



UNIVERSIDAD DE BUENOS AIRES
FACULTAD DE INGENIERÍA

LOCALIZACIÓN DE NODOS EN BASE A RSSI

GUIDO BUSCETTI CASTRO
LEONARDO GIACCONE
JOAQUIN MUGUERZA

66.48 SEMINARIO DE SISTEMAS EMBEBIDOS

14 de diciembre de 2012

Índice

1. Objetivo	3
2. Introducción	3
3. Roadmap	4
4. Estimación de Distancia en Base a RSSI	6
4.1. Introducción	6
4.2. Modelo de Propagación	6
4.3. Limitaciones del Modelo Utilizado	8
4.4. Método Alternativo de Cálculo de Distancia	8
5. Trilateración	11
5.1. Introducción	11
5.2. Punto de Partida del Problema	11
5.3. Método de Cálculo Utilizado	11
6. Estándar IEEE 802.15.4	13
6.1. Generalidades	13
6.2. Topología	13
6.3. Capa física (PHY)	13
6.4. Subcapa de Control de Acceso al Medio (MAC)	15
7. Hardware	17
7.1. Descripción de Componentes	17
7.2. Especificaciones Técnicas	20
7.3. Listado de Componentes y Costos	20
7.4. Prototipo	21
8. Software	22
8.1. Funcionamiento de la red	22
8.2. Nodos Fijos	22
8.3. Nodo Móvil	23
9. Resultados	26
9.1. Simulación de método de estimación de distancia	26
9.2. Simulación de método de trilateración	26
9.3. Validación Experimental	27
10. Conclusiones	31
Apéndice A: Esquemáticos	32
A.1. Esquemático del nodo	32
A.2. Esquemático del prototipo	33

B. Código	34
B.1. main.c	34
B.2. fijo.c	37
B.3. movil.c	39
B.4. rssи.c	44
B.5. cc2520.c	47
B.6. cc2520mac.c	52
B.7. config.h	58
B.8. includes.h	58
B.9. cc2520.h	59
B.10.cc2520mac.h	66
B.11.fijo.h	68
B.12.movil.h	68
B.13.rssi.h	69
Bibliografía	71

1. Objetivo

Se busco diseñar y desarrollar un dispositivo capaz de realizar tracking de animales. Para esto se planteo un *roadmap* compuesto de 10 puntos a través de los cuales se llegaría a obtener un sistema de nodos (3 fijos, 1 móvil y 1 encuestador) con el cual se podría precisar la posición del nodo móvil en relación a los 3 nodos fijos, utilizando mediciones de intensidad de potencia de señal (RSSI).

2. Introducción

En Argentina, Latinoamérica y gran parte del mundo existe la necesidad de actualizar constantemente la tecnología agro-ganadera. Una forma de darle más valor y hacerla más eficiente es con el uso de redes de sensores inalámbricas (WSN).

En los últimos años han tomado especial relevancia aquellos dispositivos electrónicos diseñados para tomar mediciones del medio que los rodea. Un enfoque clásico de esta temática implica que el sistema adquiere datos de sensores distribuidos en el medio, lo cual requiere en la mayoría de los casos un cableado especial que implica un alto coste de instalación. Al mismo tiempo, este enfoque requiere de un procesamiento centralizado, el cual complica aún más su implementación conforme aumenta el número de sensores o nodos a monitorear. Se presentará en este artículo el diseño e implementación de un nodo acorde con la norma de comunicaciones inalámbricas IEEE 802.15.4. Sus principales características son su tamaño reducido, el funcionamiento a batería de Liion, bajo consumo, bajo costo y capacidad de procesamiento in situ, utilizando un framework de software desarrollado especialmente para brindar servicios de red adhoc y pre procesamiento de los datos obtenidos para optimizar la estructura descentralizada de la red.

Decidimos realizar un sistema de tracking de ganado, con el fin de poder brindar información de suma importancia a los productores, dejando la posibilidad de agregar sensores en la red WSN para incorporar nuevas funcionalidades, como detectar celos, enfermedades, etc y así poder agregar valor a las exportaciones y reducir costos de los productores.

El sistema a implementar constará con nodos fijos, móviles y un nodo encuestador. Para simplificar las pruebas se utilizarán 3 nodos fijos, un nodo móvil y un nodo encuestador. El funcionamiento básico será:

1. El nodo encuestador hace un *request* de la posición del nodo móvil.
2. Los nodos toman el mensaje, y lo retransmiten.
3. El nodo móvil recibe las señales de los nodos fijos, y a partir de estas, calcula las distancias relativas a cada nodo fijo.
4. El nodo móvil realiza la trilateración y envía su ubicación a la red.
5. El nodo encuestador recibe la posición del nodo móvil y la muestra en pantalla.

3. Roadmap

El proyecto consta de 10 hitos o puntos, los cuales se detallan a continuación, así como el estado de completitud de cada uno.

1. **Informe y especificaciones técnicas, manual de usuario:** Desarrollar un informe detallado acerca del funcionamiento del nodo, y del producto, con breve descripción del software, especificaciones técnicas, y manual de uso para un usuario no experimentado. **Estado:** este informe constituye una buena parte de la documentación programada.
2. **Montaje de componentes:** Puesta en marcha, soldadura de los componentes faltantes en los nodos diseñados por Pablo Ridolfi. **Estado:** Completado. Se terminaron de soldar, y de verificar 4 placas de Pablo. La soldadura fue de componentes de montaje superficial que requieren gran precisión. Para ello se utilizaron las herramientas disponibles en el Laboratorio de Sistemas Embebidos: soldador por chorro de aire caliente, soldador con temperatura controlada, y sobre todo, lupa.
3. **Puesta en marcha de hardware:** verificar, y poner en marcha las placas de Pablo fuera de funcionamiento. Implementar una alternativa a dichos nodos. **Estado:** Completado. Todos los nodos ensamblados fueron puestos a punto. Además se implementaron nodos provisarios con fuente de alimentación propia por medio de batería.
4. **Pruebas de enlace 802.15.4:** Verificar la comunicación entre dos nodos mediante este estándar. Probar las distintas opciones provistas por el estandar para la capa PHY y la subcapa MAC. **Estado:** Completado. Se estableció comunicación con varios nodos, utilizando direccionamiento y no. Se obtuvieron datos provistos por el estándar: direcciones de destino y origen, RSSI, tipo de frame, etc.
5. **Pruebas de red:** Extender la comunicación a varios nodos. Probar las distintas topologías de red, y distintos tipos de nodos posibles. **Estado:** Completado. Se probaron las topologias, y las funcionalidades de nodos coordinadores y nodos sensores. Además se probaron los distintos tipos de frames: beacon, data, etc.
6. **Cálculo de distancias en base a RSSI. Definición de métodos de corrección:** Estimar distancias mediante algoritmos en base a las mediciones de RSSI provistas por la capa PHY del estándar. Proponer mejoras a dichos algoritmos. **Estado:** Completado. Se implementaron dos tipos de algoritmos, con distinto principio de funcionamiento, con buen resultado teórico. Los resultados prácticos no fueron óptimos por limitación de hardware. Se proponen mejoras de harware.
7. **Integración de algoritmos de localización (trilateración):** Implementación de método de trilateración en el microcontrolador. **Estado:** Completado. Se implementó el método de cuadrados mínimos lineales para el cálculo de la posición en función de distancias a nodos fijos. Se validó corriendo el algoritmo en el microcontrolador con distancias propuestas.

8. **Diagrama de las tareas de aplicación:** Diseñar las tareas de aplicación de la red: tareas para los nodos fijos, y móviles. **Estado:** Completado. Se diagramaron y programaron las rutinas correspondientes a cualquier nodo fijo y a cualquier nodo móvil, de manera tal que sean capaces de sincronizar la red, y realizar una trilateración a pedido, sin trabajo extra. Requiere de muy poca configuración previa. Permite el agregado de muchos nodos fijos y móviles de forma sencilla.
9. **Implementación de las tareas y puesta en marcha:** Puesta en marcha, y verificación experimental de las tareas diagramadas anteriormente. **Estado:** Completado. Se puso en marcha una red con tres nodos fijos, y un nodo móvil. La red se sincronizó automáticamente, recibiendo el nodo móvil mensajes de todos los nodos fijos cada un tiempo determinado, y este haciendo la trilateración los nodos de mejor señal.
10. **Puesta en marcha del nodo asociado al usuario (nodo encuestador):** Implementar un nodo conectado a una computadora que haga de interfaz entre la red y el usuario. Para ello utilizar un nodo fijo, con comunicación por medio de un puerto USB. **Estado:** No realizado.

4. Estimación de Distancia en Base a RSSI

4.1. Introducción

Gracias al avance tecnológico, en áreas de la electrónica, la computación, y la comunicación, se ha promovido el desarrollo de sensores de muy bajo consumo, portátiles, capaces de procesamiento de datos y de comunicación inalámbrica. Para muchas aplicaciones de redes inalámbricas de sensores (WSN en adelante) es de gran interés la localización de estos nodos, móviles o no. Siendo el costo de cada unidad un factor limitante, el método de localización debe resolverse con bajos recursos.

Los costos prohibitivos de la utilización del sistema GPS llevan a buscar otra alternativa. Actualmente, goza de gran popularidad en aplicaciones móviles por su bajo costo, bajo consumo, gran versatilidad y su fácil uso, el protocolo IEEE 802.15.4. Dado que el nodo utilizado, cuenta con un *transceiver* que cumple con el estándar 802.15.4, se busca aprovechar el mismo para realizar la localización.

Existen tres métodos principales para la localización mediante el uso de ondas de radio: la medición de la diferencia de tiempo de arribo (TDoA), la medición de la intensidad de potencia de la señal recibida (RSSI), y la medición del ángulo de arribo (AoA).

El método de localización basado en tiempos de arribo se basa en la medición del intervalo de tiempo que tardan dos señales, enviadas por dos emisores en distintas posiciones al mismo instante, en llegar al nodo. Dado que en la aplicación de este trabajo, se trabajan con distancia del orden de decenas de metros, por la velocidad de propagación de las ondas, esto implica medir con precisión tiempos de nanosegundos o menos. Dado que es imposible con el hardware dado, se descarta esta opción.

El método medición de ángulo de arribo se basa en la utilización de un set de antenas, dispuestas apropiadamente, que permiten calcular el ángulo desde el que fue enviada la señal. Con mediciones de distintos emisores, es posible, la trilateración. Sin embargo, este método requiere la utilización varias antenas, mas costosas, por lo tanto, impracticable.

El último método, sin embargo, no requiere hardware especial. Dentro de la norma 802.15.4 está contemplado que la capa PHY (la capa física, en este caso, implementada en el CC2520) provea para cada paquete recibido, un indicador de RSSI o de LQI (*Link Quality Indicator*). Con estos datos, y ajustando a un modelo físico de propagación de las ondas, es posible estimar la distancia. Sin embargo, las precisiones obtenidas con este método no son tan buenas como con los otros.

4.2. Modelo de Propagación

El primer modelo que se utiliza para relacionar la intensidad de potencia, con la distancia, se basa en la siguiente fórmula, basada en la ecuación de Friis[11]. Este tipo de propagación supone un modelo sumamente idílico, y contempla solo un camino directo de señal (sin considerar los posibles caminos que surgen de la reflexión con el suelo).

$$RSSI[dBm] = -10(10 \cdot n \cdot \log_{10}(d) + A) \quad (1)$$

Donde la potencia de señal inicial A describe el valor absoluto de RSSI a 1 m de distancia del

transmisor. El coeficiente de propagación n indica el deterioro de la señal. Ambos parámetros deben ser determinados empíricamente.

Potencia Inicial A

La medición de la potencia inicial A se realiza a 1 m del receptor. Esto se hace así, dado que se considera esta distancia la distancia mínima posible que se puede estimar.

Coeficiente de propagación n

Para determinar n , se miden muchos valores de RSSI hasta 40 m, de a pasos de 1 m. Luego, para cada distancia, se hace un promedio de los valores de RSSI, y luego se ajusta el valor de n con una curva de acuerdo a la ecuación 1 . Generalmente toma valores entre 1 y 4.

Una vez calculado el ajuste, basta cargar los valores de A y de n en el programa, y calcular la distancia de acuerdo a la expresión:

$$d = 10^{-(RSSI+A)/(10 \cdot n)} \quad (2)$$

Con el fin de disminuir posibles variaciones en las estimaciones, se promedian varias muestras de RSSI antes de ingresarlas en la ecuación.

Para las mediciones presentadas en la figura 1 , se obtuvo $A = 45$, $n = 2,322$.

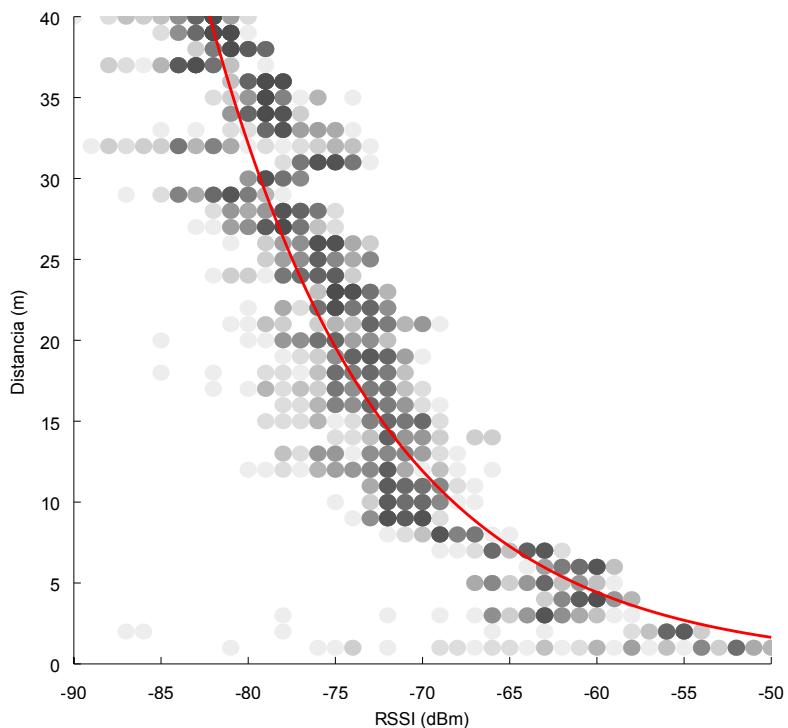


Figura 1: Mediciones de RSSI en distancias de hasta 40 m. La línea roja representa la curva que ajusta a los valores promedio.

4.3. Limitaciones del Modelo Utilizado

El método de estimación de distancia presentado, como ya se mencionó, se basa en un modelo físico de propagación directa de la señal. Sin embargo, este no es el caso para nuestra aplicación, donde los nodos se ubican aproximadamente 1 m por encima del suelo. A esta distancia del suelo, los caminos generados por las reflexiones en el mismo no son despreciables, por lo que el modelo no se ajusta a la realidad.

Un modelo más acorde a la realidad es el modelo de dos rayos[2]. Este modelo supone dos caminos para la señal: uno directo, y otro indirecto, con una reflexión en el suelo. Las señales se suman en el punto de recepción. Dado que ambas señales recorren distintas distancias, las fases son distintas, haciendo que la suma tenga máximos y mínimos, no por la degradación de la señal, sino por la interferencia.

En la figura 2 se observa este fenómeno. La curva de ajuste es coherente con las mediciones para menos de 5 m. A los 7 m de distancia, se produce interferencia destructiva, obteniendo en realidad 8dB menos de potencia de los que predice el ajuste. Este tipo de situaciones no pueden ser corregidas con el modelo actual. La interferencia puede llegar a ser disminuida a nivel hardware: una mejor antena, con un plano de tierra mas grande, disminuye las variaciones por interferencia en gran medida[4]. Por lo tanto, con el interés de evitar estos errores de cálculo, aprovechando el hardware a mano, se propone otro método.

4.4. Método Alternativo de Cálculo de Distancia

Dado que los datos experimentales muestran que la relación entre potencia y distancia no es biyectiva, entonces el método del ajuste no es óptimo, ya que presenta falencias en algunos puntos del espacio, donde la estimación de distancia llega a ser hasta 10 m desviada.

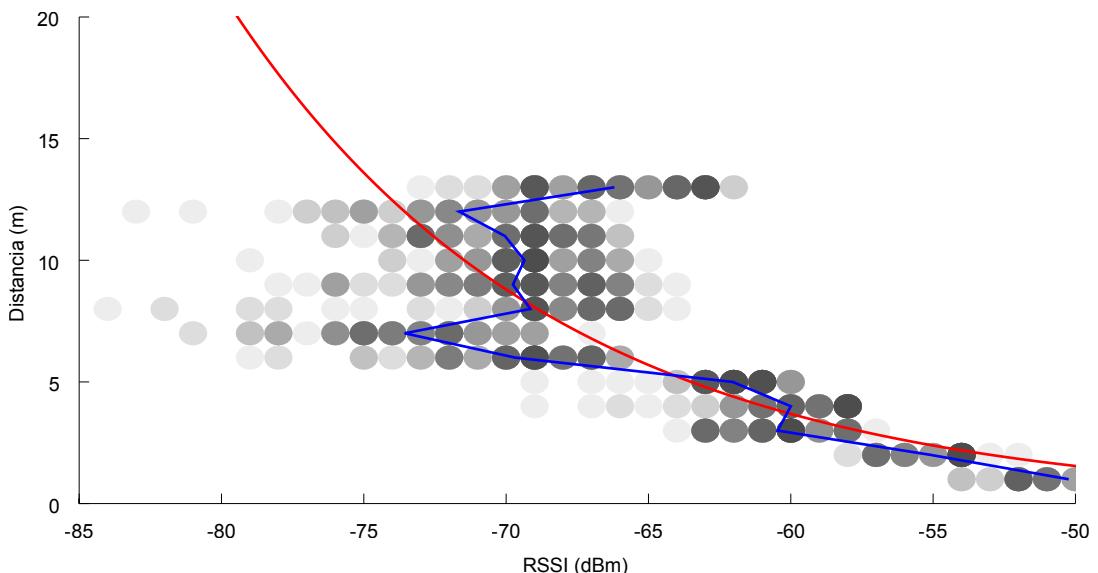


Figura 2: Mediciones de RSSI en distancias hasta 15 m. La curva roja es el ajuste, y la curva azul muestra el promedio.

Se implementa entonces otro método de estimación de distancia, basado en cálculos probabilísticos, similar al algoritmo presentado en [4]. El método se basa en obtener distancia partiendo de un conjunto de mediciones de RSSI, calculando luego la probabilidad *a priori* para cada distancia, y luego eligiendo la mayor.

Para esté método, se requieren de antemano mediciones para el calculo de las probabilidades. Se toman valores de RSSI para distancias de 1 m a 40 m, con pasos de 1 m. Luego se construyen histogramas para cada valor de RSSI. El transceiver CC2520 proveé el valor de RSSI como un número entero signado de 8 bits, por lo que puede tomar valores entre -128 y 127 [dBm]. Los resultados experimentales arrojan valores del rango de -40dBm y -90dBm, aproximadamente.

