UNIVERSIDAD POLITÉCNICA DE CARTAGENA

Escuela Técnica Superior de Ingeniería de Telecomunicación

Programación de controlador Zigbee Mimo para su uso con antenas leaky-wave en redes de sensores inalámbricas avanzadas

TRABAJO FIN DE GRADO

Autor:

Jose Mendoza García

Directores:

Rafael Asorey Cacheda

Jose Luis Gómez Tornero

A mis padres, lo hice, lo hicimos.

“No es más grande el que siempre triunfa, sino el que nunca desalienta“

Agradecimientos

Me gustaría dar las gracias en primer lugar a mis directores D. Rafael Asorey Cacheda y D. Jose Luis Gómez Tornero por su ayuda a la hora de adentrarme en ésta primera aproximación al trabajo de investigación en Ingeniería para mí y por su incondicional apoyo y dedicación. Sin olvidarme en estos agradecimientos de Eloy Andreu.

A mis compañeros que me han acompañado durante estos años, a todos ellos, gracias infinitas.

A mi familia y amigos por apoyarme hasta el final y creer siempre en mi.

Esto también es vuestro.

Índice General

Índice de ilustraciones

Índice de tablas

Capítulo Primero

Introducción

1.1 Resumen

Las redes WSN, basadas en el protocolo IEEE 802.15.4 [1] e implementadas con el estándar Zigbee [2], se usan para conectar dispositivos de bajo consumo y con una baja tasa de transmisión de datos, pudiendo cubrir áreas de corto alcance (del orden de los 100 metros como mucho) [3]. Bajo el concepto de smart WSN se engloban aquellas mejoras en las redes de sensores que permiten aumentar la eficiencia de estas, de suma importancia a la hora de reducir el consumo de baterías por parte de los sensores [4], [5]. En la mejora de la eficiencia general de una WSN intervienen muchos factores, como por ejemplo comunicaciones más eficientes con menores interferencias y mejor calidad de señal, el aumento de la cobertura radioeléctrica, la localización automática de los dispositivos desplegados en una WSN, la seguridad o la carga inalámbrica, entre otros.

En cualquier caso, el uso de antenas inteligentes (bien conectadas solamente en el coordinador o también en todos los sensores) es una de las tecnologías facilitadoras para mejorar el enlace radioeléctrico y concebir redes de sensores más eficientes, siendo muchas las propuestas realizadas en este sentido en los últimos diez años [6]- [11]. Todos los diseños de antenas inteligentes para WSN se basan en el uso de antenas directivas o sectoriales reconfigurables mediante el uso de electrónica de tipo switched beam [7], [12]-[16]. Estos mecanismos aumentan el coste y el consumo de energía de los circuitos, así como el procesado de señal para adaptar la antena a las necesidades de la red de forma dinámica, lo cual es muy negativo en el contexto de redes de sensores eficientes energéticamente y despliegue barato.

En este trabajo fin de estudio se estudiará una nueva topología para desarrollar un coordinador Zigbee con capacidad de beaming o enfoque adaptativo. Esto proporciona una elevada sencillez, tanto por el tipo de antena empleado como por el procesado de señal necesario para reconfigurar el haz. El enfoque

adaptativo se basa en el uso de antenas de onda de fuga (LWA) y esquemas de salto en frecuencia.

Las LWA son muy sencillas estructuralmente y permiten generar un haz directivo que se puede escanear en una dirección simplemente variando la frecuencia de la señal de microondas [17]. Al igual que el prisma de Newton es capaz de desviar cada longitud de onda en una dirección distinta en el espacio, las LWA son estructuras radiantes pasivas que presentan esta interesante propiedad de enfocar la energía electromagnética hacia diferentes direcciones según la frecuencia de la señal. Si bien las LWA se han usado convencionalmente para aplicaciones RADAR [18]-[20], más recientemente se han propuesto en el contexto de antenas multihaz para aplicaciones de 5G en bandas milimétricas [21]-[26], así como para comunicaciones IoT ("Internet of Things") .

El grupo de investigación GEAT de la UPCT junto con el que he desarrollado este TFG, ha propuesto el uso de LWAs para estimación del ángulo de llegada en sistemas radioeléctricos ("radio direction finding") [27]-[31]. Además, ha propuesto el uso de antenas de panel en configuración monopulso para localización en redes WLAN de tipo Wi-Fi [32]-[35]. De la misma forma, ha aplicado diseños de antenas LWA para localización en dichas redes Wi-Fi [36],[37] y redes WPAN de tipo Bluetooth [38]-[42], en la banda ISM de 2.4GHz. Más recientemente se ha propuesto también el uso de LWA para redes de sensores pasivos de tipo RFID en la banda ISM de 900 MHz [43],[44]. En cuanto a redes de sensores activos (WSN) de tipo Zigbee en la banda de 2.4GHz como las tratadas en este TFG, se hizo un primer intento de uso de las LWA para Transferencia Inalámbrica de Potencia (WPT, Wireless Power Transfer) en [45]-[47].

En cualquier caso, como se ha comentado anteriormente, el uso de LWA para localización de sensores y dispositivos móviles en redes inalámbricas IoT, se basa en el uso de esquemas de salto en frecuencia entre los canales disponibles en cada tipo de red inalámbrica. Las LWAs proporcionan inherentemente la generación de haces directivos escaneados, cuya dirección de apuntamiento varía al cambiar la frecuencia de la señal de radio. Esta

propiedad se llama "frequency-beam scanning", o escaneo del haz por variación de la frecuencia.

Así, este TFE demostrará la aplicabilidad de una LWA, que ha sido sintonizada para enfocar y dispersar los 16 canales de Zigbee de 2,4 GHz en diferentes direcciones , para desarrollar un smart Zigbee coordinator que mejore la calidad de las conexiones usando técnicas de salto en canal (channel hopping) entre los 16 canales disponibles [48] y un esquema de diversidad espacial MiMo con dos puertos. Si bien el channel hopping en Zigbee está inicialmente concebido para aumentar la robustez frente a interferencias [49]-[53], al igual que la diversidad espacial [54]-[56], en este proyecto se explorará por primera vez la diversidad espacio-frecuencial de las LWA en el contexto de una Smart WSN. Para ello, el coordinador debe implementar tecnología MiMo y poder realizar saltos en frecuencia, para controlar la frecuencia del canal en dos puertos distintos de conexión a antena. Los experimentos se realizarán en un entorno controlado (cámara anecoica).

1.2 Objetivos

-Programar un coordinador Zigbee (IEEE 802.15.4) MiMo (con capacidad de control de dos puertos de antena externos), para ser usado junto con una antena leaky-wave (LWA) sintonizada en la banda de frecuencia ISM de este protocolo de red inalámbrica de sensores (WSN, Wireless Sensor Network).

-Gestionar, usando el coordinador Zigbee, un sistema de channel hopping (salto de canal) entre los 16 canales disponibles (del canal #11, centrado en 2,405 GHz, al canal #26, en 2,480 GHz, en saltos de 5 MHz) y, utilizando sus funcionalidades MiMo o de antenna diversity, seleccionar entre los dos puertos de antena disponibles.

-Conectar los puertos de antena del coordinador a los dos terminales de una LWA, previamente diseñada y fabricada, para generar haces directivos en diferentes trayectorias mediante el cambio de frecuencia del canal.

-Medir diferentes parámetros digitales de la calidad de la conexión inalámbrica Zigbee, tales como la RSSI (Received Signal Strength Indicator), el ED (Energy Dectection), el LQI (Link Quality Indicator), el PER (Conditional Packet Error Rate), o el caudal de una conexión entre el coordinador y un nodo Zigbee.

-Finalmente, montar un demostrador que verifique, como prueba de concepto, la propuesta de un coordinador Zigbee avanzado para redes inalámbricas de sensores inteligentes (Smart WSN) basado en esquemas de radio cognitiva (channel-hopping) y LWA con capacidad de escaneo en frecuencia.

1.3 Fases

1- Programación y configuración de un coordinador Zigbee MiMo para comunicarse con otros nodos Zigbee que pueda seleccionar el canal deseado de entre los 16 disponibles, según el estándar IEEE 802.15.4, y escoger uno de los dos puertos de antena.

2- Programación del coordinador Zigbee MiMo para, una vez establecida una conexión por un canal de frecuencia y puerto de antena, medir diferentes parámetros digitales de la calidad de la conexión disponibles tales como: RSSI, ED, LQI, PER o el caudal.

3- Diseño e implementación de un protocolo de comunicaciones entre el coordinador y los sensores inalámbricos, basado en salto de canal secuencial y en la diversidad espacial que proporciona la selección independiente de dos puertos de antena distintos. Los nodos deberán sincronizarse con el esquema de channel hopping impuesto por el coordinador para establecer un enlace radioeléctrico satisfactorio en el canal escogido y el puerto de antena seleccionado. Se determinará la velocidad a la que se puede realizar el cambio secuencial de canal y puerto, sin que se pierda la conexión.

4- Medida en cámara anecoica de los diagramas de radiación analógicos (usando Vector Network Analyzer, VNA) producidos por una LWA con dos puertos, sintonizada en la banda Zigbee de 2,4 GHz, y con capacidad de

escaneo espacial de haces directivos conforme se cambia la frecuencia de la señal (channel-beam-scanning).

5- Comprobación experimental de la generación de 16 haces directivos escaneados en los diferentes canales de Zigbee (#11-#26). Determinación del campo de visión (Field of View) que barre el dispositivo.

6- Medida en cámara anecoica de los diagramas de radiación digitales (usando el coordinador Zigbee MiMo conectado a los dos puertos de la LWA y midiendo el RSSI y el ED en dB) en los diferentes canales Zigbee (#11-#26) y validación con los resultados analógicos.

7- Demostración experimental de las prestaciones potenciales del coordinador Zigbee inteligente propuesto, mediante la medida de distintos parámetros de la calidad de la conexión conseguida.

8- Comparación de estas prestaciones con las del uso de una antena normal (sin capacidad de channel-beam-scanning) usando un coordinador Zigbee MiMo convencional (con dos antenas monopolo y sin usar channel hopping).

Capitulo Segundo

Fundamentos y conceptos generales

2.1 Wireless Sensor Network (WSN)

Estas redes de sensores inalámbricas vienen impulsadas debido al reciente desarrollo de la tecnología de comunicación de sensores. Esta red consta de una gran cantidad de nodos de sensores que pueden servir para aplicaciones diversas. Estos nodos no solo pueden realizar tareas como recibir parámetros ambientales (temperatura, humedad, presión...) sino también procesar datos e interactuar entre ellos, lo que ayuda a una operatividad más eficiente en términos de consumo de energía al estar alimentados por baterías con una vida útil limitada.

Pero por otro lado este tipo de red de sensores inalámbricos e inteligentes debe hacer frente también a limitaciones como cobertura, memoria, capacidad computacional y el ancho de banda de comunicación, como grandes desafíos.

Principalmente se caracterizan por su facilidad de despliegue y por ser autoconfigurables, pudiendo convertirse en todo momento en emisor o receptor.

2.2 El estándar IEEE 802.15.4

El estándar IEEE 802.15.4 [1] es un grupo de trabajo del IEEE 802.15 y se creó para investigar una solución de baja velocidad de datos con una duración de la batería duradera y una complejidad muy baja. Las aplicaciones propias de este estándar son muy amplias y van desde sensores, control remoto, juguetes interactivos, o automatización del hogar. Los protocolos ZigBee que utilizamos en este experimento se basan en la especificación llevada a cabo por este grupo de trabajo.

Propiedad	Rango
Rango de transmisión de datos	868 MHz: 20kb/s; 915 MHz: 40kb/s; 2.4 GHz: 250 kb/s.
Alcance	10 – 20 m.
Latency	Abajo de los 15 ms.
Canales	868/915 MHz: 11 canales. 2.4 GHz: 16 canales.
Bandas de frecuencia	Dos PHY: 868/915 MHz y 2.4 GHz.
Direccionamiento	Cortos de 8 bits o 64 bits IEEE
Canal de acceso	CSMA-CA y rasurado CSMA-CA
Temperatura	El rango de temperatura industrial: -40° a +85° C

Tabla 1: Estándar 802.15.4.

2.2.1 Protocolo de comunicación ZigBee IEEE 802.15.4

El protocolo de comunicación ZigBee IEE 802.15.4 está compuesto por 16 canales en la banda ISM de 2.4 GHz donde cada canal se corresponde con una frecuencia de uso. Los canales se agrupan del 11 al 26 en saltos de ancho de banda de 5 MHz y frecuencias tal y como vemos siguiendo la siguiente figura:

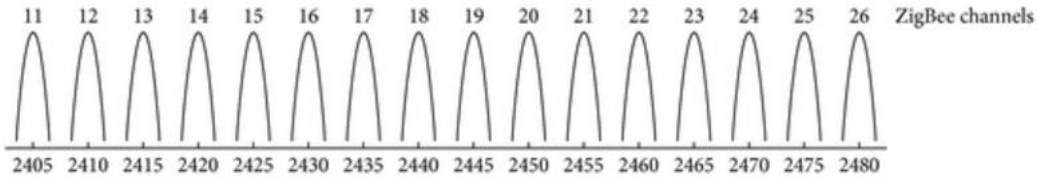


Ilustración 1:Agrupación de canales ZigBee.

Está definido en la llamada capa MAC IEEE 802.15.4.

Tiene como cometido aplicaciones que necesitan comunicaciones seguras con una gran vida útil de sus baterías gracias a su bajo consumo y una tasa de envío baja.

2.3 Channel Hopping

Antes de abordar esta técnica es interesante comprender la técnica de espectro ensanchado o Frecuency Hopping Spread Spectrum (FHSS).

Es una técnica de modulación digital en la que la señal, mediante frecuencias varias, se expande para evitar interferencias y facilitar su du detección (al ser una comunicación con saltos en frecuencia es difícil detectar la señal)

Suele ser utilizada en redes inalámbricas llevando a cabo saltos de frecuencia de la portadora.

Como hemos dicho, esta tecnología da lugar a evitar interferencias pero del mismo modo mejora la relación señal a ruido y eleva la capacidad de la señal y su privacidad.

2.4 Beamforming

El Beamforming o también llamado conformación de haces es una técnica mediante la cual se envía energía en todas las direcciones a pesar de que un solo canal esté use todo el espacio de la comunicación.

Con el fin de abordar este inconveniente lo que se idea es enfocar toda la energía hacia el receptor (beamforming), lo que da lugar a aprovechar los eficientemente los canales utilizados en ese instante mejorando de la misma manera el rendimiento de este tipo de redes inalámbricas.

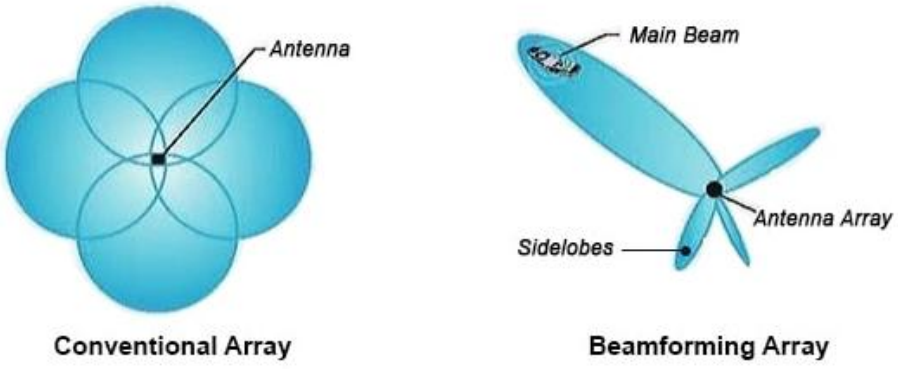


Ilustración 2:Antena convencional vs. Antena beamforming.

2.5 Antenas de onda de fuga o 'Leaky Wave'

Para este experimento se utilizará una antena 'Leaky Wave' planar impresa en tecnología 'microstrip'. El funcionamiento de las antenas leaky-wave reside en la propagación de las ondas de fuga de líneas de transmisión abiertas. Dicha línea de transmisión da lugar a una atenuación debida a la radiación, cosa que en el caso de una línea cerrada no ocurre ya que las ondas no radian hacia el exterior sino que se propagan por la superficie de la línea.

