Redes Ad hoc de sensores aplicadas a la medición de campos de viento

Autora: Yasira Pol Miñán
Tutores: Antonio Carlos Domínguez Brito
Jorge Cabrera Gámez

Curso de adaptación al Grado en Ingeniera Informática

Trabajo de Fin de Grado

Julio de 2017

ÍNDICE DE CONTENIDOS

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE TABLAS

AGRADECIMIENTOS

Quiero dedicar este apartado a plasmar mi agradecimiento a todas aquellas personas que han estado apoyándome para realizar este Trabajo de Fin de Grado.

Primero, a mi madre María, por estar siempre preocupándose por mi futuro y por insistir tanto en que me dedique a lo que realmente me gusta, sin dejar que me rinda en ningún momento. Gracias por tu empeño y tus palabras, por estar ahí siempre a mi lado, por tener la paciencia que demuestras y por ser tan buena madre. Sin ti, es posible que este Trabajo se hubiera quedado en el camino.

Luego, quiero agradecerles a mis hermanas África y Mónica y mi cuñado Jose, que me han ayudado en todo lo que han podido, ya sea moralmente y con palabras de ánimo, gestos o simplemente estando a mi lado.

También quiero agradecerle especialmente a mi compañero de aventuras José Manuel, su paciencia, sus buenas maneras, su dedicación y compañía; por guiarme, darme consejos y hacerme ver otros puntos de vista cuando llegaba a un callejón sin salida. Gracias por aguantar esas horas y jornadas infinitas de trabajo y por estar sentado a mi lado dedicándome tu atención y tu tiempo; por no dejar que me rindiera cuando llegaba a esos momentos de negatividad absoluta; por hacerme reír cuando estaba triste y por ayudarme a despejar la cabeza cuando el momento lo requería.

Por último, dar las gracias a mi tutor Antonio y a mi co-tutor Jorge, que a pesar del tiempo, me han ayudado en todo lo que han podido.

1. INTRODUCCIÓN

El viento es el principal agente que interviene en la naturaleza y capaz de provocar diversos efectos en ella.

Es por ello que los estudios sobre las mediciones de viento en distintas superficies de campo han derivado a determinados sectores profesionales a estudiar cual sería la manera más óptima de obtener dicha información, desde la AEMET hasta el Grupo de Geografía Física y Medio Ambiente de la ULPGC donde han recogido datos principalmente sobre cómo afecta el viento en función de las coordenadas.

Esto ha llevado al sector de la topografía a realizar un estudio sobre la dinámica eólica y como afecta en gran medida a terrenos sobre dunas costeras y que se recoge en el Proyecto Dunas. El proyecto ha sido desarrollado por la División de Robótica y Oceanografía Computacional del SIANI (Instituto Universitario en Sistemas Inteligentes y Aplicaciones Numéricas en Ingeniería). El objetivo del proyecto fue crear un sistema para la captura de los datos en tiempo real de los efectos del viento que consta de las siguientes características:

- Sistema preciso y exacto en la toma de datos
- Resistencia a condiciones atmosféricas adversas
- Inalámbrico
- Monitorización en tiempo real de múltiples dispositivos con un software a medida, llamado meteo.

La composición está basada en una estación base y una serie de sensores de viento cuya comunicación se realiza mediante una topología punto a punto, es decir, donde cada uno de los sensores de viento (emisores) tienen como único referente a la estación base (receptor) que a su vez disponen de un XBee[1] que ejecuta el protocolo ZigBee[2], como se muestra en la siguiente ilustración:

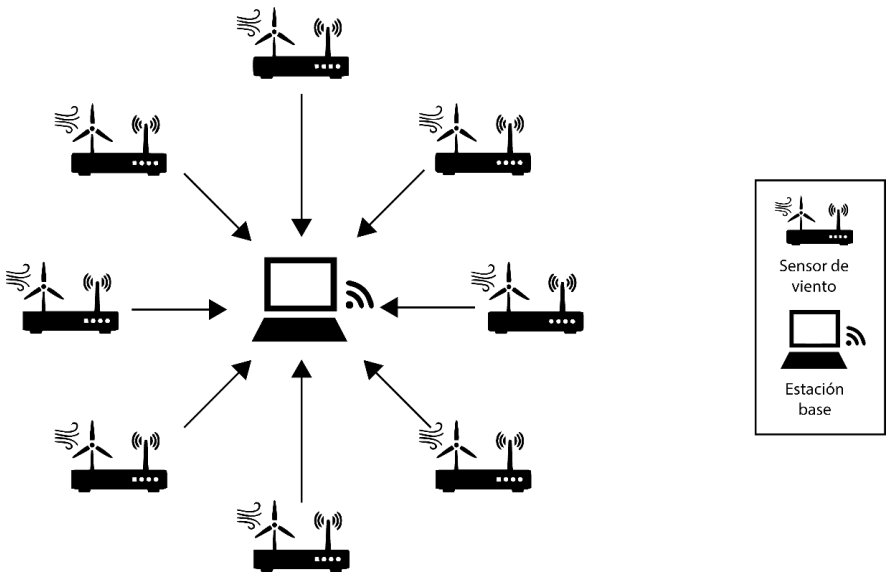


Ilustración 1: Topología punto a punto en el proyecto Dunas

El módulo de radio XBee se basa en un chip electrónico como solución integrada que brinda un medio inalámbrico para la interconexión y comunicación entre dispositivos. Estos módulos utilizan el protocolo de red llamado IEEE 802.15.4[3] para crear redes Fast point-to-multipoint (punto a multipunto) o para redes peer-to-peer (punto a punto). Fueron diseñados para aplicaciones que requieren de un alto tráfico de datos, baja latencia y una sincronización de comunicación predecible. Por lo que básicamente XBee es propiedad de Digi[4] y basado en el protocolo ZigBee, que es una alianza y un estándar de redes MESH de eficiencia energética y de costos.

Ilustración 2: Logo de ZigBee

Ilustración 3: Logo de XBee

Esta configuración posee una limitación ya que la estación base debe colocarse en una distancia muy próxima en la que todos los sensores de viento puedan comunicarse con ella, debiéndose este factor a la topología de red punto a punto.

La evolución natural del proyecto actual es la implementación de un nuevo protocolo que permita aumentar la distancia entre las radios y la estación base, para cubrir mayores distancias. Junto con esto, también habría que plantearse una nueva topología de red acorde a las nuevas capacidades que se quieren introducir en el sistema.

En la práctica, el protocolo de red inalámbrica que permitiría dotar al sistema de nuevas capacidades es DigiMesh. Éste permite crear redes de malla, donde cada nodo de la red permite enrutar paquetes y por tanto, hacer de enlace entre otros nodos de la misma, permitiendo crear "caminos" automáticamente entre el origen y el destino de la comunicación. El protocolo también se encarga de que el camino entre dos puntos sea el más óptimo dentro de la red en la que se encuentran, ya sea por el número mínimo de nodos o por la calidad de la comunicación.

Ilustración 4: Logo de DigiMesh

Respecto a las redes malla, son aquellas que mezclan las tipologías Ad-hoc y la topología infraestructura. La principal ventaja es que este tipo de topología permite unir a la red dispositivos a pesar de estar fuera del rango de cobertura de los puntos de acceso o agregadores, extendiendo así la red de manera dinámica. De esta manera, el área que se puede cubrir con las radios se vería multiplicada tantas veces como el número de nodos por la máxima distancia que permitida por cada módulo XBee.

2. OBJETIVOS

2.1. OBJETIVOS DEL PROYECTO

El objetivo principal de este trabajo consiste en reconfigurar unos módulos de radio XBee para que se comuniquen en una red de malla, usando el protocolo propietario DigiMesh de la marca Digi. Se partirá del Proyecto Dunas, en el que se adaptará el sistema de comunicaciones para usar el nuevo protocolo y así aprovechar las ventajas que éste proporciona.