Para cada posible valor (entero) de RSSI, entonces se construye un histograma, en función de la distancia en la que se obtuvo dicha medición. Normalizando la suma a la unidad se obtiene una distribución de probabilidad rudimentaria. Luego, se ajusta el conjunto de valores del histograma a una función de densidad de probabilidad similar. En este caso, se approximó por una distribución normal dado que se trata de una función matemática sencilla, aunque un ajuste por curvas del tipo $\tan x$ es mejor.

La figura 3 muestra tales histogramas, con las curvas de ajuste superpuestas. Para cada ajuste realizado (uno para cada valor de RSSI posible), se obtiene un valor de la media μ y un valor de varianza σ .

Se construye entonces una tabla, con tantas filas como posibles valores de RSSI se tengan, y con tantas columnas como posibles valores de distancia se pueden tener. Cada fila entonces, se calcula muestreando la curva que ajusta del valor de RSSI que corresponde en las distancias posibles que se pueden estimar. Por ejemplo, en nuestro caso, se muestrea desde 1 m a 40 m, con pasos de 1 m. Los valores de RSSI posibles, son todos los enteros en el rango de -40 dBm a -94 dBm.

Luego, para estimar la distancia, se suma para cada valor de RSSI medido, la fila correspondiente a un vector fila que acumula todas las sumas. Una vez hecho esto para todas las mediciones, se busca el máximo (probabilidad *a priori* máxima). Aquella posición donde se produce el máximo es la distancia elegida (ver figura 4).

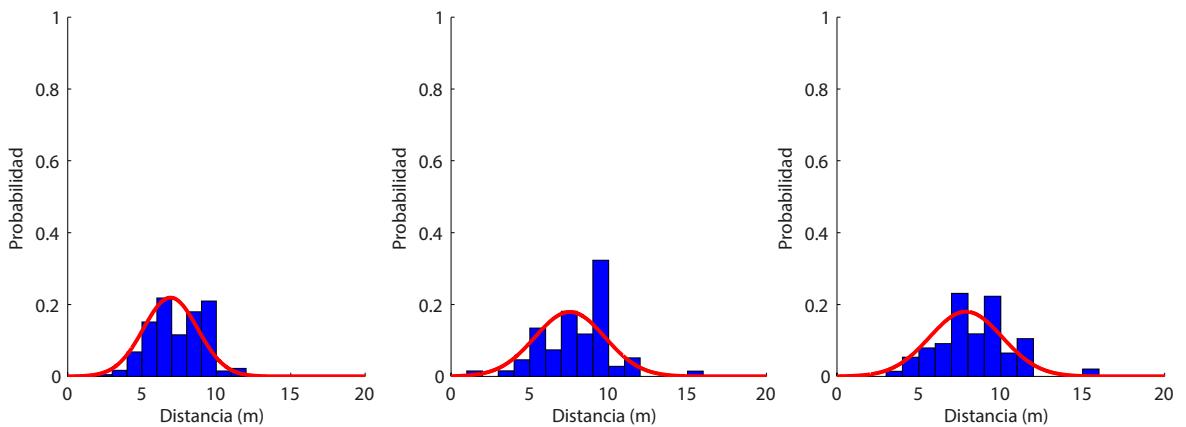


Figura 3: Histogramas para valores de RSSI (de der. a izq.) de -60, -61 y -62 dBm. La curva roja es la curva de densidad de probabilidad normal que ajusta.

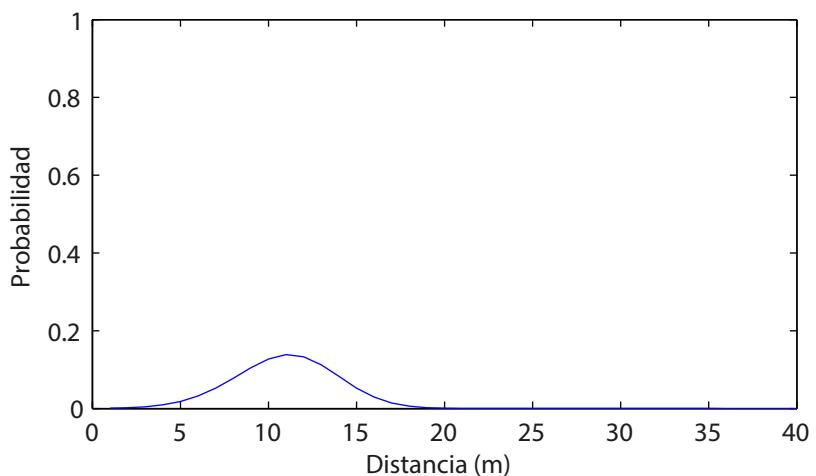


Figura 4: Vector que acumula las probabilidades. El máximo está en 11 m. Las mediciones de RSSI fueron tomadas a 10 m.

5. Trilateración

5.1. Introducción

Los sistemas de cálculo de distancia por trilateración son utilizados en aplicaciones donde otro tipo de sistemas de posicionamiento son impracticables. El cálculo de trilateración usa mediciones de distancia para determinar las tres coordenadas de posiciones desconocidas. Estos cálculos facilitan la implementación de sistemas complejos de medición de ángulos. Generalmente los datos disponibles son meras aproximaciones. A pesar de eso, pueden determinarse posiciones con buena exactitud a través del uso iterativo de métodos de cuadrados mínimos.

5.2. Punto de Partida del Problema

Para realizar la trilateración, se suponen conocidas las distancias de un punto desconocido a otros puntos de posición determinada. Las distancias obtenidas responden a la ecuación:

$$r_i = \sqrt{(x - x_i)^2 + (y - y_i)^2 + (z - z_i)^2} \quad (i = 1, 2, \dots, n), \quad (3)$$

donde i denota el número del nodo fijo, n es la cantidad total de nodos. La posición de cada nodo fijo (conocida) es (x_i, y_i, z_i) , mientras que (x, y, z) es la posición del nodo móvil a determinar.

El enfoque obvio para resolver el problema, es considerar que la posición del nodo móvil (x, y, z) es el punto en donde se intersectan todas las esferas cuyos centros son los nodos fijos. Los radios de dichas esferas son las distancias a cada nodo. La ecuación para cada esfera queda:

$$(x - x_i)^2 + (y - y_i)^2 + (z - z_i)^2 = r_i^2 \quad (4)$$

El punto de intersección se puede calcular entonces, completando las ecuaciones para todos los i posibles, y resolviendo el sistema de n ecuaciones no lineales, hasta eliminar dos coordenadas. Este enfoque no es factible de aplicarse, ya que requiere la resolución de ecuaciones lineales de alto orden.

Linealizar las ecuaciones entonces es una condición necesaria, lo que convierte al problema en el cálculo de intersección de varios planos. Si las distancias disponibles son exactas, entonces el problema tiene solución única y sin error. Para el trabajo presente, sólo se requieren conocer dos coordenadas (no se considera la coordenada z), entonces un número mínimo de tres distancias es necesario. Dado que las distancias medidas, generalmente no son exactas, entonces la resolución directa no es útil, por lo que se requiere otra metodología.

5.3. Método de Cálculo Utilizado

Dado que el sistema de ecuaciones es linealizado, entonces puede representarse como un sistema del tipo:

$$Ax = b, \quad (5)$$

donde A es una matriz conocida, b es un vector conocido, y se quiere hallar el vector x que satisface la ecuación. La linealización y posterior desarrollo de las expresiones se encuentra en

[5]. Se obtiene que el vector x contiene la posición del nódulo móvil, el vector b contiene las posiciones de los nodos fijos, y las distancias al nódulo móvil, y la matriz A contiene datos de las posiciones de los nodos fijos:

$$\mathbf{A} = \begin{bmatrix} x_2 - x_1 & y_2 - y_1 \\ x_3 - x_1 & y_3 - y_1 \\ \vdots & \vdots \\ x_n - x_1 & y_n - y_1 \end{bmatrix}, x = \begin{bmatrix} x - x_1 \\ y - y_1 \end{bmatrix}, b = \begin{bmatrix} b_{21} \\ b_{32} \\ \vdots \\ b_{n1} \end{bmatrix} \quad (6)$$

Donde $b_{j1} = 1/2 \cdot [r_i^2 - r_j^2 + d_{j1}^2]$, siendo r_i la distancia al nódulo i medida, y d_{j1} la distancia entre nodos el nódulo 1 y el nódulo j -ésimo.

El método más sencillo para la resolución del sistema, y el utilizado en este trabajo, es el de mínimos cuadrados lineales. Este método, calcula la solución real más cercana al set de distancias disponible (que generalmente no son coherentes). El vector x aproximado puede obtenerse con la siguiente expresión:

$$x = (A^T A)^{-1} A^T b \quad (7)$$

Puede pasar que la matriz $A^T A$ esté muy cercana a ser no inversible, generando grandes errores numéricos en el cálculo. Existen otros métodos que preveén esta situación, pero que son de mayor complejidad computacional. El método elegido resulta ser muy bueno para estimar posiciones dentro del área delimitada por los nodos fijos. Fuera de la misma la estimación se vuelve más inexacta.

6. Estándar IEEE 802.15.4

6.1. Generalidades

Las *Wireless personal area networks* (WPANs), son usadas para transmitir información a través de distancias relativamente cortas. A diferencia de las *Wireless local area networks* (WLANs), las conexiones a través de las WPANs involucran poca infraestructura. Esto permite el desarrollo de soluciones eficientes y de bajo costo.

El estándar 802.15.4[8] del *Institute of Electrical and Electronics Engineers* (IEEE) define la capa física (PHY) y la subcapa de control de acceso al medio (MAC) del modelo OSI para conectividad inalámbrica de bajo *datarate* a ser usado especialmente en dispositivos móviles de bajo consumo, para distancias cortas.

La norma 802.15.4 se encarga de establecer una comunicación confiable, mediante detección de error, acuse de recibo y retransmisiones; adaptado para una simple implementación, requiriendo relativamente poca potencia de procesamiento y facilitando la operación en bajo consumo.

6.2. Topología

Dependiendo del tipo de aplicación, la norma opera en dos topologías: la topología estrella y la topología punto a punto (P2P) (ver figura 5).

Para ello preveé la utilización de dos tipos de dispositivos: los dispositivos con funcionalidades completas (*Full Function Devices*), y los dispositivos con funcionalidades reducidas (*Reduced Function Devices*).

Los primeros, pueden actuar tanto como coordinadores de la red, o como dispositivos, y pueden comunicarse tanto con FFDs como con RFDs. Además pueden asociar nuevos dispositivos a la red. Pueden contar con alimentación externa o no.

En cambio, los dispositivos RFDs, son generalmente dispositivos de recursos limitados, sin alimentación externa, y que no tienen la capacidad ni de actuar como coordinadores ni de comunicarse entre ellos. Generalmente son estos los que tienen la capacidad de sensado.

6.3. Capa física (PHY)

La capa física es la encargada de transmitir y de recibir los paquetes de datos. Para ello, tiene la función de activar y desactivar la radio, evaluar el canal (CCA), seleccionar el canal de transmisión, detectar la energía (ED) y la calidad de enlace (LQI) en el mismo.

El estándar especifica tres bandas principales de frecuencia:

- **Banda de 868 MHz.**
- **Banda de 915 MHz.**
- **Banda de 2.4 GHz.**

Cada banda cuenta a su vez con varios canales, y puede tener uno o más tipos de modulación (BPSK,ASK,OQPSK), lo que le confiere distintas tasas de transmisión, que van desde 300 kb/s a los 2000 kb/s.

La norma le especifica también a la capa física, la estructura del PPDU (*PHY Protocol Data Unit*), que es el formato de los mensajes enviados por el canal (ver tabla 6.3).

Octets				
	1		variable	
Preamble	SFD	Frame length (7 bits)	Reserved (1 bit)	PSDU
SHR		PHR		PHY payload

Cuadro 1: Esquema de la estructura del PPDU.

El formato consta de tres partes principales:

- **SHR (Synchronization Header):** permite al receptor sincronizarse e identificar el inicio de una trama (SFD).
- **PHR (PHY Header):** Contiene la información de la longitud de la trama.
- **PHY payload:** Contiene una cantidad variable de bytes que componen la trama de la subcapa MAC.

Además, especifica cosas como:

- **Tiempo de conmutación de Tx a Rx y viceversa:** ambos deben ser menores o iguales a 12 símbolos.
- **Potencia de transmisión:** debe ser de por lo menos -3 dBm. Puede ser menor para reducir el consumo energético.

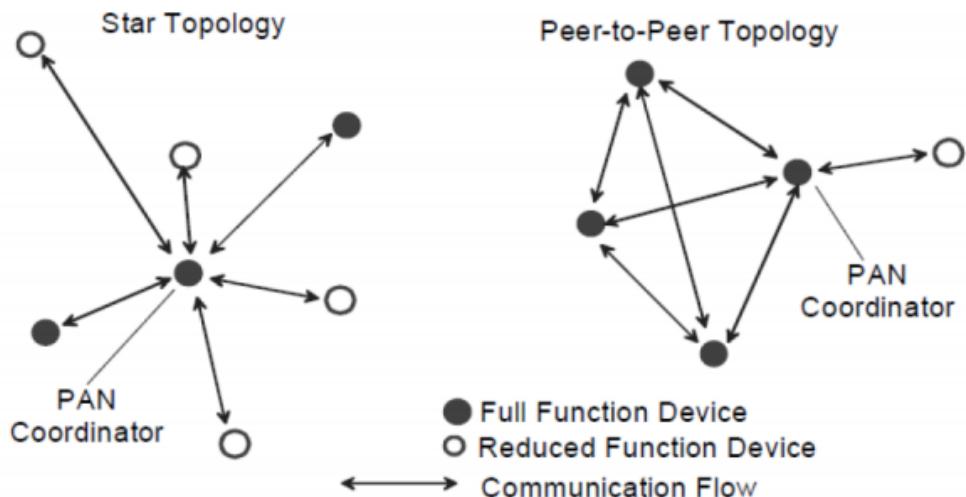


Figura 5: Topologías previstas por la 802.15.4.

- **Mediciones de energía (ED):** la norma indica la medición de energía una duración de 8 símbolos.
- **Indicador de calidad de enlace (LQI):** Para calcular el LQI se utiliza la medición de ED en el canal, o una estimación de la relación señalruído, o una combinación de ambos. Es un valor signado de 8 bits.
- **Evaluación del canal (CCA):** el escaneo dura 8 símbolos y se definen tres modos de CCA: nivel de energía sobre un valor umbral (CCA 1), detección de portadora (CCA 2), y detección de portadora con nivel de energía sobre un valor umbral (CAA 3).

6.4. Subcapa de Control de Acceso al Medio (MAC)

La subcapa MAC maneja el acceso al canal de radio físico, y es responsable de las siguientes tareas:

- Proveer un enlace confiable entre dos entidades MAC.
- Generar paquetes balizas (*beacons*) si se trata de un coordinador.
- Soporte de asociación y disociación de dispositivos a la red.
- Soporte de la seguridad de la red.
- Emplear el mecanismo CSMA-CA (*Carrier Sense Multiple Access with Collision Avoidance*) para acceder al medio.
- Manejo y mantenimiento del mecanismo GTS (*Guaranteed Time Slot*).

Para hacer efectivo el control de acceso, la norma define dos tipos de direccionamiento para la capa MAC: el direccionamiento extendido de 64 bits, y el direccionamiento corto, de 16 bits. El direccionamiento extendido constituye la dirección IEEE del dispositivo. Su valor se establece en fábrica y es único. El direccionamiento corto se utiliza en aplicaciones para reducir la cantidad de datos de cabecera enviados en las tramas. También se define un PAN ID, o número de red. Es un número de 16 bits.

Cada trama enviada debe poseer direcciones de dispositivo y de red tanto para el emisor como para el receptor. Utilizando como destino la dirección corta 0xFFFF, se envía la trama a todos los miembros de la red. De la misma forma, este valor utilizado como PAN ID de destino, indica que todas las redes aceptarán esa trama.

Cada trama MAC (la que recibe y envía la capa MAC) puede tener hasta 127 bytes de largo, y está dividida en tres partes principales (ver figura 6).

- **MAC Header:** contiene campos de control de protocolo, control de flujo y direccionamiento. Su formato varía con el tipo de trama.
- **MAC Payload:** Es la carga útil que interactúa con las capas superiores.
- **MAC Footer:** Contiene la suma de verificación (CRC-16).

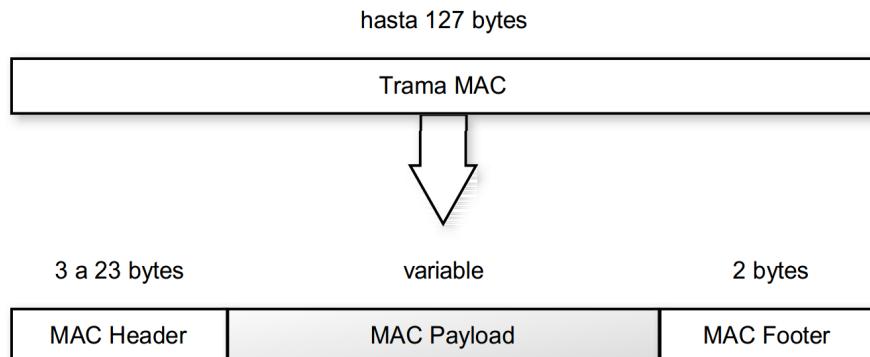


Figura 6: Trama MAC.

El MAC Header está también dividido en tres partes: el Frame Control, el número de secuencia, y el direccionamiento. El Frame Control contiene todas los datos que especifican el tipo de mensaje y otros datos necesarios para interpretar correctamente la trama, por parte del receptor. Algunos de estos parámetros son: *Frame Type, Security Enabled, Acknowledgement Request, PAN ID Compression, Destination Addressing Mode, Source Addressing Mode*.

El estándar define cuatro tipos distintos de tipos de tramas: Baliza (beacon), Datos (data), Acuse de recibo (Acknowledgment) y Comando MAC (MAC Command).

En este trabajo no se ha implementado la utilización de mensajes de Acknowledgement, ni la verificación de errores de la trama.

7. Hardware

El hardware utilizado en el proyecto, fue provisto por el Laboratorio de Sistemas Embebidos de la Facultad de Ingeniería. El diseño de los nodos es obra de Pablo Ridolfi[6]. Se trabajó con dos modelos, debido a problemas de disponibilidad de hardware. Ambos diseños tienen su mismo bloque funcional (microcontrolador+transceiver), aunque varían el modelo del microcontrolador (pero mantiene su arquitectura, Cortex-M3) y algunos componentes adicionales.

7.1. Descripción de Componentes

Todos los nodos, independientemente de su función en la red inalámbrica, están implementados sobre el mismo hardware. Los nodos cuentan con un microcontrolador, encargado de realizar todo el procesamiento de datos del nodo. La comunicación inalámbrica se realiza mediante un transceiver que cumple con el estándar IEEE 802.15.4, y un front-end que mejora el alcance y performance de la transmisión y recepción; además se cuenta con comunicación cableada, por medio de un puerto USB. Completan a la placa diversos sensores (GPS, sensor de luz, temperatura, etc.), dispositivos HMI (LEDs, pulsadores), y un circuito de alimentación por medio de una batería de Li-Pol.

Microcontrolador

El microcontrolador elegido es el modelo LPC1343 de la firma NXP. Con un núcleo Cortex-M3 de 32 bits, posee un tamaño y costos aceptables: encapsulado LQFP48, con un costo de US\$3.30 para 100 unidades[1]. Fue elegido por sobre el LPC1769, de mayores prestaciones, debido al menor consumo.