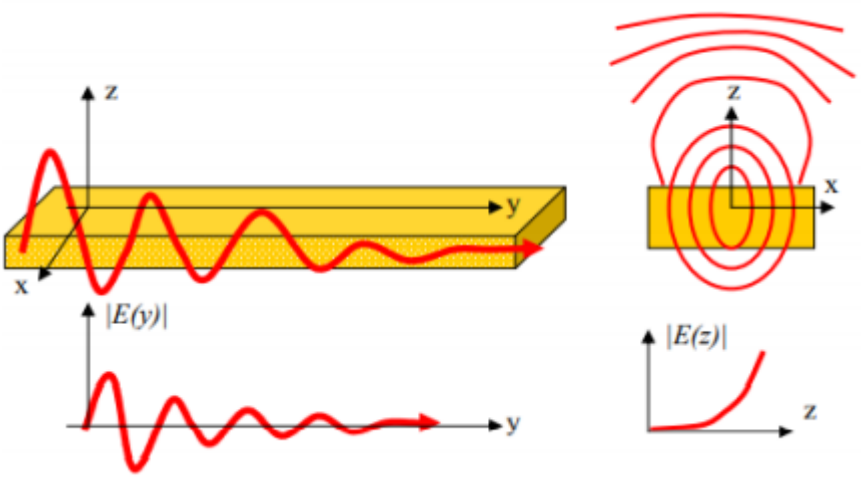


Ilustración 3: Onda de fuga.

Podemos relacionar este tipo de antenas con un prisma de Newton, donde el ángulo de radiación de la antena será diferente en base a la frecuencia de la señal.

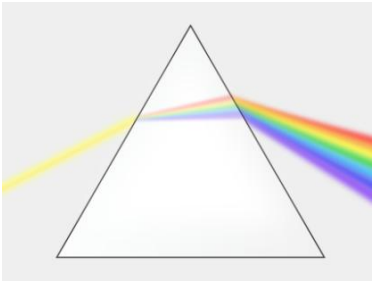


Ilustración 4: Prisma de Newton.

2.6 Indicador de fuerza de la señal recibida (RSSI)

Escala de referencia para medir el nivel de potenia de las señales recibidas por un dispositivo en redes inalámbricas. Dicha escala posee un valor en torno a 0 dBm (1 mW) y se mueve dentro de un rango negativo de valores, donde cuanto más negativo sea este valor mayor señal se pierde.

Por lo tanto, un valor ideal de este parámetro sería 0 dBm, algo complicado de obtener en la práctica. Hasta un valor de -60 dBm podríamos decir que el enlace es de buena calidad y estable, con valores por debajo de esto pueden

ocasionar cortos en el enlace dando lugar al fin de la comunicación o a la recepción de mensajes corruptos.

El parámetro RSSI tiene un nivel mínimo de -94 dBm con un rango dinámico de 87 dB. Es un parámetro compuesto por 5 bits por lo que su valor oscila entre 0 y 31. Con este parámetro se podría calcular la potencia de entrada de radio frecuencia:

$$P_{RF}[dBm] = RSSI_{BASE\_VAL}[dBm] + 3[dB] \times RSSI$$

Ilustración 5: RSSI.

2.7 Detección de energía (ED)

El parámetro ED estima el nivel de energía de las señales en el canal que el dispositivo pretende transmitir.

Si el nivel de sensibilidad del receptor se aproxima al nivel de energía de la señal, estas no podrán ser encontradas por la ED.

Este valor se promedia de la medida del nivel de energía de ocho periodos de símbolo de la señal recibida. El valor que se devuelve es un entero de 8 bits que revela el nivel de energía del canal. Posee una resolución de 1dB y 84 niveles de energía únicos (valores de 0 a 83).

Con este parámetro se podría calcular la potencia de entrada de radio frecuencia:

$$P_{RF}[dBm] = RSSI_{BASE\_VAL}[dBm] + 1[dB] \times ED\_LEVEL$$

Ilustración 6: ED.

Capítulo Tercero

Desarrollo del proyecto

Este experimento se llevó a cabo en la cámara anecoica, donde situamos nuestra placa ATMEL REB233SMAD en una mesa giratoria enfrentada en línea recta hacia el transmisor, un modulo Digi XBee Serie 1 situado en el otro extremo de la cámara.

Nuestra mesa la giraremos en una variedad de ángulos para almacenar los valores de los parámetros medidos a la vez que nuestro modulo Digi XBee Serie 1 emite para posteriormente obtener los diagramas digitales de radiación de nuestra antena y comprobar el correcto funcionamiento de la técnica channel hopping.

3.1 Descripción de componentes

Para nuestro experimento hemos utilizado los diferentes dispositivos que comentamos a continuación.

Cableado y conexiones

Para las conexiones se han utilizado distintos tipos de cables: cable ethernet RJ-45, dos adaptadores ethernet para usb y dos cables coaxiales SMA macho-hembra.

Ilustración 7: Cable coaxial SMA macho-hembra y Cable RJ-45.

-Cable RJ-45: compuesto por ocho hilos trenzados dos a dos dependiendo de la tarea de cada par. Usaremos un modelo apantallado (STP) más robusto. Puede llegar a alcanzar una velocidad de 1 Gbps.

-Cable coaxial: se utiliza para transportar señales eléctricas de frecuencia elevada. Formado por dos conductores concéntricos, e núcleo encargado de portar la información y la malla como referencia a tierra, todo ello rodeado por una cubierta aislante que los protege. Puede llegar a alcanzar velocidades de 2 Gbps.

-Adaptador Ethernet-USB.

Placa ATMEL REB23SMAD

Para este experimento utilizaremos como receptor una Placa ATMEL REB233SMAD. Como se ha comentado en anteriores apartados, dicha placa es un kit de desarrollo compuesto por dos componentes:

-Board: placa que contiene el chip ATXmega256A3 compuesto por batería, leds, botones, USART y JTAG.

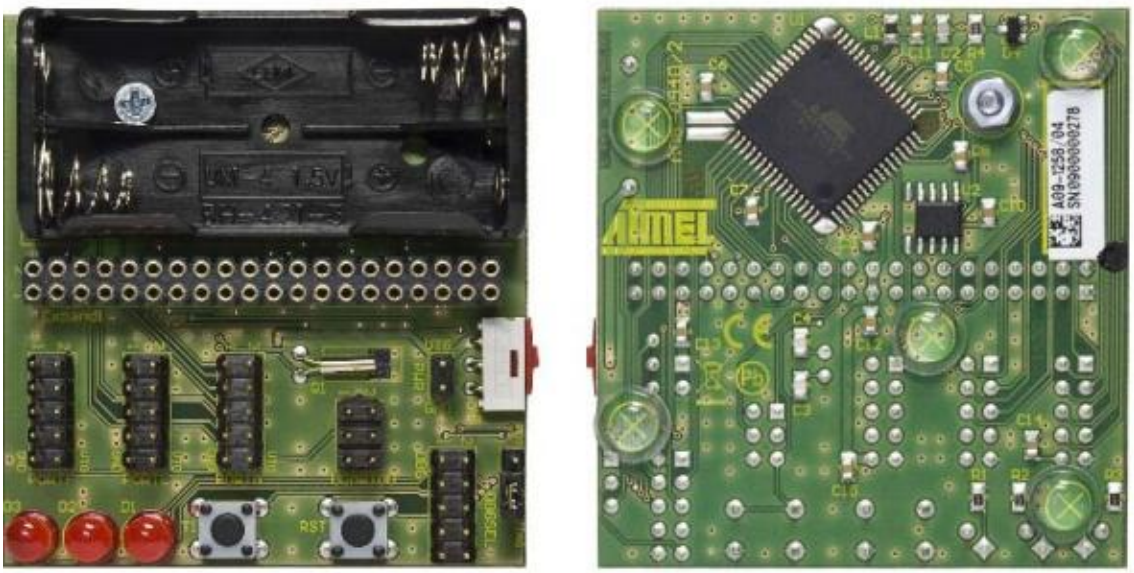


Ilustración 8: Placa ATMEL REB233SMAD.

-Transceptor: contiene el chip AT86RF233 y las antenas.

Entre ambas se comunican por el puerto GPIO y SPI. Estos dispositivos tienen una arquitectura por capas.

Para suministrar energía a este modulo se emplean baterías AAA donde el tiempo de uso es el factor que provoca el consumo de energía.

Módulo XBee Serie 1

Módulo fabricado por Digi a 2.4 GHz. Empleados en el protocolo IEEE 802.15.4 (Zigbee). Son fáciles de controlar, confiables y transparentes para la comunicación bidireccional entre microcontroladores.

Lleva a cabo una comunicación inalámbrica segura, eficaz y sencilla. No se necesita ningún ajuste para trabajar con ella, simplemente se conecta al puerto y listo.

Ilustración 9: Módulo XBee Serie 1.

Velocidad de datos	250 Kbps
Alcance en interiores	30 m
Potencia de transmisión	1mW (0 dBm)
Sensibilidad	-92 dBm
Banda de frecuencia	2.4 GHz
Velocidad de transmisión Serial	1200 bps - 250 Kbps

Tabla 2: Características Módulo XBee.

Adaptador serie USB-TTL

Se utiliza un conversor universal de USB a puerto serie para proporcionar la comunicación entre PC y la placa REB233SMAD con las siguientes características:

Ilustración 10: Adaptador serie USB-TTL.

-Conexión sencilla al PC mediante puerto USB.

-Tasa de transmisión máxima de 1Mbps.

-No necesita fuentes de alimentación independientes.

-Contiene un chip principal FTDI FT232RL con una alta estabilidad y una tensión de salida de 3.3V y 5V.

Ilustración 11: Conexión para la transmisión.

Herramienta de programación Atmel-ICE Kit

El kit Atmel-ICE de Microchip Technology es una herramienta de desarrollo para depurar y programar microcontroladores. Este módulo es totalmente compatible con Atmel Studio y debe conectarse a un PC utilizando el cable USB provisto.

Ilustración 12: Herramienta de programación Atmel-ICE kit.

Ordenador portátil HP Pavilion 15

Se necesita un ordenador portátil ya que tanto la placa como el módulo XBee van conectados a los puertos del mismo, por ello necesitamos los adaptadores y cables antes mencionados.

Este dispositivo se encarga también del desarrollo de software tanto de la placa, como del módulo XBee.

Trabajamos con tres programas. en primer lugar Microchip Studio, un entorno de desarrollo de Atmel Studio integrado para desarrollar y depurar aplicaciones de microcontroladores con el que programaremos nuestra placa usando el lenguaje de programación C.

Por otro lado, necesitamos un entorno científico para programar el módulo XBee, usaremos Spider con el lenguaje de programación Python.

Por último necesitaremos el software Matlab para recibir, almacenar y procesar los datos transmitidos donde obtendremos las mediciones de los parámetros y representar su comportamiento.

Cámara anecoica

Se define como un entorno totalmente cerrado con el fin de absorber el total de reflexiones de ondas electromagnéticas o acústicas. Esta característica y que se encuentra totalmente aislada del entorno exterior hace que dicha cámara simule un entorno ideal para que se propaguen las ondas.

Ilustración 13: Entorno de trabajo.

El diagrama de radiación del módulo XBee es omnidireccional (hace círculos), para las medidas se debe intentar que la antena se encuentre lo más recta posible hacia arriba.

Vemos en la ilustración cómo se ha colocado sobre la plataforma giratoria la antena leakywave y nuestra placa REB233SMAD para realizar un barrido de los ángulos deseados y el módulo XBee lo situaremos en el soporte situado al lado contrario de la cámara anecoica en línea recta.

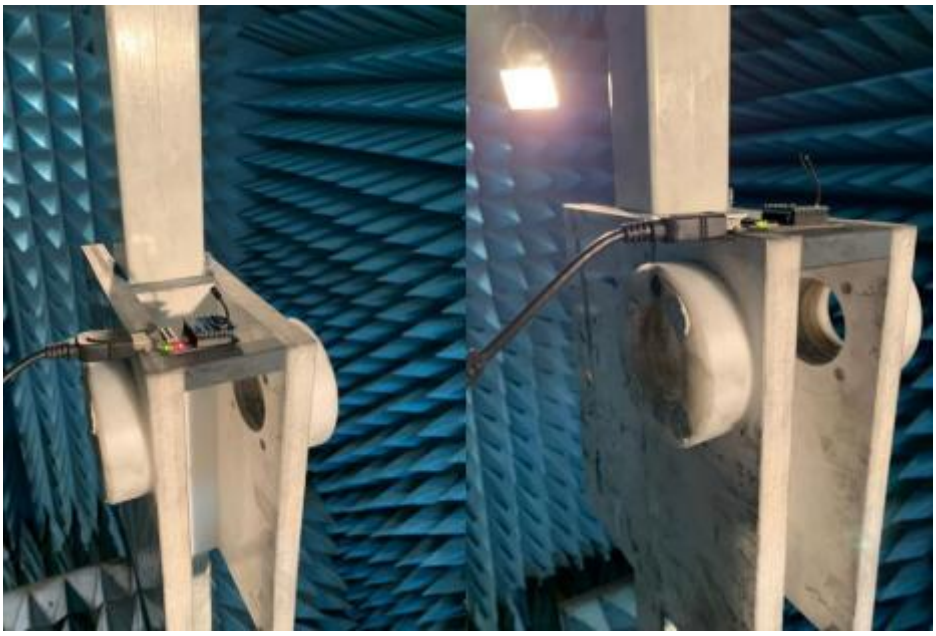


Ilustración 14: Soporte Módulo XBee.

El objetivo principal era el de realizar un barrido desde -90º hacia 90º pero debido a la inestabilidad de la placa solo pudimos tomar medidas para algunos ángulos.

Antena Leaky-Wave

Diseñada en tecnología Substrate Integrated Waveguide (SIW). En la UPCT es el primer experimento en el que se usa este tipo de antena junto con el protocolo de comunicación ZigBee para nuestro fin.

Como vemos en la siguiente ilustración, tenemos una tira de cobre que minimiza la transmisión hacia la parte posterior, dos tornillos de sujeción a la mesa giratoria y dos puertos que se conectarán a los puertos de nuestra placa REB233SMAD mediante cables coaxiales SMA macho y SMA hembra. Que dicha antena posea dos puertos viene dado por la necesidad de abarcar la mayor área de cobertura posible.

Tal y como podemos ver en la siguiente ilustración la radiación emitida por la antena cambia de aspecto dependiendo del puerto que estemos alimentando.

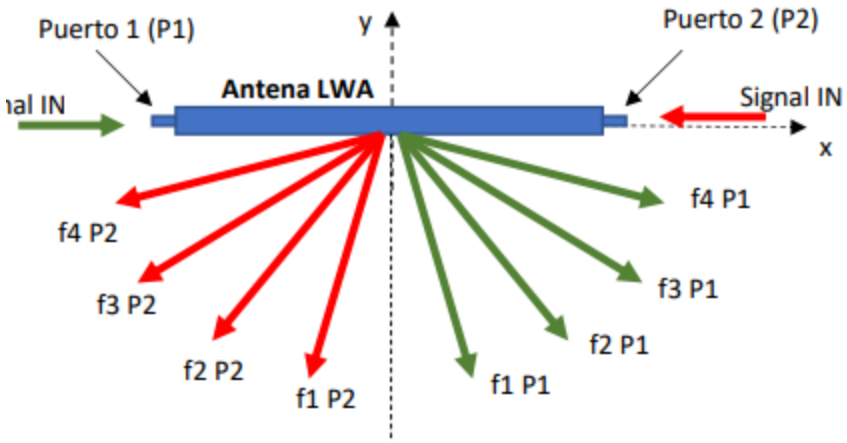


Ilustración 15: Radiación de una antena LWA alimentada por ambos puertos.

Donde la frecuencia 4 será el canal 26 y la frecuencia 1 el canal 11 ya que al alimentar por el Puerto 1 el diagrama de radiación se complementa hacia la mitad derecha de la antena, por lo que si alimentamos por el Puerto 2 será justo al revés, se complementa hacia la mitad izquierda de la antena.