Respecto al sistema y su implementación, deberá incluir las siguientes capacidades:

- Comunicación en malla con DigiMesh
- Sincronización de los dispositivos
- Envío y enrutado de paquetes entre nodos
- Adición de nodos en caliente
- Recálculo de caminos si se eliminan o fallan nodos

2.2. OBJETIVOS ACADÉMICOS

El objetivo de este trabajo es adquirir las competencias del Trabajo Fin de Grado. A continuación, se detallan aquellas competencias que son cubiertas por este trabajo.

2.2.1. CII01

Capacidad para planificar, concebir, desplegar y dirigir proyectos, servicios y sistemas informáticos en todos los ámbitos, liderando su puesta en marcha y su mejora continua y valorando su impacto económico y social.

Esta competencia queda cubierta en las secciones de Normativa y legislación, Análisis y Desarrollo e implementación donde se especifica mediante un diseño inicial el esquema que debe cumplir toda la funcionalidad del sistema, su desarrollo y su implementación mediante los requisitos software y hardware tomando en consideración cada una de las licencias junto a sus limitaciones.

2.2.1. CII18

Conocimiento de la normativa y la regulación de la informática en los ámbitos nacional, europeo e internacional.

Esta competencia queda cubierta con el conocimiento de los desarrolladores de aquellas normativas y regulaciones vigentes en los distintos ámbitos nombrados, para poder adaptar el desarrollo de cualquier proyecto a las mencionadas regulaciones actuales.

2.2.2. IC01

Capacidad de diseñar y construir sistemas digitales, incluyendo computadores, sistemas basados en microprocesador y sistemas de comunicaciones.

Esta competencia queda cubierta en la sección de Desarrollo e implementación donde se expone con detalle cada uno de los puntos que se han de seguir para el correcto desarrollo del nuevo sistema. Además se exponen los requisitos hardware que se emplearán para su consecución.

2.2.3. IC02

Capacidad de desarrollar procesadores específicos y sistemas empotrados, así como desarrollar y optimizar el software de dichos sistemas.

Esta competencia queda cubierta en la sección de Desarrollo e implementación en la que se especifica la actualización del firmware de las radios XBee.

2.2.4. IC04

Capacidad de diseñar e implementar software de sistema y de comunicaciones.

Esta competencia queda cubierta en las secciones de Análisis y Desarrollo e implementación donde se expone las principales funciones que ha de cumplir el sistema, donde se emplean las radios XBee como elementos de comunicaciones y dentro de los requisitos software el uso de la librería XBee para Arduino, XCTU para la configuración de los módulos inalámbricos XBee y la librería libxbee3 para el control de los sensores de viento (meteo).

2.2.5. IC05

Capacidad de analizar, evaluar y seleccionar las plataformas hardware y software más adecuadas para el soporte de aplicaciones empotradas y de tiempo real.

Esta competencia queda cubierta en las secciones de Análisis, Desarrollo e implementación donde se expone el estado actual del escenario planteado como punto de partida para el correspondiente desarrollo de la nueva solución a aportar. Dentro de Desarrollo e implementación se especifica los elementos hardware y software a emplear.

2.2.6. IC07

Capacidad para analizar, evaluar, seleccionar y configurar plataformas hardware para el desarrollo y ejecución de aplicaciones y servicios informáticos.

Esta competencia queda cubierta en las secciones de Análisis, Desarrollo e implementación donde se exponen los elementos hardware a emplear dentro de las funciones del sistema, siendo estos la estación base, la placa Arduino UNO y las radios XBee.

2.2.7. IC08

Capacidad para diseñar, desplegar, administrar y gestionar redes de computadores.

Esta competencia queda cubierta en las secciones de Análisis, Desarrollo e implementación donde se especifica el esquema que ha de cumplir el nuevo sistema, el cual se compone de componentes como las radios XBee con el protocolo DigiMesh que conforman una red en forma de malla.

2.2.8. TFG01

Ejercicio original a realizar individualmente y presentar y defender ante un tribunal universitario, consistente en un proyecto en el ámbito de las tecnologías específicas de la Ingeniería en Informática de naturaleza profesional en el que se sinteticen e integren las competencias adquiridas en las enseñanzas.

Esta competencia queda cubierta con la realización del presente documento y con la presentación del proyecto realizado ante un tribunal universitario.

3. ESTADO DEL ARTE

3.1. PROYECTO DUNAS

Actualmente existe una tesis doctoral llevada a cabo para la medición de la dinámica eólica de la zona de Playa del Inglés y Maspalomas. Dicho proyecto se encarga de estudiar el efecto que tiene el viento sobre el desarrollo de las Dunas de Maspalomas.

El proyecto fue desarrollado en parte gracias a la carencia en el mercado de estaciones y estructuras que cubrieran las necesidades del estudio y la recogida de datos. De este modo, se idearon las estaciones de bajo costo basadas en Arduino UNO y a las que se conectaron diferentes sensores para medir la dirección y velocidad del viento (veletas y anemómetros). También posee un zumbador que notifica si hay algún problema con la estación. Todas las estaciones se monitorizan desde una estación base, que suele ser un ordenador portátil.

El diseño de las estaciones permite que sea robusto y que resista las condiciones extremas a las que será expuesto (altas temperaturas, vientos fuertes, etc.). Además, por tener un tamaño relativamente pequeño, facilita su movilidad y manipulación en el campo de trabajo.

Para el despliegue de las estaciones, es necesario utilizar una estación total topográfica para obtener la latitud y longitud del dispositivo, datos importantes para un análisis posterior.

Una vez situados, se procede a una calibración inicial de los dispositivos. Éstos se sincronizan mediante un mensaje enviado desde la base, que indica el tiempo actual. De esta manera, todos los datos recogidos por las distintas estaciones tendrán coherencia en el tiempo. Esta sincronización se realiza mediante mensajes enviados de manera inalámbrica, utilizando el protocolo llamado ZigBee, lo cual permite abarcar áreas mayores que si estuvieran conectadas por cable.

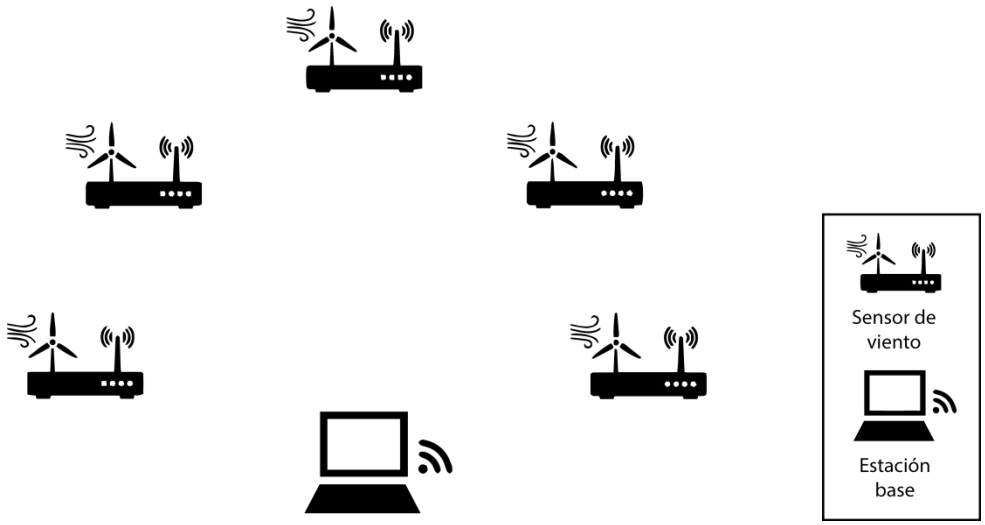


Ilustración 5: Sistema inalámbrico de toma de medidas de viento

Para la recogida de datos, se dispusieron de once estaciones idénticas, repartidas en una parcela situada en la costa de Playa del Inglés. La captura de datos se realiza en tiempo real, pues es importante para obtener datos lo más fieles a la realidad. Éstos se capturan cada 2 segundos y luego se envían inalámbricamente a la estación base, que es la encargada de grabar todos los datos que va recibiendo mediante un software realizado para tal fin. A su vez, cada estación guarda en una tarjeta MicroSD los datos recogidos, ya que, en caso de que se pierda la conexión con la estación base, los datos no se pierdan.