Algunas características del controlador son: frecuencia de clock de hasta 72 MHz, 32 kbytes de memoria flash de programa, 8 kbytes de SRAM, ISP, diversas interfaces de comunicación (UART, USB, SPI, I²C), 42 GPIO, 10bit ADC, DAC, tres modos de consumo reducido.

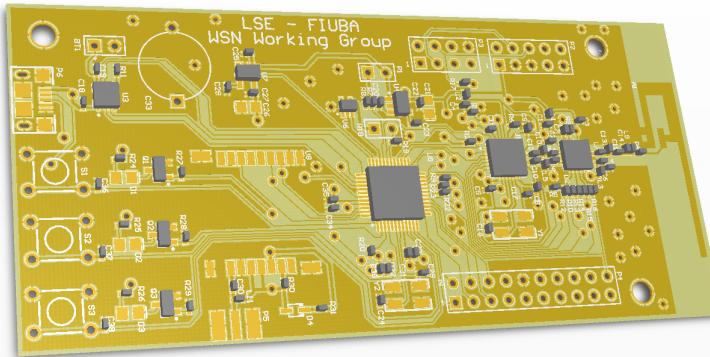


Figura 7: Render 3D del nodo.

Transceiver+FrontEnd

Para la transmisión y recepción se utiliza el transceiver CC2520[9] de Texas Instruments y un frontend compatible, el CC2591[10].

El CC2520 es un transceiver RF que opera de acuerdo a Zigbee/IEEE 802.15.4 en la banda de 2.4GHz. Cuenta con un bajo nivel de consumo: 18.5 mA durante recepción, , 33.6 mA durante la transmisión a +5 dBm y menos de 1 μ A en estado desocupado. Acepta tensiones de alimentación de 1.8 V a 3.8 V. En su interior, cuenta con un controlador, que se encarga de brindar datos acerca de RSSI/LQI, asegurar el canal de transmisión libre automáticamente, hacer comprobación de redundancia cíclica (CRC), filtrar frames de acuerdo al campo de dirección, enviar automáticamente mensajes de acuso de recibo (Acknowledgment), y opcionalmente, encriptar mensajes de acuerdo al estándar. Además tiene para comunicarse, un puerto SPI, 6 pines de entrada/salida configurables y generador de interrupciones.

El transceiver tiene un gran nivel de configuración, pudiendo setear el nivel de consumo, elegir las salidas/entradas de los pines ante diversos eventos (recepción de frame, error de comunicación, buffer de recepción rebalsado, entre otros). Tiene además un buffer FIFO de recepción y otro de transmisión, de 128 bytes.

El CC2591 se trata de un front-end compatible con el trasnceiver utilizado. La principiar característica del mismo es que centra su diseño para aplicaciones de bajo consumo, y de bajo

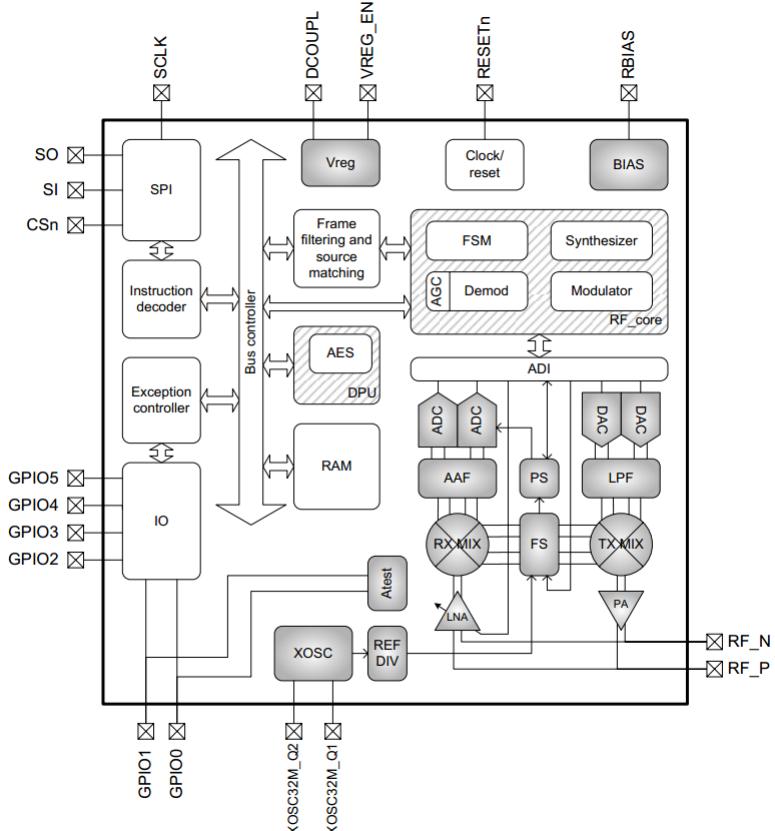


Figura 8: Diagrama en bloques del CC2520.

costo. Requiere la utilización de muy pocos componentes externos, y logra extender la potencia de transmisión a 22 dBm, y mejora la sensibilidad de recepción en 6 dB.

Trabajando en conjunto logran[7]:

- Sensibilidad de recepción de -97.7 dBm, con 1 % PER.
- Potencia de salida de hasta 17 dBm, con un consumo de 136mA.

Alimentación

Dado que el nodo debe funcionar en forma autónoma, se optó por implementar un circuito de alimentación a batería con su correspondiente cargador. El circuito integrado elegido para esta función es el BQ24080 de la firma Texas Instruments, que además posee una interfaz digital que permite al CPU conocer el estado de carga así como poner en modo de ahorro de energía al cargador.

La batería a utilizar será de Li-ion, 3.7V, 900mAh (valores típicos para una batería de celular). Al mismo tiempo, se utilizan reguladores lineales con muy baja caída de tensión (Low Dropout, o LDO, con caídas del orden de 120mV), para lograr un aprovechamiento óptimo de la batería.

El criterio para la elección de estos reguladores se basa en su tamaño reducido, su mínima cantidad de componentes externos necesarios y su alta inmunidad al ruido. Utilizar una fuente conmutada en estos casos no es recomendable debido principalmente a la complejidad de los filtros requeridos para eliminar el ruido que podría interferir en la etapa de RF.

Obsérvese la utilización de un regulador independiente para el módulo GPS. Esto permite al microcontrolador desactivar este módulo, cuyo consumo ronda los 40mA en forma continua. El circuito de carga se alimenta a través de un conector USB microB, que es el adoptado por las compañías de telefonía celular, con lo cual adquirir una fuente de alimentación de 5V resulta accesible ya que cualquier cargador de teléfono celular cumple con esta condición. El mismo puerto USB que se utiliza para la carga de la batería, es el que realiza la comunicación con entre el host y el microcontrolador LPC1343.

Sensores y HMI

La placa cuenta además con tres LEDs conectados a pines GPIO del micro, y con tres pulsadores, de manera que se tiene una sencilla interfaz para visualizar estados e interactuar con el controlador.

Además, los nodos cuentan con tiras de pines del tipo hembra, conectados a varios pines del microcontrolador: GPIO, ADC, DAC, UART, ya que fue pensado para conectarles módulos expandibles con diversos tipos de sensores.

El GPS, previsto en el diseño, no está aún implementado. Dado el costo y el consumo del mismo, fue descartado para la aplicación presente. Su implementación está planeada para sólo algunos nodos, los cuales podrían cumplir la función de sincronizar una red de sensores.

7.2. Especificaciones Técnicas

En resumen, las especificaciones técnicas de cada nodo son:

Microcontrolador LPC1343

- Procesador CortexM3 72MHz.
- 32 kB de memoria Flash.
- 8 kB de memoria RAM.
- Consumo estático 72MHz: 17 mA.
- Consumo estático 12MHz: 4 mA.
- Consumo estático en modo *deep powerdown*: 220 nA.

Transceiver CC2520+CC2591

- Consumo Tx 22 dBm: 145 mA.
- Consumo Rx: 22 mA.
- Consumo en estado *idle*: 1.3 μ A.
- Sensibilidad: 98 dBm + (6 dBm) = 104 dBm.
- Link Budget: 104 dBm + 22 dBm = 126 dB.

Generales

- Autonomía, con una batería de 900 mAh, transmisión el 20 % del tiempo, GPS encendido el 50 % del tiempo, CPU con clock máximo: 10.63 horas.
- Alimentación externa: 4.5 a 6.5 V.

7.3. Listado de Componentes y Costos

La estimación de costos se realizó en función de los precios FOB de la firma DigiKey para los componentes de montaje superficial, salvo el módulo GPS que fue presupuestado por la firma Cika Electrónica y el costo de fabricación de los PCB que fue presupuestado por la firma Eleprint. Debido a que el GPS no es necesario en todas las aplicaciones, se plantean diferentes opciones en la Tabla 2 a continuación:

	Con GPS	Sin GPS
10 unidades	U\$S 91.09	U\$S 69.83
100 unidades	U\$S 68.76	U\$S 38.03

Cuadro 2: Comparación de costos de nodos.

7.4. Prototipo

El proyecto comenzó con la implementación de hardware de los nodos de Pablo Ridolfi. En el Laboratorio de Sistemas Embebidos había varias placas en proceso de armado. Se procedió entonces a la soldadura de los componentes, y a la verificación y puesta en marcha de las placas ya disponibles, con lo que se logró 4 placas operativas. Sin embargo, por cuestiones ajenas a nosotros, sólo una de ellas quedó a nuestra disponibilidad.

Para solucionar esta complicación, y poder desarrollar el proyecto ante la ausencia del hardware propuesto, se nos brindó la posibilidad de utilizar otro prototipo. Se trata de otro módulo, diseñado y provisto también por Pablo Ridolfi, y que consta de un CC2520, montado con todos sus componentes necesarios, y una antena sobre un PCB. Esta pequeña placa cuenta con 16 pines, que están unidos al SPI, a los GPIO y a la alimentación del transceiver.

Aprovechando los kits de desarrollo propuestos en el curso, se conectaron los módulos de comunicación con un stick LPCXpresso 1769; y se implementó una pequeña fuente de alimentación comutada, para alimentar a los nuevos nodos por medio de baterías de 9 V.

A pesar de utilizar distinto microcontrolador, el desarrollo realizado en software es completamente portable a los nodos con el LPC1343. Sólo una variación en performance es esperada, sin perder ninguna funcionalidad, ya que no se utilizan funciones adicionales propias del LPC1769.

8. Software

8.1. Funcionamiento de la red

Como ya se explico previamente, la red cuenta con 3 nodos fijos, 1 nodo movil y 1 encuestador.

Cuando el usuario desea conocer la posicion del nodo movil, le envia una peticion al nodo fijo identificado como Nodo 1. Este la recibe, y emite un mensaje a todos los nodos con sus coordenadas X e Y. Para evitar que todos los nodos transmitan a la vez y se produzca interferencia entre ellos, el resto de los nodos fijos evaluan si el remitente es el nodo inmediatamente anterior a ellos, si no lo es, vuelven al estado SLEEP. De esta manera se asegura de que solo el nodo 2 efectue una accion distinta.

Al recibir el mensaje, el nodo 2 realiza la misma accion que el nodo 1 anteriormente, es decir, envia su posicion a todos los nodos en la red. Esto se repite a lo largo de todos los nodos fijos en la red. Una vez que los 3 hayan emitido su posicion, el nodo movil esta en condiciones de realizar la trilateracion para calcular su posicion y reenviar el dato al nodo encuestador.

8.2. Nodos Fijos

Dado que los nodos no contaban con el modulo GPS, fue necesario ubicarlos en posiciones conocidas y cargarle las mismas manualmente a la hora de compilar el codigo. Por simplicidad se los ubico a 15 mts de distancia formando entre ellos un triangulo rectangulo como se muestra en la siguiente figura:

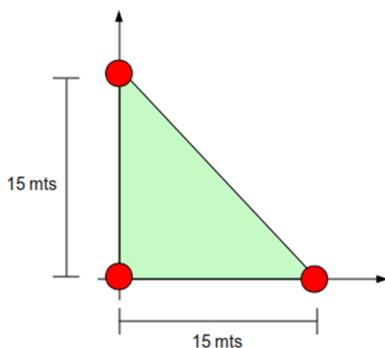


Figura 9: Esquema de la red de nodos fijos.

Cada nodo cuenta con una direccion unica e irrepetible a lo largo de toda la red. Esta direccion es la que establece el orden en que se envian/reciben los mensajes para que circulen por la red.

Todos los nodos estan cargados con el mismo software, los unicos parametro que cambian son la direccion y una constante que indica si son fijos o moviles.

El firmware de cada nodo fijo se puede plasmar en el diagrama de la figura 10.

8.3. Nodo Móvil

El nodo móvil es el encargado de recibir la posición de cada nodo fijo, una vez que obtuvo datos de 3 nodos fijos diferentes, realiza la trilateración y obtiene su posición. Luego, envía la posición calculada al nodo encuestador. Ver diagrama en figura 11.

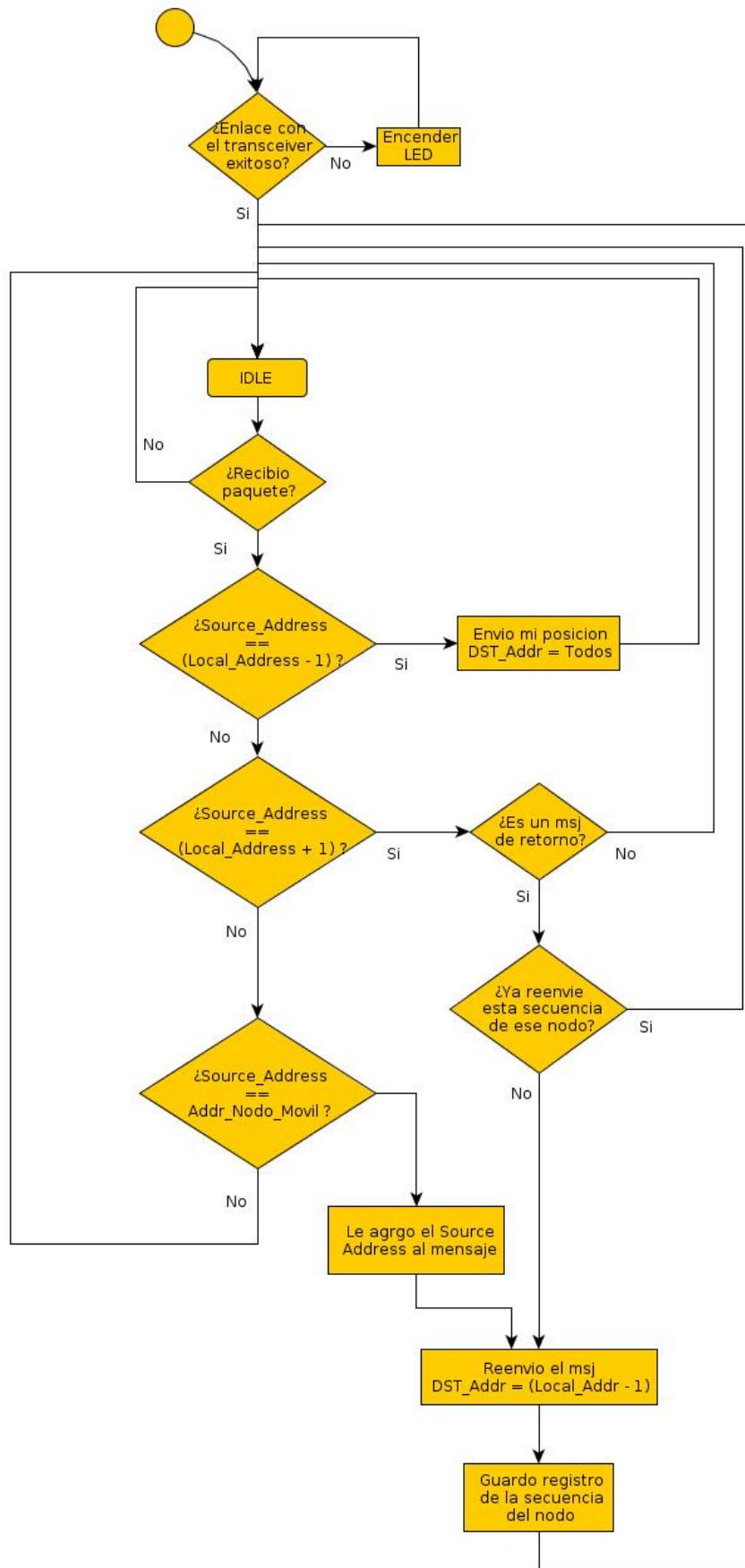


Figura 10: Diagrama de flujo de un nodo fijo.

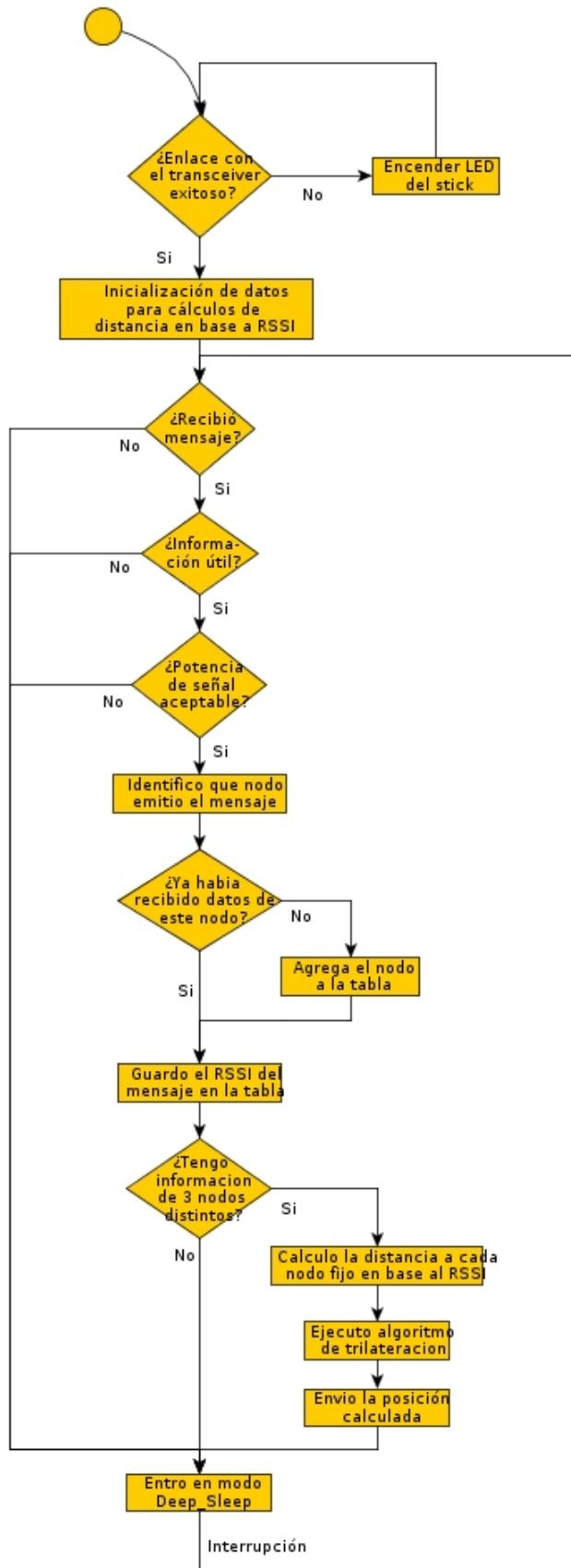


Figura 11: Diagrama de flujo de un nodo móvil.²⁵

9. Resultados

A continuación se muestran los resultados obtenidos tanto con simulaciones hechas en MATLAB y en el microcontrolador, como con mediciones experimentales.

