siguientes.

- **Blind Node Find Response (Cluster ID: 0x0014)**

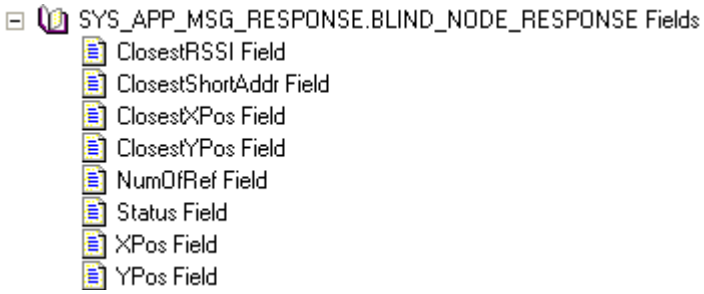


Figura 37. Campos del ClusterID 0x0014

Byte Index	Description	Value
0	Status	0 – Success 1 – Not enough Reference Node responding
1 & 2	Blind Node's calculated X position	Bits 15-2 – whole meters Bits 1-0 – 0.25 meters
3 & 4	Blind Node's calculated Y position	Bits 15-2 – whole meters Bits 1-0 – 0.25 meters
5	Number of Reference Nodes used in calculation	0 – 8
6 & 7	Closest (based on RSSI) Reference Node's short address	0x0000 – 0xFFFA, 0xFFFE is invalid
8 & 9	Closest Reference Node's X position	Bits 15-2 – whole meters Bits 1-0 – 0.25 meters
10 & 11	Closest Reference Node's Y position	Bits 15-2 – whole meters Bits 1-0 – 0.25 meters
12	Closest Reference Node's RSSI	

Tabla 9. Disposición de los Bytes y los campos del ClusterID 0x0014

El Diagrama de Flujo que describe el comportamiento del controlador de mensajes entrantes quedaría de la siguiente manera.

Rutina de atención a mensaje recibido.

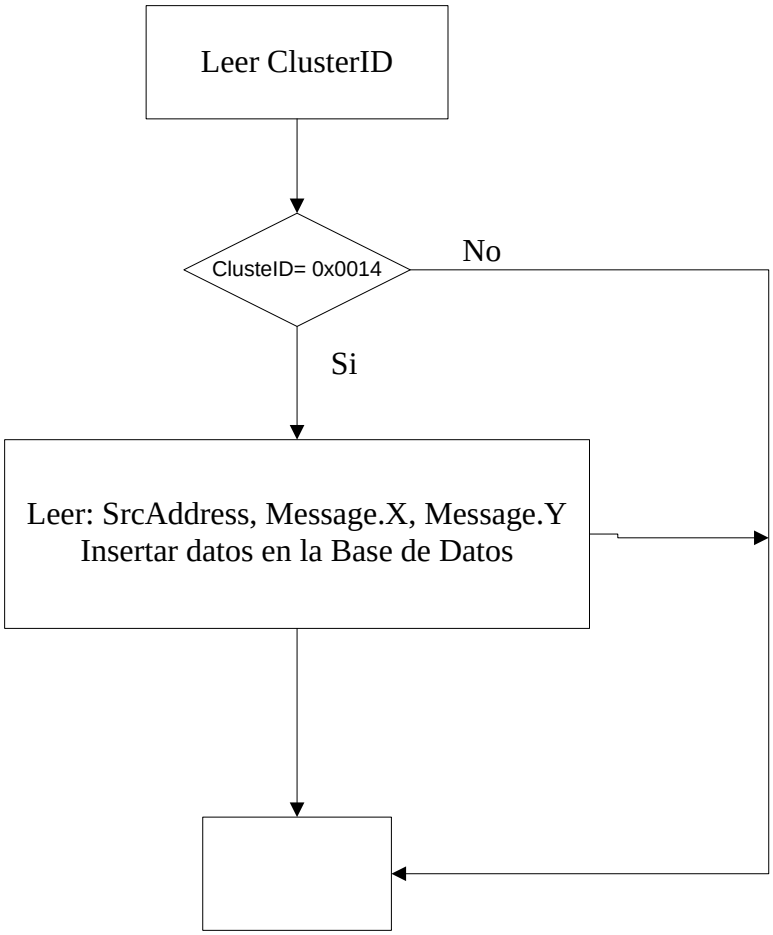


Figura 38. Diagrama de Flujo del manejador de mensajes entrantes

8.2.2.2 Algoritmo Round-Robin.

Algoritmo principal en el cual la aplicación se queda testeando si hay tareas pendientes. En el momento que llegue un mensaje, saltará el manejador de mensajes entrantes, si no, seguirá testeando tareas pendientes.

- **Tarea1: Reference Node Configuration (Cluster ID: 0x0015)**

En el momento que haya que configurar un ReferenceNode se tendrá que enviar el ClusterID 0x0015.

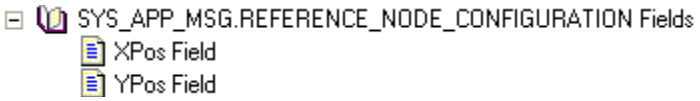


Figure 39. Campos del ClusterID 0x0015

Byte Index	Description	Value
0 & 1	Reference Node's X position	Bits 15-2 – whole meters Bits 1-0 – 0.25 meters
2 & 3	Reference Node's Y position	Bits 15-2 – whole meters Bits 1-0 – 0.25 meters

Tabla 10. Disposición de Bytes y campos del ClusterID 0x0015

- **Tarea2: Blind Node Configuration (Cluster ID: 0x0016)**

En el momento que haya que configurar las Variables *A*, *n* de los BlindNodes, se tendrá que enviar el ClusterID 0x0016.

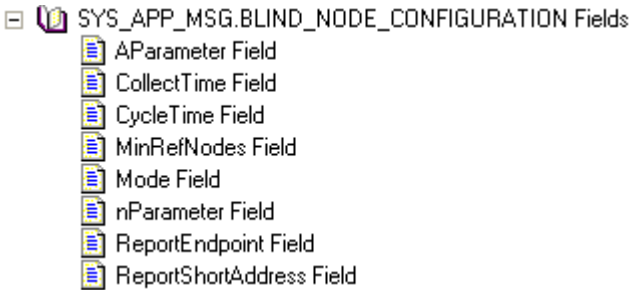


Figura 40. Campos del ClusterID 0x0016

Byte Index	Description	Value
0	Blind Node's A parameter. "A" is defined as the absolute value of the average power in dBm received at a close-in reference distance of one meter from the transmitter, assuming an omni-directional radiation pattern.	
1	Blind Node's N parameter, which is the path loss index that describes the rate at which the signal power decays with increasing distance from the transmitter.	
2	Operating Mode	0 – Polled. Waits for Blind Node Requests to do a find & rsp 1 – Automatically initiate a location find and response.
3 & 4	Collect Time – the number of milliseconds to wait for Reference Node Responses after sending the request.	In 100 millisecond increments
5 & 6	Cycle Time – Low byte first. The number of milliseconds to wait before starting the calculation cycle. Only valid in the Auto Operating Mode.	In 100 millisecond increments
7 & 8	Report Short Address – low byte first. Destination address for Blind Node Response messages in AUTO mode. In POLL mode, the response is returned to the requestor's address.	0x0000 – 0xFFFF
9	Report Endpoint – Destination endpoint for Blind Node Response message in AUTO mode.	
10	Minimum Reference Nodes to use to calculate location.	1 – 16