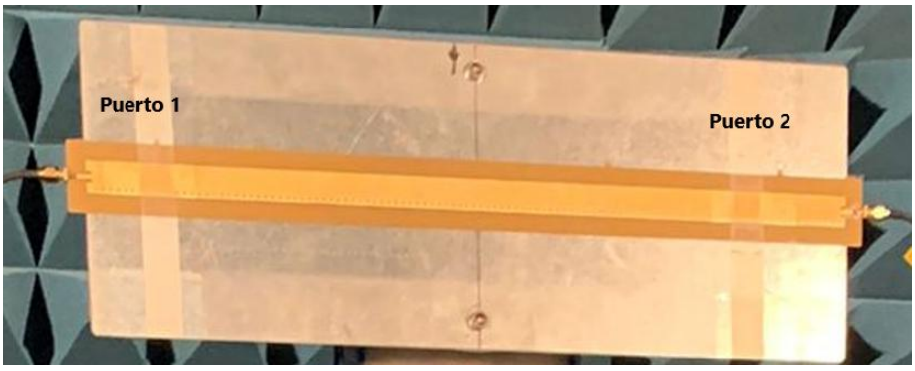


Ilustración 16: Antena LWA.

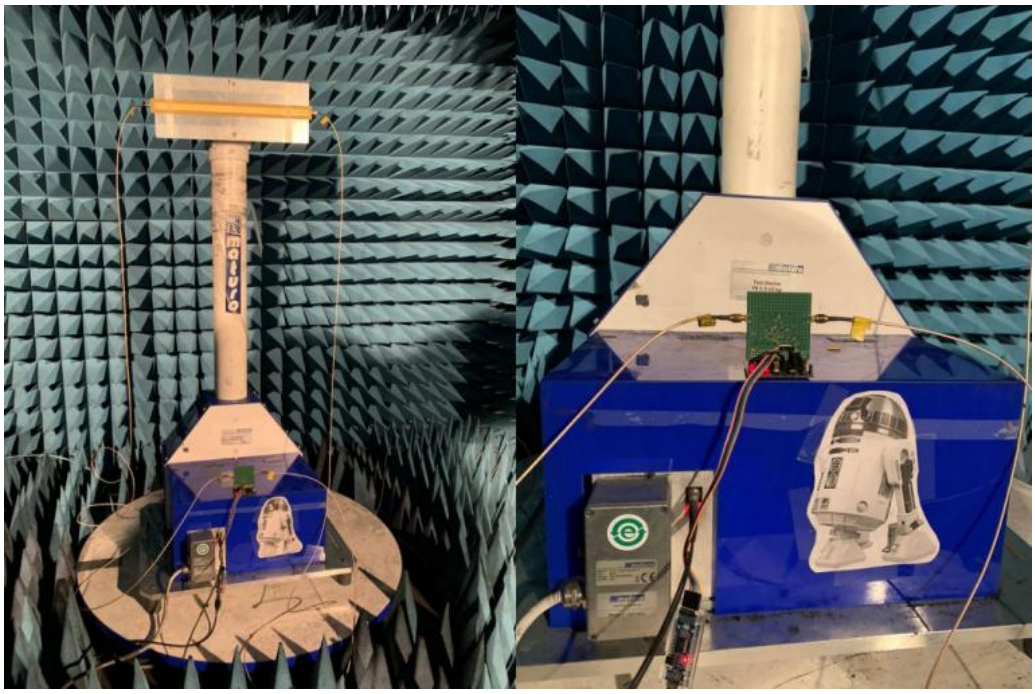


Ilustración 17: Conexión entre Antena LWA y placa REB233SMAD.

Mesa giratoria

Su función principal es la de orientar la antena Leaky-Wave a los ángulos deseados desde uno de los fondos de la cámara con el fin de obtener el diagrama de radiación.

Se utilizó un dispositivo para girar la plataforma a los ángulos deseados.

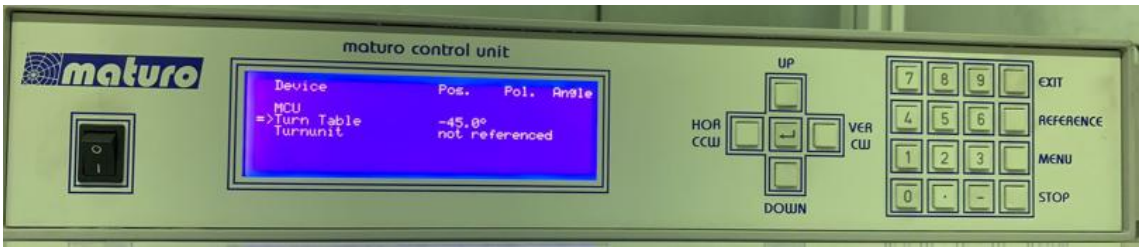


Ilustración 18: Dispositivo para manejar la mesa giratoria.

3.2 Lenguajes de programación

Matlab

Entorno para resolver problemas científicos y de ingeniería. Optimiza altas prestaciones de análisis numérico, cálculo matricial y procesamiento de señales y gráficos.[57]

- Lenguaje de alto nivel para cálculos científicos y de ingeniería.
- Entorno de escritorio optimizado para la exploración iterativa, el diseño y la solución de problemas.
- Gráficas para visualizar datos y herramientas para crear diagramas personalizados.
- Aplicaciones para ajustar curvas, clasificar datos, analizar señales, ajustar sistemas de control y muchas otras tareas.
- Toolboxes complementarias para una amplia variedad de aplicaciones científicas y de ingeniería.
- Herramientas para crear aplicaciones con interfaces de usuario personalizadas.
- Interfaces para C/C++, Java®, .NET, Python, SQL, Hadoop y Microsoft® Excel®.
- Opciones de implementación libres de derechos para compartir programas de MATLAB con los usuarios finales.

Python

Es un lenguaje de programación muy popular y fácil de interpretar debido a su legibilidad del código. Permite programación multiparadigma y soporta orientación a objetos, programación funcional y programación imperativa.

Diseñado para ser leído con facilidad el contenido de los bloques de código es delimitado mediante espacios o tabuladores diferenciándose de otros lenguajes de programación que usan normalmente llaves {}.

C

Lenguaje de programación orientado en gran parte a la implementación de sistemas operativos. Es el lenguaje de programación popular para crear software de aplicaciones y sistemas. Es un lenguaje simple, rápido y eficiente. No dispone de orientación orientada a objetos.

XCTU

Es un programa con una interfaz gráfica de fácil uso con la que poder interactuar con los módulos RF como nuestro módulo XBee.

Vemos en la siguiente ilustración la ventana principal donde podemos configurar la XBee y los parámetros de la comunicación.

El modulo XBee con el paso del tiempo saturaba y se calentaba por lo que podíamos hacer dos cosas, o conectar y desconectar dicho módulo del puerto USB correspondiente o desde esta plataforma volver a leer dicho módulo.

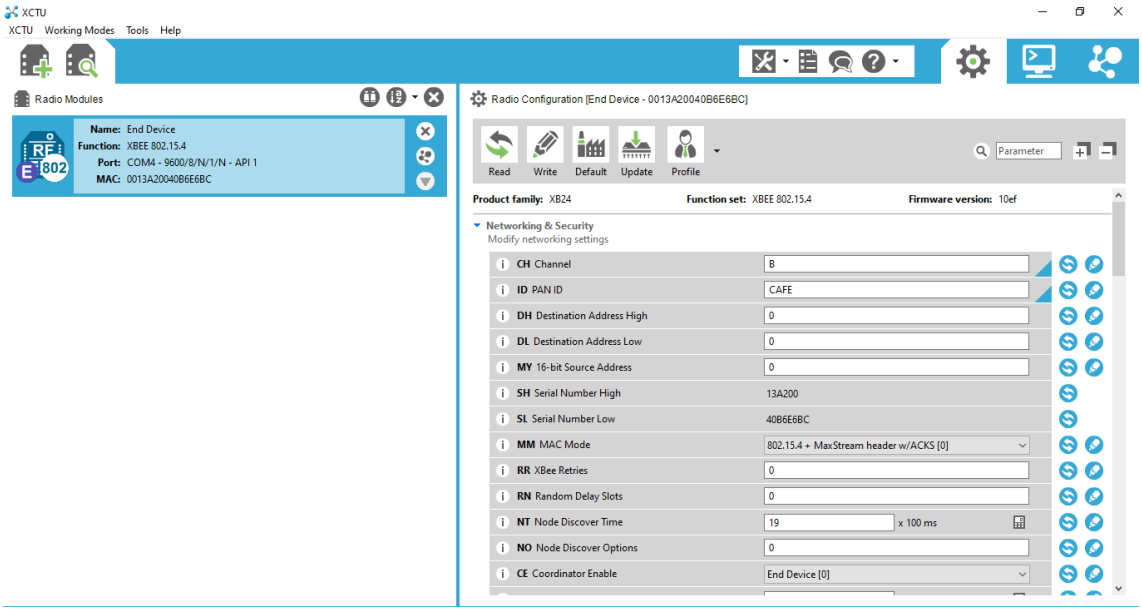


Ilustración 19: XCTU.

3.3 Pruebas experimentales

En este esquema resumen de la conexión de los distintos dispositivos utilizados vemos como vamos a trabajar.

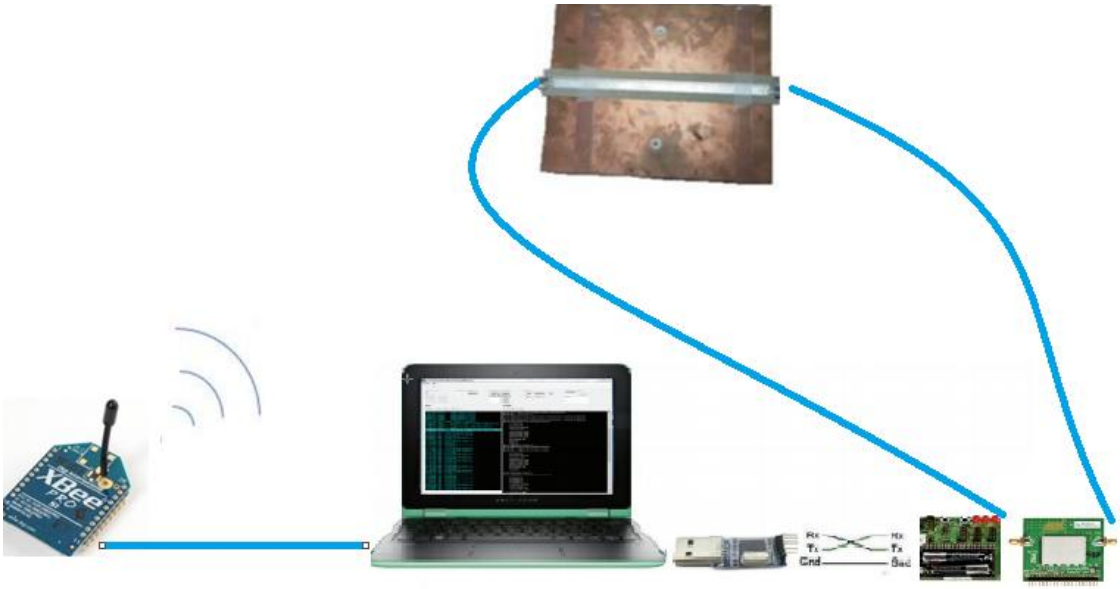


Ilustración 20: Hardware utilizado en el experimento.

Nuestro módulo Xbee va a estar buscando redes y, como sabemos que tenemos 16 canales (protocolo de comunicación Zigbee), es decir, 16 frecuencias distintas, empieza a buscar por cada una y va comprobando la ID de esa red (canal), si la ID es igual que la PANID solicita conectarse al Coordinador.

Es decir, el Coordinador establece una red, el sensor escanea todos los canales y pide permiso de conexión al Coordinador, este le asignará una dirección y se producirá la comunicación.

Durante la comunicación el sensor enviará diez paquetes en dos bloques de cinco al Coordinador, el cual recibirá y leerá parámetros de calidad de señal (RSSI,ED...) y cuando haya recibido los diez paquetes ambos cambiaran de canal y se volverá a repetir el proceso (channel hopping).

Además para cada canal en el Coordinador, que recibe diez paquetes, cada cinco paquetes recibirá por una antena distinta. Los primeros 5 paquetes por la antena 1 y los otros cinco por la antena 2 (diversidad de antenas)

Entonces la XBee envía tramas y recibe la otra placa cinco tramas por un lado, cinco por otro y cambia de canal, y se vuelve a repetir el proceso.

3.3.1 Manual de operación

Con el fin de conectar nuestra placa REB233SMAD al ordenador lo que se hizo fue utilizar nuestro conversor USB-TTL y conectarlo con los puertos tierra, recepción y transmisión. Una vez realizado esto podemos conectar el USB del conversor al puerto USB del ordenador habilitado para ello, tal y como vemos en la siguiente ilustración.

Ilustración 21: Conexión USB-TTL.

Lo que queda ahora es ejecutar el software desarrollado en el entorno MATLAB con el fin de comenzar la comunicación y empezar a recibir las tramas para su posterior análisis. Se puede consultar dicho software en el ANEXO A.

Lo primero que visualizaremos en la consola será lo siguiente:

```
Serial Port Object : Serial-COM4

Communication Settings
   Port:               COM4
   BaudRate:           9600
   Terminator:         'LF'

Communication State
   Status:             closed
   RecordStatus:       off

Read/Write State
   TransferStatus:     idle
   BytesAvailable:     0
   ValuesReceived:     0
   ValuesSent:         0
```

Ilustración 22: Comunicación abierta.

Donde vemos que la comunicación se establece correctamente y el dispositivo se mantiene "escuchando" a la espera de recibir lo que le transmita el módulo XBee.

En paralelo, y en nuestro entorno Spider (recordemos que es donde ejecutamos las tareas de nuestro módulo XBee como podemos ver en el ANEXO B) hacemos lo mismo y obtenemos lo siguiente:

```
 +-----------------------------------------
-+
 | XBee Python Library Set/Get parameters Sample
|
 +-----------------------------------------
-+

Node ID:                    Router
PAN ID:                     CA FE
Destination address high:   00 00 00 00
Destination address low:    00 00 00 00
Channel:                    0B
Scan Channel:               3F FE
Role.END_DEVICE
AI:                         0

Cached parameters
-----------------------------------------
--
64-bit address:   0013A20040B6E6BC
16-bit address:   FFFE
Node Identifier:  Router
Firmware version: 10 EF
Hardware version: XB24-Axx-xx
```

Ilustración 23: Comunicación XBee

Lo cual nos dice que todo funciona correctamente y se va a proceder al envió de tramas tal y como vemos a continuación:

Observamos cómo según lo estipulado, se envían diez tramas en dos paquetes de cinco.

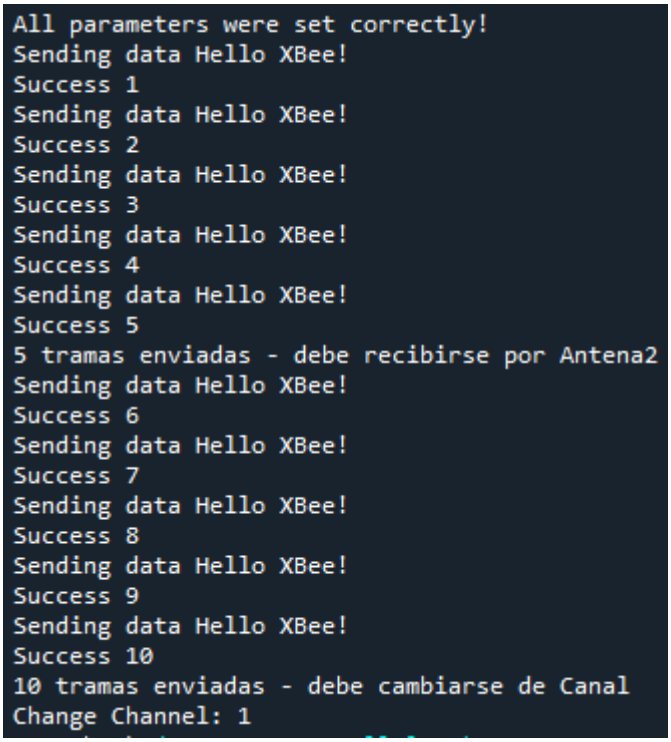


Ilustración 24: Envío de tramas.

Y en la consola de MATLAB comenzaremos a visualizar como comienza a llegar toda la información.