9.1. Simulación de método de estimación de distancia

Para simular el correcto funcionamiento de los algoritmos de estimación de distancia en base a RSSI se toman 20 muestras de RSSI para una distancia dada (se utilizan las mismas muestras que las usadas en la calibración), y se ingresan en los dos algoritmos posibles. Los resultados, en la figura 12.

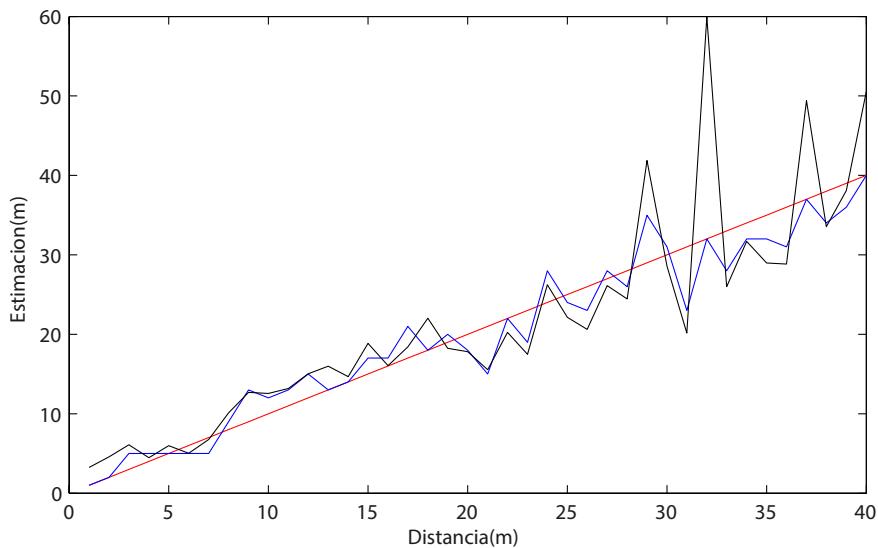


Figura 12: Distancia vs. Estimaciones. En rojo la distancia real, en azul la estimación con el método estadístico y en negro con el ajuste.

Los resultados obtenidos en general fueron mejores con el método estadístico que con el otro método. El error promedio para cada método: 2.15 m para el primero, y 4.29 para el segundo. Además, el error máximo del primer método fue de 8.00 m y el del ajuste de 27.94 m. Posibles explicaciones de estos picos (a pesar de ser casos aislados) son físicas: mal desempeño de hardware, o alguna alteración circunstancial de la medición.

9.2. Simulación de método de trilateración

Para validar el método de trilateración, se simuló el algoritmo de trilateración en MATLAB, y en el microcontrolador. Se eligieron posiciones arbitrarias, considerando tres nodos fijos situados en $(0, 0)$, $(15, 0)$ y $(0, 15)$, y se comparan los errores obtenidos.

Para poder simular los errores cometidos por la trilateración con mediciones inexactas se hace lo siguiente. A las distancias exactas se les suma un ruido blanco de media cero, y varianza 2 (en mediciones experimentales se obtuvo una varianza de 2 m.). Se simulan muchos puntos, y

Posición	MATLAB		LPC	
	Posición	Error	Posición	Error
(0, 0)	(0,000, 0,000)	(0,000, 0,000)	(0,000, 0,000)	(0,000, 0,000)
(15, 0)	(15,000, 0,000)	(0,000, 0,000)	(15,000, 0,000)	(0,000, 0,000)
(0, 15)	(0,000, 15,000)	(0,000, 0,000)	(0,000, 15,000)	(0,000, 0,000)
(1, 1)	(1,000, 1,000)	(0,000, 0,000)	1,000, 1,000)	(0,000, 0,000)
(5, 5)	(5,000, 5,000)	(0,000, 0,000)	(5,000, 5,000)	(0,000, 0,000)
(20, 20)	(20,000, 20,000)	(0,000, 0,000)	(20,000, 20,000)	(0,000, 0,000)
(-10, -10)	(-10,000, -10,000)	(0,000, 0,000)	(-10,000, -10,000)	(0,000, 0,000)
(300, 300)	(300,000, 300,000)	(0,000, 0,000)	(300,000, 300,000)	(0,000, 0,000)

Cuadro 3: Resultados de trilateración para distancias exactas según distintos métodos.

se grafican los puntos estimados. Se especifica también el radio de un círculo tal que el percentil 80 % queda dentro del mismo (ver figura 13). Puede observarse, que a medida que se aleja del perímetro, a pesar de contar con la misma varianza de ruido en las mediciones, el radio es mucho mayor. Los resultados obtenidos por MATLAB (con doble precisión) son casi idénticos a los obtenidos con el LPC.

9.3. Validación Experimental

A pesar de los buenos resultados preliminares obtenidos en las simulaciones, este no fue el caso durante las mediciones experimentales. Dado que el algoritmo de trilateración se trata solamente de operaciones matemáticas, que descansan sobre una medición de distancia, fue sobre esta última etapa que se hizo mayor hincapié. Para ello, las pruebas iniciales se realizaron únicamente con dos nodos, un emisor y un receptor.

Los resultados bajos ciertas condiciones resultaron aceptables : se obtenían errores de posición que promediaban los 2 m, con picos de hasta 5 m o más. Sin embargo, a pesar de controlar el ambiente (las calibraciones están hechas, tal como se explicó más arriba, para un ambiente controlado, donde no haya múltiples caminos de señal), se encontró que en ciertas zonas de alejamiento entre nodo emisor y receptor las estimaciones eran muy erradas.

Debido a las variaciones en los posicionamientos de las antenas entre medición y medición, se sospechó de la direccionalidad de la antena impresa como causante de estos errores. Para corroborar esta hipótesis, se midió el diagrama de radiación de la antena, a 3 m de distancia, en posición horizontal. Los resultados se ven en la figura 14. El diagrama de radiación no resultó ser óptimo, teniendo un mínimo de -70.71 dBm, y un máximo de -56.72 dBm, una diferencia de casi 15 dBm. Semejante diferencia para una misma distancia es incorregible con ningún método por software, ya que es imposible discernir la pérdida de potencia por atenuación de la pérdida por posicionamiento de la antena.

Las estimaciones con los algoritmos, en función del ángulo se presentan en la figura 15. Los resultados son muy variados. Para el ángulo 0 (ángulo entre antenas para el que fueron calibrados los métodos de estimación), las estimaciones dan 2 m con el método estadístico, y 3.55 m con el ajuste de curva. En general el primer método obtiene mejores resultados, con

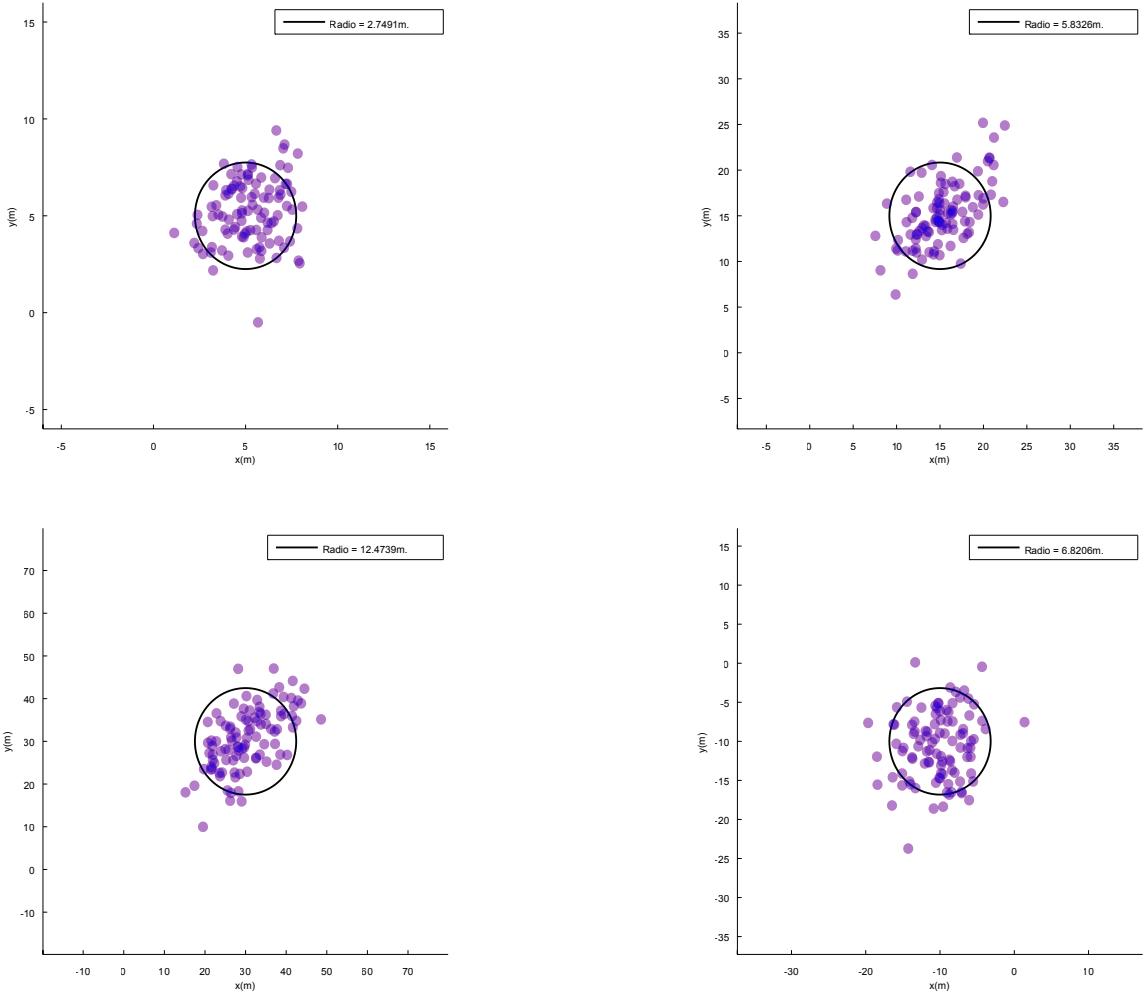


Figura 13: Trilateración con mediciones imprecisas: MATLAB (rojo), LPC (azul). Las posiciones originales (de izq. a der., empezando por arriba): $(5, 5)$, $(15, 15)$, $(30, 30)$ y $(-10, -10)$. Los radios obtenidos: $2,74m$, $5,83m$, $12,47m$ y $6,82m$.

una media de error de 5.84 m, contra un promedio de 7.27 m del otro método. Sin embargo, y debido a las variaciones drásticas del RSSI en función del ángulo, las estimaciones varían aún más: se estima 12.80 m en el peor caso (casi 10 m de diferencia).

Con este resultado obtenido se descartó entonces la posibilidad de ver el proyecto en funcionamiento entero. Esto debido a la poca uniformidad del patrón de radiación de las antenas, y dado que la red prevista tiene tres nodos fijos, y uno móvil, es imposible incluso orientar a todas las antenas y calibrar a los algoritmos para dichas orientaciones. El problema que termina limitando la implementación del proyecto es a nivel hardware, bastaría instalar otro tipo de antena más isotropica en los nodos.

Sin embargo, y para demostrar que aunque sea, en alguna de las tres direcciones a los nodos fijos se puede estimar bien, se instaló la red, y se hicieron mediciones de campo. Para la mayoría de las mediciones, hubo siempre una de las distancias con menos de 2 m de error. Las distancias

a otros nodos, por otra parte, tuvieron gran nivel de desviación. La trilateración subsiguiente no tiene mucho sentido, ya que los set de distancias son absurdos, pero es presentada a modo de ilustración (ver tabla 4). La posición de nodos fijos es la misma usanda anteriormente.

Posición		Distancia exacta a nodos			Promedio de Distancias a nodos estimadas			Trilateración	
x	y	d_1	d_2	d_3	d_1	d_2	d_3	x	y
1,500	1,500	2,121	13,583	13,583	2,408	5,224	25,961	6,783	-14,773
1,500	4,500	4,743	14,230	10,606	6,253	5,813	32,000	7,676	-25,330
1,500	7,500	7,648	15,443	7,648	34,007	6,964	32,988	44,433	9,775
1,500	10,500	10,606	17,102	4,743	38,522	29,023	32,000	28,887	22,833
4,500	10,500	11,423	14,849	6,364	28,715	14,267	7,378	28,200	33,170
7,500	10,500	12,903	12,903	8,746	11,473	9,825	5,000	8,669	11,054
10,500	10,500	14,849	11,423	11,423	9,376	23,662	5,744	-8,233	9,330
13,500	10,500	17,102	10,606	14,230	22,338	32,000	5,000	-10,000	23,300
4,500	1,500	4,743	10,606	14,230	6,000	32,000	5,000	-25,433	7,866
7,500	1,500	7,648	7,648	15,443	9,260	32,000	5,605	-23,775	9,311
10,500	1,500	10,606	4,743	17,102	24,572	32,000	2,000	-6,506	27,493

Cuadro 4: Resultados obtenidos durante mediciones. El método de estimación de distancia utilizado fue el estadístico.

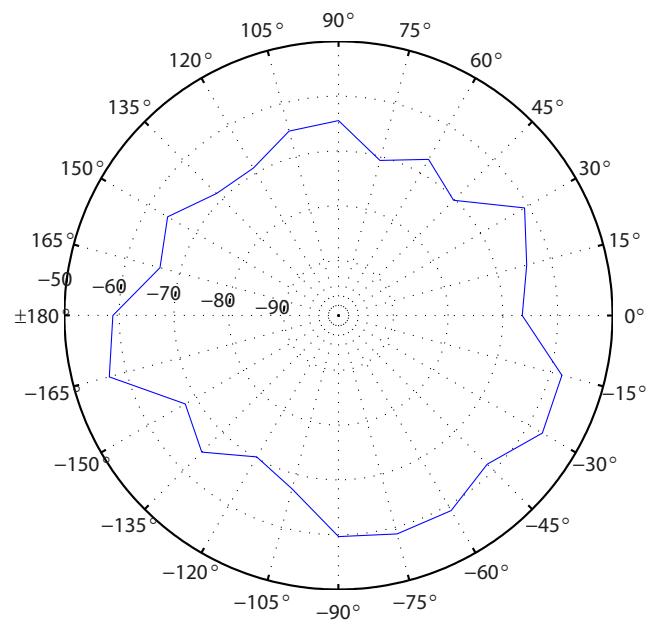


Figura 14: Diagrama de radiación de la antena a 3 m, en posición vertical.

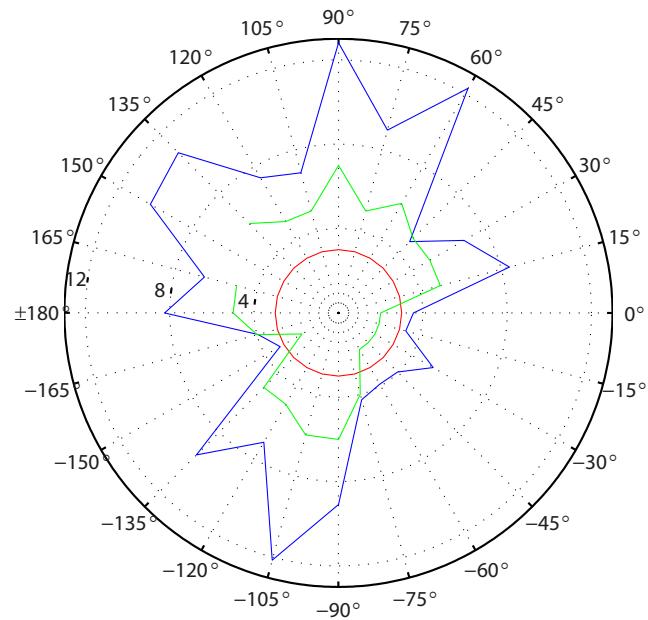


Figura 15: Estimaciones de distancia en función del ángulo. Rojo: 3 m, Azul: Ajuste, Verde: Estadístico.

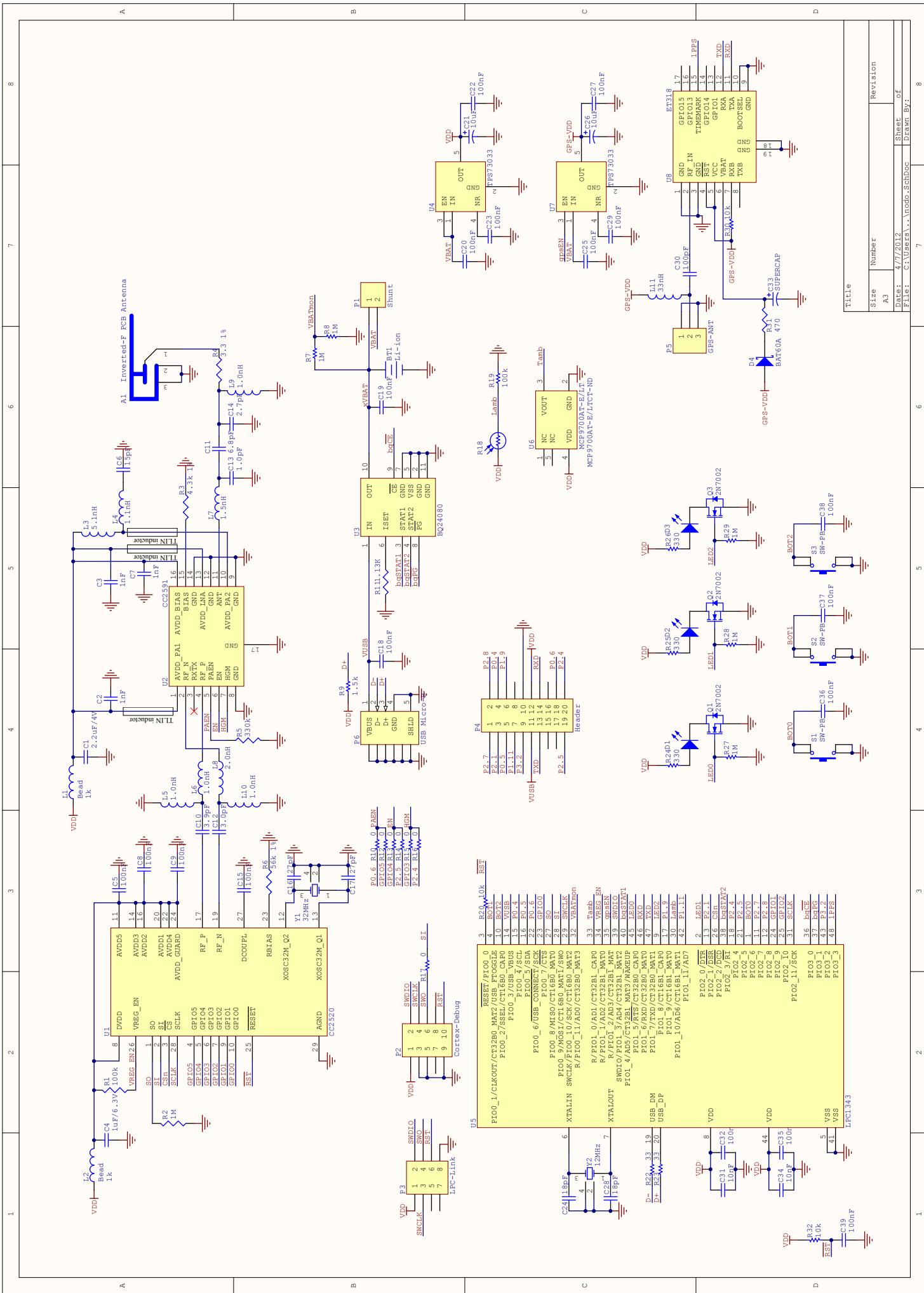
10. Conclusiones

A lo largo del desarrollo del trabajo, pudimos aplicar gran parte de los conocimientos adquiridos durante el curso. Con los cuales pudimos desarrollar un prototipo para la localización de animales a través de RSSI. A pesar de que no tuvimos los mejores resultados el proceso fue muy productivo y pudimos identificar fallas en el hardware, las cuales nos van a permitir mejorar y continuar con el proyecto.