Tabla 11. Disposición de Bytes y campos del ClusterID 0x0016

El siguiente Diagrama de Flujo describe el comportamiento del algoritmo:

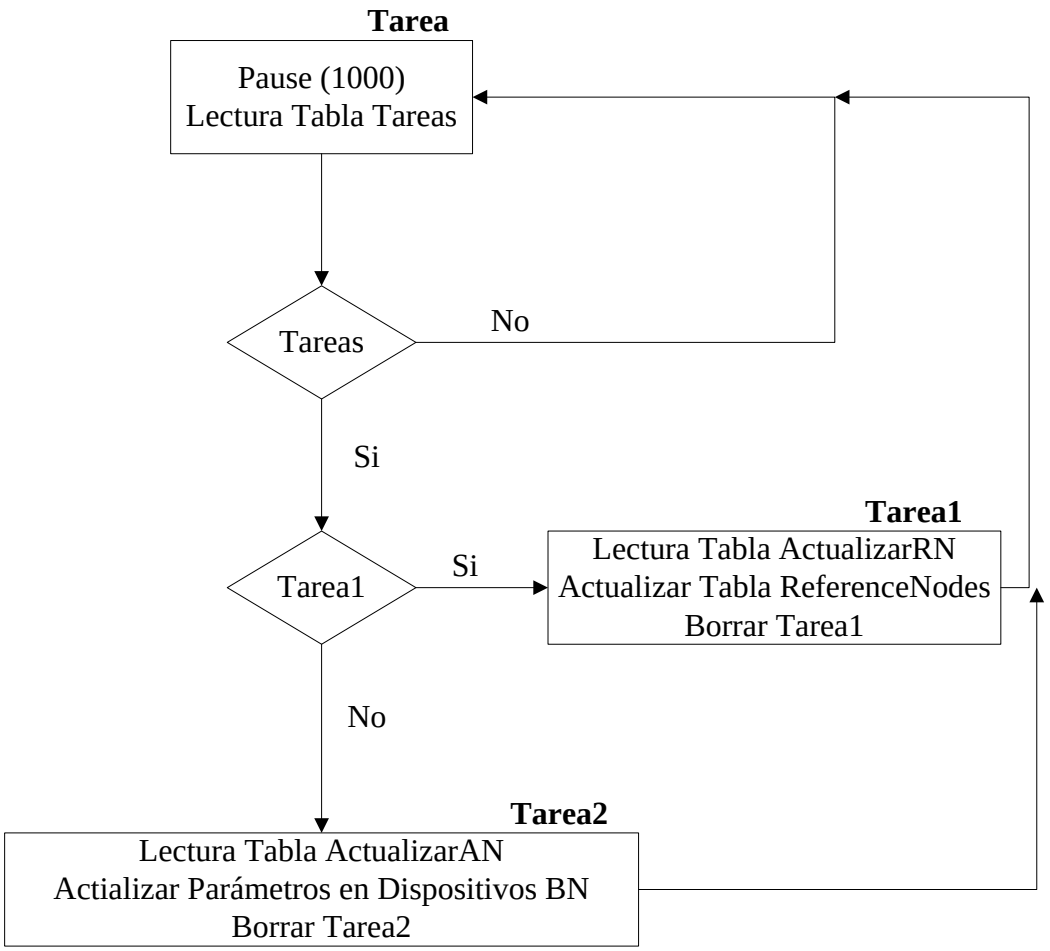


Figura 41. Diagrama de Flujo del algoritmo principal

8.3 Implementación Software

9.3.1 Implementación Java Script y ejecución en Z-Tool

9.3.1.1 Creación "Archivo.zjs"

El formato de los archivos Java Script a ejecutar en Z_Tool deben de tener la extensión "zjs" de tal forma que se debe crear un archivo de texto con dicha extensión que acepte Z_Tool.

Para ello debemos abrir un archivo de texto con la aplicación Z_Converter.

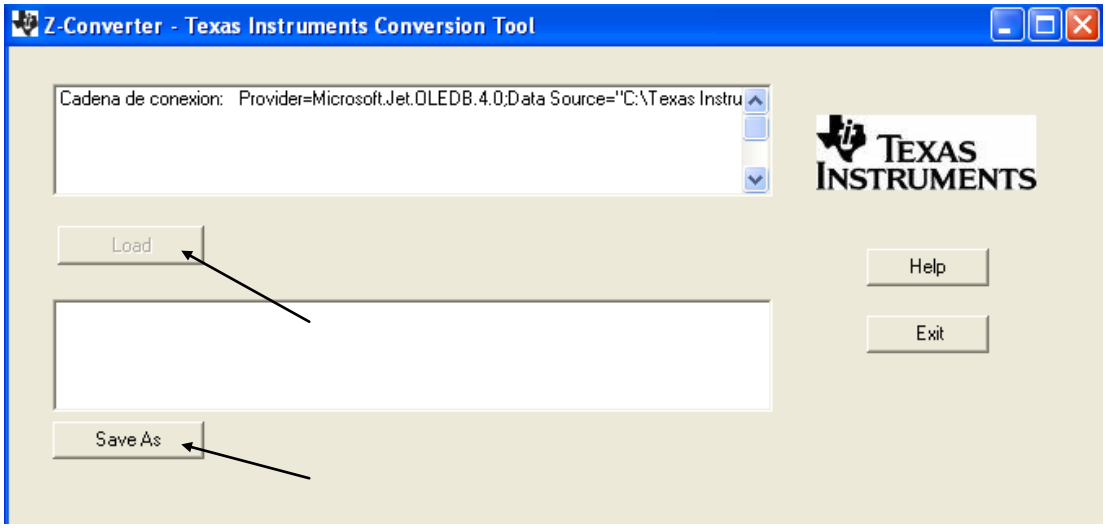


Figura 42. Interfaz Z_Converter

A continuación deberemos salvarlo como: "Nombre archivo.zjs"

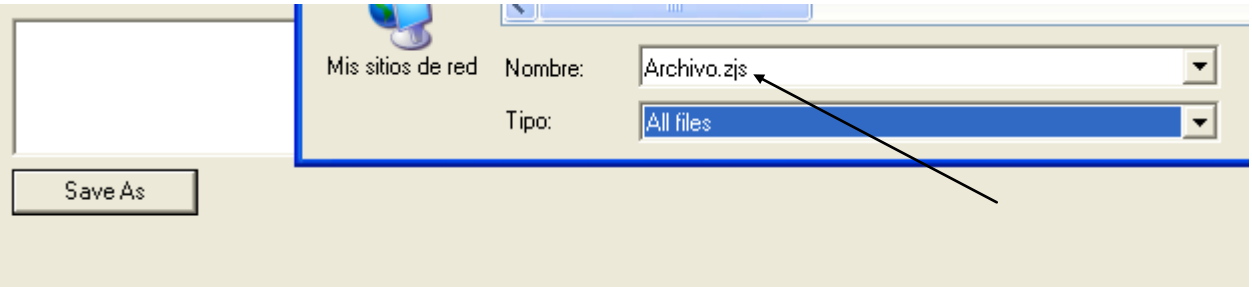


Figura 43. Crear Archivo.zjs

De este modo ya tenemos el archivo sobre el cual podremos comenzar a implementar la aplicación y ejecutarla en Z_Tool.