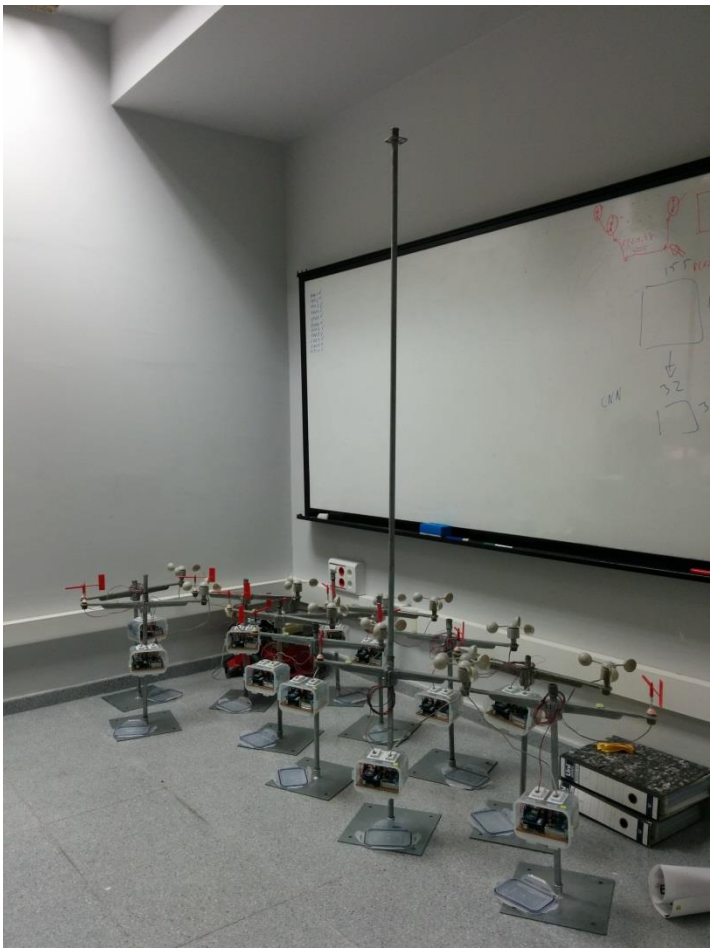


Ilustración 6: Vista del conjunto de dispositivos en el laboratorio

El software que se ejecuta en la estación base permite monitorizar el estado de todas las estaciones, indicando si éstas están funcionando o no y si hay cambios en las mediciones de los sensores. Además, como se ha mencionado, recoge todos los datos que envían las estaciones y los almacena.

3.1.1. OTRAS INICIATIVAS

3.1.1.1. LUCES URBANAS INTELIGENTES[5]

En la ciudad de Mississauga (Toronto) han reemplazado las luces urbanas de alta presión de sodio (HPS en inglés) por luces LED para reducir la energía consumida y los costes de mantenimiento.

Se conectaron 50.000 lámparas, adaptándolas con controles que permiten agruparlas en zonas gestionadas de manera centralizada mediante un módulo Digi XBee. Éstos se conectan a otros dispositivos que actúan como puertas de enlace, que a su vez se conectan con el sistema central de gestión de luces de la ciudad, tal y como se muestra en la siguiente ilustración:

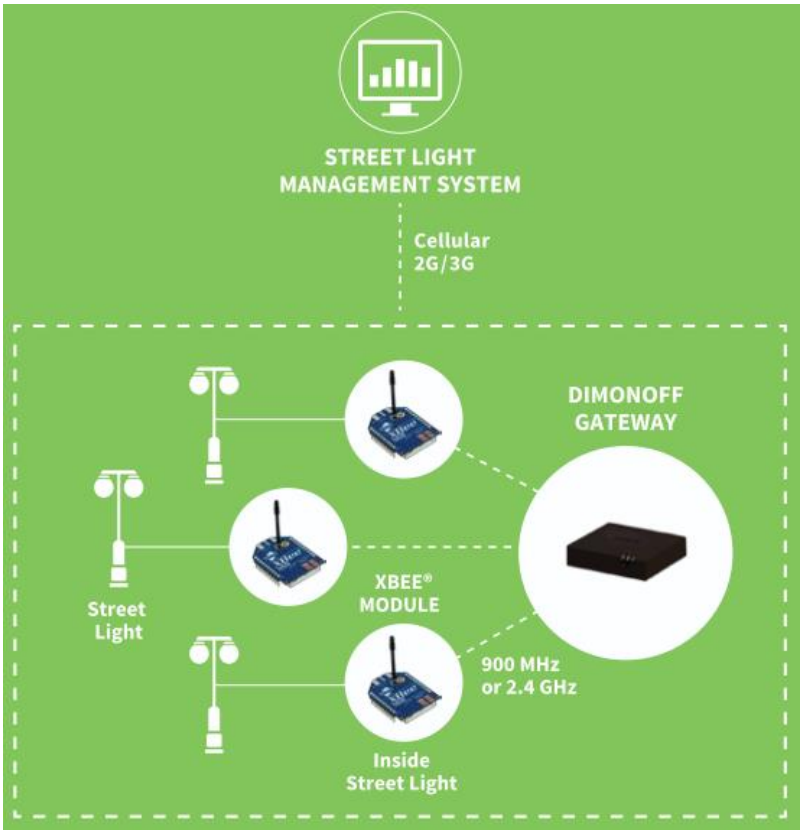


Ilustración 7: Red de luces urbanas usando XBee

Los resultados ofrecidos se pueden dividir en 3 aspectos:

- **Optimización de la seguridad**: las lámparas inteligentes permiten monitorizar el estado de la misma en carreteras, parques, estaciones y otras áreas públicas.
- **Costes operacionales bajos**: permitiendo la comunicación y el control centralizado de las luces, la ciudad puede reducir el consumo de energía, mejorando el mantenimiento y reduciendo las emisiones de CO_2.
- **Mantenimiento predictivo**: al tener la lámpara inteligente, el departamento de mantenimiento puede programar mantenimientos y recursos de manera más eficiente, pudiendo resolver fallos en 15 minutos.

3.1.1.2. SISTEMA DE CONTROL DE RIEGO[6]

La implantación de sistemas de control y adquisición de datos en el sector agrícola está suponiendo en la actualidad una inversión cada vez más importante, dadas las prestaciones que ofrecen. De esta manera, el productor puede disponer de un sistema de control de riego que maximizará la producción, consiguiéndose así un ahorro significativo del agua. Debido a que el empleo de sistemas cableados, supone grandes costes de mantenimiento, cada vez se opta más por soluciones inalámbricas de largo alcance.

En este caso, se ha implementado una arquitectura de comunicación inalámbrica que permite transmitir las respectivas señales de control y monitorización entre una estación de control (y bombeo) y un cabezal de riego, encargados ambos del proceso de riego en una finca. Ambas partes se encuentran distanciadas 1 Km aproximadamente.