Para realizar tracking de animales o cualquier otro objeto, es necesario poder obtener su ubicación de forma precisa (o con un rango de error aceptable) para realizar esto, en nuestro caso, debíamos contar con los valores de RSSI con un rango de error bajo. Sin embargo estos eran muy variables, por lo que las distancias calculadas presentaban mucha distorsión y por ende la posición calculada con la trilateración era errónea. El diseño e implementación de la antena resultó ser fundamental en el desarrollo del hardware, en un principio no lo tuvimos en cuenta y nos centramos en la programación de los algoritmos, lo cual nos llevó a no obtener los resultados esperados. Realizamos gran cantidad de pruebas y pudimos concluir que para tener el hardware en condiciones óptimas para que nuestros algoritmos no presenten errores debemos cambiar o re adaptar la antena del transceiver.

Gracias a que se presentaron estas fallas durante el desarrollo del proyecto, investigamos sobre opciones alternativas para el tracking de ganado. Como el uso de un sistema simil GPS, en el cual una antena de alto poder de recepción de señal recibe las señales de cada tag (nodo móvil) y mediante circuitería y algoritmos adecuados de sincronización se logra medir la distancia a la antena, considerando la velocidad de propagación de la señal. Por lo que concluimos que el tracking por RSSI no es la única opción a considerar.

El tracking de animales es un campo con muy poca explotación hoy en día en nuestro país, por esto vamos a resolver los problemas de hardware y seguir mejorando nuestros algoritmos con el fin de poder tener un primer prototipo funcionando y poder lograr expandir el uso de esta tecnología en la Argentina.



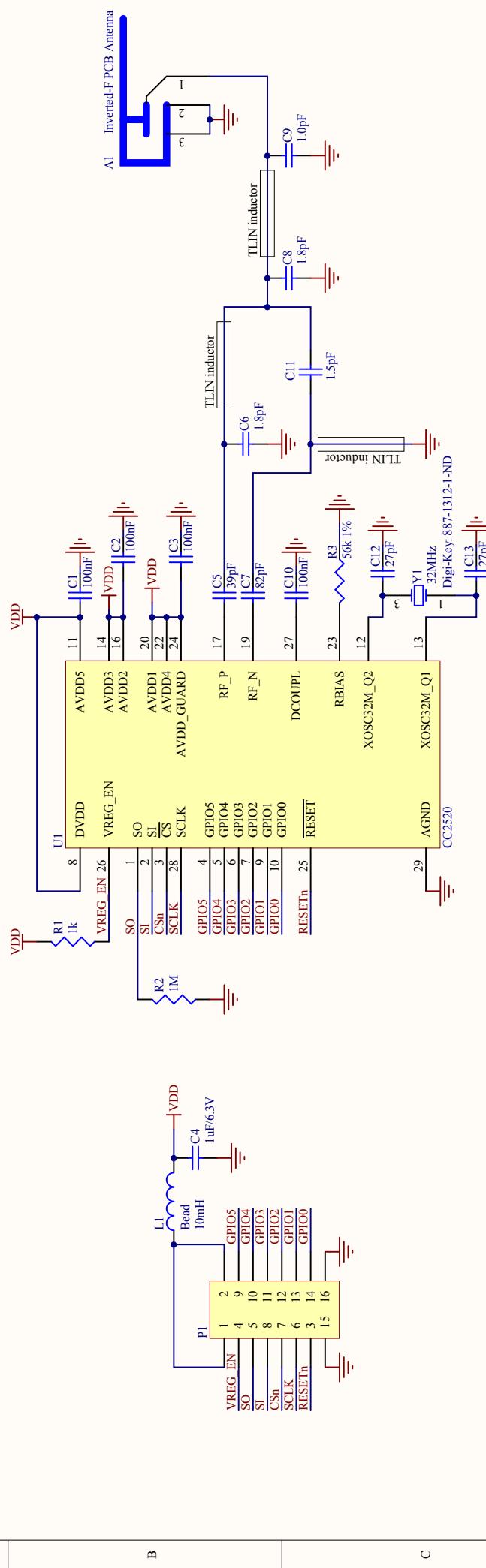
4

3

2

1

A



4

3

2

1

D

B. Código

B.1. main.c

En él se realizan las configuraciones de hardware y de interrupciones necesarias, y se llama a la rutina correspondiente, según el número de dirección.

```
1 #include "includes.h"
2
3
4 int ledcont , pausecont;
5
6 void SysTick_Handler( void )
7 {
8     if(ledcont)
9     {
10         ledcont--;
11         if(ledcont==0)
12             GPIO_ClearValue(LED_PORT, LED_BIT);
13     }
14     if(pausecont) pausecont--;
15 }
16
17 void pausems( int t )
18 {
19     pausecont = t;
20     while( pausecont );
21 }
22
23 void ledFlash( int t )
24 {
25     GPIO_SetValue(LED_PORT, LED_BIT);
26     ledcont = t;
27 }
28
29 // MAIN PARA NODO MOVIL:
30
31 #ifdef MOVIL
32
33 int main ( void )
34 {
35     int8_t init = -1;
36     // Configuro el tick del sistema
37     SysTick_Config(SystemCoreClock/1000);
38
39     // Seteo el LED
40     GPIO_SetDir(LED_PORT, LED_BIT, 1);
41     GPIO_ClearValue(LED_PORT, LED_BIT);
42
43     // Inicializo direcciones del tranceiver
44     GPIO_SetValue(LED_PORT, LED_BIT);
45     while( init<0)
46     {
```

```

47     init = ccInit(PANID, ADDR_LOCAL, ADDR_LOCAL, 0);
48 }
49 GPIO_ClearValue(LED_PORT, LED_BIT);
50
51 // Calculo la tabla para el ajuste probabilistico
52 calc_tabla(tabla);
53
54 // Inicio la interrupcion
55 NVIC_EnableIRQ(EINT3_IRQn);
56
57 // Inicio la tarea del nodo movil
58 rutina_movil();
59
60 // En teoria nunca debe llegar a esta instancia
61 for(;;);
62}
63#endif
64
65#ifndef FIJO
66// Main para nodo FIJO
67int main (void)
68{
69    int8_t init=-1;
70
71    // Configuro el tick del sistema
72    SysTick_Config(SystemCoreClock/1000);
73
74    // Seteo el LED
75    GPIO_SetDir(LED_PORT, LED_BIT, 1);
76    GPIO_ClearValue(LED_PORT, LED_BIT);
77
78    GPIO_SetValue(LED_PORT, LED_BIT);
79    // Inicializo direcciones del tranceiver
80    while(init<0)
81    {
82        init = ccInit(PANID, ADDR_LOCAL, ADDR_LOCAL, 0);
83    }
84    GPIO_ClearValue(LED_PORT, LED_BIT);
85
86    // Inicio la interrupcion externa
87    NVIC_EnableIRQ(EINT3_IRQn);
88
89#if (ADDR_LOCAL == 2)
90    // Si es el primer nodo, seteo el timer que sincroniza red
91    TIM_TIMERCFG_Type TIM_ConfigStruct;
92    TIM_MATCHCFG_Type TIM_MatchConfigStruct;
93
94    // Seteo escala de timer en 1 milosegundo
95    TIM_ConfigStruct.PrescaleOption = TIM_PRESCALE_USVAL;
96    TIM_ConfigStruct.PrescaleValue = 1000;
97
98    // Uso Timer 0
99    TIM_MatchConfigStruct.MatchChannel = 0;

```

```

100 // Habilito la interrupcion
101 TIM_MatchConfigStruct.IntOnMatch = TRUE;
102 // Habilito el Reset
103 TIM_MatchConfigStruct.ResetOnMatch = TRUE;
104 TIM_MatchConfigStruct.StopOnMatch = FALSE;
105 TIM_MatchConfigStruct.ExtMatchOutputType = TIM_EXTMATCHNOTHING;
106 // Seteo valor final 10 segundos
107 TIM_MatchConfigStruct.MatchValue = 3000;
108
109 TIM_Init(LPC_TIM0, TIM_TIMER_MODE, &TIM_ConfigStruct);
110 TIM_ConfigMatch(LPC_TIM0,&TIM_MatchConfigStruct);
111
112 // Inicio la interrupcion
113 NVIC_EnableIRQ(TIMER0_IRQn);
114
115 #endif
116 // Inicio la tarea del nodo fijo
117 rutina_fijo();
118
119 // En teoria nunca debe llegar a esta instancia
120 for(;;);
121 }
122 #endif
123
124
125 void EINT3_IRQHandler( void )
126 {
127 // Limpio la interrupcion
128 GPIO_ClearInt(0,(1<<1));
129
130 // Limpio el flag en cc2520
131 ccCmd(BCLR, EXCFLAG1, 0, 0);
132
133 // Seteo flag de mensaje nuevo
134 mje_nuevo = 1;
135 }
136
137 void TIMER0_IRQHandler ( void )
138 {
139 // Verifico que haya sido la interrupcion
140 if (TIM_GetIntStatus(LPC_TIM0,TIM_MR0_INT) == SET)
141 {
142 // "Creo" un mensaje recibido
143 d_rx.fcf = MAC_FRAME_TYPE_DATA;
144 d_rx.dst.shortAddr.panid = PANID;
145 d_rx.dst.shortAddr.addr = ADDR_LOCAL;
146 d_rx.payload[0] = (uint8_t)0;
147 d_rx.pl_length = 1;
148 d_rx.src.shortAddr.panid = PANID;
149 d_rx.src.shortAddr.addr = ADDR_LOCAL-1;
150 mje_nuevo = 1;
151 mje_simulado = 1;
152

```

```

153    }
154
155    // Limpio el flag de interrupcion
156    TIM_ClearIntPending(LPC_TIM0,TIM_MR0_INT);
157}

```

B.2. fijo.c

En él se implementa la rutina de los nodos fijos.

```

1  /*
2  * fijo.c
3  *
4  *   Created on: 14/11/2012
5  *       Author: joaquin
6  */
7 #include "fijo.h"
8
9
10 void rutina_fijo (void)
11 {
12     uint32_t aux;
13     uint16_t addr_aux;
14     uint8_t reg_nodos[10];
15     uint8_t i;
16
17     // Inicio Timer
18     TIM_Cmd(LPC_TIM0,ENABLE);
19
20     // Inicializo el registro
21     for(i=0;i<10;i++)
22     {
23         reg_nodos[i] = 0;
24     }
25
26     // Configuro mensajes que mando
27
28     d_tx.dst.shortAddr.panid = d_tx.src.shortAddr.panid = PANID;
29     d_tx.src.shortAddr.addr = ADDR_LOCAL;
30
31     d_tx.fcfc = ccWrapperFCF(MAC_FRAME_TYPE_DATA, 0, 0, 1, 0,
32                               MAC_ADDR_MODE_SHORT, 0, MAC_ADDR_MODE_SHORT);
33
34     // Comienzo rutina infinita
35
36     while(1)
37     {
38         if (mje_nuevo)
39         {
40             mje_nuevo = 0;
41
42             // Leo el tranceiver

```

```

43| if (!mje_simulado) d_rx = ccFrameRx();
44|
45| mje_simulado = 0;
46|
47| // Veo si es mensaje util
48| if (macFCFGetFrameType(d_rx.fcf) == MAC_FRAME_TYPE_DATA)
49|
50|
51| // Veo si es mensaje del nodo anterior
52| if (d_rx.src.shortAddr.addr == ADDR_LOCAL -1 )
53|
54|     // Si es mensaje , envio posicion del nodo a todos
55|
56|     d_tx.dst.shortAddr.addr = 0xFFFF;
57|     d_tx.payload[0] = POS_X; // Mando posicion X
58|     d_tx.payload[1] = POS_Y; // Mando posicion Y
59|     d_tx.pl_length = 2;
60|
61|     // Espero un pequeno delay antes de enviar
62|     for (aux=0;aux<10000;aux++);
63|
64|     // Envio posicion
65|     ccFrameTx(d_tx);
66|
67|     // Prendo LED de envio
68|     GPIO_SetValue(LED_PORT,LED_BIT);
69|     for (aux=0;aux<10000;aux++);
70|     GPIO_ClearValue(LED_PORT,LED_BIT);
71}
72| else if (d_rx.src.shortAddr.addr == ADDR_LOCAL +1 )
73|
74|     // Verifico si es mensaje de retorno
75|     if (d_rx.pl_length > 2)
76|
77|         // Veo si ya reenvie esa secuencia
78|         memcpy(&addr_aux,d_rx.payload,2);
79|         if (d_rx.payload[2] != reg_nodos[addr_aux - 10])
80|
81|             // Si no la reenvie , la mando al anterior
82|
83|             // Copio el payload
84|             memcpy(d_tx.payload,d_rx.payload,d_rx.pl_length);
85|
86|             // Reenvio al anterior
87|             d_tx.dst.shortAddr.addr = ADDR_LOCAL -1;
88|             ccFrameTx(d_tx);
89|
90|             // Prendo LED de envio
91|             GPIO_SetValue(LED_PORT,LED_BIT);
92|             for (aux=0;aux<10000;aux++);
93|             GPIO_ClearValue(LED_PORT,LED_BIT);
94|
95|             // Registro el envio

```

```

96         reg_nodos [addr_aux - 10] = d_rx.payload [2];
97     }
98 }
99
100 else if (d_rx.src.shortAddr.addr >=10 )
101 {
102     // Si es mensaje de nodo movil , adapto el mensaje
103
104     // Copio direccion del nodo a 2 primeros bytes
105     memcpy(d_tx.payload , &(d_rx.src.shortAddr.addr) , 2);
106     // Copio el numero de secuencia del payload
107     memcpy((d_tx.payload)+2, d_rx.payload , 1);
108     // Luego copio el payload con la posicion del movil
109     memcpy((d_tx.payload)+3, (d_rx.payload)+1, sizeof(float32_t)*2);
110
111     d_tx.p1_length = sizeof(float32_t)*2 + 3;
112
113     // Reenvio
114     d_tx.dst.shortAddr.addr = ADDR_LOCAL -1;
115     ccFrameTx(d_tx);
116
117     // Prendo LED de envio
118     GPIO_SetValue(LED_PORT,LED_BIT);
119     for (aux=0;aux<10000;aux++);
120     GPIO_ClearValue(LED_PORT,LED_BIT);
121
122     // Registro el envio
123     reg_nodos [d_rx.src.shortAddr.addr - 10] = d_rx.payload [0];
124 }
125
126 }
127
128 }
129
130
131 // Deshabilita el systick para que no despierte al sleep
132 SYSTICK_Cmd(DISABLE);
133
134 // Entra en modo Deep-Sleep
135 CLKPWR_Sleep();
136
137 // Reanuda el programa
138
139 // Habilita nuevamente el systick
140 SYSTICK_Cmd(ENABLE);
141
142
143
144 }
145
146 }
```

B.3. movil.c

En él se implementa la rutina del nódulo móvil, con el algoritmo de trilateración incluido.

```
1 #include "movil.h"
2
3 void rutina_movil (void)
4 {
5     // Verificaciones y declaraciones
6     frameData data;          // Data que leo
7     nodo_fijo_t nodo_vec[20]; // Defino nodos fijos
8     uint8_t i,cant_nodos=0;   // Cantidad de nodos fijos
9     float32_t posicion[2];    // Posicion del nodo móvil
10    uint8_t j,lista,k;
11    uint32_t aux;
12
13    while(1)
14    {
15        if (mje_nuevo)
16        {
17            mje_nuevo = 0;
18            data = ccFrameRx(); // Leo data del receptor
19
20            // Prendo led por breve tiempo
21            GPIO_SetValue(LED_PORT,LED_BIT);
22            for (aux=0;aux<10000;aux++);
23            GPIO_ClearValue(LED_PORT,LED_BIT);
24
25
26            // Verifico que sean datos útiles
27            if(macFCFGetFrameType(data.fcf)==MAC_FRAME_TYPE_DATA)
28            {
29                // Busco nodo que envio mensaje
30
31                i=0;
32                lista=0;
33
34                // Si el RSSI es malo lo descarto
35                if(data.rssi != 0)
36                {
37                    // printf("%d\n",data.rssi);
38
39                    while(i<=cant_nodos && !lista)
40                    {
41                        // Si se encuentra, se guarda el RSSI
42                        if (data.src.shortAddr.addr == nodo_vec[i].numero)
43                        {
44                            k = i; // Guardo el numero del nodo
45                            nodo_vec[i].rssi[nodo_vec[i].last_rssi] = data.rssi;
46                            nodo_vec[i].last_rssi++;
47                            if (nodo_vec[i].last_rssi == RSSI_MAX) nodo_vec[i].last_rssi=0;
48                            lista=1;
49                        }
50                        i++;
51                    }
52                }
53            }
54        }
55    }
56}
```

```

51 }
52
53 // Si no encontro el nodo, defino uno nuevo
54 if (!listo)
55 {
56     nodo_vec[cant_nodos].numero = (data.src).shortAddr.addr;      // Guardo numero
57     nodo_vec[cant_nodos].posicion[0] = (float32_t) data.payload[0]; // Guardo posicion X
58     nodo_vec[cant_nodos].posicion[1] = (float32_t) data.payload[1]; // Guardo posicion Y
59     nodo_vec[cant_nodos].last_rssi = 0;                            // Inicio last_rssi
60     nodo_vec[cant_nodos].rss[i[nodo_vec[cant_nodos].last_rssi]] = data.rssi; // Guardo rssi
61     nodo_vec[cant_nodos].last_rssi++;
62     for(j=0;j<RSSI_MAX;j++) nodo_vec[cant_nodos].rss[j] = 0; // Inicializo rssi en 0
63     k = cant_nodos;                                              // Guardo el numero de nodo
64     cant_nodos++;                                                 // Aumento cantidad de nodos
65 }
66
67 // Calculo las distancias a cada nodo (si recibo de 3)
68
69 if (k==2)
70 {
71     i=0;
72     for(i=0;i<cant_nodos;i++)
73     {
74         nodo_vec[i].dist = (float32_t) RSSI_to_dist_1(nodo_vec[i].rss , RSSI_MAX); // Calculo distancia
75         printf("Distancia a nodo %d : %.2fm. Ultimo RSSI:%d\n",i+1, nodo_vec[i].dist , nodo_vec[i].rss[nodo_vec[i].last_rssi -1]);
76     }
77     nodo_vec[k].prom_rssi = (float32_t) promediar(nodo_vec[k].rss , RSSI_MAX); // Calculo promedio rssi
78 }
79
80 // Pregunto si hago trilateracion
81
82 // Caso simple : con 3 nodos fijos UNICAMENTE, hago trilateracion
83
84 if (cant_nodos >= 3 && k==2)
85 {
86     trilateracion(nodo_vec,cant_nodos,posicion);
87     printf("Posicion (%.2f,%2f)\n",posicion[0],posicion[1]);
88 }
89
90 // Mando la posicion
91 enviar_posicion((uint8_t*) posicion , sizeof(float32_t)*2,ADDRENCUEST);
92 }
93 }
94 }
```

```

95
96 // Deshabilita el systick para que no despierte al sleep
97 SYSTICK_Cmd(DISABLE) ;
98
99 // Entra en modo Deep-Sleep
100 CLKPWR_DeepSleep() ;
101
102 // Aqui se reanuda el programa
103
104 // Habilita nuevamente el systick
105 SYSTICK_Cmd(ENABLE) ;
106 }
107
108 }
109
110
111 void trilateracion ( nodo_fijo_t * nodo_vec , uint8_t cant_nodos , float32_t *
112 posicion )
113 {
114     uint8_t i ,i1 ,i2 ,i3 ;
115
116     float32_t Adata[2*2];
117     float32_t Atdata[2*2];
118     float32_t bdata[2*1];
119     float32_t distref [2];
120
121     arm_status s;
122
123 // Seleccion de nodos para trilaterar
124     i1=0;
125     i2=0;
126     i3=0;
127
128 // Busco los tres nodos con mejor RSSI.
129
130 for ( i=0;i<cant_nodos ;i++)
131 {
132     if ( nodo_vec [ i ] . prom_rssi>nodo_vec [ i3 ] . prom_rssi )
133     { if ( nodo_vec [ i ] . prom_rssi>nodo_vec [ i2 ] . prom_rssi )
134     {
135         if ( nodo_vec [ i ] . prom_rssi>nodo_vec [ i1 ] . prom_rssi )
136         {
137             i3=i2 ;
138             i2=i1 ;
139             i1=i ;
140         }
141         else
142         {
143             i3=i2 ;
144             i2=i ;
145         }
146     }