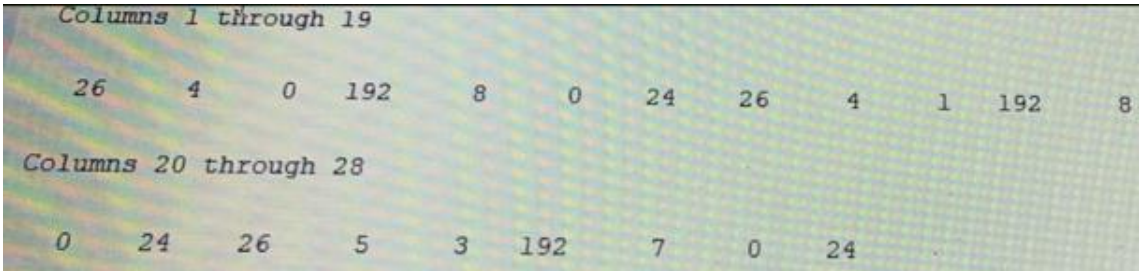


Ilustración 25: Llegada de datos.

Al tener configurada la placa para analizar solamente cinco parámetros durante diez tramas y 16 canales organizo lo datos de la siguiente manera:

11	142	0	224	36
11	14	1	128	36
11	14	2	192	36
11	142	3	224	36
11	14	4	160	36
11	5	5	128	36
11	5	6	128	33
11	5	7	128	36
11	5	8	128	33
11	5	9	128	33
12	4	0	128	33
12	4	1	128	33
12	4	2	128	33
12	4	3	128	33
12	4	4	128	33
12	5	5	128	36
12	5	6	128	33
12	5	7	128	33
12	5	8	128	33
12	5	9	128	33
13	4	0	128	33
13	4	1	128	33
13	4	2	128	33
13	4	3	128	33
13	4	4	128	33

13	5	5	128	33
13	5	6	128	33
13	5	7	128	33
13	5	8	128	36
13	5	9	128	33
14	4	0	128	33
14	4	1	128	33
14	4	2	128	33
14	4	3	128	33
14	4	4	128	33
14	5	5	128	33
14	5	6	128	33
14	5	7	128	33
14	5	8	128	33
14	5	9	128	33
15	4	0	128	33
15	4	1	128	33
15	4	2	128	33
15	4	3	128	33
15	4	4	128	33
15	5	5	128	33
15	5	6	128	33
15	5	7	128	33
15	5	8	128	33
15	5	9	128	33
16	4	0	128	33
16	4	1	128	32
16	4	2	128	33
16	4	3	128	33

16	4	4	128	33
16	5	5	128	33
16	5	6	128	33
16	5	7	128	33
16	5	8	128	32
16	5	9	128	33
17	4	0	128	32
17	4	1	128	32
17	4	2	128	32
17	4	3	128	32
17	4	4	128	32
17	5	5	128	32
17	5	6	128	32
17	5	7	128	32
17	5	8	128	32
17	5	9	128	32
18	4	0	128	31
18	4	1	128	31
18	4	2	128	32
18	4	3	128	32
18	4	4	128	31
18	5	5	128	31
18	5	6	128	31
18	5	7	128	32
18	5	8	128	31
18	5	9	128	32
19	4	0	128	30
19	4	1	128	30
19	4	2	128	31

19	4	3	128	30
19	4	4	128	30
19	5	5	128	30
19	5	6	128	30
19	5	7	128	30
19	5	8	128	30
19	5	9	128	30
20	4	0	128	30
20	4	1	128	30
20	4	2	128	30
20	4	3	128	30
20	4	4	128	30
20	5	5	128	30
20	5	6	128	30
20	5	7	128	30
20	5	8	128	30
20	5	9	128	30
21	4	0	128	30
21	4	1	128	30
21	4	2	128	30
21	4	3	128	30
21	4	4	128	30
21	5	5	128	30
21	5	6	128	30
21	5	7	128	30
21	5	8	128	30
21	5	9	128	30
22	4	0	128	29
22	4	1	128	29

22	4	2	128	30
22	4	3	128	30
22	4	4	128	29
22	5	5	128	29
22	5	6	128	30
22	5	7	128	29
22	5	8	128	29
22	5	9	128	30
23	4	0	128	28
23	4	1	128	28
23	4	2	128	28
23	4	3	128	28
23	4	4	128	28
23	5	5	128	28
23	5	6	128	28
23	5	7	128	28
23	5	8	128	28
23	5	9	128	28
24	4	0	128	27
24	4	1	128	27
24	4	2	128	27
24	4	3	128	27
24	4	4	128	27
24	5	5	128	27
24	5	6	128	27
24	5	7	128	27
24	5	8	128	27
24	5	9	128	27
25	4	0	128	27

25	4	1	128	27
25	4	2	128	27
25	4	3	128	27
25	4	4	128	27
25	5	5	128	27
25	5	6	128	27
25	5	7	128	27
25	5	8	128	27
25	5	9	128	27
26	4	0	128	26
26	4	1	128	26
26	4	2	128	26
26	4	3	128	26
26	4	4	128	26
26	5	5	128	26
26	5	6	128	26
26	5	7	128	27
26	5	8	128	26
26	5	9	128	26

Tabla 3: Tabla de datos recibidos.

Donde la primera columna hace referencia al número de canal en el que nos encontramos, como vemos, hay diez veces ese número ya que son diez el número de tramas que se envían por cada canal.

La segunda columna hace referencia a la antena por la cual recibimos las tramas, si recibimos por la ANTENA 0 un 5 y si recibimos por la ANTENA 1 otro valor, normalmente un 4.

La tercera columna nos indica el número de tramas por cada canal (de 0 a 9) ya que envíanos diez tramas.

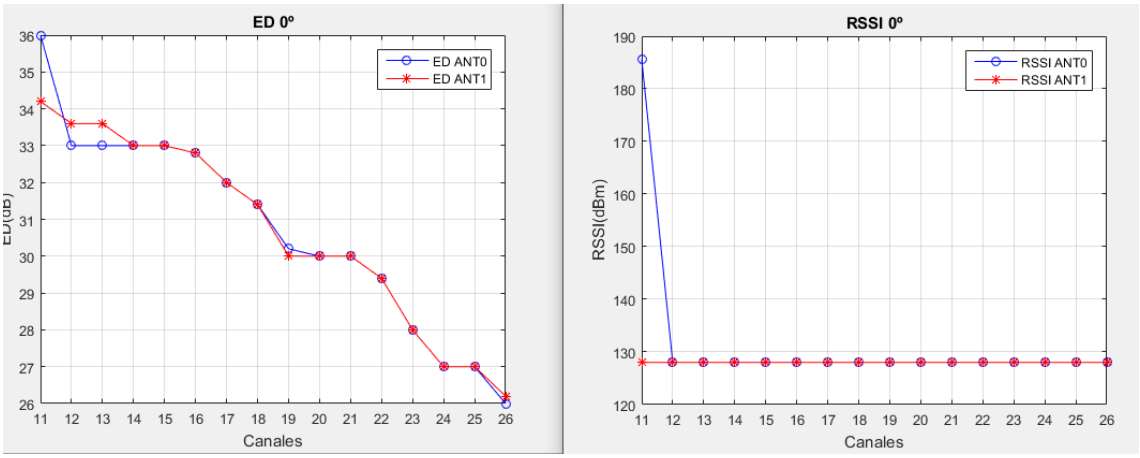
La cuarta y quinta columna corresponden al valor de RSSI y ED, respectivamente, para cada trama de cada canal.

Una vez que tenemos estos datos organizados, es momento de representarlos gráficamente con el fin de hacer un análisis de los resultados, tal y como abordaremos en el apartado siguiente.

3.4 Resultados

Se han realizado mediciones de los parámetros Deteción de energía (ED) e Indicador de fuerza de la señal recibida (RSSI) para distintos ángulos y analizando en gráficas la relación de la tramas enviadas por cada antena (ANT0 y ANT1) y para cada canal.

<u>0 grados:</u>



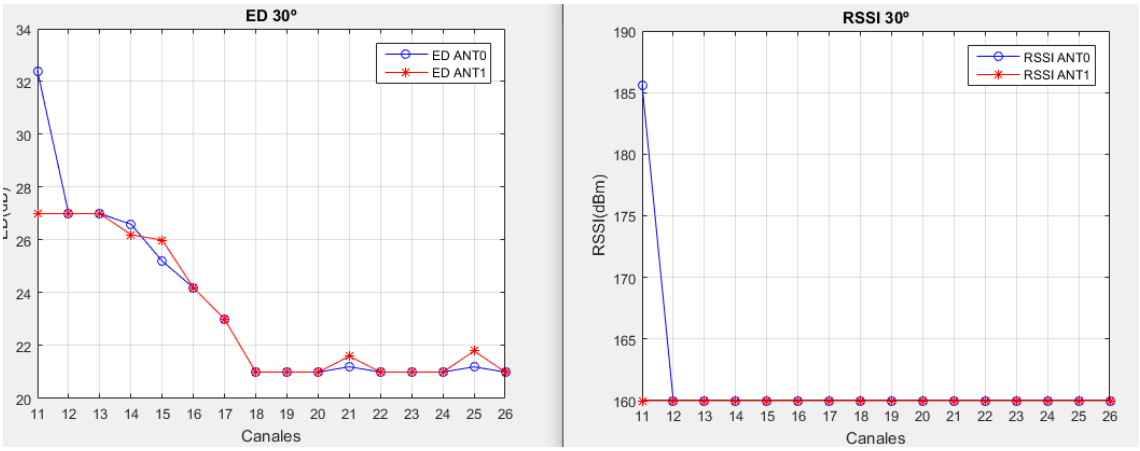
Como vemos en estas gráficas y veremos en las siguientes el valor del parámetro RSSI no se encuentra entre los valores teóricos estipulados. Además, es siempre constante. Esto se podría explicar ya que en nuestro programa MATLAB la funcion de lectura utiliza una precisión de 8 bits como mínimo (uint8), donde transformamos 8 bits en un vakir decimal. Por ello nuestro parámetro ED si que es correcto al tener 8 bits. Podemos decir que nuestro parámetro RSSI al tener cinco bits hay tres que aparentemente rellena con ceros.

Por ejemplo en esta gráfica vemos que tiene un valor de 128 (10000000), si lo pasamos a formato de 5 bits (10000) que nos da 16, un valor dentro del rango teórico que nos facilita el fabricante.

Que este valor sea constante a lo largo de todos los canales se podría explicar diciendo que la frecuencia cambia poco de un salto de canal a otro (5 MHz).

Para el valor del parámetro ED observamos que es prácticamente idéntico para ambas antenas y que este disminuye a la vez que aumenta la frecuencia (aumenta el canal). Que disminuya quiere decir que las mediciones tienen a lo que debería, pero que sean valores idénticos para ambas antenas hace pensar que la placa no está llevando a cabo la diversidad de antenas.

30 grados

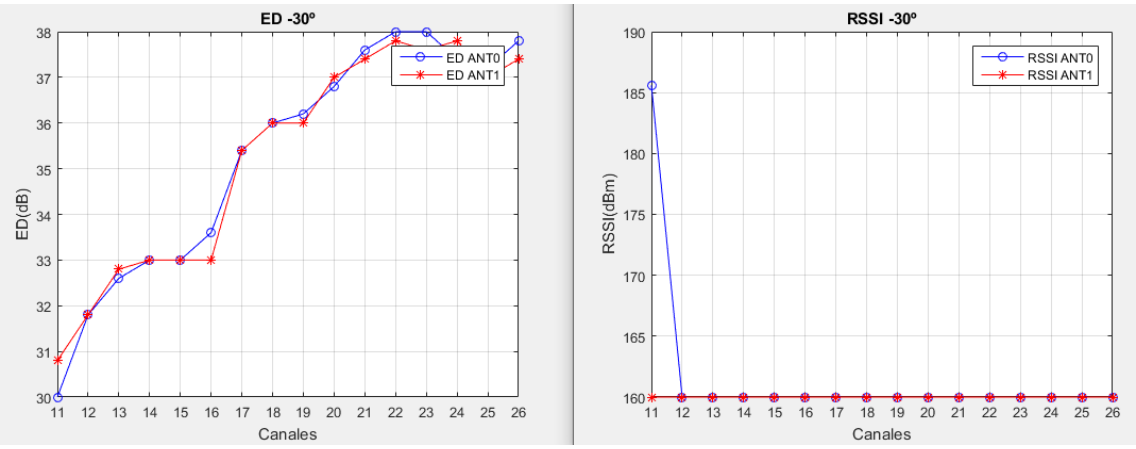


Para 30 grados el parámetro ED disminuye igualmente hasta que se estabiliza en el canal 18. Estas dos líneas no deberían coincidir en base a lo que sabemos de la teoría viendo los lóbulos de radiación de la LWA y el diagrama omnidireccional del módulo XBee.

La línea de la ANTENA 0 si estaría bien , pero según los diagramas de radiación la ANTENA 1 debería de tener una forma acampanada. Lo que me hace reforzar la idea de que la placa no lleve a cabo la diversidad de antenas.

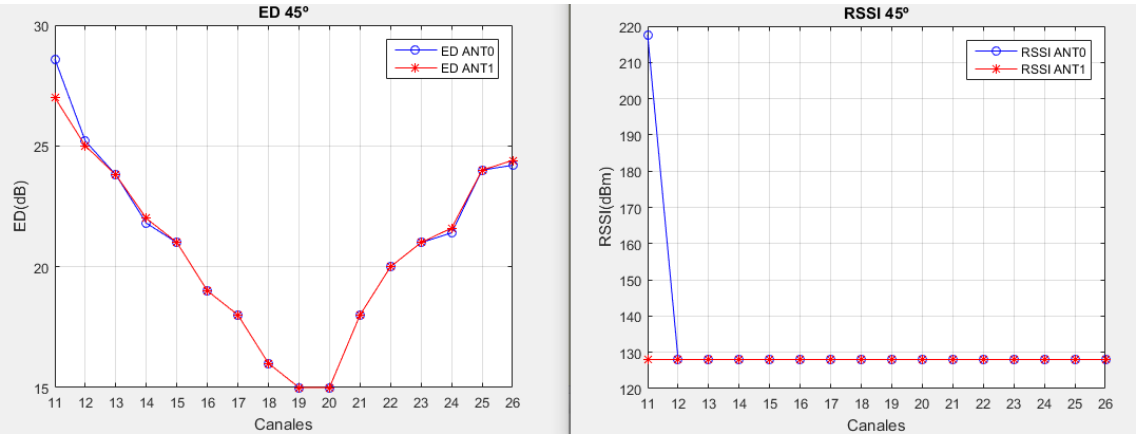
El parámetro RSSI se mantiene constante en 20 si seguimos la explicación anterior.

-30 grados



Evidentemente, para su ángulo negativo su comportamiento es el inverso para la mayoría de canales para el parámetro ED y no cambia para el RSSI.

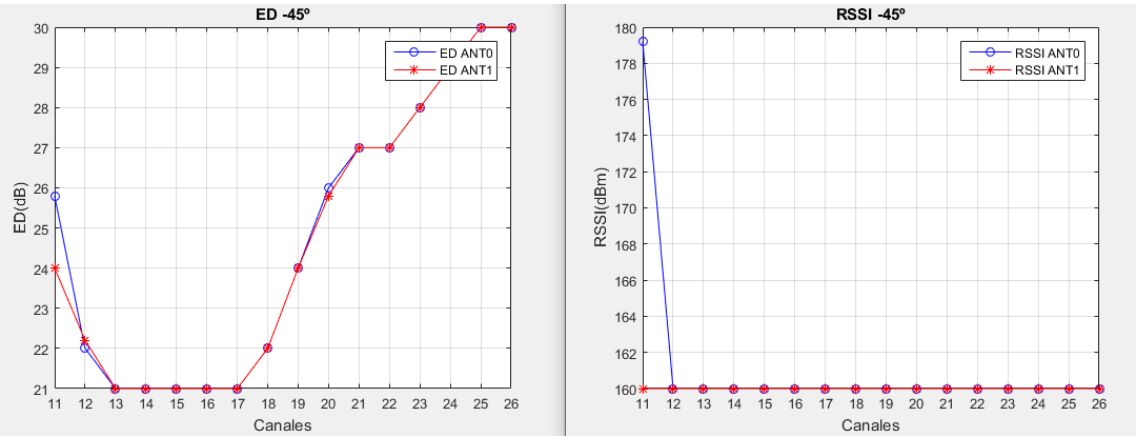
45 grados



Para 45 grados el nivel de RSSI es 16.