8.3.1.2 Implementar y ejecutar en Z_Tool

Primeramente debemos abrir el archivo.zjs en la ventana de ejecución de Script, para ello en el menú Tools seleccionaremos Script Windows. Aprarecerá el siguiente interfaz.

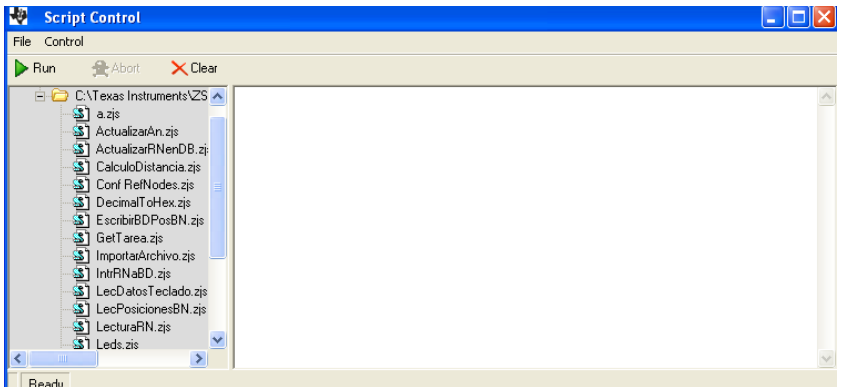


Figura 44. Interfaz Script Windows

A continuación haciendo click en el botón derecho del ratón sobre el Archivo.zjs, aparecerá un menú emergente donde seleccionaremos editar. Por defecto tenemos el Notepad de Windows como editor.

Una vez editado lanzaremos el Jscript haciendo clic sobre el botón Run.

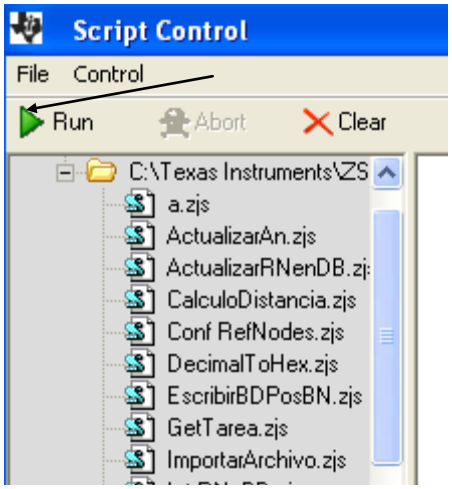


Figura 45. Lanzar Script

8.3.1.2 Implementación motor del proyecto

Z_Tool incorpora una librería denominada ZEngine, la cual nos permitirá interactuar con el sistema Location. Dicha librería debemos incorporarla en la cabecera del Script *"import TI.ZPI1;"*.A continuación describiremos los métodos y eventos utilizados en este proyecto.

8.3.1.2.1 Métodos y Eventos de ZEngine

- **ZEngine.add_OnMessageZPI1(this.MessageHandler);**

Función que permite suscribirse a los mensajes entrantes cuyo manejador de evento será la función siguiente:

- **Function MessageHandler(zportName:String, id:MESSAGE_ID, msg:Object)**

```
{

 if    ((id == "SYS_APP_MSG_RESPONSE")&&(msg.ClusterID==20))
        //Mensaje entrante es un Blind Node Response, ClusterID=0x0014
      {
      //msg es un objeto del tipo SYS_APP_MSG_RESPONSE

   var reg : SYS_APP_MSG_RESPONSE.BLIND_NODE_RESPONSE=msg.Message;

     //Creamos una variable con el objeto Blind_Node_Response del mensaje


        var BNa= msg.SrcAddress;  // dirección de origen del BlindNode
        var BN=BNa.toString(16);  // pasamos el entero a un string en base 16(HEX)

        var PosX=reg.XPos;          //Posición X del BN en Hexadecimal
        var PosY=reg.YPos;          //Posición Y del BN en Hexadecimal

        var a= HexToDecimal(PosX);  // Conversión a decimal
        var b= HexToDecimal(PosY);
```

//Escribir posiciones en la BD

```
EscribirPosXBN((a*100),BN);   //Escribir en la base de datos las posiciones X e Y
EscribirPosYBN((b*100),BN);


    }
  }
```

- **Pause(int milliseconds)**

Función que pausa el Scrip el tiempo indicado en milisegundos

- **Send(string deviceName, object message)**

Función que envía el objeto, Cluster al dispositivo especifico.

deviceName: Introducir el alias que se puso al configurar el puerto serie en Z_Tool, de este modo el Cluster enviado llega al Dongle acoplado en la SmartRF04EB y lo gestiona. En nuestro caso el Alias asignado a nuestro dispositivo es Device1.

8.3.1.2.1 Main (Round Robin)

```
import System;
import System.Text;
import TI.ZPI1;

class ZScriptSampleClass

{

   //-------------------------Funcion Start---------------------------------//


   function Start()

   {


          // subscribe to message handler

          ZEngine.add_OnMessageZPI1(this.MessageHandler);




          //Bucle principal, mirar en la tabla tareas si  tareas pendientes

          var Tarea;
          while (Tarea!=7){       //Tarea!=7 Bucle infinito donde testeara las tareas pendientes
```

```
ZEngine.Pause(1000);

Tarea = GetTareas();

switch(Tarea)

{

case Tarea=1: //Configuracion posicion RN

{

        // Configuracicon de las posiciones de los Ref Nodes

        //costructor, instanciar una variable del tipo ClusterID 0x15 (dec=21)
        //REFERENCE_NODE_CONFIGURATION

        var mes : SYS_APP_MSG.REFERENCE_NODE_CONFIGURATION = new
SYS_APP_MSG.REFERENCE_NODE_CONFIGURATION();

        //leer posicion del RN en la BD

        var sql="SELECT Direccion,PosX,PosY FROM ActualizarRN ";

        var adOpenDynamic=3;

        var adLockOptimistic=3;

        var rs = new ActiveXObject("ADODB.Recordset");

        rs.open(sql,conn,adOpenDynamic,adLockOptimistic);

if(!rs.bof)
```

```
{

        var a = DecimalToHex((rs.fields(1).value)/100);
        var b = DecimalToHex((rs.fields(2).value)/100);
        rs.MoveFirst();
        mes.XPos=a;
        mes.YPos=b;
        var dir =rs.fields(0).value;

        ZEngine.WriteLog("A verX Hex" + a );
        ZEngine.WriteLog("A verY Hex" + b );
        ZEngine.WriteLog("direccion" + dir );

      // Constructor, Instanciar una variable del tipo SYS_APP_MSG


 var reg : SYS_APP_MSG = new SYS_APP_MSG();

reg.AppEndpoint=203;
reg.ClusterID=21;          //Cluster, aplicación a la que se refiere
reg.DestAddress =parseInt(dir,16);
reg.DestEndpoint=210;    // EndPoint perteneciente a Dispositivo ReferenceNode
reg.MsgLen=4;
reg.Message=mes;          //Variable con el Cluster a enviar


      //Enviar al Dispositivo1 cuyo alias es Device1, setting--> serial port-->alias

      ZEngine.Send("Device1", reg);
```

```
            //Introducir  en la BD la posicion del RN


            ActualizarRNX(dir,rs.fields(1));
            ActualizarRNY(dir,rs.fields(2));

            //Borramos la tarea

            BorrarTarea(Tarea);    //Borra la tarea de la lista de tareas

                                }
}

   break;




case Tarea=2:  //Configuracion parametros A,N

   {

            // ENVIAR PETICION DE CONFIGURACION DEL BLIND NODE

             // Constructor, Instanciar una variable del tipo SYS_APP_MSG

var registro : SYS_APP_MSG = new SYS_APP_MSG();
 registro.AppEndpoint=203;
 registro.ClusterID=24;
```

```
registro.DestAddress =parseInt('0x7970',16);
registro.DestEndpoint=211;
registro.MsgLen=0;



        //Enviar al Dispositivo1 cuyo alias es Device1, setting--> serial port-->alias

ZEngine.Send("Device1", registro);
BorrarTarea(Tarea);

        }       }
}
}
}
```

El código completo se proporcionará en el anexo del proyecto.