```

```

147     else
148     {
149         i3=i ;
150     }
151 }
152 }
153
154
155 // ALGORITMO DE TRILATERACION
156
157 // Trilatero con i1 ,i2 ,i3
158
159 // Calculo las distancias entre nodos
160 distref[0] = dist2d(nodo_vec[i2].posicion ,nodo_vec[i1].posicion );
161 distref[1] = dist2d(nodo_vec[i3].posicion ,nodo_vec[i1].posicion );
162
163 // Creo la Matriz A
164 Adata[0] = nodo_vec[i2].posicion [0] - nodo_vec[i1].posicion [0];
165 Adata[1] = nodo_vec[i2].posicion [1] - nodo_vec[i1].posicion [1];
166 Adata[2] = nodo_vec[i3].posicion [0] - nodo_vec[i1].posicion [0];
167 Adata[3] = nodo_vec[i3].posicion [1] - nodo_vec[i1].posicion [1];
168
169 // Creo el vector b
170 bdata[0] = (nodo_vec[i1].dist*nodo_vec[i1].dist - nodo_vec[i2].dist*nodo_vec[
171     i2 ].dist + distref[0]*distref[0])/2;
171 bdata[1] = (nodo_vec[i1].dist*nodo_vec[i1].dist - nodo_vec[i3].dist*nodo_vec[
172     i3 ].dist + distref[1]*distref[1])/2;
173
173 arm_matrix_instance_f32 matA = {2,2,Adata};
174 arm_matrix_instance_f32 matAt = {2,2,Atdata};
175 arm_matrix_instance_f32 matB = {2,1,bdata};
176
177 // Calculo A transpuesta
178 s = arm_mat_trans_f32(&matA,&matAt);
179
180 // Matriz Auxiliar 1
181 float32_t aux1data[2*2];
182 arm_matrix_instance_f32 mataux1 = {2,2,aux1data};
183
184 // Aux 1 = At*A
185 s = arm_mat_mult_f32(&matAt,&matA,&mataux1);
186
187 // Invierto la matriz auxiliar
188 float32_t auxinvdata[2*2];
189 arm_matrix_instance_f32 matauxinv = {2,2,auxinvdata};
190
191 s = arm_mat_inverse_f32(&mataux1,&matauxinv);
192
193 // Matriz auxiliar 2
194 float32_t aux2data[2*2];
195 arm_matrix_instance_f32 mataux2 = {2,2,aux2data};
196
197 // Aux 2 = AuxInv * At

```

```

198 s = arm_mat_mult_f32(&matauxinv,&matAt,&mataux2) ;
199 // Aux1 = Aux2 * b
200 s = arm_mat_mult_f32(&mataux2,&matB,&mataux1) ;
202
203 // Lo guardo en posicion
204 posicion[0] = aux1data[0] + nodo_vec[i1].posicion[0];
205 posicion[1] = aux1data[1] + nodo_vec[i1].posicion[1];
206
207 }
208
209
210 void enviar_posicion ( uint8_t * payload , uint8_t size , uint16_t dst_address )
211 {
212
213 frameData d;
214
215 d.dst.shortAddr.panid = PANID;
216 d.src.shortAddr.panid = PANID;
217 d.dst.shortAddr.addr = dst_address;
218 d.src.shortAddr.addr = ADDR_LOCAL;
219
220 d.fcfc = ccWrapperFCF(MAC_FRAME_TYPE_DATA, 0, 0, 1, 0,
221 MAC_ADDR_MODE_SHORT, 0, MAC_ADDR_MODE_SHORT);
222
223 // Mando el numero de secuencia
224 d.payload[0] = sec_num;
225 // Incremento el numero de secuencia
226 sec_num++;
227
228 memcpy((d.payload)+1, payload, size);
229 d.pl_length = size+1;
230
231 ccFrameTx(d);
232
233 }
234
235 float32_t dist2d ( float32_t * r1 , float32_t * r2 )
236 {
237 float32_t x = r1[0] - r2[0];
238 float32_t y = r1[1] - r2[1];
239
240 x *= x;
241 y *= y;
242
243 x += y;
244
245 arm_sqrt_f32(x,&x);
246
247 return x;
248 }
```

B.4. rssi.c

En él se implementan los dos algoritmos de estimación de distancia en función del RSSI.

```
1 /*  
2 * rssi.c  
3 *  
4 *   Created on: 29/10/2012  
5 *       Author: Joaquin  
6 */  
7  
8 #include "rssi.h"  
9  
10 // Pasar RSSI a distancia usando metodo probabilistico  
11 float64_t RSSI_to_dist_1 (int8_t * RSSI_vec , uint8_t size)  
12 {  
13     float32_t suma[cols_tabla];  
14     uint8_t i , dist , rssi_index ;  
15  
16     // Inicializo el vector suma  
17     for(i=0;i<cols_tabla ;i++)  
18     {suma[ i]=0;}  
19  
20     // Inicio el loop sobre el vector de RSSI  
21     for(i=0;i<size ;i++)  
22     {  
23  
24         rssi_index = (uint8_t) ((int8_t)rssi_max - RSSI_vec[ i]);  
25  
26         // Sumo las probabilidades para ese RSSI  
27         if (rssi_index >=0 && rssi_index <filas_tabla )  
28         {  
29             sumar_vectores(suma,tabla+((uint32_t)rssi_index)*cols_tabla ,cols_tabla );  
30         }  
31     }  
32  
33     // Busco el mÁximo  
34     dist = buscar_max(suma,dist_max);  
35     dist++;  
36  
37     return (float64_t) dist;  
38  
39 }  
40  
41 // Pasar RSSI a distancia ajustando la curva  
42 float64_t RSSI_to_dist_2 (int8_t * RSSI_vec , uint8_t size)  
43 {  
44     float64_t dist ,prom ,exponente ;  
45  
46     // Hago un promedio de todos los RSSI  
47     prom = promediar (RSSI_vec ,size );  
48  
49     // Ajusto con la curva  
50     exponente = -(prom+AUSTE_A)/(10*AUSTE_N);
```

```

51    dist = pow(( float64_t )10,exponente);
52
53    return dist;
54}
55}
56
57
58 void sumar_vectores( float32_t * a, float32_t * b, uint8_t size)
59{
60    uint8_t i;
61
62    for ( i=0;i<size ; i++)
63    {
64        a[ i]+=b[ i];
65    }
66}
67
68 uint8_t buscar_max ( float32_t * vec , uint8_t size )
69{
70    uint8_t i ,index ;
71    index=0;
72
73    for ( i=0;i<size ; i++)
74    {
75        if ( vec [ index]<vec [ i] )
76            index=i ;
77    }
78
79    return index ;
80}
81
82 float64_t promediar ( int8_t * vec , uint8_t size )
83{
84    float64_t prom=0.0;
85    uint8_t i ;
86
87    for ( i=0;i<size ; i++)
88    {
89        prom += ( float64_t ) vec [ i]/ size ;
90    }
91
92    return prom;
93}
94
95 void calc_tabla ( float32_t * tabla )
96{
97    uint8_t i ,j ;
98    float32_t pi = 3.14159265358979;
99    float32_t expon ,mod;
100
101 // Inicializo la tabla de ajustes:
102 float32_t tabla_ajuste_u [] =
103     {0,0,0,0,0,0,0,0,0,1,1,1,1,1,1.326000000000000,1.971000000000000,1.873000000000000,1.86

```

```

103 float32_t tabla_ajuste_s [ filas_tabla ] =
104     { 0.0100000000000000 , 0.0100000000000000 , 0.0100000000000000 , 0.0100000000000000 , 0.0100000000000000 ,
105
106 // Calculo cada elemento de la tabla
107 for (j=0;j<filas_tabla ;j++)
108 {
109     for (i=0;i<cols_tabla ;i++)
110     {
111         expon = ( i+1 - tabla_ajuste_u [ j ]) / tabla_ajuste_s [ j ];
112         expon = -0.5*expon*expon ;
113         mod =(float32_t) sqrt(2*pi)*tabla_ajuste_s [ j ];
114         tabla [ j*cols_tabla+i ] = (float32_t) (exp((float64_t)expon)/mod);
115     }
116 }
117
118 }
```

B.5. cc2520.c

En él se implementan las funciones necesarias para la comunicación entre micro y cc2520. Provisto por Pablo Ridolfi.

```

1 /*
2 * cc2520.c
3 * Interfaz con CC2520 conectado a SSP1
4 *
5 * Created on: 30/01/2012
6 * Author: pablo
7 */
8
9 #include "includes.h"
10
11 u8 ccCmd(u8 cmd, u16 addr, u8 n, void * data)
12 {
13     uint8_t txdata[n+10];
14     uint8_t rxdata[n+10];
15     u8 length, j;
16     int i;
17
18     if (n>128) return -1;
19
20     txdata[0] = cmd;
21
22     switch(cmd)
23     {
24         case SFLUSHTX:
25         case SFLUSHRX:
26         case STXON:
27         case STXONCCA:
```

```

28|     case SRXON:
29|     case SSAMPLECCA:
30|     case SNOP:
31|         length = 1;
32|         break;
33|
34|     case TXBUF:
35|         length = n+1;
36|         for (i=0; i<n; i++) txdata [ i+1 ] = (( u8* ) data ) [ i ];
37|         break;
38|
39|     case RXBUF:
40|         for (i=1; i<n+1; i++) txdata [ i ] = 0;
41|         length = n+1;
42|         break;
43|
44|     case BCLR:
45|     case BSET:
46|         if (addr > 0x1F) return -1;
47|         if (n > 7) return -1;
48|         txdata [ 1 ] = (( addr&0x1F )<<3) | ( n&0x7 );
49|         length = 2;
50|         break;
51|
52|     case MEMWR:
53|         txdata [ 0 ] |= (( addr&0xFFFF )>>8);
54|         txdata [ 1 ] = ( u8 ) ( addr&0xFF );
55|         for (i=0; i<n; i++) txdata [ i+2 ] = (( u8* ) data ) [ i ];
56|         length = n+2;
57|         break;
58|
59|     case MEMRD:
60|         txdata [ 0 ] |= (( addr&0xFFFF )>>8);
61|         txdata [ 1 ] = ( u8 ) ( addr&0xFF );
62|         for (i=2; i<n+2; i++) txdata [ i ] = 0;
63|         length = n+2;
64|         break;
65|
66|     case REGRD:
67|         txdata [ 0 ] |= addr&0x3F;
68|         for (i=1; i<n+1; i++) txdata [ i ] = 0;
69|         length = n+1;
70|         break;
71|
72|     case REGWR:
73|         txdata [ 0 ] |= addr&0x3F;
74|         for (i=0; i<n; i++) txdata [ i+1 ] = (( u8* ) data ) [ i ];
75|         length = n+1;
76|         break;
77|
78|     default:
79|         return -1;
80| }

```

```

81
82 SSP_DATA_SETUP_Type sspData;
83
84 sspData.length = length;
85 sspData.rx_data = rxdata;
86 sspData.tx_data = txdata;
87
88 SSEL_Clr();
89
90 for (j=0;j<100;j++);
91 //pausems(1);
92
93 SSP_ReadWrite(LPC_SSP1, &sspData, SSP_TRANSFER_POLLING);
94
95 for (j=0;j<100;j++);
96 //pausems(1);
97 SSEL_Set();
98
99 switch(cmd)
100 {
101     case SFLUSHTX:
102     case SFLUSHRX:
103     case STXON:
104     case STXONCCA:
105     case SRXON:
106     case SSAMPLECCA:
107     case SNOP:
108     case TXBUF:
109     case BCLR:
110     case BSET:
111         break;
112
113     case RXBUF:
114     case REGRD:
115     case REGWR:
116         for (i=0; i<n; i++) ((u8*)data)[i] = rxdata[i+1];
117         break;
118
119     case MEMWR:
120     case MEMRD:
121         for (i=0; i<n; i++) ((u8*)data)[i] = rxdata[i+2];
122         break;
123
124     default:
125         return -1;
126     }
127
128     return rxdata[0];
129 }
130
131 u8 ccRegRd(u8 reg, u8 n, void * out)
132 {
133     if (reg < 0x040)

```

```

134     return ccCmd(REGRD, reg, n, out);
135 else
136     return ccCmd(MEMRD, reg, n, out);
137 }
138
139 u8 ccRegWr(u8 reg, u8 n, void * in)
140 {
141 if (reg < 0x040)
142     return ccCmd(REGWR, reg, n, in);
143 else
144     return ccCmd(MEMWR, reg, n, in);
145 }
146
147
148 /* Actualiza registros del CC2520 siguiendo Tabla 21 de la hoja de datos */
149 void ccConfig(void)
150 {
151     u8 i;
152
153     //Potencia de salida
154     //i=0x32; //0dBm
155     i=0xF7; //5dBm
156     ccCmd(MEMWR, TXPOWER, 1, &i);
157     i=0xF8;
158     ccCmd(MEMWR, CCACTRL0, 1, &i);
159     i=0x85;
160     ccCmd(MEMWR, MDMCTRL0, 1, &i);
161     i=0x14;
162     ccCmd(MEMWR, MDMCTRL1, 1, &i);
163     i=0x3F;
164     ccCmd(MEMWR, RXCTRL, 1, &i);
165     i=0x5A;
166     ccCmd(MEMWR, FSCTRL, 1, &i);
167     i=0x2B;
168     ccCmd(MEMWR, FSCAL1, 1, &i);
169     i=0x11;
170     ccCmd(MEMWR, AGCCTRL1, 1, &i);
171     i=0x10;
172     ccCmd(MEMWR, ADCTEST0, 1, &i);
173     i=0x0E;
174     ccCmd(MEMWR, ADCTEST1, 1, &i);
175     i=0x03;
176     ccCmd(MEMWR, ADCTEST2, 1, &i);
177
178     //ch. 1
179     i=0x10;
180     ccRegWr(FREQCTRL, 1, &i);
181
182 }
183
184 /* parametros: PANID y direcciones locales */
185 s8 ccInit(u16 panid, u16 shortAddr, u64 extAddr, u8 autoAck)
186 {

```

```

187 SSP_CFG_Type sspCfg;
188 PINSEL_CFG_Type pinCfg;
189 uint8_t aux;
190
191
192 //P0.0: RESETn
193 LPC_GPIO0->FIODIR |= 1;
194 LPC_GPIO0->FIOSET = 1;
195
196 //P0.24: VREG_EN
197 LPC_GPIO0->FIODIR |= 1<<24;
198 LPC_GPIO0->FIOSET = 1<<24;
199
200 //P0.1: GPIO5 (poner a GND o Vcc)
201 LPC_GPIO0->FIODIR |= 1<<1;
202
203 //P0.9: MOSI1
204 pinCfg.Portnum = PINSEL_PORT_0;
205 pinCfg.Pinnum = PINSEL_PIN_9;
206 pinCfg.Pinmode = PINSEL_PINMODE_PULLUP;
207 pinCfg.Funcnum = PINSEL_FUNC_2;
208 pinCfg.OpenDrain = PINSEL_PINMODE_NORMAL;
209 PINSEL_ConfigPin(&pinCfg);
210
211 //P0.8: MISO1
212 pinCfg.Pinnum = PINSEL_PIN_8;
213 PINSEL_ConfigPin(&pinCfg);
214
215 //P0.7: SCK1
216 pinCfg.Pinnum = PINSEL_PIN_7;
217 PINSEL_ConfigPin(&pinCfg);
218
219 //P0.6: SSEL1 -> Controlado por software
220 // pinCfg.Pinnum = PINSEL_PIN_6;
221 // PINSEL_ConfigPin(&pinCfg);
222 LPC_GPIO0->FIODIR |= 1<<6;
223 SSEL_Set();
224
225 //Parametros del puerto
226 sspCfg.CPHA = SSP_CPHA_FIRST;
227 sspCfg.CPOL = SSP_CPOL_HI;
228 sspCfg.ClockRate = 1000000;
229 sspCfg.Databit = SSP_DATABIT_8;
230 sspCfg.FrameFormat = SSP_FRAME_SPI;
231 sspCfg.Mode = SSP_MASTER_MODE;
232
233 SSP_Init(LPC_SSP1, &sspCfg);
234 SSP_Cmd(LPC_SSP1, ENABLE);
235
236 u8 id;
237 ccRegRd(CHIPID, 1, &id);
238 if(id != 0x84) return -1;
239