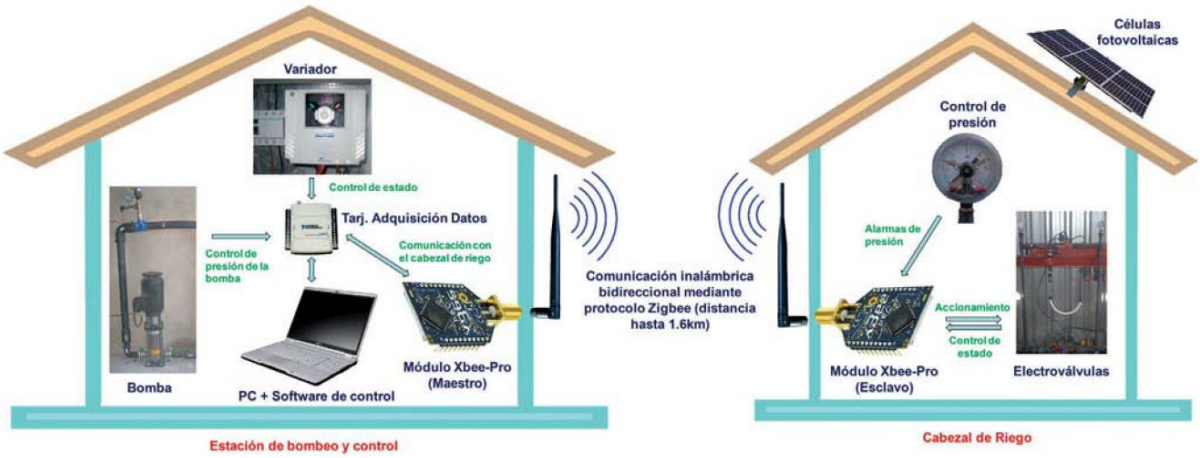


Ilustración 8: Estructura del sistema de riego

Como se puede ver en la ilustración anterior, se pueden distinguir los siguientes componentes:

- Un sistema de bombeo provisto de variador, responsable de que le llegue agua al cabezal de riego.
- Un computador donde se encuentra instalado la aplicación de control.
- Una tarjeta de adquisición de datos que traduce las señales entre los componentes del sistema de riego y el computador.
- Un módulo de comunicaciones XBee-Pro junto con una antena de alta ganancia que permite la comunicación inalámbrica con el cabezal de riego.

A su vez, el cabezal de riego consta de los siguientes elementos:

- Varias electroválvulas que serán accionadas por la aplicación desde la estación de control y bombeo
- Un sistema de alarma por presión que avisa a la estación de control y bombeo tanto en caso de sobrepresión como de presión baja
- Paneles fotovoltaicos provisto de batería para alimentar los dispositivos del cabezal de riego

- Otro módulo XBee-Pro junto con una antena de alta ganancia que permite la comunicación inalámbrica con la estación de control y bombeo.

El sistema de comunicaciones a larga distancia implementado consta principalmente de dos módulos XBee-Pro 802.15.4 de Digi, configurados como coordinator y end device. Los módulos se comunican entre sí de forma transparente para el usuario.

Esta implementación permite dotar al sistema de control de riego de una comunicación libre de errores y sin necesidad de mantenimiento, proporcionando de esta manera una alternativa viable a las redes cableadas que se usan en la actualidad. A su vez, el empleo de antenas de elevada ganancia garantiza una comunicación fiable entre ambos módulos, de manera que la distancia existente entre la plataforma de control y el cabezal de riego no es un problema.

3.1.1.3. PROYECTO ADI-BIDEA

El objetivo del proyecto ADI-BIDEA[7] es dotar a las infraestructuras viales vizcaínas de consciencia para percibir su entorno y actuar en consecuencia, garantizando una movilidad segura y sostenible.

Para ello se pretende crear un ecosistema o red de sistemas inteligentes, apoyándose en la infraestructura vial ya existente, capaz de adecuar la iluminación y la señalización vial tanto a las condiciones atmosféricas en cada momento, como a la propia ocupación de la carretera y a la vez, convertir la propia vía en fuente de información útil para los usuarios y conductores.

Se ha diseñado y probado una red de comunicaciones inalámbricas en malla (mesh) autoconfigurable para el intercambio de información entre los puntos estáticos en la infraestructura. Dicha red se apoya en comunicaciones de corto alcance para el escenario de iluminación inteligente. Todo ello se ha realizado utilizando dispositivos de la casa Libelium[8] funcionando a 2,4 GHz (DigiMesh Pro 2).

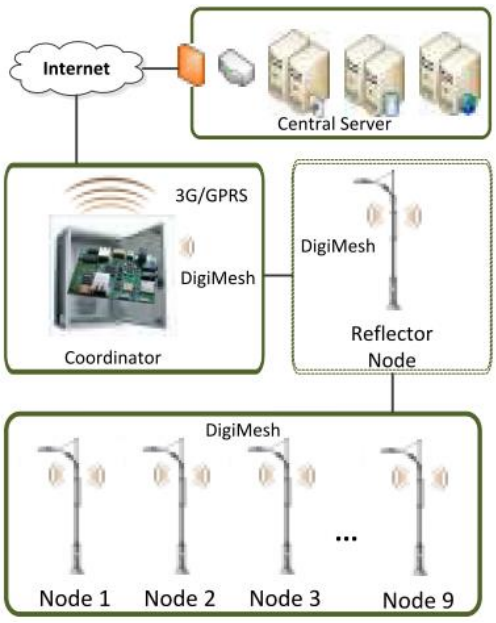


Ilustración 9: Arquitectura proyecto ADI-BIDEA

Como se puede apreciar en la imagen anterior, la arquitectura diseñada se basa en que existe un nodo coordinador, otro concentrador y un número variable de nodos finales. El nodo coordinador es en el que radica la lógica de los sistemas y decide las actuaciones a efectuar según una situación detectada u otra, incluyendo su reporte al centro de Control y Monitorización remota. Por su parte, el concentrador recibe y concentra todos los mensajes dirigidos al coordinador. Se establece que cada instalación constituirá una red en sí misma, diferente en cada caso y que existirá un concentrador para cada una de ellas.

Por último, todos los nodos finales intercambian mensajes mediante tecnología inalámbrica creando una red DigiMesh.

3.1.1.4. MONITORIZACIÓN DE TURBINAS DE VIENTO RESIDENCIALES[9]

En todo el mundo se están desplegando turbinas de viento para ayudar a reducir el consumo de combustibles fósiles.

Los usuarios de estas turbinas pueden monitorizar la energía creada por éstas y los fabricantes necesitan controlar el estado de las máquinas y proveer datos a los gobiernos para incentivar el uso de las mismas.

Ilustración 10: Red ZigBee para monitorizar turbinas de viento residenciales

Por esta razón, un fabricante ha diseñado un dispositivo XBee embebido en sus propias turbinas. Como se puede observar en la imagen anterior, esto permite comunicar las turbinas con pasarelas exteriores que se conectan a su vez con centrales de monitorización vía IP, mostrando al usuario los datos en tiempo real y además alertando al fabricante de cualquier incidencia ocurrida en la turbina.

Esta red interconectada es de bajo coste, poco intrusiva y segura. Esta solución ofrece beneficios significativos tanto para el fabricante como para el usuario.

3.2. TECNOLOGÍAS

3.2.1. REDES SENSORIALES INALÁMBRICAS (WSN)

Las redes sensoriales inalámbricas o WSN por su siglas en inglés, es una tecnología que está siendo empleada en un número creciente de aplicaciones y que puede considerarse como la tecnología subyacente en el concepto de Internet de las cosas o Internet of Things (IoT), definido como red de objetos cotidianos interconectados.

Actualmente existe la demanda en el mercado de conectarlo todo a internet, y tenerlo todo controlado. Las redes sensoriales no se quedan al margen de esta tendencia, donde la red de sensores aportará información importante al resto de nodos del Internet de las cosas.