8.3.2 Implementación MATLAB

El código de las respuestas a los eventos sobre el interfaz grafico creado se proporcionará en el anexo del proyecto, destacando en este apartado la implementación de la función que calcula la distancia entre dos puntos (X1, Y1, X2, Y2)

```
x = Distancia(Nx, Ny,Camarax,Camaray);
```

Siendo uno de los puntos uno de los BlindNodes que actúa como cámara.

```
function [ x ] = Distancia( a, b, c, d )
%UNTITLED Summary of this function goes here
%   Detailed explanation goes here

    Base =abs( a - c); //Valor absoluto
    Altura = abs(b - d);  //Valor absoluto
    x =sqrt( (Base * Base) + ( Altura * Altura)); //h^2 = Base^2 + Altura^2

end
```

Así como la función de la recta que pasa por dos puntos, para visualizar en el gráfico la línea recta entre la cámara y el actor seleccionado.

```
function [ y] = Lineal( x,x1,y1,x2,y2 )
%UNTITLED2 Summary of this function goes here
%   Detailed explanation goes here

y = y1 + ((y2 - y1)*((x - x1)/(x2-x1)));


end
```

Cuya sección del código donde se usa la función se muestra a continuación

```
                        .

                        .

                        .

                  //Lectura BN

 [Struct Table] = adodb_query(DB, 'select  * from BlindNodes');



hold all

                  //Plot de los RN y BN

for i=2:length(Struct.posx)

    px= (double(Struct.posx{i})/100);
```

```
    py= (double(Struct.posy{i})/100);

plot(R1x,R1y,'rx',R2x,R2y,'rx',R3x,R3y,'rx',R4x,R4y,'rx',px,py,'bs',Ca
marax,Camaray,'bo','LineWidth',2,...
        'MarkerEdgeColor','k',...
        'MarkerFaceColor','g',...
        'MarkerSize',10);

    axis square
    text(double(px), double(py),'     ACTOR');
    text(double(px+0.5), double(py), num2str(i-1));
end


 valor = get(handles.popupmenu3,'Value');
 set(handles.text17,'String',valor);
 f=double(Struct.posx{valor+1})/100;


 if f >Camarax

    a= Camarax:0.1:f;

 else
    a= f:0.1:Camarax;
 end


 Nx = double(Struct.posx{valor+1})/100;
 Ny = double(Struct.posy{valor+1})/100;

 x = Distancia(Nx, Ny,Camarax,Camaray);
 set(handles.edit16,'String',x);

 set(handles.PosX,'String',double(Struct.posx{valor+1})/100);
 set(handles.PosY,'String',double(Struct.posy{valor+1})/100);


            //Plot de la linea entre cámara y actor

 plot(a,Lineal(a,Nx,Ny,Camarax,Camaray),'--');
```

CAPITULO 9: ANÁLISIS E INTERPRETACIÓN DE RESULTADOS

Para la siguiente prueba, se ha calibrado los valores *A*=38 y *n*=25 obtenidos mediante prueba y error al situar los nodos finales a un metro de cada uno de los nodos de referencia.

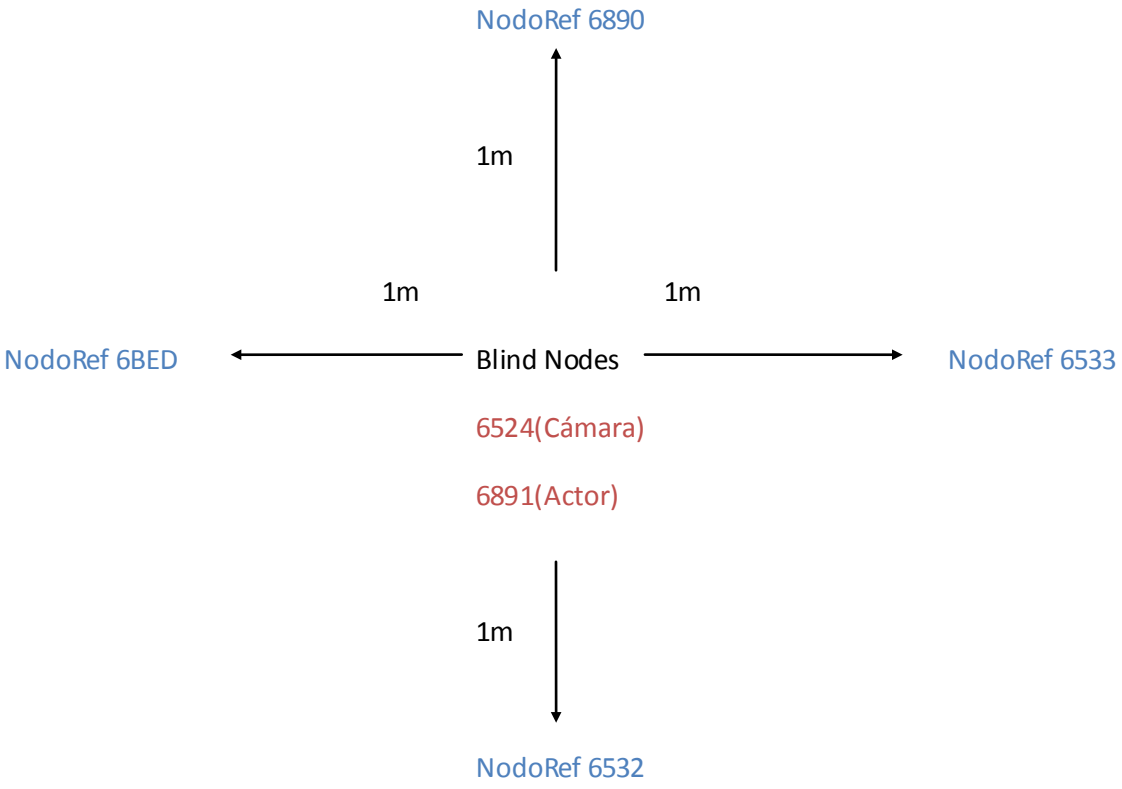


Figura 46. Configuración valores A y n

Una vez calibrados los valores *A* y *n*, se ha realizado la prueba de localización y calculo de distancias entre Cámara y Actor configurando los nodos de referencia en las posiciones que se muestran en la tabla 12 así como los resultados y errores obtenidos.