El ED cae en pico en la mitad de la comunicación para posteriormente volver ascender en valores. Esta media la obtenemos un poco corrupta ya que debería de ser un descenso más constante o plano.

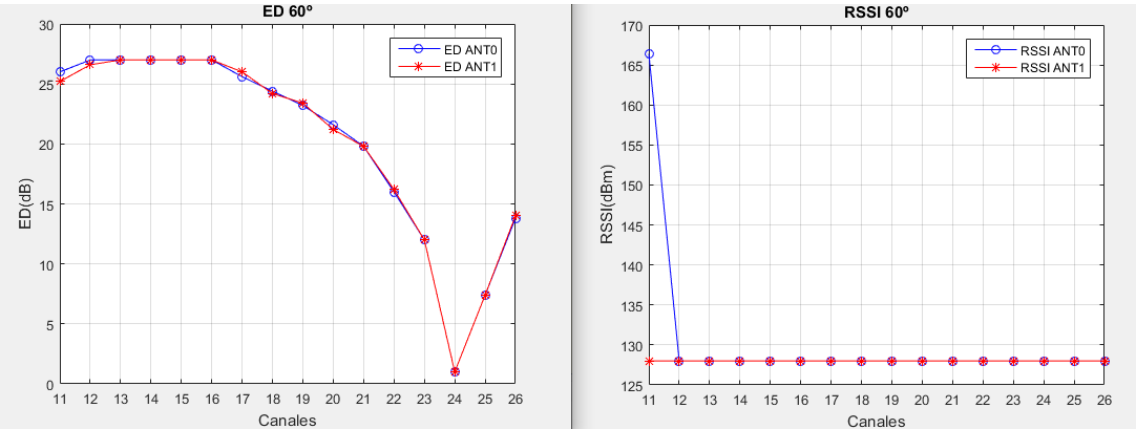
-45 grados



Para el ángulo inverso tenemos un comportamiento similar.

RSSI = 20.

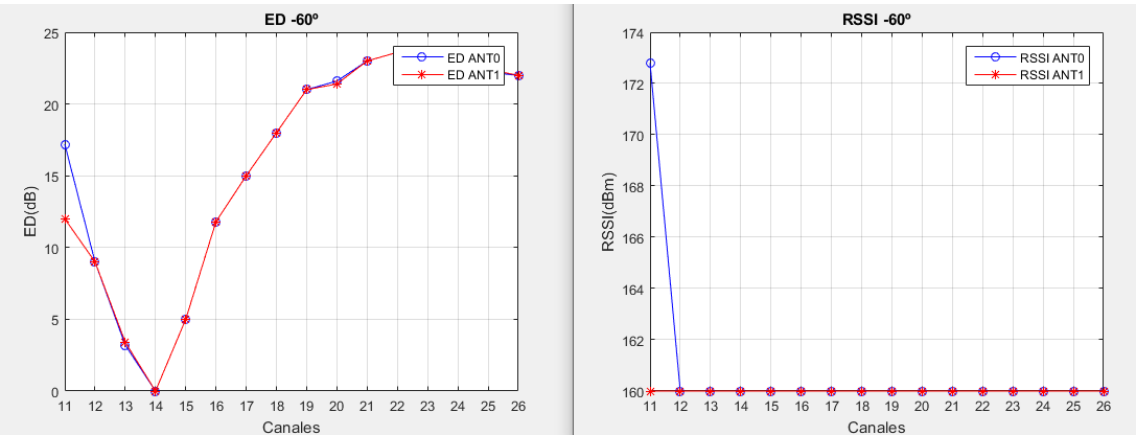
60 grados



RSSI = 16.

Para 60 grados el parámetro ED tiene un buen comportamiento, a pesar de que realiza un salto de los canales 24 al 26.

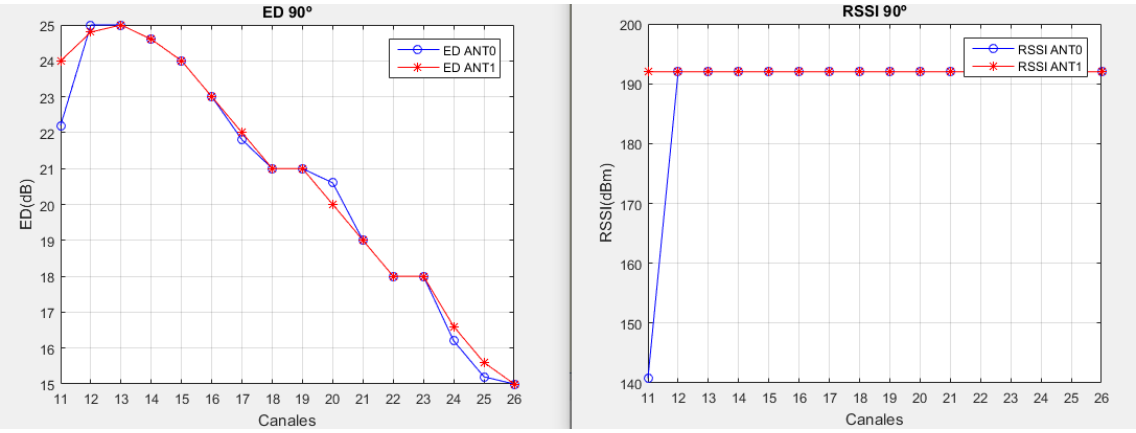
-60 grados



Comportamiento inverso que para 60 grados.

RSSI = 20.

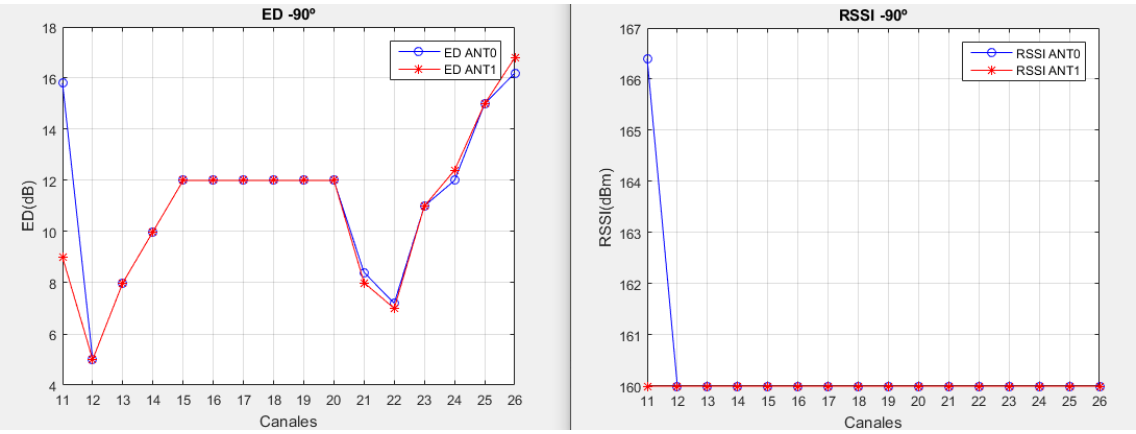
90 grados



RSSI = 24.

Comportamiento correcto para el parámetro ED.

-90 grados



RSSI = 20.

Para -90 grados el parámetro ED toma valores corruptos, la gráfica debería de seguir la misma progresión que para el ángulo positivo.

Capítulo Cuarto

Conclusiones y líneas futuras

4.1 Conclusiones

En este experimento se ha utilizado una antena LWA junto a una placa REB233SMAD y un módulo XBee que debíamos de programar para aprovechar las propiedades de una LWA radiando en ángulos distintos y demostrar así la eficiencia de este modelo en ZigBee.

Lo primero que se hizo fue programar la placa REB233SMAD para que pudiese cambiar de canal sin perder la conexión con nuestro módulo XBee. A la vez se debía de almacenar diversos parámetros que nos diesen información sobre el funcionamiento de la antena.

Por otro lado, en el entorno Matlab se realizó un programa encargado de recibir, almacenar y analizar la información.

A pesar de que el comportamiento de las graficas parezca correcto por lo general, parece ser que la placa no realiza la diversidad de antenas por lo que tenemos prácticamente la misma gráfica dos veces. El software está programado para que la realice, además el bit de selección de la antena si cambia, lo que aumentan mis pensamientos de la poca efectividad de la placa. Para el ED por ejemplo, para 30 grados debería de salir por un puerto muy alto y por otro muy bajo, pero salen muy parecido ambos. Se podría comprobar en la cámara anecoica la diversidad de antenas dejando una antena sin conectar (o conectada a una carga adaptada de 50 ohmios) y viendo los resultados.

Con todo ello podemos ver cómo al variar el ángulo de observación las distribuciones de ED respecto del canal varían sensiblemente, lo cual sienta las bases para usar esta información en la localización angular como hemos hecho en otros trabajos con Bluetooth, Wi-Fi y más recientemente con RFID.

Las características de las antenas 'leakywave' nos permite poder realizar experimentos de forma eficiente en aplicaciones inalámbricas como en nuestro caso, sistemas de localización.

4.2 Líneas futuras

Para este proyecto se ha demostrado la capacidad de utilizar la técnica de channel hoppig con una antena LWA para dar lugar a un sistema de localización de un receptor. Este sistema no ha podido quedar demostrado con garantías debido a la inestabilidad de nuestra placa REB233SMAD. Con el fin de mejorar estos resultados se ha mencionado la posibilidad de cambiar dicho componente problemático por otro tipo de tecnología. También desconectar un puerto de la antena para verificar si se realiza o no la diversidad de antenas.

La estabilidad necesaria que no hemos conseguido nos lleva a obtener resultados no esperados. De igual manera, este experimento nos ayuda a verificar que es posible implementar este sistema de localización utilizando el protocolo de comunicación ZigBee.

Bibliografía

[1]-IEEE 802.15.4 estándar, disponible en: http://www.ieee802.org/15/pub/TG4.html y http://www.standards.ieee.org/about/get/802/802.15html

[2]- ZigBee Alliance:, disponible en: http://www.zigbee.org

[3]- "What's The Difference Between IEEE 802.15.4 And ZigBee Wireless?", disponible: https://www.electronicdesign.com/unused/article/21796046/whats-the-differencebetween-ieee-802154-and-zigbee-wireless

[4]- "Smart Wireless Sensor Networks", Edited by Tan Yen Kheng, Ed. IntechOpen, Dec. 2010. DOI:10.5772/660, ISBN: 978-953-307-261-6 https://www.intechopen.com/books/smart-wireless-sensor-networks

[5]- "Smart Sensors Networks. Communication Technologies and Intelligent Applications", Series, Editor Fatos Xhafa, Academic Press Elsevier Ed. 2017, ISBN: 978-0-12-809859-2

https://www.sciencedirect.com/book/9780128098592/smart-sensors-networks

[6]- Boudour, G., Lecointre, A., Berthou, P., Dragomirecu, D., & Gayraud, T. (2007), "On designing sensor networks with smart antennas," IFAC Proceedings Volumes, 40(22), 349–356. doi:10.3182/20071107-3-fr-3907.00049 https://www.sciencedirect.com/science/article/pii/S147466701635176X

[7]- G. Giorgetti, A. Cidronali, S. K. S. Gupta and G. Manes, "Exploiting Low-Cost Directional Antennas in 2.4 GHz IEEE 802.15.4 Wireless Sensor Networks," 2007 European Conference on Wireless Technologies, Munich, 2007, pp. 217-220. doi: 10.1109/ECWT.2007.4403985

https://ieeexplore.ieee.org/document/4403985

[8]- A. Kucuk, K. & Kavak, "Connectivity Analysis for Wireless Sensor Networks with Antenna Array Integrated Central Node", A. Wireless Pers Commun (2013) 72: 1361. https://doi.org/10.1007/s11277-013-1083-2 https://link.springer.com/article/10.1007/s11277-013-1083-2

[9]- C. Lin and T. Kokkinos, "Cross-Layer Solutions for Extended-Range Wireless Sensor Networks," in IEEE Sensors Journal, vol. 13, no. 3, pp. 1044-1054, March 2013. doi: 10.1109/JSEN.2012.2234737

https://ieeexplore.ieee.org/document/6392194

[10]- T. N. Le, A. Pegatoquet, T. Le Huy, L. Lizzi and F. Ferrero, "Improving Energy Efficiency of Mobile WSN Using Reconfigurable Directional Antennas," in IEEE Communications Letters, vol. 20, no. 6, pp. 1243-1246, June 2016. doi: 10.1109/LCOMM.2016.2554544

https://ieeexplore.ieee.org/abstract/document/7452550

[11]- Wang, Qiu et al. “On Connectivity of Wireless Sensor Networks with Directional Antennas.” Sensors (Basel, Switzerland) vol. 17,1 134. 12 Jan. 2017, doi:10.3390/s17010134

https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5298707/

[12]- Dai, H.-N., Ng, K.-W., Li, M., & Wu, M.-Y. (2011), “An overview of using directional antennas in wireless networks. International Journal of Communication Systems,” 26(4), 413–448. doi:10.1002/dac.1348
https://onlinelibrary.wiley.com/doi/abs/10.1002/dac.1348

[13]- L. Catarinucci, S. Guglielmi, R. Colella and L. Tarricone, "Compact Switched-Beam Antennas Enabling Novel Power-Efficient Wireless Sensor Networks," in IEEE Sensors Journal, vol. 14, no. 9, pp. 3252-3259, Sept. 2014. doi: 10.1109/JSEN.2014.2326971

https://ieeexplore.ieee.org/document/6822538

[14]- Wang, R., Wang, B.-Z., Huang, W.-Y., & Ding, X. “Compact Reconfigurable Antenna with an Omnidirectional Pattern and Four Directional Patterns for Wireless Sensor Systems”. Sensors, 16(4), 552, 2016 doi:10.3390/s16040552
https://www.mdpi.com/1424-8220/16/4/552/html

[15]- A. A. Niyi Dihissou, A. Diallo, P. L. Thuc and R. Staraj, "Directive and Reconfigurable Antenna for Wireless Sensor Network to Improve Link Quality between Nodes," 2019 IEEE Sensors Applications Symposium (SAS), Sophia Antipolis, France, 2019, pp. 1-5. doi: 10.1109/SAS.2019.8706061
https://ieeexplore.ieee.org/document/8706061

[16]- S. A., P. Mathur, A. R. Chandran, N. Timmons, J. Morrison and S. Raman, "2.45 GHz Pattern Reconfigurable Antenna for Wireless Sensor Network applications," 2019 URSI Asia-Pacific Radio Science Conference (AP-RASC), New Delhi, India, 2019, pp. 1-4. doi: 10.23919/URSIAP-RASC.2019.8738453
https://ieeexplore.ieee.org/document/8738453

[17]- A. A. Oliner, “Leakage from higher modes on microstrip line with application to antennas,” Radio Sci., vol. 22, pp. 907–912, Nov. 1987.
https://ieeexplore.ieee.org/document/7774299

[18]- S. Matsuzawa, K. Sato, Y. Inoe and T. Nomura, "Steerable Composite Right/LeftHanded Leaky Wave Antenna for Automotive Radar Applications," 2006 European Microwave Conference, Manchester, 2006, pp. 1155-1158. https://ieeexplore.ieee.org/document/4058031

[19]- C. Rusch, J. Schäfer, H. Gulan, P. Pahl and T. Zwick, "Holographic mmW-Antennas With TE_0 and TM_0 Surface Wave Launchers for Frequency-Scanning FMCW-Radars," in IEEE Transactions on Antennas and Propagation, vol. 63, no. 4, pp. 1603-1613, April 2015.

https://ieeexplore.ieee.org/document/7031934

[20]- D. A. Schneider, M. Rösch, A. Tessmann and T. Zwick, "A Low-Loss W-Band FrequencyScanning Antenna for Wideband Multichannel Radar Applications," in IEEE Antennas and Wireless Propagation Letters, vol. 18, no. 4, pp. 806-810, April 2019. https://ieeexplore.ieee.org/document/8664098

[21]- M. Ettorre, R. Sauleau and L. Le Coq, "Multi-Beam Multi-Layer Leaky-Wave SIW Pillbox Antenna for Millimeter-Wave Applications," in IEEE Transactions on Antennas and Propagation, vol. 59, no. 4, pp. 1093-1100, April 2011. https://ieeexplore.ieee.org/document/5705554