Las redes sensoriales utilizan varios medios físicos como medio de transmisión. La luz puede ser un medio para transmitir datos, aunque son las tecnologías de comunicación basadas en señales ópticas, principalmente en el infrarrojo (Bluetooth, Wi-Fi, ZigBee, 3G, etc.), las más utilizadas. En la siguiente imagen se muestra un ejemplo de diferentes redes de sensores:

Ilustración 11: Wireless Sensor Network (WSN) [10]

Las posibilidades que ofrece una tecnología como esta yacen en la realización de aplicaciones donde es complicado el cableado o como mejora de proyectos ya existentes donde la tecnología inalámbrica ofrece un valor añadido.

Los últimos estudios de redes sensoriales quieren solucionar los grandes problemas de las redes inalámbricas:

- Baja eficiencia al no utilizar el número óptimo de saltos entre dos nodos.
- No se utiliza una recuperación de errores óptima.
- Alto consumo energético, y un mal ajuste de la potencia hace que se gaste más energía de la necesaria.

Algunas de las características de las redes de sensores inalámbricas son:

- **Patrón de flujo de datos**: cada nodo envía los datos recogidos a un almacén central de datos. Los almacenes en sí pueden a su vez estar distribuidos.
- **Restricciones energéticas**: uno de los principales problemas a los que se enfrenta este tipo de tecnología es la alimentación de cada nodo. Por tanto, los nodos deben estar diseñados para que su consumo sea mínimo, con ciclos de trabajo reducidos para poder hibernarlos el máximo tiempo posible.

Este tipo de redes pretende cambiar la manera de registrar la información a nivel global, con miles de sensores conectados entre sí. Gracias a su capacidad inalámbrica, pueden abarcar grandes áreas y además pueden conectarse a internet, de manera que la información esté disponible casi en tiempo real.

3.2.2. TOPOLOGÍAS DE RED INALÁMBRICAS

3.2.2.1. PUNTO A PUNTO

Se conoce como redes punto a punto[11] las que responden a un tipo de arquitectura de red en las que cada canal de datos se usa para comunicar únicamente dos nodos, en clara oposición a las que se basan en redes multipunto, en las cuales cada canal de datos se puede usar para comunicarse con diversos nodos (ilustración 10).

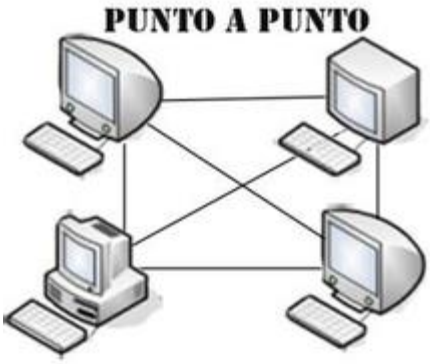


Ilustración 12: Topología punto a punto

En una red de estas características los dispositivos actúan como pares entre sí, donde cada dispositivo puede tomar el rol de emisor o la función de receptor. A continuación, se muestra las principales características que la caracterizan:

- Se utiliza en redes de largo alcance (WAN).
- Los algoritmos de encaminamiento suelen ser complejos y el control de errores se realiza entre los nodos intermedios además de los extremos.
- Las estaciones reciben sólo mensajes que les entregan los nodos de la red. Estos previamente identifican a la estación receptora a partir de la dirección de destino del mensaje.
- La conexión entre los nodos se puede realizar con uno o varios sistemas de transmisión de diferente velocidad trabajando en paralelo.
- Los retardos se deben al tránsito de los mensajes a través de los nodos intermedios.
- La conexión extremo a extremo se realiza a través de los nodos intermedios, por lo que depende de su fiabilidad.
- La seguridad es inherente a la propia estructura en malla de la red en la que cada nodo se conecta a dos o más nodos.
- Los costos del cableado dependen del número de enlaces entre las estaciones. Cada nodo tiene por lo menos dos interfaces.

Además, se recoge una serie de características que reflejan las ventajas y desventajas de esta tipología de red (tabla 1).

Ventajas	Desventajas
Fáciles de configurar	Administración no centralizada
Menor complejidad	No son muy seguras
Menor costo dado que no se necesita dispositivos de red ni servidores dedicados	Todos los dispositivos pueden actuar como cliente y como servidor, lo que puede ralentizar su funcionamiento
	No son escalables
	Reducen su rendimiento

Tabla 1: Ventajas/desventajas de tipología punto a punto

3.2.2.2. REDES MESH

Una red Mesh (malla en español) es una topología de red en la que cada nodo de la red está conectado a los otros nodos que hay alrededor de él. Cada nodo coopera en la transmisión de información. Los beneficios más importantes de este tipo de redes son:

- ***Routing***: con esta técnica, los mensajes se propagan a lo largo de toda la ruta saltando de nodo a nodo hasta que llegan a su destino final.
- **Creación de redes *Ad-hoc***: es un proceso automático que crea la red completa con todos los nodos sobre la marcha, sin intervención del usuario.
- ***Self-healing***: este proceso detecta automáticamente si un nodo se ha caído y reconfigura toda la red para reparar rutas inválidas.

Con una red Mesh, la distancia entre dos nodos no es relevante, siempre y cuando haya nodos entre ellos que permitan el paso de mensajes. Cuando un nodo quiere comunicarse con otro, la red automáticamente calcula el camino más óptimo.

En la siguiente imagen se muestra un ejemplo de red de este tipo, en la que los nodos se interconectan entre sí:

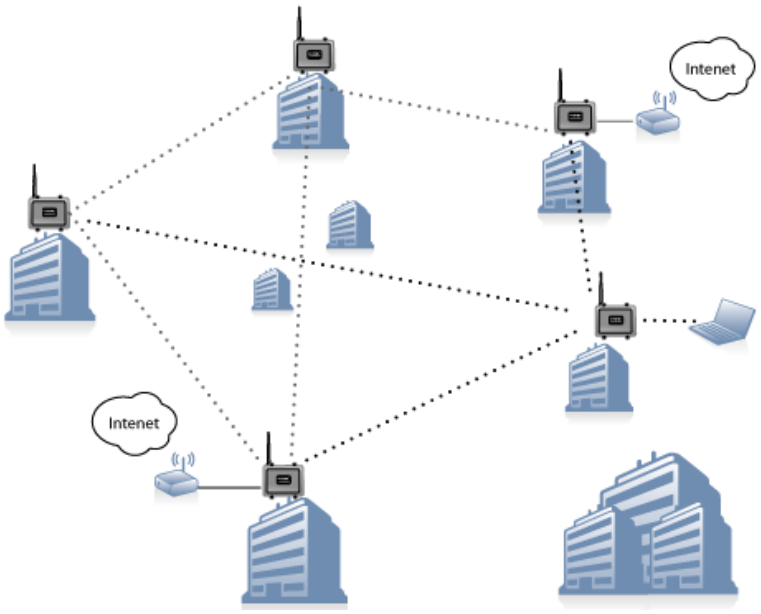


Ilustración 13: Ejemplo de red Mesh (malla)

Además, las redes Mesh son fiables y ofrecen redundancia. Si un nodo no puede seguir funcionando, ya sea porque se ha retirado de la red o por circunstancias físicas que bloqueen la comunicación, el resto de nodos pueden seguir comunicándose, ya sea directamente o usando otros nodos intermedios.

Una desventaja de este tipo de redes es que usan más ancho de banda para la administración y por tanto la capacidad de enviar datos se ve reducida. Otro punto a tener en cuenta es que, dependiendo del caso, pueden tener una configuración más compleja.