```

```

240 ccConfig() ;
241
242 if (autoAck) ccCmd(BSET, FRMCTRL0, 5, 0) ;
243
244 ccSetLocalExtAddr(extAddr) ;
245 ccSetLocalPANID(panid) ;
246 ccSetLocalShortAddr(shortAddr) ;
247
248 // Configuro la el pin P0.1 como entrada
249 GPIO_SetDir(0,1<<1,0) ;
250
251 // Configuro el pin 5 para que informe nuevo frame
252 aux = 0x09 ;
253 ccRegWr(GPIOCTRL5,1,&aux) ;
254
255 // Configuro la interrupcion de GPIO
256 GPIO_IntCmd(0,1<<1,0) ;
257
258 // Activo la radio
259 ccCmd(SRXON, 0, 0, 0) ;
260
261 return 0;
262 }
```

B.6. cc2520mac.c

En él se implementan las funciones de la subcapa MAC para la comunicación en la red.
Provisto por Pablo Ridolfi.

```

1 /*
2 * cc2520-mac.c
3 *
4 * Created on: 03/02/2012
5 * Author: pablo
6 */
7
8 #include "includes.h"
9
10 void memorycpy( void * dst, void * src, int n)
11 {
12     int i;
13     for( i=0; i<n; i++)
14         ((char*)dst)[i] = ((char*)src)[i];
15 }
16
17 u8 ccSetLocalShortAddr(u16 addr)
18 {
19     return ccCmd(MEMWR, RAMSHORTADR, 2, &addr);
20 }
21
22 u16 ccGetLocalShortAddr( void )
23 {
```

```

24    u16 r;
25    ccCmd(MEMRD, RAMSHORTADR, 2, &r);
26    return r;
27 }
28
29 u8 ccSetLocalPANID(u16 panid)
30 {
31     return ccCmd(MEMWR, RAM.PANID, 2, &panid);
32 }
33
34 u16 ccGetLocalPANID(void)
35 {
36     u16 r;
37     ccCmd(MEMRD, RAM.PANID, 2, &r);
38     return r;
39 }
40
41 u8 ccSetLocalExtAddr(u64 addr)
42 {
43     return ccCmd(MEMWR, RAM.IEEEADR, 8, &addr);
44 }
45
46 u64 ccGetLocalExtAddr(void)
47 {
48     u64 r;
49     ccCmd(MEMRD, RAM.IEEEADR, 8, &r);
50     return r;
51 }
52
53 u8 ccSetSrcShortAddr(u8 n, u32 panid_addr)
54 {
55     if(n>23) return -1;
56     return ccCmd(MEMWR, SRC.MATCH_TABLE+4*n, 4, &panid_addr);
57 }
58
59 u32 ccGetSrcShortAddr(u8 n)
60 {
61     u32 r;
62     if(n>23) return -1;
63     ccCmd(MEMRD, SRC.MATCH_TABLE+4*n, 4, &r);
64     return r;
65 }
66
67 u8 ccSetSrcExtAddr(u8 n, u64 addr)
68 {
69     if(n>11) return -1;
70     return ccCmd(MEMWR, SRC.MATCH_TABLE+8*n, 8, &addr);
71 }
72
73 u64 ccGetSrcExtAddr(u8 n)
74 {
75     u32 r;
76     if(n>11) return -1;

```

```

77    ccCmd(MEMRD, SRC_MATCH_TABLE+8*n, 8, &r);
78    return r;
79 }
80
81 u16 ccWrapperFCF(macFrameType_e frameType, u8 securityEnabled, u8 framePending,
82                    u8 ackRequested,
83                    u8 PANIDcompression, macAddrMode_e destAddrMode, u8 frameVersion,
84                    macAddrMode_e sourceAddrMode)
85 {
86     u16 fcf = 0;
87
88     fcf |= frameType & 0x7;
89     fcf |= (securityEnabled & 1) << 3;
90     fcf |= (framePending & 1) << 4;
91     fcf |= (ackRequested & 1) << 5;
92     fcf |= (PANIDcompression & 1) << 6;
93     fcf |= (destAddrMode & 0x3) << 10;
94     fcf |= (frameVersion & 0x3) << 12;
95     fcf |= (sourceAddrMode & 0x3) << 14;
96
97     return fcf;
98 }
99
100 s8 ccFrameTx(frameData d)
101 {
102     u8 txbuf[128];
103     int i=0;
104
105     //comprobacion de errores
106     if ((macFCFGGetDestAddrMode(d.fcf)==0)&&(macFCFGGetSrcAddrMode(d.fcf)==0)&&(d.fcf
107         &PAN_ID_COMPRESSION_FLAG))
108         return -1;
109
110     //armo la trama
111     memorycpy(txbuf+1, &d.fcf, 2);
112     txbuf[3] = d.seqn;
113
114     i=4;
115     switch(macFCFGGetDestAddrMode(d.fcf))
116     {
117         case MAC_ADDR_MODE_NO_PAN_ADDR:
118             break;
119         case MAC_ADDR_MODE_1_RESERVED:
120             return -2;
121         case MAC_ADDR_MODE_SHORT:
122             memorycpy(txbuf+i, &(d.dst.shortAddr.panid), 2);
123             memorycpy(txbuf+i+2, &(d.dst.shortAddr.addr), 2);
124             i+=4;
125             break;
126         case MAC_ADDR_MODE_EXTENDED:
127             memorycpy(txbuf+i, &(d.dst.extAddr.panid), 2);
128             memorycpy(txbuf+i+2, &(d.dst.extAddr.addr), 8);
129             i+=10;

```

```

127         break;
128     }
129     switch( macFCFGetSrcAddrMode(d.fcf) )
130     {
131         case MAC_ADDR_MODE_NO_PAN_ADDR:
132             break;
133         case MAC_ADDR_MODE_1_RESERVED:
134             return -3;
135         case MAC_ADDR_MODE_SHORT:
136             if( d.fcf & PAN_ID_COMPRESSION_FLAG )
137             {
138                 memorycpy( txbuf+i , &(d.src.shortAddr.addr) , 2 );
139                 i+=2;
140             }
141             else
142             {
143                 memorycpy( txbuf+i , &(d.src.shortAddr.panid) , 2 );
144                 memorycpy( txbuf+i+2, &(d.src.shortAddr.addr) , 2 );
145                 i+=4;
146             }
147             break;
148         case MAC_ADDR_MODE_EXTENDED:
149             if( d.fcf & PAN_ID_COMPRESSION_FLAG )
150             {
151                 memorycpy( txbuf+i , &(d.src.extAddr.addr) , 8 );
152                 i+=8;
153             }
154             else
155             {
156                 memorycpy( txbuf+i , &(d.src.extAddr.panid) , 2 );
157                 memorycpy( txbuf+i+2, &(d.src.extAddr.addr) , 8 );
158                 i+=10;
159             }
160             break;
161     }
162
163 //aca iria el aux security header (no implementado aun...)
164
165 //si el payload no entra, error
166 if((i+d.pl_length+1) > 127) return -4;
167
168 memorycpy( txbuf+i , d.payload , d.pl_length );
169
170 txbuf[0] = i+d.pl_length+1;
171
172 //cargo fifo tx
173 ccCmd(SFLUSHTX, 0, 0, 0);
174 ccCmd(TXBUF, 0, txbuf[0]-1, txbuf);
175
176 //CSMA-CA
177 //primero, leo bit SAMPLED_CCA
178 u8 rssiv, ccaok;
179 u16 timeout1=0;

```

```

180    u16 timeout2;
181    do
182    {
183        timeout1++;
184        if (timeout1==0xFFFF)
185            return -7;
186        ccCmd(SRXON, 0, 0, 0);
187        timeout2=0;
188        do
189        {
190            timeout2++;
191            if (timeout2==0xFFFF)
192                return -8;
193
194            ccCmd(SSAMPLECCA, 0, 0, 0);
195            ccRegRd(RSSISTAT, 1, &rssiv);
196            ccRegRd(FSMSTAT1, 1, &ccaok);
197        }while (!(rssiv == 1)&&(ccaok & 0x08));
198        ccCmd(STXONCCA, 0, 0, 0);
199        ccRegRd(FSMSTAT1, 1, &ccaok);
200    }while (!(ccaok & 0x08));
201
202    u8 exc[3];
203    do
204    {
205        if (i==0xFFFF) return -6;
206        ccRegRd(EXCFLAG0, 3, exc);
207        i++;
208    }while (!(exc[0] & TX_FRM_DONE_BIT));
209
210 //limpio bit TX_FRM_DONE
211 ccCmd(BCLR, EXCFLAG0, 1, 0);
212
213 txbuf[0]=0;
214
215 return 0;
216}
217
218 frameData ccFrameRx(void)
219{
220    u8 i;
221    u8 rxcont;
222    frameData d;
223    u8 rxbuff[128];
224
225 // ccCmd(SRXON, 0, 0, 0);
226
227 // do
228 // {
229 // ccRegRd(EXCFLAG1, 1, &i);
230 // }while (!(i & RX_FRM_DONE_BIT));
231
232 // ccCmd(BCLR, EXCFLAG1, 0, 0);

```

```

233|
234 ccCmd(MEMRD, RXFIFOCNT, 1, &rxcont);
235|
236 ccCmd(RXBUF, 0, rxcont, rxbuf);
237 ccCmd(SFLUSHRX, 0, 0, 0);
238|
239 memorycpy(&(d.fcf), rxbuf+1, 2);
240 memorycpy(&(d.seqn), rxbuf+3, 1);
241|
242 i=4;
243 switch(macFCFGetDestAddrMode(d.fcf))
244 {
245     case MAC_ADDR_MODE_1_RESERVED: //este deberia filtrarse por hardware
246         return d;
247     case MAC_ADDR_MODE_NO_PAN_ADDR:
248         break;
249     case MAC_ADDR_MODE_SHORT:
250         memorycpy(&(d.dst.shortAddr.panid), rxbuf+i, 2);
251         memorycpy(&(d.dst.shortAddr.addr), rxbuf+i+2, 2);
252         i+=4;
253         break;
254     case MAC_ADDR_MODE_EXTENDED:
255         memorycpy(&(d.dst.extAddr.panid), rxbuf+i, 2);
256         memorycpy(&(d.dst.extAddr.addr), rxbuf+i+2, 8);
257         i+=10;
258         break;
259 }
260 switch(macFCFGetSrcAddrMode(d.fcf))
261 {
262     case MAC_ADDR_MODE_1_RESERVED:
263         return d;
264     case MAC_ADDR_MODE_NO_PAN_ADDR:
265         break;
266     case MAC_ADDR_MODE_SHORT:
267         if(d.fcf & PAN_ID_COMPRESSION_FLAG)
268         {
269             memorycpy(&(d.src.shortAddr.addr), rxbuf+i, 2);
270             i+=2;
271         }
272         else
273         {
274             memorycpy(&(d.src.shortAddr.panid), rxbuf+i, 2);
275             memorycpy(&(d.src.shortAddr.addr), rxbuf+i+2, 2);
276             i+=4;
277         }
278         break;
279     case MAC_ADDR_MODE_EXTENDED:
280         if(d.fcf & PAN_ID_COMPRESSION_FLAG)
281         {
282             memorycpy(&(d.src.extAddr.addr), rxbuf+i, 8);
283             i+=8;
284         }
285         else

```

```

286    {
287        memorycpy(&(d.src.extAddr.panid), rdbuf+i, 2);
288        memorycpy(&(d.src.extAddr.addr), rdbuf+i+2, 8);
289        i+=10;
290    }
291    break;
292 }
293
294 d.rssi = rdbuf[rxcont-2]-76; //ajusto segun hoja de datos
295 d.corr_crc = rdbuf[rxcont-1];
296
297 /*aca iria el aux security header (no implementado aun...)*/
298
299 memorycpy(d.payload, rdbuf+i, rxcont-i-2);
300 d.pl_length = rxcont-i-2;
301
302 return d;
303 }
```

B.7. config.h

En él se realizan las configuraciones de la red: direcciones y posiciones de nodos fijos.

```

1 /*
2 * config.h
3 *
4 * Created on: 05/11/2012
5 * Author: joaquin
6 */
7
8 // Aca van los defines caracteristicos de cada nodo
9
10 // Cambiar Address Local para cada nodo (menor a 10 es fijo , mayor o igual movil
11 #define ADDRLOCAL      2
12
13 // Si es un nodo fijo , setearle la posicion
14 #define POS_X 13        // Posicion X de nodo fijo
15 #define POS_Y 23        // Posicion Y de nodo fijo
16
17 #define PANID 1
18 #define ADDR_ENCUEST   1
19
20 #if (ADDRLOCAL<=8)
21     #define FIJO
22 #elif (ADDRLOCAL >=10)
23     #define MOVIL
24 #endif
```

B.8. includes.h

```

1  /*
2   *  includes.h
3   *
4   *  Created on: 31/01/2012
5   *      Author: pablo
6   */
7
8 #ifndef INCLUDES_H_
9 #define INCLUDES_H_
10
11 #ifdef __USE_CMSIS
12 #include "LPC17xx.h"
13#endif
14
15 #include <cr_section_macros.h>
16 #include <NXP/crp.h>
17
18 // Incluyo configuracion de la RED
19 #include "config.h"
20
21 // Incluyo defines (de hardware y software)
22 #define LED_PORT 0
23 #define LED_BIT (1<<22)
24
25 typedef uint8_t u8;
26 typedef int8_t s8;
27 typedef uint16_t u16;
28 typedef int16_t s16;
29 typedef uint32_t u32;
30 typedef int32_t s32;
31 typedef int64_t s64;
32 typedef uint64_t u64;
33 typedef char ascii;
34
35 // Incluyo drivers del micro
36 #include "lpc17xx_ssp.h"
37 #include "lpc17xx_pinsel.h"
38 #include "lpc17xx_gpio.h"
39 #include "lpc17xx_clkpwr.h"
40 #include "lpc17xx_systick.h"
41 #include "lpc17xx_timer.h"
42
43 // Incluyo definiciones matematicas
44 #include "math.h"
45 #include "arm_math.h"
46
47 // Incluyo headers de funciones
48 #include "cc2520.h"
49 #include "cc2520-mac.h"
50 #include "rss.h"
51
52 #include "movil.h"
53 #include "fijo.h"

```

```

54
55 void pausems( int t );
56
57 // Defino variables globales
58 uint8_t mje_nuevo , mje_simulado ;
59
60 #endif /* INCLUDES_H_ */

```

B.9. cc2520.h

```

1  /*
2  * cc2520.h
3  *
4  *   Created on: 31/01/2012
5  *       Author: pablo
6  */
7
8 #ifndef CC2520_H_
9 #define CC2520_H_
10
11 #define SSEL_Set() {LPC_GPIO0->FIOSET = 1<<6;}
12 #define SSEL_Clr() {LPC_GPIO0->FIOCLR = 1<<6;}
13
14 /*
15
16                                     Target Specific Defines
17
18 */
19 /* strobe command registers */
20 #define SNOP                      0x00
21 #define IBUFLD                    0x02
22 #define SIBUFEX                   0x03
23 #define SSAMPLECCA                0x04
24 #define SRES                      0x0F
25 #define MEMRD                     0x10
26 #define MEMWR                     0x20
27 #define RXBUF                     0x30
28 #define RXBUFCP                   0x38
29 #define RXBUFMOV                  0x32
30 #define TXBUF                     0x3A
31 #define TXBUFCP                   0x3E
32 #define RANDOM                    0x3C
33 #define SXOSCON                  0x40
34 #define STXCAL                    0x41
35 #define SRXON                     0x42
36 #define STXON                     0x43
37 #define STXONCCA                  0x44
38 #define SRFOFF                    0x45

```

```

39 #define SXOSCOFF          0x46
40 #define SFLUSHRX          0x47
41 #define SFLUSHTX          0x48
42 #define SACK              0x49
43 #define SACKPEND          0x4A
44 #define SNACK              0x4B
45 #define SRXMASKBITSET      0x4C
46 #define SMRMASKBITCLR      0x4D
47 #define RXMASKAND          0x4E
48 #define RXMASKOR           0x4F
49 #define MEMCP              0x50
50 #define MEMCPR              0x52
51 #define MEMXCP              0x54
52 #define MEMXWR              0x56
53 #define BCLR               0x58
54 #define BSET                0x59
55 #define CTR_UCTR            0x60
56 #define CBCMAC              0x64
57 #define UCBCMAC             0x66
58 #define CCM                 0x68
59 #define UCCM                0x6A
60 #define ECB                 0x70
61 #define ECBO                0x72
62 #define ECBX                0x74
63 #define ECBXO               0x76
64 #define INC                 0x78
65 #define ABORT               0x7F
66 #define REGRD               0x80
67 #define REGWR               0xC0
68
69 /* configuration registers */
70 #define FRMFILT0            0x00
71 #define FRMFILT1            0x01
72 #define SRCMATCH             0x02
73 #define SRCSHORTEN0          0x04
74 #define SRCSHORTEN1          0x05
75 #define SRCSHORTEN2          0x06
76 #define SRCEXTEN0            0x08
77 #define SRCEXTEN1            0x09
78 #define SRCEXTEN2            0x0A
79 #define FRMCTRL0             0x0C
80 #define FRMCTRL1             0x0D
81 #define RXENABLE0            0x0E
82 #define RXENABLE1            0x0F
83 #define EXCFLAG0             0x10
84 #define EXCFLAG1             0x11
85 #define EXCFLAG2             0x12
86 #define EXCMASKA0            0x14
87 #define EXCMASKA1            0x15
88 #define EXCMASKA2            0x16
89 #define EXCMASKB0            0x18
90 #define EXCMASKB1            0x19
91 #define EXCMASKB2            0x1A

```

```

92 #define EXCBINDEX0          0x1C
93 #define EXCBINDEX1          0x1D
94 #define EXCBINDY0           0x1E
95 #define EXCBINDY1           0x1F
96 #define GPIOCTRL0           0x20
97 #define GPIOCTRL1           0x21
98 #define GPIOCTRL2           0x22
99 #define GPIOCTRL3           0x23
100 #define GPIOCTRL4          0x24
101 #define GPIOCTRL5          0x25
102 #define GPIOPOLARITY        0x26
103 #define GPIOCTRL            0x28
104 #define DPUCON              0x2A
105 #define DPUSTAT             0x2C
106 #define FREQCTRL             0x2E
107 #define FREQTUNE             0x2F
108 #define TXPOWER              0x30
109 #define TXCTRL               0x31
110 #define FSMSTAT0             0x32
111 #define FSMSTAT1             0x33
112 #define FIFOCTRL             0x34
113 #define FSMCTRL              0x35
114 #define CCACTRL0              0x36
115 #define CCACTRL1              0x37
116 #define RSSI                  0x38
117 #define RSSISTAT             0x39
118 #define TXFIFO_BUF           0x3A
119 #define RXFIRST              0x3C
120 #define RXFIFOCNT            0x3E
121 #define TXFIFOCNT            0x3F
122 #define CHIPID                0x40
123 #define VERSION               0x42
124 #define EXTCLOCK              0x44
125 #define MDMCTRL0              0x46
126 #define MDMCTRL1              0x47
127 #define FREQEST               0x48
128 #define RXCTRL                0x4A
129 #define FSCTRL                0x4C
130 #define FSCAL0                 0x4E
131 #define FSCAL1                 0x4F
132 #define FSCAL2                 0x50
133 #define FSCAL3                 0x51
134 #define AGCCTRL0               0x52
135 #define AGCCTRL1               0x53
136 #define AGCCTRL2               0x54
137 #define AGCCTRL3               0x55
138 #define ADCTEST0               0x56
139 #define ADCTEST1               0x57
140 #define ADCTEST2               0x58
141 #define MSMTEST0                0x5A
142 #define MSMTEST1               0x5B
143 #define DACTEST0                0x5C
144 #define DACTEST1               0x5D