9.1 Tabla posiciones nodos de referencia

La siguiente tabla muestra las direcciones lógicas y las coordenadas en las que se han situado los nodos de referencia para la prueba de la que se han recogido diez muestras, quedando un rectángulo de 2,5m x 4 m sobre el que se han ido situando los nodos ciegos y recogiendo las lecturas de sus posiciones.

Nodos	Dirección Lógica	Coordenadas (X,Y)
1	6890	(0-1.25)
2	6532	(2-0)
3	6533	(4-1.25)
4	6BED	(2-2.5)

Tabla 12 . Nodos de Referencia

9.2 Tabla lecturas Nodo Cámara

La siguiente tabla muestra las coordenadas reales y calculadas del nodo final que ejercería como cámara y cuya dirección lógica en la red ZigBee es 6534

Num. Veces	X(m)		Error absoluto	Y(m)		Error absoluto
	Real	Calculado		Real	Calculado	
1	1	0,75	0,25	1	1	0
2	3,75	3,75	0	1,25	1,5	0,25
3	3,5	3,5	0	1,25	1,5	0,25
4	2,25	2	0,25	1,8	1,25	0,55
5	2	2	0	1,25	1,75	0,5
6	2	2,25	0,25	2,4	2,5	0,1
7	2,25	2,25	0	0,15	0	0,15
8	0,95	1	0,05	1,95	2	0,05
9	2,05	1,75	0,3	2,4	2,25	0,15
10	3,75	3,5	0,25	1,4	1	0,4
media			0,13			0,24

Tabla 13 . Errores absolutos Nodo Cámara 6534

9.3 Tabla lecturas Nodo Actor

Tabla en la cual se muestran las lecturas de las coordenadas calculadas por el motor de localización y las posiciones reales del nodo final que vendría a representar al actor y cuya dirección lógica en la red ZigBee es 6891

Num. Veces	X(m)		Error absoluto	Y(m)		Error absoluto
	Real	Calculado		Real	Calculado	
1	2	2	0	0,25	0	0,25
2	2	2,25	0,25	1,8	1,5	0,3
3	0,25	0,5	0,25	1,5	1,75	0,25
4	1,3	1	0,3	1	1	0
5	0,75	1	0,25	1,25	1,5	0,25
6	2	2	0	0,25	0,25	0
7	1,75	2	0,25	0,15	0,25	0,1
8	1,65	1,75	0,1	1,5	1,5	0
9	1	1	0	1,1	0,75	0,35
10	3,75	3,25	0,5	1,05	1	0,05
media			0,19			0,15

Tabla 14 . Errores absolutos nodo Actor 6891

9.4 Tabla lecturas Distancias entre Cámara y Actor

La siguiente Tabla muestra las distancias reales entre Cámara y Actor, así como la distancia obtenida a través de las coordenadas obtenidas por el motor de localización.

Num. Veces	Distancia		Error absoluto
	Real	Calculado	
1	1,5	1,6	0,1
2	1,8	1,5	0,3
3	3,46	3,1	0,36
4	1,35	1,04	0,31
5	1,6	1,04	0,56
6	2,16	2,26	0,1
7	0,55	0,35	0,2
8	0,85	0,9	0,05
9	1,7	1,68	0,02
10	0,43	0,25	0,18
Error Relativo			0,22

Tabla 15. Errores absolutos entre las distancias de los nodos finales

CAPITULO 10: CONCLUSIONES Y BIBLIOGRAFÍA

10.1 Conclusiones

10.1.1 Problemas en la práctica con RSSI

- Las características de propagación de la señal varían constantemente.
- El RSSI depende del nivel de batería del Nodo.
- Evitar obstáculos que atenúen la señal.
- La sensibilidad del transceptor.

10.1.2 Recomendaciones

- Cuantos más nodos de referencia se coloquen, mayor es la precisión obtenida en las coordenadas, por lo tanto se recomienda que se distribuyan en forma de malla o cuadrante para optimizar la recepción en el área de trabajo.
- Para optimizar la precisión, la altura de los nodos de referencia debe ser la misma que la de los nodos ciegos, ya que al variar la altura pueden haber inconsistencias en los datos obtenidos.
- Para evitar interferencias, las paredes no deben exceder de un grosor de 20 cm, o tener un recubrimiento de metal.

10.1.3 Valoración general

Dado que el margen de error oscila alrededor de 20 cm en las mediciones de distancias entre dos nodos ciegos (Cámara y Actor) y la prueba realizada se ha puesto en marcha bajo condiciones ideales (sin obstáculos, con baterías nuevas y recién calibrados los valores *A* y *n*) como conclusión general se podría destacar que es difícil considerar este sistema como válido para una aplicación real de sistema de enfoque automático.

El sistema es muy influenciable por el entorno y varía constantemente, además de la atenuación e influencia de los obstáculos en las mediciones de los valores RSSI. En ocasiones los valores medidos saltan bruscamente a posiciones alejadísimas teniendo que implementarse un sistema de filtrado de falsas lecturas, que no se ha realizados y simplemente se ha esperado a que se estabilice por sí mismo.

El tamaño de los nodos también se debería tener en cuenta ya que resultaría difícil ocultarlos para evitar su aparición en pantalla, sería engorroso para los actores y la atenuación de la señal del cuerpo humano daría lecturas muy dispares en función de de la orientación del mismo.

10.2 Bibliografía

10.2.1 Documentación hardware

Texas Instrument, *CC2430 Data Sheet (rev. 2.1)* SWRS036F

Texas Instrument, *CC2430DK Development Kit User Manual (Rev. 1.0)* SWRU133

Texas Instrument, *CC2431DK Development Kit User Manual (Rev. 1.5)* SWRU76D

10.2.2 Documentación Z-Stack

Texas Instrument, *Z-Stack Developer's Guide, (*F8W-2006-0022)

Texas Instrument, Z-Stack Developer's Guide, SWRA176

Texas Instrument, *Z-Stack/Z-Tool Serial Port Interface,*F8W-2003-0001

10.2.3 Documentación Location Profile

Texas Instrument, *Z-Stack Location Profile,*

Texas Instrument, *Z-StackMonitor and Test API,* SWRA198

10.2.4 Ejemplos Software Z-Stack

Texas Instrument, *CC2430 Software Examples User's Guide,* SWRU178B

Texas Instrument, *Z-Stack Sample Applications,* F8W-2006-0023

Texas Instrument, CC2431DK Quick Start Instructions, SWRU080

10.2.5 OSAL

Texas Instrument, *Z-StackOS Abstraction Layer Application Programming Interface*, F8W-2003-0002

10.2.6 Proyectos

Lenin Eduardo Guerra, Ingeniero en electrónica y computación, Tesis de Grado, *Diseño e implementación de un prototipo para la localización de un objeto en movimiento mediante ZigBee.* Escuela superior politécnica de Chimborazo (Ecuador), Noviembre 2010.

Cynthia Vanessa Moreno Pallo, Proyecto previo a la obtención del título de ingeniero en electrónica y telecomunicaciones, *Construcción de una red ZigBee prototipo para la adquisición de datos desde transmisores de corriente de dos hilos.* Escuela politécnica nacional, facultad de ingeniería electrónica y telecomunicaciones, Marzo de 2009.

Sergio Lillo Moreno, Proyecto fin de carrera, Ingeniería técnica superior en telecomunicaciones, *Desarrollo de un entorno para la configuración y monitorización de redes ZigBee.* Escuela técnica superior de ingeniería de telecomunicación, universidad de Málaga, 2010.