3.2.3. PROTOCOLOS INALÁMBRICOS

3.2.3.1. PUNTO A PUNTO

Una red punto-multipunto se consigue mediante conexiones uno-a-uno o uno-a-muchos, proporcionando múltiples rutas desde un lugar a muchos otros.

Los siguientes puntos muestran una serie de protocolos que se basan en este principio.

3.2.3.1.1. IEEE 802.15.4

El estándar IEEE 802.15.4 especifica la capa física y control de acceso y es ideal para aplicaciones que requieran baja latencia y un tiempo de comunicación predecible.

El propósito del estándar es definir los niveles de red básicos para dar servicio a un tipo específico de red inalámbrica de área personal (WPAN) centrada en la habilitación de comunicación entre dispositivos ubicuos con bajo coste y velocidad (en contraste con Wi-Fi). Se enfatiza el bajo coste de comunicación con nodos cercanos y sin infraestructura o con muy poca, para favorecer aún más el bajo consumo.

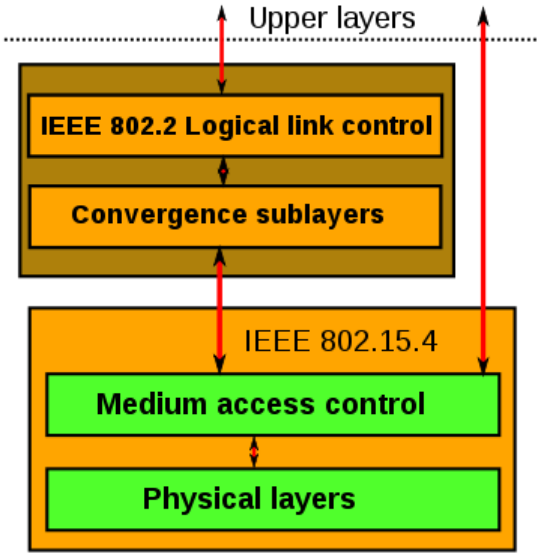


Ilustración 14: Capas IEEE 802.15.4

La arquitectura se apoya en los distintos los niveles, basados en el modelo OSI. Como se puede comprobar en la imagen anterior, los niveles inferiores se definen en el estándar e interaccionan con el resto de niveles por medio de un subnivel de control de enlace lógico, que acceda a MAC a través de un subnivel de convergencia. La implementación puede basarse en dispositivos externos o integrarlo todo en dispositivos autónomos.

Opera en una de tres posibles bandas de frecuencia de uso no regulado:

- **868-868,8 MHz**: Europa. Permite 3 canales de comunicación.
- **902-928 MHz**: Norte América. Hasta 30 canales.
- **2,4 GHz**: uso en todo el mundo. Hasta 16 canales.

El estándar define dos tipos de nodo en la red. El primero es el dispositivo de funcionalidad completa (FFD). Puede funcionar como coordinador de una red de área personal o como un nodo normal. Implementa un modelo general de comunicación que le permite establecer un intercambio con cualquier otro dispositivo. Puede, además, encaminar mensajes, en cuyo caso se le denomina coordinador.

Contrapuestos a éstos están los dispositivos de funcionalidad reducida (RFD). Se plantean como dispositivos muy sencillos con recursos y necesidades de comunicación muy limitadas. Por ello, sólo pueden comunicarse con FFD's y nunca pueden ser coordinadores.

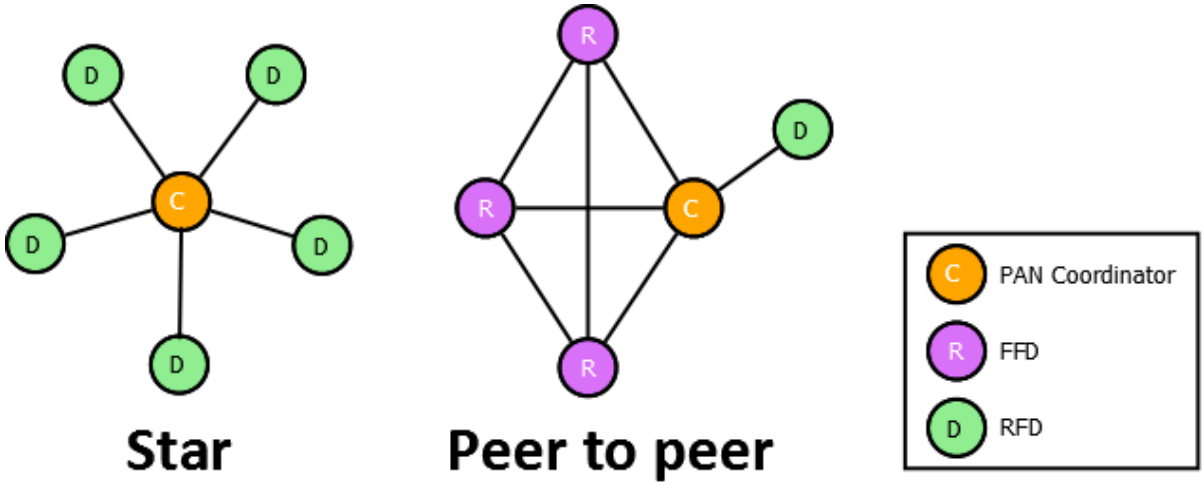


Ilustración 15: Tipos de redes permitidas en el IEEE 802.15.4

Como se puede observar en la ilustración anterior, las redes de nodos pueden construirse como redes punto a punto o en estrella. En cualquier caso, toda red necesita al menos un FFD que actúe como su coordinador. Las redes están compuestas por grupos de dispositivos separados por distancias suficientemente reducidas; cada dispositivo posee un identificador único de 64 bits, aunque si se dan ciertas condiciones de entorno en éste pueden utilizarse identificadores cortos de 16 bits. Probablemente éstos se utilizarán dentro del dominio de cada WPAN separada.

3.2.3.1.2. IEEE 802.11 (Wi-Fi [12])

Esta nueva tecnología surgió por la necesidad de establecer un mecanismo de conexión inalámbrica que fuese compatible entre distintos dispositivos. Buscando esa compatibilidad, varias empresas se unieron para crear la llamada Alianza Wi-Fi. El objetivo de la misma fue designar una marca que permitiese fomentar más fácilmente la tecnología inalámbrica y asegurar la compatibilidad de equipos.

Ilustración 16: Logotipo Wi-Fi

Este estándar certifica la interoperabilidad de equipos según la norma IEEE 802.11b, bajo la marca Wi-Fi. Esto quiere decir que el usuario tiene la garantía de que todos los equipos que tengan el sello Wi-Fi pueden trabajar juntos sin problemas, independientemente del fabricante de cada uno de ellos. El siguiente esquema refleja una gran variedad de dispositivos que se interconectan utilizando Wi-Fi:

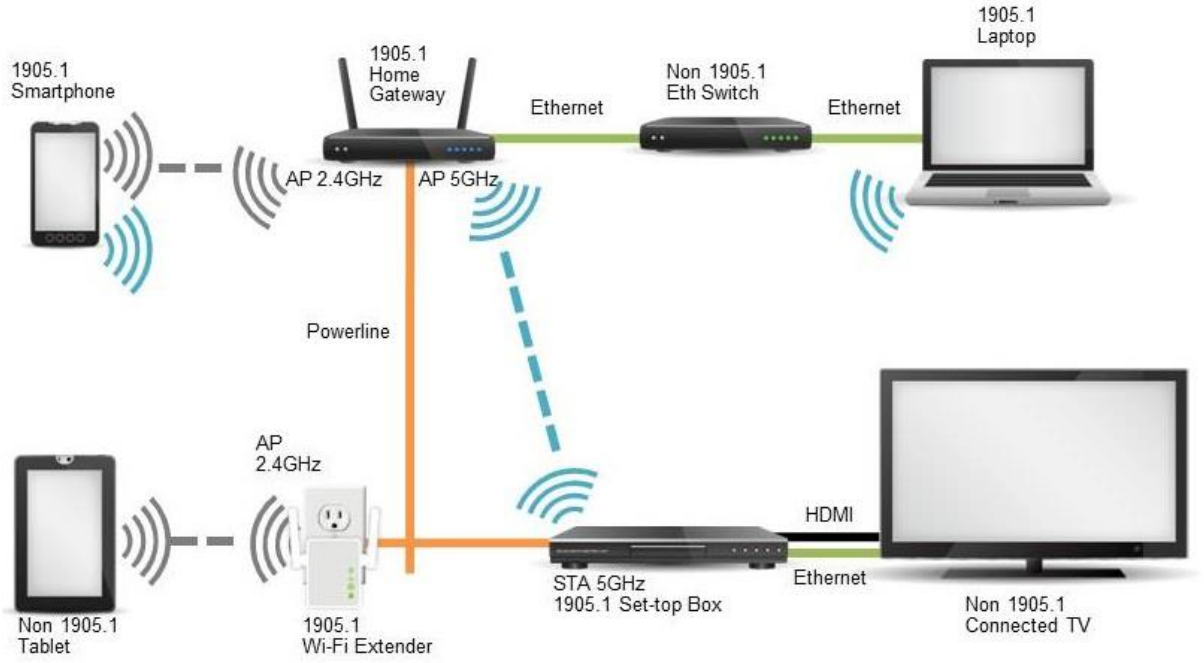


Ilustración 17: Red Wi-Fi

La peculiaridad de Wi-Fi es que en lo único que se diferencia de una red Ethernet es en cómo se transmiten las tramas o paquetes de datos; el resto es idéntico. Por tanto, una red local inalámbrica es completamente compatible con todos los servicios de las redes locales Ethernet.

Este estándar tiene una variedad de subprotocolos representados por los sufijos `a/b/g/n/ac` con diferentes anchos de banda. Los estándares `b, g` y `n` disfrutan de una aceptación internacional debido a que la banda de 2,4 GHz está disponible casi universalmente. El estándar `ac`, conocido como WIFI 5 opera en la banda de 5 GHz y disfruta de una operatividad con canales relativamente limpios, pues no hay otras tecnologías que usen esta banda.

3.2.3.1.3. IEEE 802.15.1 (BLUETOOTH)

El Bluetooth [13] es una especificación industrial para Redes Inalámbricas de Área Personal (WPAN) que posibilita la transmisión de voz y datos entre diferentes dispositivos mediante un enlace por radiofrecuencia en la banda de los 2.4 GHz. Los principales objetivos que se pretenden conseguir con esta norma son:

- Facilitar las comunicaciones entre equipos móviles.
- Eliminar los cables y conectores entre éstos.
- Ofrecer la posibilidad de crear pequeñas redes inalámbricas y facilitar la sincronización de datos entre equipos personales

Ilustración 18: Logotipo de Bluetooth

Es un protocolo de comunicaciones diseñado especialmente para dispositivos de bajo consumo, que requieren corto alcance de emisión y basados en transceptores de bajo costo.

Los dispositivos que incorporan este protocolo pueden comunicarse entre sí cuando se encuentran dentro de su alcance. Las comunicaciones se realizan por radiofrecuencia de forma que los dispositivos no tienen que estar alineados y pueden incluso estar en habitaciones separadas si la potencia de transmisión es suficiente.

Clase	Potencia máxima permitida (mW)	Potencia máxima permitida (dBm)	Alcance (aproximado)
Clase 1	100 mW	20 dBm	~100 metros
Clase 2	2.5 mW	4 dBm	~5-10 metros
Clase 3	1 mW	0 dBm	~1 metro

Tabla 2: Clases de dispositivoos Bluetooth

Como se puede comprobar en el cuadro anterior, los dispositivos se clasifican como "Clase 1", "Clase 2" o "Clase 3" en referencia a su potencia de transmisión.

El Bluetooth se utiliza principalmente en productos tales como teléfonos, impresoras, módems y auriculares. Su uso es adecuado cuando puede haber dos o más dispositivos en un área reducida sin grandes necesidades de ancho de banda, algo parecido a lo que se muestra en la siguiente ilustración:

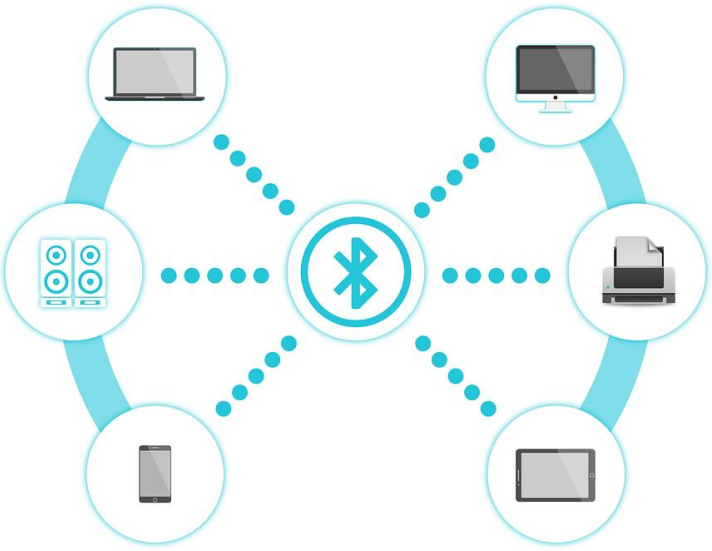


Ilustración 19: Red Bluetooth

Bluetooth simplifica el descubrimiento y configuración de los dispositivos, ya que estos pueden indicar a otros los servicios que ofrecen, lo que permite establecer la conexión de forma rápida.

3.2.3.2. MESH

Las topologías de red tipo Mesh se usan en aplicaciones en las que la distancia entre dos puntos puede estar más allá del rango de dos radios que estén la red, pero que usan puntos intermedios que permiten reenviar cualquier mensaje entre ellos.

En los siguientes puntos se explicarán diferentes protocolos que siguen la filosofía explicada.

3.2.3.2.1. ZIGBEE

El protocolo ZigBee es un protocolo de comunicaciones estándar y abierto, diseñado específicamente para aplicaciones que necesiten baja densidad de datos y ahorro de energía.

Ilustración 20: Logotipo de ZigBee

ZigBee surge de la necesidad de desarrollar una tecnología inalámbrica fiable, pero de baja transferencia de datos. Su principal propósito es ofrecer una solución completa para este tipo de redes construyendo los niveles superiores de la pila de protocolos que el estándar 802.15.4 no cubre. De esta forma, consigue complementar a otros protocolos inalámbricos ya asentados, como Wi-fi y Bluetooth.

Técnicamente, el estándar define un conjunto de protocolos y comunicación de baja velocidad de datos y corto alcance. Usa una técnica de modulación Direct Sequence Spread Spectrum [14] (DSSS), que consiste en utilizar un código de pseudorruido para "modular" digitalmente una señal portadora, de tal forma que aumente el ancho de banda de la transmisión y reduzca la densidad de potencia espectral. La señal resultante tiene un espectro muy parecido al del ruido, de tal forma que a todos los radiorreceptores les parecerá ruido menos al que va dirigida la señal.

Las principales ventajas que ofrece ZigBee son:

- Al ser un estándar abierto que permite la interoperabilidad entre diferentes fabricantes, basado en el 802.15.4.
- Permite reducir gastos al utilizar dispositivos finales con funcionalidades reducidas.
- Permite actualizaciones over-the-air.
- Gran soporte desde la industria y utilizado por múltiples empresas.
- Bajo coste.
- Bajo consumo.
- Redes flexibles y extensibles.