[22]- Y. J. Cheng, W. Hong, K. Wu and Y. Fan, "Millimeter-Wave Substrate Integrated Waveguide Long Slot Leaky-Wave Antennas and Two-Dimensional Multibeam Applications," in IEEE Transactions on Antennas and Propagation, vol. 59, no. 1, pp. 40- 47, Jan. 2011.

https://ieeexplore.ieee.org/document/5617222

[23]- A. Hakkarainen, J. Werner, M. Costa, K. Leppanen and M. Valkama, "High-Efficiency Device Localization in 5G Ultra-Dense Networks: Prospects and Enabling Technologies," 2015 IEEE 82nd Vehicular Technology Conference (VTC2015-Fall), Boston, MA, 2015, pp. 1-5.

https://ieeexplore.ieee.org/document/7390965

[24]-M. Steeg, B. Khani, V. Rymanov and A. Stöhr, "Novel 50–70 GHz compact PCB leaky-wave antenna with high broadside efficiency and low return loss," 2016 41st International Conference on Infrared, Millimeter, and Terahertz waves (IRMMW-THz), Copenhagen, 2016, pp. 1-2.

https://ieeexplore.ieee.org/document/7758489

[25]-B. Husain, M. Steeg and A. Stöhr, "Estimating direction-of-arrival in a 5G hot-spot scenario using a 60 GHz leaky-wave antenna," 2017 IEEE International Conference on Microwaves, Antennas, Communications and Electronic Systems (COMCAS), Tel-Aviv, 2017, pp. 1-4.

https://ieeexplore.ieee.org/document/8244845

[26]- Steeg M., Szczęsny M., Stöhr A. (2019) Multiuser 5G Hot-Spots Based on 60 GHz Leaky-Wave Antennas and WDM Radio-over-Fiber. In: Electronic Navigation Research Institute (eds) Air Traffic Management and Systems III. EIWAC 2017. Lecture Notes in Electrical Engineering, vol 555. Springer, Singapore, 2019. https://link.springer.com/chapter/10.1007/978-981-13-7086-1_16

[27]- A. J. Martinez-Ros, J. L. Gómez-Tornero, and G. Goussetis, “Frequency scanning leaky wave antenna for positioning and identification of RFID tags,” in Proc. IEEE Int. Conf. RFID-Technol. Appl. (RFID-TA), Sep. 2011, pp. 451–456.

[28]- M. Poveda-García, D. Cañete-Rebenaque, and J.L. Gómez-Tornero, “Frequency-Scanned Monopulse Pattern Synthesis Using Leaky-Wave Antennas for Enhanced Power-Based Direction-of-Arrival Estimation”, *IEEE Transactions on Antennas and Propagation*, vol. 67, no. 11, pp. 7071-7086, Nov. 2019.

[29]- A. Gil-Martinez, M. Poveda-Garcia, and J. L. Gómez-Tornero, “Direct synthesis of frequency-scanned monopulse half-width microstrip leaky-wave antennas,” in Proc. Eur. Conf. Antennas Propag. (EuCAP 2020), pp. 1–4, 2020.

[30]- M. Poveda-García, E. Andreu-García, J. García-Fernández, D. Cañete-Rebenaque, J.L. Gómez-Tornero, “Frequency-scanned leaky-wave antenna topologies for two-dimensional direction of arrival estimation in IoT wireless networks,” in Proc. 15th Eur. Conf. Antennas Propag. (EuCAP 2021), Mar, 2021, pp. 1–4.

[31]- M. Poveda-García, and J.L. Gómez-Tornero, “Ambiguity Resolution in Amplitude-Monopulse Systems using Broad-Beam Patterns”, *IEEE Antennas and Wireless Propagation Letters*, vol. 20, no. 4, pp. 503-507, April 2021.

[32]- R. Guzmán-Quirós, A.J. Martínez-Salas, J. L. Gómez-Tornero, and J. García-Haro, "Integration of Directional Antennas in an RSS Fingerprinting-Based Indoor Localization System," in *Sensors, Special Issue Sensors for Indoor Mapping and Navigation*, vol. 16, no. 4, pp. 1-23, December 2015.

[33]- J.L. Gómez-Tornero, D. Cañete-Rebenaque, J.A. López-Pastor, and A. Martínez-Salas “Hybrid Analog-Digital Processing System for Amplitude-Monopulse RSSI-based MiMo WiFi Direction-of-Arrival Estimation”, *IEEE Journal of Selected Topics in Signal Processing. Special Issue on Hybrid Analog - Digital Signal Processing for Hardware-Efficient Large Scale Antenna Arrays*, vol.12, no.3, pp. 529-540, June 2018

[34]- J.A. López-Pastor, A. Gómez-Alcaraz, D. Cañete-Rebenaque, A.S. Martínez-Sala, and J.L. Gómez-Tornero, “Near-Field Monopulse DoA Estimation for Angle-Sensitive Proximity WiFi Readers”, *IEEE Access Journal Special Section on Emerging Trends, Issues and Challenges for Array Signal Processing and Its Applications in Smart City*, vol. 7, no. 1, pp. 88450- 88460, June. 2019.

[35]- J.A. López-Pastor, P. Arqués-Lara, J.J Franco-Peñaranda, A.J. García-Sánchez and J.L. Gómez-Tornero, “Wi-Fi RTT-based active monopulse RADAR for single access point localization”, *IEEE Access*, vol. 9, pp. 34755-34766, Feb. 2021.

[36]- A. Gil-Martínez, Y. El Gholb, M. Poveda-García, J.L. Gómez-Tornero, and N. El Amrani El Idrissi, “An array of leaky wave antennas for indoor smart wireless access point applications,” in Proc. 7th International Conference on Wireless Networks and Mobile Communications (WINCOM'19), Fez, Morocco, Nov. 2019.

[37]- A. Gil-Martínez, M. Poveda-García, and J.L. Gómez-Tornero, “Wi-Fi Direction Finding with Frequency-Scanned Antenna and Channel-Hopping Scheme” , IEEE Sensor Journal, in press 2021.

[38]- M. Poveda-García, A. Gómez-Alcaraz, A. Gil-Martínez, D. Cañete-Rebenaque, A. S. Martinez-Sala and J. L. Gómez-Tornero, "Frequency-Scanned Active Monopulse Radar based on Bluetooth Low Energy Devices using an Array of Two Planar Leaky-Wave Antennas," 2019 IEEE-APS Topical Conference on Antennas and Propagation in Wireless Communications (APWC), Granada, Spain, 2019, pp. 390-393. https://ieeexplore.ieee.org/document/8870454

[39]- M. Poveda-García, A. Gómez-Alcaraz, D. Cañete-Rebenaque A. S. Martínez-Sala, J.L. Gómez-Tornero, “RSSI-based Direction-of-Departure Finding of Bluetooth Devices using Frequency-Scanned Monopulse Leaky-Wave Antenna Arrays”, IEEE Access Journal Special Section on Emerging Trends, Issues and Challenges for Array Signal Processing and Its Applications in Smart City, under review.

[40]- M. Poveda-García, A. Gómez-Alcaraz, D. Cañete-Rebenaque A. S. Martínez-Sala, J.L. Gómez-Tornero, “RSSI-Based Direction-of-Departure Estimation in Bluetooth Low Energy Using an Array of Frequency-Steered Leaky-Wave Antennas”, *IEEE Access Journal Special Section on Emerging Trends, Issues and Challenges for Array Signal Processing and Its Applications in Smart City*, vol. 8, pp. 9380-9394, Jan. 2020.

[41]- M. Poveda-Garcia, A. G. Martinez and J. L. Gomez-Tornero, "Frequency-scanned focused leaky-wave antennas for direction-of-arrival detection in proximity BLE sensing applications," in Proc. 14th Eur. Conf. Antennas Propag. (EuCAP 2014), Mar. 2020, pp. 1–4.

[42]- M. Poveda-García, E. Andreu-García, J. García-Fernández, D. Cañete-Rebenaque, J.L. Gómez-Tornero, “Frequency-scanned leaky-wave antenna topologies for two-dimensional direction of arrival estimation in IoT wireless networks,” in Proc. 15th Eur. Conf. Antennas Propag. (EuCAP 2021), Mar, 2021, pp. 1–4.

[43]- A. Gil-Martínez, M. Poveda-García, J. García-Fernández, D. Cañete-Rebenaque, and J.L. Gómez-Tornero,,"Design of passive beam-scanning monopulse leaky-wave antennas for direction finding in UHF RFID", in Proc. IEEE Int. Conf. RFID-Technol. Appl. (RFID-TA), Oct. 2021, pp. 1–4

[44]- A. Gil-Martínez, M. Poveda-García, D. Cañete-Rebenaque, and J.L. Gómez-Tornero, “Frequency-scanned monopulse antenna for RSSI-based direction finding of UHF RFID tags”, *IEEE Antennas and Wireless Propagation Letters*, in press Jul. 2021.

[45]- J.L. Gómez-Tornero, M. Poveda-García, R. Guzmán-Quirós, J. C. Sánchez-Arnause, “Design of Ku-Band Wireless Power Transfer System to Empower Light Drones,” 2016 IEEE Wireless Power Transfer Conference (WPTC2016), Aveiro, Portugal, 5-6 May. 2016.

[46]- M. Poveda-García, J. Oliva-Sánchez, R. Sanchez-Iborra, D. Cañete-Rebenaque, J.L. Gómez-Tornero, “Dynamic Wireless Power Transfer for Cost-Effective Wireless Sensor Networks using Frequency-Scanned Beaming”, *IEEE Access Journal Special Section on Wirelessly Powered Networks*, vol. 7, pp. 8081-8094, Jan. 2019.

[47]- M. Poveda-García and J.L. Gómez-Tornero, “Beaming Efficiency of 1-D Frequency-Scanned Based Radiative WPT System for Wireless Sensor Networks,” 2019 IEEE Wireless Power Transfer Conference (WPTC 2019), 17-21 Jun. 2019.

[48]- “Time Slotted Channel Hopping”, available online in https://en.wikipedia.org/wiki/Time_Slotted_Channel_Hopping

[49]- L. Varga, M. Heusse, R. Guizzetti and A. Duda, "Why is frequency channel diversity so beneficial in wireless sensor networks?," 2016 Wireless Days (WD), Toulouse, 2016, pp. 13. doi: 10.1109/WD.2016.7461497 https://ieeexplore.ieee.org/document/7461497

[50]- Jae-Ho Lee, Sung-Jae Kho, Uk-Jin Jang, Youn-Sik Hong, “An Improved Method of Avoiding RF Congestion in Indoor Environments,” Procedia Engineering, Vol. 154, pp. 223-228, 2016, https://doi.org/10.1016/j.proeng.2016.07.455. https://www.sciencedirect.com/science/article/pii/S1877705816318446

[51]- “A method of improving PRR for WiFi interference avoidance in ZigBee networks in indoor environments,” International Journal of Advanced Intelligence Paradigms (IJAIP) 2018 Vol.11 No.1/2 pp.176. https://www.inderscience.com/info/inarticle.php?artid=92953

[52]- Fernández de Gorostiza, E.; Berzosa, J.; Mabe, J.; Cortiñas, R., “A Method for Dynamically Selecting the Best Frequency Hopping Technique in Industrial Wireless Sensor Network Applications,” Sensors 2018, 18, 657. https://doi.org/10.3390/s18020657

https://www.mdpi.com/1424-8220/18/2/657

[53]- A. R. Vedral, T. Kruse and J. F. Wollert, "Development and Performance Evaluation of an Antenna Diversity Module for Industrial Communication based on IEEE 802.15.4," 2007 IEEE Conference on Emerging Technologies and Factory Automation (EFTA 2007), Patras, 2007, pp. 177-186. doi: 10.1109/EFTA.2007.4416769 https://ieeexplore.ieee.org/document/4416769

[54]- Husain Rehmani Mubashir, Abderrezak Rachedi, Stéphane Lohier, Thierry Alves, Benoit Poussot. “Intelligent Antenna Selection Decision in IEEE 802.15.4 Wireless Sensor Networks: An Experimental Analysis,” Computers and Electrical Engineering, Elsevier, 2014, 40 (2), pp.443-455. https://www.sciencedirect.com/science/article/pii/S0045790613003030

[55]- A. A. Bavarva and P. V. Jani, "Improve the channel performance of Wireless Multimedia Sensor Network using MIMO properties," 2015 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Kochi, 2015, pp. 277-282. doi: 10.1109/ICACCI.2015.7275621 https://ieeexplore.ieee.org/document/7275621

[56]- Willig, A. (2008). How to exploit spatial diversity in wireless industrial networks. Annual Reviews in Control, 32(1), 49–57. doi:10.1016/j.arcontrol.2008.03.005 https://www.sciencedirect.com/science/article/pii/S1474667016351291

[57]- Descripción del producto MATLAB:

https://es.mathworks.com/help/matlab/learn_matlab/product-description.html

ANEXO A

Toma de muestras

Con este script desarrollado en entorno MATLAB establezco la comunicación y activo la recepción de datos para después organizar todos esos datos en matrices que faciliten el análisis y su representación gráfica. Este script se ejecuta para medir los parámetros en cada ángulo deseado.

```
s = serial('COM4','Baudrate',9600) % Configuración y creación de un
                                   instrumento objeto
fopen(s) % Comenzar comunicación
fwrite(s,'CN') % Escribimos un ENTER para activar la recepción de los
                 datos
b=zeros(50,16);% (parámetros x tramas),(canales)
i=1;
while(1)
a=fread(s,[1 50],'char')
b(:,i)=a;
i=i+1;
end

%% Recoloca
c=b';
r=zeros(160,5);%(Tramas),(Parámetros)

for  i = 1:16
    for  j = 1:50
        if j<=5
            r(1+10*(i-1),j) = c(i,j);
        end
        if j>5 && j<=10
           r(2+10*(i-1),j-5) = c(i,j);
        end
          if j>10 && j<=15
            r(3+10*(i-1),j-10) = c(i,j);
          end
              if j>15 && j<=20
            r(4+10*(i-1),j-15) = c(i,j);
              end
              if j>20 && j<=25
            r(5+10*(i-1),j-20) = c(i,j);
              end
              if j>25 && j<=30
            r(6+10*(i-1),j-25) = c(i,j);
              end
              if j>30 && j<=35
            r(7+10*(i-1),j-30) = c(i,j);
              end
              if j>35 && j<=40
            r(8+10*(i-1),j-35) = c(i,j);
              end
             if j>40 && j<=45
```

```
            r(9+10*(i-1),j-40) = c(i,j);
             end
               if j>45 && j<=50
            r(10+10*(i-1),j-45) = c(i,j);
              end

    end
end

%% DibujaRSSI
contador= 5;
channel = 11:26;
for j=1:16
    RSSI_ANT0=0;
    RSSI_ANT1=0;
    for  i=1:160
        if r(i,1)== channel(j)
            if r(i,2)==5;
            RSSI_ANT1= RSSI_ANT1 + r(i,4);
            else
                 RSSI_ANT0= RSSI_ANT0 + r(i,4);
            end
        end
    end
    RSSI_ANT0= RSSI_ANT0/contador;
    RSSIBuena(1,j)=RSSI_ANT0;
    RSSI_ANT1= RSSI_ANT1/contador;
    RSSIBuena(2,j)=RSSI_ANT1;
end
PTO1=RSSIBuena(1,:);
PTO2 = RSSIBuena(2,:);
figure;
plot(channel,PTO1,'-ob');
hold on;
plot(channel,PTO2,'-*r');
legend('RSSI ANT0','RSSI ANT1')
title('RSSI');
xlabel('Canales');
ylabel('RSSI(dBm)');
grid on;
xlim([11 26]);
set(gca,'Xtick',[11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26]);

%% DibujaED
contador= 5;
channel = 11:26;
for j=1:16
    ED_ANT0=0;
    ED_ANT1=0;
    for  i=1:160
        if r(i,1)== channel(j)
            if r(i,2)==5;
            ED_ANT1= ED_ANT1 + r(i,4);
            else
                 ED_ANT0= ED_ANT0 + r(i,4);
            end
        end
    end
    ED_ANT0= ED_ANT0/contador;
    EDBuena(1,j)=ED_ANT0;
    ED_ANT1= ED_ANT1/contador;
```

```
    EDBuena(2,j)=ED_ANT1;
end
PTO1=EDBuena(1,:);
PTO2 = EDBuena(2,:);
figure;
plot(channel,PTO1,'-ob');
hold on;
plot(channel,PTO2,'-*r');
legend(ED ANT0',ED ANT1')
title('ED');
xlabel('Canales');
ylabel('ED(dB)');
grid on;
%ylim([170 200]);
xlim([11 26]);
set(gca,'Xtick',[11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26]);
```