Sergio Arévalo Barazas, Proyecto fin de carrera, Ingeniería de telecomunicaciones, *Programación de un nodo conmutador para la gestión remota de redes ZigBee.* Escuela Técnica superior de ingeniería de telecomunicación, universidad de Málaga, Noviembre de 2011.

Ivan Barneda Faudot, Proyecto fin de carrera, Ingeniería de telecomunicaciones, *ZigBee aplicado a la transmisión de datos de sensores biométricos.* Universidad autónoma de Barcelona, 5 de Septiembre de 2008.

CAPITULO 11: ANEXOS

ANEXO A

Alternativa de desarrollo del proyecto

Otra alternativa de diseño e implementación software consistiría en prescindir de la Base de Datos y Z-Tool. Realizando todo el proyecto a través de Matlab o VisualStudio (Visual Basic) en plataforma .NET. Gestionando la comunicación con el dispositivo Dongle acoplado a la placa SmartRF a través del puerto serie configurado en dichos programas.

La comunicación vía puerto serie está sujeta a un tipo de formato de tramas, los cuales se describen en la documentación *Z-Stack/Z-Tool Serial Port Interface*, referenciada en la Bibliografía.

ANEXO B

Puesta en marcha: Ejemplo de aplicación Z-Location Engine

Chipcon de Texas Instrument nos proporciona una aplicación que nos permite monitorizar y configurar los distintos dispositivos de una red ZigBee sobre la cual esta implementado el sistema Location. Una buena toma de contacto a la hora de familiarizarse con el hardware, programar dispositivos y comprender el sistema monitoriazado.

Software proporcionado por Chipcon:

- Archivos .Hex para programar módulos CC2431 y CC2430 a través de SmartRF Flash Programmer, asumiendo los roles de dispositivos descritos en la memoria del proyecto.
- Aplicación Z-Location Engine para monitorizar los distintos dispositivos.

La puesta en marcha del sistema queda detallada en la documentación: CC2431 *Quick Start instruction*, referenciada en la Bibliografía.