```

```

145 #define ATEST 0x5E
146 #define DACTEST2 0x5F
147 #define PTEST0 0x60
148 #define PTEST1 0x61
149 #define RESERVED 0x62
150 #define DPUTEST 0x7A
151 #define ACTTEST 0x7C
152 #define RAM_BIST_CTRL 0x7E
153
154 /* RAM memory spaces */
155 #define RAM_PANID 0x3F2
156 #define RAMSHORTADR 0x3F4
157 #define RAM_IEEEADR 0x3EA
158
159 /* Src matching table */
160 #define SRC_MATCH_TABLE 0x380
161 #define SRCSSHORTPENDE0 0x3E7
162 #define SRCEXTPENDE0 0x3E4
163 #define SRCRESINDEX 0x3E3
164
165 /* status byte */
166 #define XOSC16M_STABLE (1 << 7)
167 #define RSSI_VALID (1 << 6)
168 #define EXCEPTION_A (1 << 5)
169 #define EXCEPTION_B (1 << 4)
170 #define DPU_H_ACTIVE (1 << 3)
171 #define DPU_L_ACTIVE (1 << 2)
172 #define TX_ACTIVE (1 << 1)
173 #define RX_ACTIVE (1 << 0)
174
175 /* CHIPID */
176 #define CHIPID_RESET_VALUE 0x84
177
178 /* CHIP VERSION */
179 #define REV_A 0
180
181 /* FRMCTRL0 */
182 #define FRMCTRL0_RESET_VALUE 0x40
183 #define RX_MODE(x) ((x) << 2)
184 #define RX_MODE_INFINITE_RECEPTION RX_MODE(2)
185 #define RX_MODE_NORMAL_OPERATION RX_MODE(0)
186 #define AUTOACK_BV (1 << 5)
187
188 /* FRMCTRL1 */
189 #define FRMCTRL1_RESET_VALUE 0x01
190 #define PENDING_OR_BV (1 << 2)
191
192 /* FIFOPCTRL */
193 #define FIFOP_THR_RESET_VALUE 64
194
195 /* FRMFILT0 */
196 #define FRMFILT0_RESET_VALUE 0x0D
197 #define MAX_FRAME_VERSION_MASK (3 << 2)

```

```

198 #define MAX_FRAME_VERSION          (1 << 3)
199 #define PAN_COORDINATOR_BV        (1 << 1)
200 #define ADR_DECODE_BV            (1 << 0)
201
202 /* FREQCTRL */
203 #define FREQCTRL_BASE_VALUE      0
204 #define FREQCTRL_FREQ_2405MHZ    11
205
206 /* FSMSTAT1 */
207 #define SAMPLED_CCA_BV          (1 << 3)
208
209 /* TXPOWER */
210 #define TXPOWER_BASE_VALUE       0
211
212 /* FSMSTATE */
213 #define FSM_FFCTRL_STATE_RX_MASK 0x3F
214 #define FSM_FFCTRL_STATE_RX_INF   31
215
216 /* SRCMATCH */
217 #define SRCMATCH_RESET_VALUE     0x07
218 #define SRC_MATCH_EN             (1 << ( 0 ))
219 #define AUTOPEND                 (1 << ( 1 ))
220 #define PEND_DATAREQ_ONLY        (1 << ( 2 ))
221
222 /* SRCRESINDEX */
223 #define AUTOPEND_RES             (1 << ( 6 ))
224
225 /* FRMFILT1 */
226 #define FRMFILT1_RESET_VALUE     0x78
227
228 /* MDMCTRL1 */
229 #define MDMCTRL1_RESET_VALUE     0x2E
230 #define CORR_THR_MASK            0x1F
231
232 /* GPIO directional control */
233 #define GPIO_DIR_RADIO_INPUT     0x80
234 #define GPIO_DIR_RADIO_OUTPUT    0x00
235
236 /* GPIO command strobes */
237 #define GPIO_CMD_SIBUFEX         0x00
238 #define GPIO_CMD_SRXMASKBITCLR   0x01
239 #define GPIO_CMD_SRXMASKBITSET    0x02
240 #define GPIO_CMD_SRXON           0x03
241 #define GPIO_CMD_SSAMPLECCA      0x04
242 #define GPIO_CMD_SACK             0x05
243 #define GPIO_CMD_SACKPEND        0x06
244 #define GPIO_CMD_SNACK           0x07
245 #define GPIO_CMD_STXON           0x08
246 #define GPIO_CMD_STXONCCA        0x09
247 #define GPIO_CMD_SFLUSHRX         0x0A
248 #define GPIO_CMD_SFLUSHTX         0x0B
249 #define GPIO_CMD_SRXFIFOPOP      0x0C
250 #define GPIO_CMD_STXCAL          0x0D

```

```

251 #define GPIO_CMD_SRFOFF 0x0E
252 #define GPIO_CMD_SXOSCOFF 0x0F
253
254 /* GPIO exceptions */
255 #define EXCEPTION_ECG_EXT_CLOCK 0x00
256 #define EXCEPTION_RF_IDLE 0x01
257 #define EXCEPTION_TX_FRM_DONE 0x02
258 #define EXCEPTION_TX_ACK_DONE 0x03
259 #define EXCEPTION_TX_UNDERFLOW 0x04
260 #define EXCEPTION_TX_OVERFLOW 0x05
261 #define EXCEPTION_RX_UNDERFLOW 0x06
262 #define EXCEPTION_RX_OVERFLOW 0x07
263 #define EXCEPTION_RXENABLE_ZERO 0x08
264 #define EXCEPTION_RX_FRM_DONE 0x09
265 #define EXCEPTION_RX_FRM_ACCEPTED 0x0A
266 #define EXCEPTION_SRC_MATCHL_DONE 0x0B
267 #define EXCEPTION_SRC_MATCHH_FOUND 0x0C
268 #define EXCEPTION_FIFOP 0x0D
269 #define EXCEPTION_SFD 0x0E
270 #define EXCEPTION_DPU_DONE_L 0x0F
271 #define EXCEPTION_DPU_DONE_H 0x10
272 #define EXCEPTION_MEMADDR_ERROR 0x11
273 #define EXCEPTION_USAGE_ERROR 0x12
274 #define EXCEPTION_OPERAND_ERROR 0x13
275 #define EXCEPTION_SPI_ERROR 0x14
276 #define EXCEPTION_RF_NO_LOCK 0x15
277 #define EXCEPTION_RX_FRM_ABORTED 0x16
278 #define EXCEPTION_RXBUFMOV_TIMEOUT 0x17
279 #define EXCEPTION_UNUSED 0x18
280 #define EXCEPTION_CHANNEL_A 0x21
281 #define EXCEPTION_CHANNEL_B 0x22
282 #define EXCEPTION_CHANNEL_COMP_A 0x23
283 #define EXCEPTION_CHANNEL_COMP_B 0x24
284 #define EXCEPTION_RFC_FIFO 0x27
285 #define EXCEPTION_RFC_FIFOP 0x28
286 #define EXCEPTION_RFC_CCA 0x29
287 #define EXCEPTION_RFC_SFD_SYNC 0x2A
288 #define EXCEPTION_RFC_SNIFTER_CLK 0x31
289 #define EXCEPTION_RFC_SNIFTER_DATA 0x32
290
291 /* GPIO exception bit in EXCFLAG0 */
292 #define RF_IDLE_BIT (1<<0)
293 #define TX_FRM_DONE_BIT (1<<1)
294 #define TX_ACK_DONE_BIT (1<<2)
295 #define TX_UNDERFLOW_BIT (1<<3)
296 #define TX_OVERFLOW_BIT (1<<4)
297 #define RX_UNDERFLOW_BIT (1<<5)
298 #define RX_OVERFLOW_BIT (1<<6)
299 #define RXENABLE_ZERO_BIT (1<<7)
300
301 /* GPIO exception bit in EXCFLAG1 */
302 #define RX_FRM_DONE_BIT (1<<0)
303 #define RX_FRM_ACCEPTED_BIT (1<<1)

```

```

304 #define SRC_MATCH_DONE_BIT          (1<<2)
305 #define SRC_MATCH_FOUND_BIT        (1<<3)
306 #define FIFOP_BIT                  (1<<4)
307 #define SFD_BIT                    (1<<5)
308 #define DPU_DONE_L_BIT             (1<<6)
309 #define DPU_DONE_H_BIT             (1<<7)
310
311 /* GPIO exception bit in EXCFLAG2 */
312 #define MEMADDR_ERROR_BIT          (1<<0)
313 #define USAGE_ERROR_BIT            (1<<1)
314 #define OPERAND_ERROR_BIT          (1<<2)
315 #define SPLERROR_BIT               (1<<3)
316 #define RF_NO_LOCK_BIT              (1<<4)
317 #define RX_FRM_ABORTED_BIT         (1<<5)
318 #define RXBUFMOV_TIMEOUT_BIT       (1<<6)
319 #define UNUSED_BIT                 (1<<7)
320
321 /* Clear GPIO exception in EXCFLAG0 */
322 #define RF_IDLE_FLAG                ((1<<0)^0xFF)
323 #define TX_FRM_DONE_FLAG            ((1<<1)^0xFF)
324 #define TX_ACK_DONE_FLAG           ((1<<2)^0xFF)
325 #define TX_UNDERFLOW_FLAG          ((1<<3)^0xFF)
326 #define TX_OVERFLOW_FLAG            ((1<<4)^0xFF)
327 #define RX_UNDERFLOW_FLAG          ((1<<5)^0xFF)
328 #define RX_OVERFLOW_FLAG            ((1<<6)^0xFF)
329 #define RXENABLEZEROFLAG          ((1<<7)^0xFF)
330
331 /* Clear GPIO exception in EXCFLAG1 */
332 #define RX_FRM_DONE_FLAG            ((1<<0)^0xFF)
333 #define RX_FRM_ACCEPTED_FLAG       ((1<<1)^0xFF)
334 #define SRC_MATCH_DONE_FLAG         ((1<<2)^0xFF)
335 #define SRC_MATCH_FOUND_FLAG        ((1<<3)^0xFF)
336 #define FIFOP_FLAG                  ((1<<4)^0xFF)
337 #define SFD_FLAG                    ((1<<5)^0xFF)
338 #define DPU_DONE_L_FLAG             ((1<<6)^0xFF)
339 #define DPU_DONE_H_FLAG             ((1<<7)^0xFF)
340
341 /* Clear GPIO exception in EXCFLAG2 */
342 #define MEMADDRERRORFLAG          ((1<<0)^0xFF)
343 #define USAGEERRORFLAG             ((1<<1)^0xFF)
344 #define OPERANDERRORFLAG          ((1<<2)^0xFF)
345 #define SPLERRORFLAG               ((1<<3)^0xFF)
346 #define RFNOLOCKFLAG               ((1<<4)^0xFF)
347 #define RXFRMABORTEDFLAG          ((1<<5)^0xFF)
348 #define RXBUFMOVTIMEOUTFLAG        ((1<<6)^0xFF)
349 #define UNUSEDFLAG                 ((1<<7)^0xFF)
350
351
352 /* Funciones */
353
354 s8 ccInit(u16 panid, u16 shortAddr, u64 extAddr, u8 autoAck);
355 u8 ccCmd(u8 cmd, u16 addr, u8 n, void * data);
356 u8 ccRegRd(u8 reg, u8 n, void * out);

```

```

357| u8 ccRegWr(u8 reg, u8 n, void * in);
358|
359| #endif /* CC2520_H_ */

```

B.10. cc2520mac.h

```

1  /*
2  * cc2520-mac.h
3  *
4  *   Created on: 03/02/2012
5  *       Author: pablo
6  */
7
8 #ifndef CC2520_MAC_H_
9 #define CC2520_MAC_H_
10
11 #include "includes.h"
12
13 typedef enum
14 {
15     MAC_FRAME_TYPE_BEACON=0,
16     MAC_FRAME_TYPE_DATA=1,
17     MAC_FRAME_TYPE_ACK=2,
18     MAC_FRAME_TYPE_MAC_CMD=3,
19     MAC_FRAME_TYPE_4_RESERVED,
20     MAC_FRAME_TYPE_5_RESERVED,
21     MAC_FRAME_TYPE_6_RESERVED,
22     MAC_FRAME_TYPE_7_RESERVED
23 }macFrameType_e;
24
25 typedef enum
26 {
27     MAC_ADDR_MODE_NO_PAN_ADDR=0,
28     MAC_ADDR_MODE_1_RESERVED=1,
29     MAC_ADDR_MODE_SHORT=2,
30     MAC_ADDR_MODE_EXTENDED=3
31 }macAddrMode_e;
32
33 typedef union
34 {
35     struct
36     {
37         u16 panid;
38         u16 addr;
39     }shortAddr;
40     struct
41     {
42         u16 panid;
43         u64 addr;
44     }extAddr;
45 }macaddr_t;
46

```

```

47| typedef struct _frameData
48| {
49|
50|     u16 fcf;
51|     u8 seqn;
52|     macaddr_t dst;
53|     macaddr_t src;
54|     s8 rssi;
55|     u8 corr_crc;
56|     u8 p1_length;
57|     u8 payload[128];
58| }frameData;
59|
60| #define macFCFGGetDestAddrMode(x) (((x)&0xC00)>>10)
61| #define macFCFGGetSrcAddrMode(x) (((x)&0xC000)>>14)
62| #define macFCFGGetFrameVersion(x) (((x)&0x3000)>>12)
63| #define macFCFGGetFrameType(x) ((x)&0x07)
64|
65| #define PAN_ID_COMPRESSION_FLAG (1<<6)
66|
67| u8 ccSetLocalShortAddr(u16 addr);
68| u16 ccGetLocalShortAddr(void);
69| u8 ccSetLocalPANID(u16 panid);
70| u16 ccGetLocalPANID(void);
71| u8 ccSetLocalExtAddr(u64 addr);
72| u64 ccGetLocalExtAddr(void);
73| u8 ccSetSrcShortAddr(u8 n, u32 panid_addr);
74| u32 ccGetSrcShortAddr(u8 n);
75| u8 ccSetSrcExtAddr(u8 n, u64 addr);
76| u64 ccGetSrcExtAddr(u8 n);
77| u16 ccWrapperFCF(u8 frameType, u8 securityEnabled, u8 framePending, u8
    ackRequested,
78|                 u8 PANIDcompression, u8 destAddrMode, u8 frameVersion, u8 sourceAddrMode);
79| s8 ccFrameTx(frameData d);
80| frameData ccFrameRx(void);
81|
82| void memorycpy(void * dst, void * src, int n);
83|
84| #endif /* CC2520_MAC_H_ */
```

B.11. fijo.h

```

1 /*
2  * fijo.h
3  *
4  *   Created on: 14/11/2012
5  *       Author: joaquin
6  */
7 #ifndef FIJO_H_
8 #define FIJO_H_
9 #include "cc2520-mac.h"
10
```

```

11 void rutina_fijo (void);
12
13 frameData d_rx, d_tx;
14
15 #endif

```

B.12. movil.h

```

1 /*
2 * movil.h
3 *
4 * Created on: 03/11/2012
5 * Author: joaquin
6 */
7
8 #ifndef MOVIL_H_
9 #define MOVIL_H_
10
11
12 #include "includes.h"
13 // #include "cc2520-mac.h"
14 // #include "arm_math.h"
15 // #include "config.h"
16
17 #define RSSI_MAX 20
18
19 struct {
20     uint8_t numero;
21     float32_t posicion[2];
22     int8_t rssi[RSSI_MAX];
23     uint8_t last_rssi;
24     float32_t prom_rssi;
25     float32_t dist;
26 } typedef nodo_fijo_t;
27
28 uint8_t sec_num;
29
30 void rutina_movil (void);
31 void trilateracion (nodo_fijo_t * nodo_vec, uint8_t cant_nodos, float32_t *
32     posicion);
33 void enviar_posicion (uint8_t * payload, uint8_t size, uint16_t dst_address);
34 float32_t dist2d (float32_t * r1, float32_t * r2);
35
# endif

```

B.13. rssih.h

```

1 /*
2 * rssih.h
3 *

```

```

4 * Created on: 29/10/2012
5 * Author: Joaquin
6 */
7
8 #include "includes.h"
9 //#include "math.h"
10
11 #define dist_max 15
12 #define dist_min 1
13
14 #define rssи_max -40
15 #define rssи_min -94
16 #define filas_tabla (rssи_max-rssи_min+1)
17 #define cols_tabla (dist_max-dist_min+1)
18
19 #define AJUSTE_A 41.9
20 #define AJUSTE_N 2.088
21
22 float64_t RSSI_to_dist_1 (int8_t * RSSI_vec , uint8_t size);
23 float64_t RSSI_to_dist_2 (int8_t * RSSI_vec , uint8_t size);
24 void sumar_vectores (float32_t * a, float32_t * b, uint8_t size);
25 uint8_t buscar_max (float32_t * vec, uint8_t size);
26 float64_t promediar (int8_t * vec, uint8_t size);
27 void calc_tabla (float32_t * tabla);
28
29 float32_t tabla[filas_tabla*cols_tabla];

```

Bibliografía

- [1] Costos de LPC1343 en digikey. <http://www.digikey.com/product-detail/en/LPC1343FBD48,151/568-4945-ND/2180456/>. Verificado: 12/12/2012.
- [2] Paolo Barsocchi, Antonio Blasco Bonito, and Stefano Chessa. Localizacion in open fields by using rssi on ieee 802.15.4. *Proc. of the Eurosensors XXIII Conference*, 2009.
- [3] Ana Laura Diedrichs. IEEE802.15.4. Notas de clase para Introducción a Redes de Sensores Inalámbricos - Carrera de Especialización de Sistemas Embebidos - FIUBA.
- [4] Juergen Graefenstein and M. Essayed Bouzouraa. Robust method for outdoor localization of a mobile robot using received signal strength in low power wireless networks. *IEEE International Conference on Robotics and Automation*, 2008.
- [5] William S. Murphy Jr. and Willy Hereman. Determination of a position in three dimensions using trilateration and approximate distances. Noviembre 1999.
- [6] Pablo Ridolfi, Sergio Scaglia, Ariel Lutemberg, and Pedro Martos. Diseño e implementación de un nodo compatible con 802.15.4 para redes inalámbricas de sensores. *Congreso Argentino de Sistemas Embebidos*, 2012.
- [7] Charlotte Seem and Espen Slette. *Application Note AN065: Using CC2591 Front End with CC2520*. Texas Instruments.
- [8] IEEE Computer Society. Part 15.4: Low-rate wireless personal area networks (LR-WPANs). In *IEEE Standard for Local and metropolitan area networks*. IEEE, 2011.
- [9] Texas Instruments. *CC2520 Datasheet*, Diciembre 2007.
- [10] Texas Instruments. *CC2591 Datasheet*, Junio 2008.
- [11] Xinwei Wang, Ole Bischoff, Rainer Laur, and Steffen Paul. Localization in wireless ad-hoc sensor networks using multilateration with rssi for logistic applications. *IEEE International Conference on Robotics and Automation*, 2009.