ANEXO B

Programar Módulo XBee

Este script ha sido desarrollado con la herramienta Spider con lenguaje de programación Python donde programo el módulo XBee. Cabe destacar los imports de los paquetes necesarios como *'utils'* ya que hay ciertas funciones que no se les puede pasar un string sino un bit array. Con este código el módulo XBee enviará al Coordinador 10 tramas en dos paquetes de 5 recibiendo uno por cada antena.

```
#### IMPORTS ####

from digi.xbee.devices import XBeeDevice,ZigBeeDevice

from digi.xbee.util import utils

import time

#### DEFINE PARAMETERS ####

PORT = "COM3"

BAUD_RATE = 9600


PARAM_NODE_ID = "NI"

PARAM_PAN_ID = "ID"

PARAM_DEST_ADDRESS_H = "DH"

PARAM_DEST_ADDRESS_L = "DL"

PARAM_CHANNEL = "CH"

PARAM_SCAN_CHANNEL = "SC"

PARAM_RSSI = "DB"

PARAM_VALUE_NODE_ID = "Router"  #Nombre de este dispositivo que
                                reconocerá la variable REMOTE_NODE_ID del otro
                                dispositvo

PARAM_VALUE_PAN_ID = utils.hex_string_to_bytes("CAFE")

PARAM_VALUE_DEST_ADDRESS_H = utils.hex_string_to_bytes("00")

PARAM_VALUE_DEST_ADDRESS_L = utils.hex_string_to_bytes("FFFF")
```

```
PARAM_VALUE_CHANNEL = []

CHANNELS = "0B0C0D0E0F10111213141516171819 1A"

for i in utils.hex_string_to_bytes(CHANNELS):

    PARAM_VALUE_CHANNEL.append(i)


PARAM_VALUE_SCAN_CHANNEL = utils.hex_string_to_bytes("3FFF")


DATA_TO_SEND = "Hello XBee!"

REMOTE_NODE_ID = "Coordinator" #Name to receptor device


#### MAIN FUNCTION ####

def main():

    print(" +---------------------------------------------+")

    print(" | XBee Python Library Set/Get parameters Sample |")

    print(" +---------------------------------------------+\n")


        device = XBeeDevice(PORT, BAUD_RATE)

    try:

        device.open() #Opens the communication with the XBee and loads information about it.

        # Get parameters.

        device.set_pan_id(PARAM_VALUE_PAN_ID)

        device.set_parameter(PARAM_NODE_ID,
bytearray(PARAM_VALUE_NODE_ID, 'utf8'))

        device.set_parameter(PARAM_PAN_ID, PARAM_VALUE_PAN_ID)

        device.set_parameter(PARAM_CHANNEL,
utils.int_to_bytes(PARAM_VALUE_CHANNEL[0]))

        device.execute_command("AC")
```

```
print("Node ID:                    %s" %
device.get_parameter(PARAM_NODE_ID).decode())

        print("PAN ID:                     %s" %
utils.hex_to_string(device.get_parameter(PARAM_PAN_ID)))

        print("Destination address high:    %s" %
utils.hex_to_string(device.get_parameter(PARAM_DEST_ADDRESS_H)))

        print("Destination address low:     %s" %
utils.hex_to_string(device.get_parameter(PARAM_DEST_ADDRESS_L)))

        print("Channel:                    %s" %
utils.hex_to_string(device.get_parameter(PARAM_CHANNEL)))

        print("Scan Channel:               %s" %
utils.hex_to_string(device.get_parameter(PARAM_SCAN_CHANNEL)))

        print(device.get_role())

        print("AI:                         %s" %
utils.bytes_to_int(device.get_parameter("AI")))

    print("")

print("Cached parameters\n" + "-" * 50)

        print("64-bit address:   %s" % device.get_64bit_addr())

        print("16-bit address:   %s" % device.get_16bit_addr())

        print("Node Identifier:  %s" % device.get_node_id())

        print("Firmware version: %s" %
utils.hex_to_string(device.get_firmware_version()))

        print("Hardware version: %s" %
device.get_hardware_version().description)

        print("")

        print("All parameters were set correctly!")

time.sleep(4)



        device.flush_queues()



        ch = 1

        seq_aux = 0 # 5 para antena1 y 5 para antena2 (cuando llegue a 10)
```

```
while True:
            seq_aux = seq_aux + 1
            print("Sending data %s" % (DATA_TO_SEND))
            device.send_data_broadcast(DATA_TO_SEND)
            print("Success %s" % (seq_aux))


            time.sleep(0.2)


            if seq_aux == 10:
                time.sleep(1)
                print("10 tramas enviadas - debe recibirse por Antena2")


            if seq_aux == 20:
                time.sleep(1)
                print("20 tramas enviadas - debe cambiarse de Canal")
                seq_aux = 0


                device.set_parameter(PARAM_CHANNEL,
utils.int_to_bytes(PARAM_VALUE_CHANNEL[ch]))
                #device.execute_command("AC")
                print("Change Channel: %s" %  ch)
                ch = ch + 1


                if ch == 17:
                    exit(1)
                time.sleep(4)
    finally:
        if device is not None and device.is_open():
            device.close()
if __name__ == '__main__':
    main()
```

ANEXO C

Software placa ATMEL REB233SMAD

En este script del programa Microchip Studio se ha desarrollado el software con el cuál programaremos nuestra placa REB233SMAD. En términos generales, se realizan funciones como el cambio de canal de la forma que queremos (#11-#26), seleccionar el número de tramas que enviaremos en total y por cada antena (5+5=10) y limpieza de buffer a casa paso para evitar que se satura nuestra placa además de seleccionar los parámetros que vamos a querer medir.

Este script lo ejecutaremos cada vez que queramos modificar algo de lo mencionado anteriormente y gracias a nuestra Herramienta de Programación Atmel-ICE Kit introduciremos dicho script en la placa de la manera que explico en el ANEXO D.

```
#include <asf.h>
#include <string.h>
#include <stdio.h>
#include "stdio/stdio_serial/stdio_serial.h"
#include "tal.h"
#include "tal_helper.h"
#include "tal_internal.h"
#include "common_sw_timer.h"
#include "ieee_const.h"
#include "tal_constants.h"
#include "compiler.h"
#include "usart_serial.h"
#include "conf_sio2host.h"

// MACROS

#define DEBOUNCE_DELAY_MS (200)
       //se usa para definir el retraso requerido para verificar el rebote
       pulsador
//#define DEFAULT_CHANNEL CCPU_ENDIAN_TO_LE16(11)
       //se utiliza para definir el canal definido por IEEE 802.15.4
#define FRAME_CONTROL_FIELD CCPU_ENDIAN_TO_LE16(0x8001)//se  utiliza para definir
el campo Frame Control definido por IEEE802.15.4
#define  DEFAULT_PAN_ID  CCPU_ENDIAN_TO_LE16(0xCAFE)//se utiliza para definir el
PAN ID de la red, es decir, el identificador de la red
#define  SOURCE_ADDRESS  CCPU_ENDIAN_TO_LE16(0xBABE)//se  usa  para  definir  la
dirección del nodo
#define CHANNEL_AUX (1)
```

```
//Estructura de la trama que se transmite
typedef struct
{
	uint8_t phr; /**< PHY header (frame length field). */
	uint16_t fcf; /**< Frame control field */
	uint8_t seq; /**< Frame sequence number. */
	uint16_t span; /**< source PAN identifier */
	uint16_t saddr; /**< source address */
	char data[5]; /**< Frame payload */
	uint16_t crc; /**< CRC16 field of the frame. */
} app_frame_t;

static void app_alert(void);
void app_task(void);
void usart_inizializate (void);
void conf_channel(void);

// Inicializacion de variables
bool tx_ready_flag = 1;           //estado de tx del paquete
static uint8_t seq_num = 0; //numero de secuencia de la trama
frame_info_t tx_frame_info; // contiene la estructura de la trama de tx utilizada
                            por la capa TAL
app_frame_t tx_frame, rx_frame; //contienen datos de la trama tx y rx
uint8_t msg[5] = "Hello"; // contiene los datos que se van a transmitir
int num_frames = 0;
uint8_t *received_byte;
uint8_t *rssi;

uint8_t *ant_sel;
uint8_t *pmu;
uint8_t DEFAULT_CHANNEL = CCPU_ENDIAN_TO_LE16(11);
uint8_t lqi;
uint8_t ed;
//static uint8_t stx = 2;
//static uint8_t etx = 3;
int ch = 0;

// FUNCION PRINCIPAL
// Inicializamos el vector IRQ, Reloj, Plac, Temporizador Software y TAL Layer.
int main(void)
{
	uint8_t temp;
	irq_initialize_vectors();
	board_init();

	sysclk_init();

	usart_inizializate();
	/* Inicializamos la placa.
	* El archivo conf_board.h contiene la configuración inicial.
	*/
	LED_On(LED0);
	delay_ms(2000);
	LED_Off(LED0);

	/* Inicializamos el temporizador software.
	* The conf_hw_timer.h,conf_common_sw_timer.h and app_config.h file
	* contains the configuration for the timer initialization.
	*/
	sw_timer_init();

	/* Inicializamos capa TAL */
```

```
	if (tal_init() != MAC_SUCCESS) {
		/* si algo sale mal durante la inicialización */
		app_alert();
	}

//	/* Configuramos Enable/Disable la Diversidad de Antenas*/
//	trx_bit_write(SR_ANT_CTRL, ANT_CTRL_1);
	//Controlamos la antena a elegir -> FUNCIONA
//	trx_bit_write(SR_PDT_THRES, THRES_ANT_DIV_ENABLE);		//
//	trx_bit_write(SR_ANT_DIV_EN, ANT_DIV_DISABLE);			//Algoritmo
Ant Div DISABLE
//	//trx_bit_write(SR_ANT_EXT_SW_EN, ANT_EXT_SW_DISABLE);//Switch	entre
antenas Ant Div DISABLE - NO HACE FALTA SEGUN EL PDF

	//Otra manera de configurar la Diversidad de Antenas (tal_helper.c)
	//tal_ant_div_config(ANT_DIV_DISABLE, ANT_CTRL_0);

	/* Configuración del canal por defecto */
	temp = DEFAULT_CHANNEL;
	tal_pib_set(phyCurrentChannel, (pib_value_t *)&temp);

	/* Channel page default configuration*/
	temp = TAL_CURRENT_PAGE_DEFAULT;
	tal_pib_set(phyCurrentPage, (pib_value_t *)&temp);

	/* Tx power default configurations */
	temp = TAL_TRANSMIT_POWER_DEFAULT;
	tal_pib_set(phyTransmitPower, (pib_value_t *)&temp);

	/* Sets transceiver state */
	set_trx_state(CMD_RX_ON);
	pal_global_irq_enable();

	while(1)
	{
		pal_task(); /* Handle platform specific tasks, like serial
			       interface */
		tal_task(); /* Handle transceiver specific tasks */
		app_task(); /* Application task */
	}
}

void usart_inizializate (void){
	// Debe de estar inicializado sysclk_init();

	static usart_serial_options_t usart_options = {
		.baudrate = USART_HOST_BAUDRATE,
		.charlength = USART_HOST_CHAR_LENGTH,
		.paritytype = USART_HOST_PARITY,
		.stopbits = USART_HOST_STOP_BITS
	};

	usart_serial_init(USART_HOST, &usart_options);
	//subsección serial_basic_use_case_usage_flow Flujo de trabajo
	// Esperar la recepción de un carácter:
	usart_serial_getchar(USART_HOST, &received_byte); // Pulsa enter para
							  comenzar
	//Enviar el carácter de vuelta:
	//usart_serial_putchar(USART_HOST, received_byte);
	//usart_serial_write_packet(USART_HOST, "\n App V1.0 Diversidad de Antenas
	y RSSI", strlen("\n App V1.0 Diversidad de Antenas y RSSI"));
	// De momento el printf no funciona
```

```
	//usart_serial_write_packet(USART_HOST, 2, strlen(2));
}

void conf_channel(void){

	if (DEFAULT_CHANNEL > CCPU_ENDIAN_TO_LE16(26))
	{
		DEFAULT_CHANNEL = CCPU_ENDIAN_TO_LE16(11);
		usart_serial_write_packet(USART_HOST,  "\n  Ch  hop",  strlen("\n  Ch
hop"));

	}

}


void app_task(void)
{

		if (!ioport_get_pin_level(GPIO_PUSH_BUTTON_0)) // Si pulsamos el
                                                       botón
		{
			delay_ms(DEBOUNCE_DELAY_MS); // Espera para evitar el rebote
                                         del pulsador
			if (!ioport_get_pin_level(GPIO_PUSH_BUTTON_0)) // Si sigue
                                                           pulsado la
                                                           función  carga
                                                           los  datos  que
                                                           deben ser tx
			{
				if (tx_ready_flag == 1) //Si está disponible para
                                        transmitir
				{
					tx_ready_flag = 0; // No puede recibir tramas
					tx_frame_info.msg_type = DATAREQUEST;
					tx_frame_info.msduHandle = seq_num;
					tx_frame.phr = (sizeof(app_frame_t)); //
					tx_frame.fcf                                    =
CCPU_ENDIAN_TO_LE16(FRAME_CONTROL_FIELD);
					tx_frame.seq = seq_num;
					tx_frame.span                                   =
CCPU_ENDIAN_TO_LE16(DEFAULT_PAN_ID);
					tx_frame.saddr=
CCPU_ENDIAN_TO_LE16(SOURCE_ADDRESS);
					memcpy(&tx_frame.data, &msg, sizeof(msg));
					tx_frame_info.mpdu = (uint8_t *)&tx_frame;
					seq_num++;



                                        tal_tx_frame(&tx_frame_info,
CSMA_UNSLOTTED,false); // Esta Api transmite la trama

                                                uint8_t ant_aux;
                                                ant_aux=trx_bit_read(SR_ANT_SEL);

                                                if(ant_aux==0){
                                                        LED_Toggle(LED0);
                                                        LED_Off(LED2);
                                                }

                                                if (ant_aux==1)
```

```
                                        {
                                                LED_Off(LED0);
                                                LED_Toggle(LED2);
                                        }

                                }
                        }
                }
        }

        // La capa TAL invoca esta función para informar sobre el estado de la tx
de tramas a la siguiente capa, en este caso es la capa aplicación
void tal_tx_frame_done_cb(retval_t status, frame_info_t *frame)
{
                if (status == MAC_SUCCESS) // comprueba el estado de la transmisión
                                           y si tiene éxito el seq_num se
                                           incrementa
                {
                        seq_num++;

                        if (seq_num==num_frames) // Cuando se han transmitido 10
                                                    tramas, se enciende el LED1
                        {
                                LED_Toggle(LED1);
                        }
                }
                /* Buffer libre, que se uso para la recepcion de tramas */
                bmm_buffer_free((buffer_t *)(frame->buffer_header));


                /* Sets transceiver state */
                set_trx_state(CMD_RX_ON); // Establece el estado del transmisor
                tx_ready_flag = 1; // Disponible para tx y rx
}

// Función invocada por la capa TAL cuando se recibe una trama
void tal_rx_frame_cb(frame_info_t *frame)
{
                //if (CRC16_VALID == pal_trx_bit_read(SR_RX_CRC_VALID)) //Comprueba
si es una trama IEEE802.15.4 valida. Daba error, creo que porque falta conf_pal.h
                //{
                        //Asumimos que es una trama valida.
                        memset(&rx_frame,0,sizeof(rx_frame));
                        memcpy(&rx_frame,frame->mpdu, sizeof(rx_frame));

                        LED_Off(LED2); //Apagamos LED2 para posteriormente comprobar
                                           si se han recibido 10 tramas


                        if  (rx_frame.span  ==  CCPU_ENDIAN_TO_LE16(DEFAULT_PAN_ID))//
&& rx_frame.saddr == CCPU_ENDIAN_TO_LE16(SOURCE_ADDRESS)) // Comprueba el PAN ID
y la dirección de fuente, si coinciden cambia el LED0
                        {
                                LED_Toggle(LED1);

                                uint8_t ant_aux;

                                ant_aux = trx_bit_read(SR_ANT_SEL);

                                if(ant_aux==0){
                                        LED_On(LED0);
                                        LED_Off(LED2);
```

```
                                //usart_serial_write_packet(USART_HOST,      "\t
Frame rx Antena 1 RSSI:", strlen("Frame rx Antena 1 RSSI:"));

                        }

                        if (ant_aux==1)
                        {
                                LED_Off(LED0);
                                //LED_On(LED2);
                                //usart_serial_write_packet(USART_HOST,      "\n
Frame rx Antena 2 RSSI:", strlen("\n Frame rx Antena 2 RSSI:"));

                        }

                        /* Se muestra por pantalla
                        * TRAMA comienza con STX y finaliza con ETX
                        * STX - CH - ANT - NUM_FRAMES - RSSI - ED - ETX
                        */

                        tal_trx_reg_read(RG_PHY_ED_LEVEL, (uint8_t *)&ed);
                        tal_trx_reg_read(RG_PHY_RSSI, (uint8_t *)&rssi);
                        tal_trx_reg_read(RG_ANT_DIV,  (uint8_t *)&ant_sel);
                        //tal_trx_reg_read(RG_LQI,  (uint8_t *)&lqi);
                        //tal_trx_reg_read(RG_PHY_PMU_VALUE,          (uint8_t
*)&pmu);

                        //usart_serial_putchar(USART_HOST, stx);
                        usart_serial_putchar(USART_HOST, DEFAULT_CHANNEL);
                        usart_serial_putchar(USART_HOST, ant_sel);
                        usart_serial_putchar(USART_HOST, num_frames);
                        usart_serial_putchar(USART_HOST, rssi);
                        usart_serial_putchar(USART_HOST, ed);
                        //usart_serial_putchar(USART_HOST, lqi);
                        //usart_serial_putchar(USART_HOST, pmu);
                        //usart_serial_putchar(USART_HOST, etx);

                        num_frames++;

                        if (num_frames == 5){ // Se han recibido 10 tramas
                                //LED_On(LED1);
                                //usart_serial_write_packet(USART_HOST, "\t
                                10", strlen("\t 20"));

                                tal_ant_div_config(ANT_DIV_DISABLE,
ANT_CTRL_1);  //Cambia a antena2
                                //usart_serial_write_packet(USART_HOST, "\t ¡",
                                strlen("\t !"));
                                bmm_buffer_free((buffer_t             *)(frame-
>buffer_header));
                        }

                        if (num_frames == 10){ // Se han recibido 20 tramas
                                //LED_Off(LED1);
                                //usart_serial_write_packet(USART_HOST, "\t
                                20", strlen("\t 20"));

                                num_frames = 0; //Restablecemos el numero de
                                                frames

                                tal_ant_div_config(ANT_DIV_DISABLE,
ANT_CTRL_0); //Volvemos a la antena1 por defecto
```

```
                                //usart_serial_write_packet(USART_HOST, "\t !",
                                        strlen("\t !"));

                                bmm_buffer_free((buffer_t               *)(frame-
>buffer_header)); //NECESARIO PARA QUE CAMBIE DE CANAL

                                ch = ch + 1;

                                if (ch == 1)
                                {
                                        DEFAULT_CHANNEL=  CCPU_ENDIAN_TO_LE16(12)
;
                                        LED_On(LED2);
                                        bmm_buffer_free((buffer_t       *)(frame-
>buffer_header)); //NECESARIO PARA QUE CAMBIE DE CANAL

                                }
                                if (ch == 2)
                                {
                                        DEFAULT_CHANNEL=  CCPU_ENDIAN_TO_LE16(13)
;
                                        LED_Off(LED2);
                                        bmm_buffer_free((buffer_t       *)(frame-
>buffer_header)); //NECESARIO PARA QUE CAMBIE DE CANAL

                                }
                                if (ch == 3)
                                {
                                        DEFAULT_CHANNEL=  CCPU_ENDIAN_TO_LE16(14)
;
                                        LED_On(LED2);
                                        bmm_buffer_free((buffer_t       *)(frame-
>buffer_header)); //NECESARIO PARA QUE CAMBIE DE CANAL

                                }
                                if (ch == 4)
                                {
                                        DEFAULT_CHANNEL=  CCPU_ENDIAN_TO_LE16(15)
;
                                        LED_Off(LED2);
                                        bmm_buffer_free((buffer_t       *)(frame-
>buffer_header)); //NECESARIO PARA QUE CAMBIE DE CANAL

                                }
                                if (ch == 5)
                                {
                                        DEFAULT_CHANNEL=  CCPU_ENDIAN_TO_LE16(16)
;
                                        LED_Off(LED2);
                                        bmm_buffer_free((buffer_t       *)(frame-
>buffer_header)); //NECESARIO PARA QUE CAMBIE DE CANAL

                                }
                                if (ch == 6)
                                {
                                        DEFAULT_CHANNEL=  CCPU_ENDIAN_TO_LE16(17)
;
                                        LED_Off(LED2);
                                        bmm_buffer_free((buffer_t       *)(frame-
>buffer_header)); //NECESARIO PARA QUE CAMBIE DE CANAL

                                }
```

```
                                    if (ch == 7)
                                    {
                                           DEFAULT_CHANNEL=  CCPU_ENDIAN_TO_LE16(18)
;
                                           LED_Off(LED2);
                                           bmm_buffer_free((buffer_t       *)(frame-
>buffer_header)); //NECESARIO PARA QUE CAMBIE DE CANAL

                                    }
                                    if (ch == 8)
                                    {
                                           DEFAULT_CHANNEL=  CCPU_ENDIAN_TO_LE16(19)
;
                                           LED_Off(LED2);
                                           bmm_buffer_free((buffer_t       *)(frame-
>buffer_header)); //NECESARIO PARA QUE CAMBIE DE CANAL

                                    }
                                    if (ch == 9)
                                    {
                                           DEFAULT_CHANNEL=  CCPU_ENDIAN_TO_LE16(20)
;
                                           LED_Off(LED2);
                                           bmm_buffer_free((buffer_t       *)(frame-
>buffer_header)); //NECESARIO PARA QUE CAMBIE DE CANAL

                                    }
                                    if (ch == 10)
                                    {
                                           DEFAULT_CHANNEL=  CCPU_ENDIAN_TO_LE16(21)
;
                                           LED_Off(LED2);
                                           bmm_buffer_free((buffer_t       *)(frame-
>buffer_header)); //NECESARIO PARA QUE CAMBIE DE CANAL

                                    }
                                    if (ch == 11)
                                    {
                                           DEFAULT_CHANNEL=  CCPU_ENDIAN_TO_LE16(22)
;
                                           LED_Off(LED2);
                                           bmm_buffer_free((buffer_t       *)(frame-
>buffer_header)); //NECESARIO PARA QUE CAMBIE DE CANAL

                                    }
                                    if (ch == 12)
                                    {
                                           DEFAULT_CHANNEL=  CCPU_ENDIAN_TO_LE16(23)
;
                                           LED_Off(LED2);
                                           bmm_buffer_free((buffer_t       *)(frame-
>buffer_header)); //NECESARIO PARA QUE CAMBIE DE CANAL

                                    }
                                    if (ch == 13)
                                    {
                                           DEFAULT_CHANNEL=  CCPU_ENDIAN_TO_LE16(24)
;
                                           LED_Off(LED2);
                                           bmm_buffer_free((buffer_t       *)(frame-
>buffer_header)); //NECESARIO PARA QUE CAMBIE DE CANAL
```

```
                                        }
                                        if (ch == 14)
                                        {
                                                DEFAULT_CHANNEL=  CCPU_ENDIAN_TO_LE16(25)
;
                                                LED_Off(LED2);
                                                bmm_buffer_free((buffer_t       *)(frame-
>buffer_header)); //NECESARIO PARA QUE CAMBIE DE CANAL

                                        }
                                        if (ch == 15)
                                        {
                                                DEFAULT_CHANNEL=  CCPU_ENDIAN_TO_LE16(26)
;
                                                LED_Off(LED2);
                                                bmm_buffer_free((buffer_t       *)(frame-
>buffer_header)); //NECESARIO PARA QUE CAMBIE DE CANAL
                                                ch=0;
                                        }


                                        }

                                        //conf_channel(); // Entra a ver si el canal
                                        está bien definido [11-26]
                                        /* Si el canal está entre 11-26 sigue con esta
                                        instrucción
                                        * Si es mas de 26 lo restablece a 11
                                        */
                                        tal_pib_set(phyCurrentChannel,      (pib_value_t
*)&DEFAULT_CHANNEL); //Esta instrucción lo fija

                                        /* Sets transceiver state */
                                        set_trx_state(CMD_RX_ON); // Establece el
                                                                  estado del transmisor
                                        tx_ready_flag = 1; // Disponible para tx y rx

                                }
                                bmm_buffer_free((buffer_t *)(frame->buffer_header));

                        }
                //}
                /* free buffer that was used for frame reception */
                bmm_buffer_free((buffer_t  *)(frame->buffer_header));  //  Libera  el
buffer
}

        /* Alerta de que algo ha salido mal */
static void app_alert(void)
{
                while (1)
                {
                        #if LED_COUNT > 0
                        LED_Toggle(LED0);
                        #endif
                        #if LED_COUNT > 1
                        LED_Toggle(LED1);
                        #endif
                        #if LED_COUNT > 2
                        LED_Toggle(LED2);
                        #endif
        }}
```