ANEXO C

Código de programa algoritmo principal (Round-Robin)

```
import System;
import System.Text;
import TI.ZPI1;


class ZScriptSampleClass

{

        // define states

        enum TEST_STATE

        {
        ProcesandoDato,

        MirandoTarea,
        }

  //--------------------------Funcion Start----------------------------------//


  function Start()

  {

        //inicializar el estado a mirando tarea
        currentState = TEST_STATE.ProcesandoDato;   //Cambiar valor a mirando tarea
```

```
            // subscribe to message handler

        ZEngine.add_OnMessageZPI1(this.MessageHandler);


            //Bucle principal, mirar en la tabla tareas si  tareas pendientes

            var Tarea;
while (Tarea!=7){       //Tarea!=7 Bucle infinito donde testeara las tareas pendientes


        if (currentState == TEST_STATE.MirandoTarea)
            {

            ZEngine.Pause(1000);
            Tarea = GetTareas();
             switch(Tarea)

        {

          case Tarea=1: //Configuracion posicion RN

            {

                        // Configuracicon de las posiciones de los Ref Nodes

//costructor, instanciar una variable del tipo ClusterID 0x15 (dec=21) REFERENCE_NODE_CONFIGURATION

            var mes : SYS_APP_MSG.REFERENCE_NODE_CONFIGURATION = new
SYS_APP_MSG.REFERENCE_NODE_CONFIGURATION();
```

```
            //leer posicion del RN en la BD


            var sql="SELECT Direccion,PosX,PosY FROM ActualizarRN ";
            var adOpenDynamic=3;
            var adLockOptimistic=3;
            var rs = new ActiveXObject("ADODB.Recordset");
            rs.open(sql,conn,adOpenDynamic,adLockOptimistic);


 if(!rs.bof)

{

             var a = DecimalToHex((rs.fields(1).value)/100);
             var b = DecimalToHex((rs.fields(2).value)/100);
            rs.MoveFirst();
            mes.XPos=a;
            mes.YPos=b;
            var dir =rs.fields(0).value;


             ZEngine.WriteLog("A verX Hex" + a );
             ZEngine.WriteLog("A verY Hex" + b );
             ZEngine.WriteLog("direccion" + dir );


            // Constructor, Instanciar una variable del tipo SYS_APP_MSG

var reg : SYS_APP_MSG = new SYS_APP_MSG();

 reg.AppEndpoint=203;
```

```
reg.ClusterID=21;
reg.DestAddress =parseInt(dir,16);
reg.DestEndpoint=210;
reg.MsgLen=4;
reg.Message=mes;


            //Enviar al Dispositivo1 cuyo alias es Device1, setting--> serial port-->alias

ZEngine.Send("Device1", reg);

            //Introducir  en la BD la posicion del RN


            ActualizarRNX(dir,rs.fields(1));
            ActualizarRNY(dir,rs.fields(2));







            //Borramos la tarea

            BorrarTarea(Tarea);    //Borra la tarea de la lista de tareas

                                    }
}

   break;
```

```
    case Tarea=2:  //Configuracion parametros A,N

    {


            // ENVIAR PETICION DE CONFIGURACION DEL BLIND NODE
             // Constructor, Instanciar una variable del tipo SYS_APP_MSG



var registro : SYS_APP_MSG = new SYS_APP_MSG();

 registro.AppEndpoint=203;
 registro.ClusterID=24;
 registro.DestAddress =parseInt('0x7970',16);
 registro.DestEndpoint=211;
 registro.MsgLen=0;




            //Enviar al Dispositivo1 cuyo alias es Device1, setting--> serial port-->alias

ZEngine.Send("Device1", registro);
BorrarTarea(Tarea);

            }
    break;
}
}
}
```

```
}

        //----------Manejador de mensajes OTA----------------------------------------------//

   function MessageHandler(zportName:String, id:MESSAGE_ID, msg:Object)

   {

                currentState = TEST_STATE.ProcesandoDato;

        if ((id == "SYS_APP_MSG_RESPONSE")&&(msg.ClusterID==20))  //Mensaje entrante es un Blind Node
Response

        {

            //msg es un objeto del tipo SYS_APP_MSG_RESPONSE

            var reg : SYS_APP_MSG_RESPONSE.BLIND_NODE_RESPONSE = msg.Message;

                        var BNa= msg.SrcAddress;
                        var BN=BNa.toString(16);  // pasamos el entero a un estring de base 16

                       var PosX=reg.XPos;
                       var PosY=reg.YPos;

                       var a= HexToDecimal(PosX);
                       var b= HexToDecimal(PosY);
```

```
                    //Escribir posiciones en la BD

                    EscribirPosXBN((a*100),BN);   //Escribir en la base de datos las posiciones X e Y
                    EscribirPosYBN((b*100),BN);




                    currentState = TEST_STATE.MirandoTarea;
          }

               else if ((id == "SYS_APP_MSG_RESPONSE")&&(msg.ClusterID==22))  //Mensaje entrante es
un Blind Node configuration Response

               {

                    //regTarea2 es de tipo Blind node confifiguration

                     var regTarea2 : SYS_APP_MSG_RESPONSE.BLIND_NODE_CONFIGURATION =
msg.Message;




                    //Acceso a la BD


                    var sql="SELECT ValorA, ValorN FROM ActualizarAN ";
                    var adOpenDynamic=3;
                    var adLockOptimistic=3;
                    var rs = new ActiveXObject("ADODB.Recordset");
                    rs.open(sql,conn,adOpenDynamic,adLockOptimistic);
```

```
                rs.MoveFirst();
                var valorA = ((rs.fields(0).value)/10); //Lectura del valor A
                var valorN = rs.fields(1).value;       //Lectura del valor N


                regTarea2.nParameter = valorN;
                regTarea2.AParameter = DecimalToHexParamA(valorA);


               // Constructor, Instanciar una variable del tipo SYS_APP_MSG

var reg2 : SYS_APP_MSG = new SYS_APP_MSG();

 reg2.AppEndpoint=203;
 reg2.ClusterID=22;
 reg2.DestAddress =parseInt('0x7970',16);
 reg2.DestEndpoint=211;
 reg2.MsgLen=11;
 reg2.Message=regTarea2;



//Enviar al Dispositivo1 cuyo alias es Device1, setting--> serial port-->alias

               ZEngine.Send("Device1", reg2);
                currentState = TEST_STATE.MirandoTarea;

}

else{

               currentState = TEST_STATE.MirandoTarea;
```

```
                    }
}
```

```
//----------------------------Funcion GetTares--------------------------//

function GetTareas(){ //Lectura de tareas pendientes

var sql="SELECT Tareas FROM Tareas WHERE Activacion=TRUE;";
var adOpenDynamic=2;
var adLockOptimistic=3;
var rs = new ActiveXObject("ADODB.Recordset");
//var sqlArr=new Array();
//var i=0;var tArr=[];
var NumTarea=3;


rs.open(sql,conn,adOpenDynamic,adLockOptimistic);


 if(!rs.bof)
        {rs.MoveFirst();
        return(rs.fields(0).value);
        rs.close();

        }
        rs.close();
```

```
}



//----------------------------Funcion Borrar Tarea-------------------------//

function BorrarTarea(tarea){

var a=tarea
var sql="UPDATE Tareas set Activacion=FALSE WHERE Tareas="+a+"";
var adOpenDynamic=2;
var adLockOptimistic=3;
var rs = new ActiveXObject("ADODB.Recordset");
var sqlArr=new Array();
var i=0;var tArr=[];
var NumTarea;
rs.open(sql,conn, adOpenDynamic, adLockOptimistic);



}



//----------------------------Funcion HexToDecimal-------------------------//

function HexToDecimal(Numero){

var binH = Numero
binH >>= 2;
var binL = Numero;

var i = 0;
```

```
var binarray= new Array;

ZEngine.WriteLog("paso" + i);

while (  (binL.toString(2)[i]) != undefined)

        {binarray[i] =(binL.toString(2)[i]);
        i++;}


var a = parseInt(binarray[i-1]);

var b = parseInt(binarray[i-2]);


ZEngine.WriteLog("a= " + a);
ZEngine.WriteLog("b= " + b);

var terminacion=new Array;
terminacion[0]=b;
terminacion[1]=a;

var Numero=0;



if ( (!terminacion[0]) && terminacion[1]) //01 ba

{ZEngine.WriteLog("biennnn 01");
```

```
            Numero= (binH + 0.25);

            ZEngine.WriteLog("El numero es                " + Numero);
            return(Numero);

            }

    if ( (!terminacion[0]) && (!terminacion[1])) //00 ba

{ZEngine.WriteLog("biennnn 00");
            Numero=binH;
            ZEngine.WriteLog("El numero es                " + Numero);
            return(Numero);

            }

    if ( terminacion[0] && terminacion[1]) //11

{ZEngine.WriteLog("biennnn 11");
            Numero= (binH + 0.75);

            ZEngine.WriteLog("El numero es                " + Numero);
            return(Numero);

            }
    if ( terminacion[0] && (!terminacion[1])) //10

{ZEngine.WriteLog("biennnn 10");

            Numero= (binH + 0.5);
```

```
            ZEngine.WriteLog("El numero es                    " + Numero);
            return(Numero);
            }
 }

//--------------------------------Funcion Decimal to Hexadecimal Parametro A------------------//




function DecimalToHexParamA(Num){


var DecH = parseInt(Num,10);

ZEngine.WriteLog("Decim " + DecH);

ZEngine.WriteLog("Decim " + DecH.toString(2));

DecH <<=1;
ZEngine.WriteLog("Decim " + DecH.toString(2));

//Parte decimal

var NumDec =Num - (parseInt(Num,10));
ZEngine.WriteLog("Decim " + NumDec);
```

```
if (NumDec==0.5)
        { DecH = (DecH | 1);

ZEngine.WriteLog("numBinario " + DecH.toString(2));

ZEngine.WriteLog("numHDec " + DecH);
var bin= DecH.toString(2);

var Hex=DecH.toString(16);
return DecH;
}

if (NumDec==0)
        { DecH = (DecH | 0);

ZEngine.WriteLog("numBinario " + DecH.toString(2));

ZEngine.WriteLog("numHDec " + DecH);
var bin= DecH.toString(2);

var Hex=DecH.toString(16);
ZEngine.WriteLog("numHex " + DecH.toString(16));

return DecH;
}
}
```

```
//-------------------------Funcion Decimal To Hex--------------------//


function DecimalToHex(Numero){

var DecH = parseInt(Numero,10);
ZEngine.WriteLog("Decim " + DecH);
ZEngine.WriteLog("Decim " + DecH.toString(2));
DecH <<=2;
ZEngine.WriteLog("Decim " + DecH.toString(2));




//Parte decimal

var NumDec =Numero - (parseInt(Numero,10));
ZEngine.WriteLog("Decim " + NumDec);



if (NumDec==0.25)
        { DecH = (DecH | 1);

ZEngine.WriteLog("numBinario " + DecH.toString(2));
ZEngine.WriteLog("numHDec " + DecH);
var bin= DecH.toString(2);
var Hex=DecH.toString(16);
return DecH;

}
```

```
if (NumDec==0.5)
        { DecH = (DecH | 2);

ZEngine.WriteLog("numBinario " + DecH.toString(2));

ZEngine.WriteLog("numHDec " + DecH);
var bin= DecH.toString(2);

var Hex=DecH.toString(16);
return DecH;
}


if (NumDec==0.75)
        { DecH = (DecH | 3);

ZEngine.WriteLog("numBinario " + DecH.toString(2));

ZEngine.WriteLog("numHDec " + DecH);
var bin= DecH.toString(2);

var Hex=DecH.toString(16);
return DecH;
}


if (NumDec==0)
        { DecH = (DecH | 0);

ZEngine.WriteLog("numBinario " + DecH.toString(2));

ZEngine.WriteLog("numHDec " + DecH);
```

```
var bin= DecH.toString(2);

var Hex=DecH.toString(16);
return DecH;
}

ZEngine.WriteLog("numHex " + Hex);

//-----------------------------------Funcion Escribir PosX en BN----------------------//

}

function EscribirPosXBN(PosX,BN){

var sql="UPDATE BlindNodes set PosX="+PosX+" WHERE Direccion='"+BN+"'";
var adOpenDynamic=2;
var adLockOptimistic=3;
var rs = new ActiveXObject("ADODB.Recordset");

rs.open(sql,conn, adOpenDynamic, adLockOptimistic);

}
```

```
//---------------------------------Funcion Escribir PosY BN---------------------------//



function EscribirPosYBN(PosY,BN){



var sql="UPDATE BlindNodes set PosY="+PosY+" WHERE Direccion='"+BN+"'";
var adOpenDynamic=2;
var adLockOptimistic=3;
var rs = new ActiveXObject("ADODB.Recordset");

rs.open(sql,conn, adOpenDynamic, adLockOptimistic);
        }




//---------------------------------Funcion Borrar RN a actualizar----------------------//



function BorrarRNactualizar(){



var sql=" DELETE  FROM ActualizarRN ";
var adOpenDynamic=2;
var adLockOptimistic=3;
```

```
var rs = new ActiveXObject("ADODB.Recordset");
rs.open(sql,conn, adOpenDynamic, adLockOptimistic);

}

//---------------------------------Funcion actualizar RN en BD--------------------------//



function ActualizarRNX(Dir,PosX){



var sql="UPDATE ReferenceNodes set PosX="+PosX+" WHERE Direccion='"+Dir+"'";
var adOpenDynamic=2;
var adLockOptimistic=3;
var rs = new ActiveXObject("ADODB.Recordset");
rs.open(sql,conn, adOpenDynamic, adLockOptimistic);


}

//---------------------------------Funcion actualizar RN en BD--------------------------//



function ActualizarRNY(Dir,PosY){
```

```
        var sql="UPDATE ReferenceNodes set PosY="+PosY+" WHERE Direccion="'+Dir+"'";
        var adOpenDynamic=2;
        var adLockOptimistic=3;
        var rs = new ActiveXObject("ADODB.Recordset");
        rs.open(sql,conn, adOpenDynamic, adLockOptimistic);


}
}



var zTestScript:ZScriptSampleClass = new ZScriptSampleClass();

//Conexion a la BD

        var wscript = new ActiveXObject("wscript.shell");
        var path = wscript.CurrentDirectory;
        var mdb = path + "\\BD_Proyecto\\BD_Proyecto.mdb";
        var conn = new ActiveXObject("adodb.connection");
        var connString = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source="+ mdb;
        conn.open(connString);


zTestScript.Start();
```