ANEXO D

Actualización software placa ATMEL REB233SMAD

Una vez modificado y ejecutado nuestro software (ANEXO B), cargaremos el mismo en la placa de la siguiente manera:

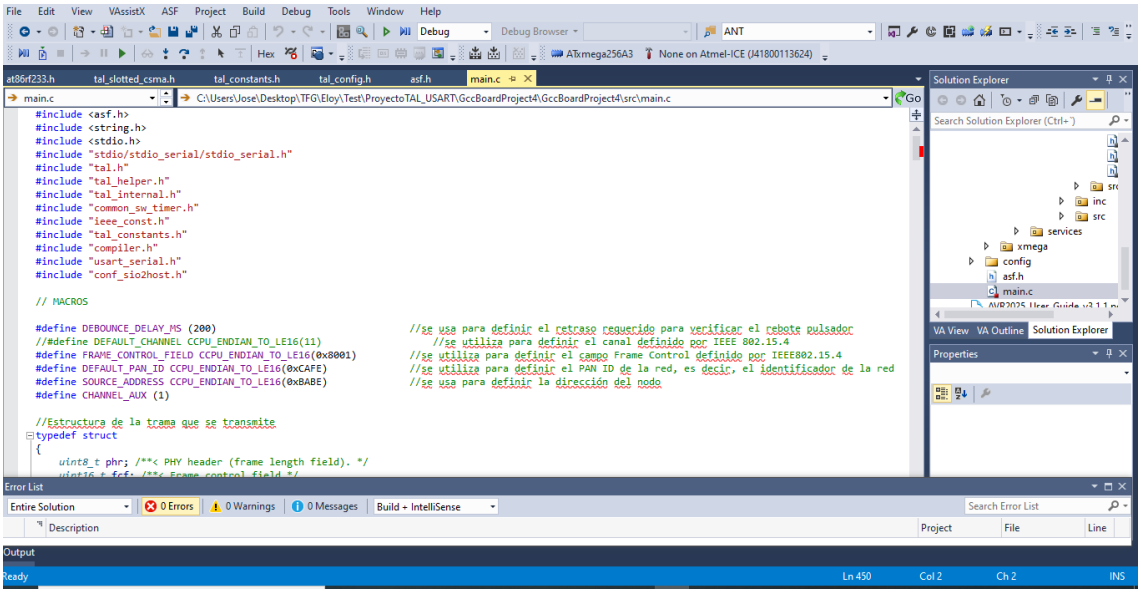


Ilustración 26: Microchip Studio.

En la barra de herramientas de nuestro programa Microchip Studio pulsamos sobre *Device Programming* o pulsamos *Ctrl+Shift+P*.

Ilustración 27: Device Programming.

Tras ello nos aparece la ventana que muestro a continuación, donde tendremos que seleccionar el tipo de herramienta que estamos utilizando, la interfaz JTAG y pulsad en Apply.

Para leer el mismo debemos de pulsar en Read y que encuentre el microcontrolador. Como vemos, aparece nuestro chip *ATXmega256A3* e incluso el nivel de batería del dispositivo.

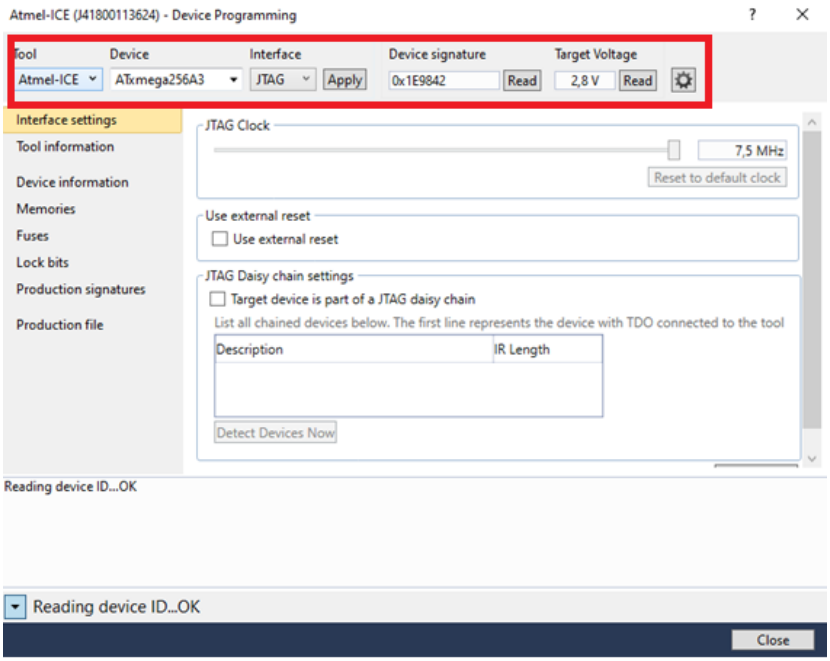


Ilustración 28: Selección chip ATxmega256A3.

Por último y en la pestaña *Memories*, si pulsamos sobre *Program* cargaremos nuestro software en nuestro microcontrolador.

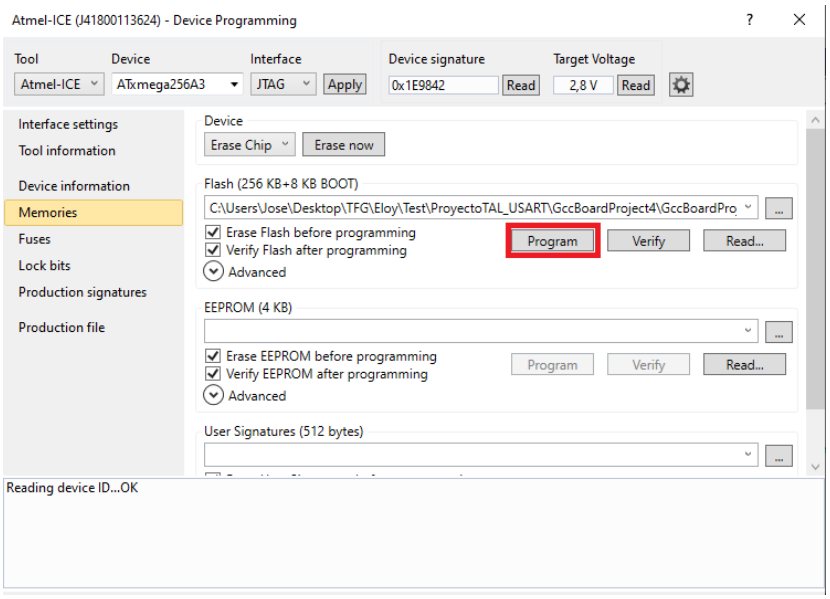


Ilustración 29: Programando la placa.