ANEXO D

Código de programa Matlab

El siguiente código corresponde al código ejecutado en respuesta del evento del boton Start/Stop.

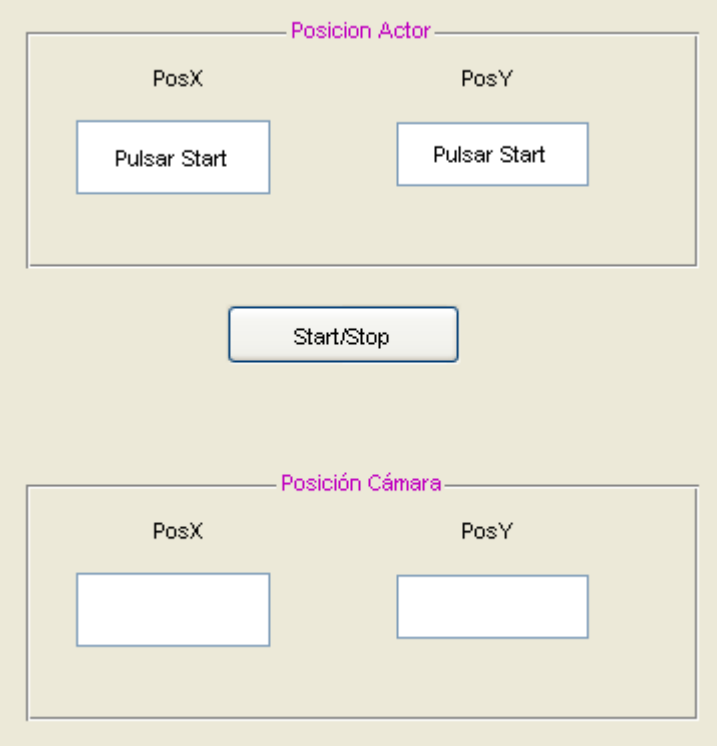


Figura 47 : Interfaz grafico de usuario

```
% --- Executes on button press in start.
function start_Callback(hObject, eventdata, handles)
% hObject    handle to start (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


estado = get(handles.start,'Value');
```

```
disp(estado);
cla reset

while (estado==1)


   pause(0.5);
   cla reset

%                                   Conexion BD

DB = adodb_connect('PROVIDER=Microsoft.Jet.OLEDB.4.0; Data
Source=C:\Texas Instruments\ZStack-1.4.3-1.2.0\Tools\Z-
Tool\BD_Proyecto\BD_Proyecto.mdb;');


%                                    Lectura RN

 record = adodb_query(DB, 'select  * from ReferenceNodes ');


 R1x =(double(record.posx{1})/100);
 R1y =(double(record.posy{1})/100);

 R2x =(double(record.posx{2})/100);
 R2y =(double(record.posy{2})/100);

 R3x =(double(record.posx{3})/100);
 R3y =(double(record.posy{3})/100);

 R4x =(double(record.posx{4})/100);
 R4y =(double(record.posx{4})/100);




%                                     Nodo CAMARA

record7 = adodb_query(DB, 'select  * from BlindNodes where
Direccion="6534"');


Camarax=(double(record7.posx)/100);
Camaray=(double(record7.posy)/100);



set(handles.PosXCamara,'String',num2str(Camarax));
set(handles.PosYCamara,'String',num2str(Camaray));
```

```
 %                                  Lectura del resto de BN

 [Struct Table] = adodb_query(DB, 'select  * from BlindNodes');



hold all

for i=2:length(Struct.posx)

    px= (double(Struct.posx{i})/100);
    py= (double(Struct.posy{i})/100);

plot(R1x,R1y,'rx',R2x,R2y,'rx',R3x,R3y,'rx',R4x,R4y,'rx',px,py,'bs',Ca
marax,Camaray,'bo','LineWidth',2,...
        'MarkerEdgeColor','k',...
        'MarkerFaceColor','g',...
        'MarkerSize',10);

    axis square
    text(double(px), double(py),'     ACTOR');
    text(double(px+1), double(py), num2str(i-1));


end


 valor = get(handles.popupmenu3,'Value');
 set(handles.text17,'String',valor );
 f=double(Struct.posx{valor+1})/100;


 if f >Camarax

    a= Camarax:0.1:f;

 else
    a= f:0.1:Camarax;
 end


 Nx = double(Struct.posx{valor+1})/100;
 Ny = double(Struct.posy{valor+1})/100;

 x = Distancia(Nx, Ny,Camarax,Camaray);
 set(handles.edit16,'String',x);

 set(handles.PosX,'String',double(Struct.posx{valor+1})/100);
```

```
 set(handles.PosY,'String',double(Struct.posy{valor+1})/100);


 plot(a,Lineal(a,Nx,Ny,Camarax,Camaray),'--');

  b= size(a);
  c= floor(b(2)/2);

  text(a(c),Lineal(a(c),Nx,Ny,Camarax,Camaray), num2str(x));

  text (Camarax, Camaray, '   CAMARA');
  hold off

DB.release;


estado = get(handles.start,'Value');
end
    cla reset

% Hint: get(hObject,'Value') returns toggle state of start
```