3.2.3.2.1.1. NODOS

El estándar IEEE 802.15.4 y ZigBee definen tres tipos diferentes de nodos. Sus roles son prácticamente los mismos pero la nomenclatura es distinta, tal y como se puede apreciar en la siguiente ilustración:

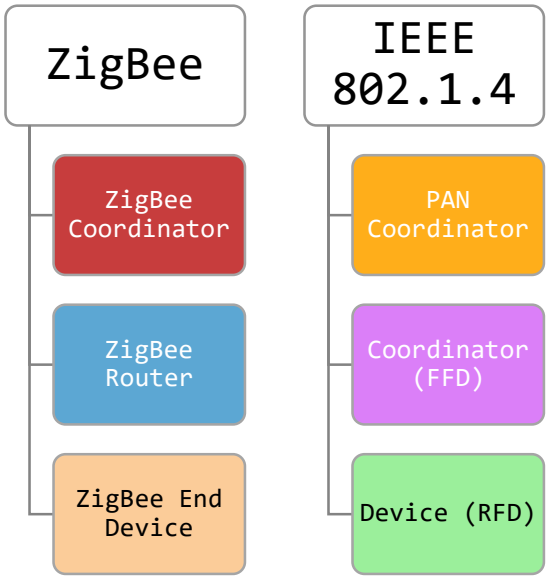


Ilustración 21: Tipos de nodos IEEE 8052.1.4 y XBee

Los **coordinadores** son los nodos más completos. Debe haber solamente un coordinador, pues es el dispositivo que establece y administra la red, además de controlar el enrutado de los datos. Es capaz de almacenar información sobre la propia red, incluyendo claves de seguridad.

Los **routers** actúan como nodos intermediarios, interconectando los dispositivos mediante técnicas de encaminamiento y direccionamiento, reenviando los datos entre los nodos hasta su destino final.

Los **dispositivos finales** (end devices) son dispositivos de funcionalidad reducida y se comunican con coordinadores o routers. Son elementos pasivos de la red, pues normalmente responden a peticiones de dispositivos superiores. De este modo, el dispositivo pasa la mayor parte del tiempo en estado de hibernación, ahorrando energía.

En la siguiente ilustración, que muestra un ejemplo de red ZigBee, se pueden apreciar los distintos tipos de nodos y sus roles:

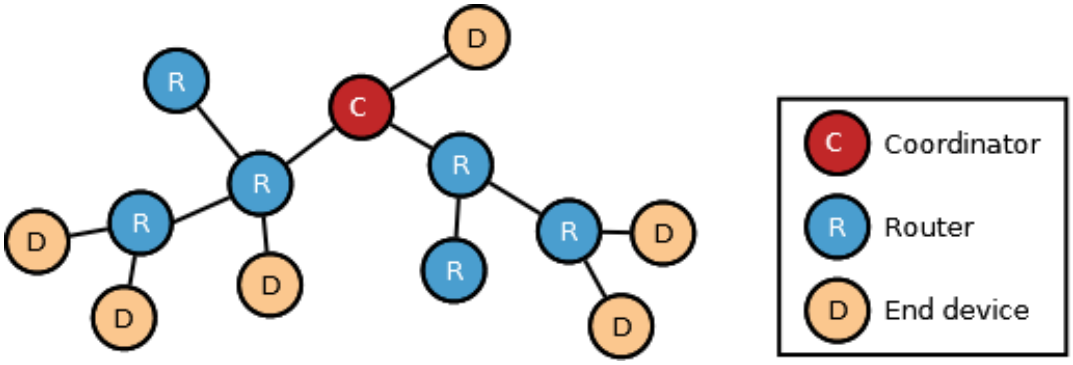


Ilustración 22: Red ZigBee con diferentes tipos de nodos

3.2.3.2.1.2. TOPOLOGÍAS DE RED

Zigbee permite tres topologías de red:

- **Pair**: es la forma más sencilla para crear una red, ya que con dos nodos es suficiente. Uno de ellos debe ser obligatoriamente un coordinador. El otro puede ser bien un router o bien un end device.
  La siguiente imagen muestra un ejemplo de red Pair, con 2 nodos implicados:

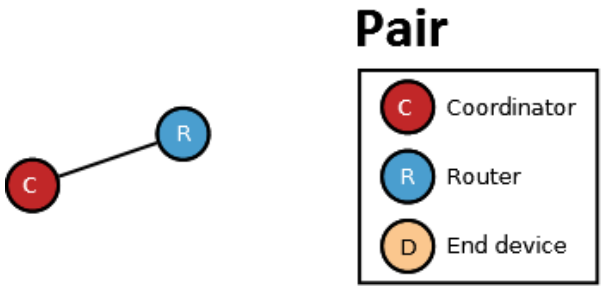


Ilustración 23: Topología de red "Pair" (ZigBee)

- **Star**: en esta topología el coordinador es el centro de la red y es el que se conecta en círculo con los demás dispositivos. De este modo, todos los mensajes deben pasar por el coordinador. Dos dispositivos finales no pueden comunicarse entre sí directamente.
  En el siguiente gráfico se muestra un ejemplo de red tipo Star muy sencilla:

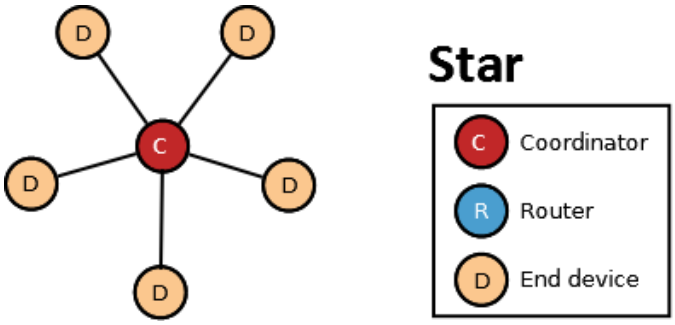


Ilustración 24: Topología de red "Star" (ZigBee)

- **Mesh**: en esta configuración entran en juego los nodos de tipo router, además del obligatorio nodo coordinador. Se trata de una topología no jerárquica en el sentido de que cualquier dispositivo puede interactuar con cualquier otro.
  Este tipo de topología permite el coordinador rehaga los caminos si en un momento un nodo falla en la comunicación, por lo que se reestructura automáticamente sin interrupciones en la red.
  Para clarificar el concepto, se muestra la siguiente imagen con una red Mesh, en la que los nodos coordinador y router se interconectan entre sí:

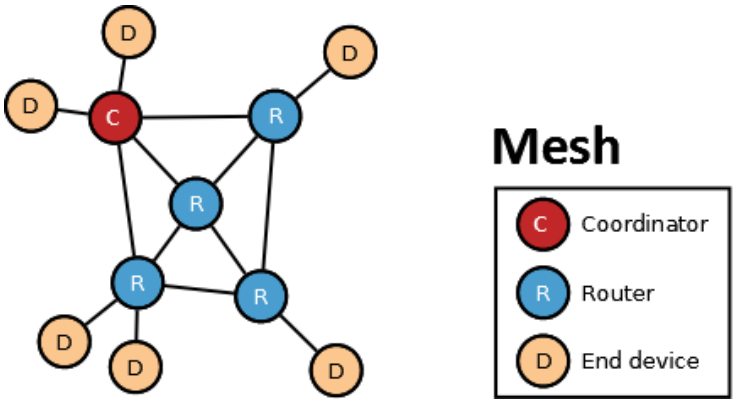


Ilustración 25: Topología de red "Mesh" (ZigBee)

- **Cluster Tree**: es una variación de la topología mesh. En este caso, los routers forman una columna vertebral con los dispositivos finales, que están agrupados en torno a los routers.
  En la siguiente imagen se puede apreciar este tipo de topología, en la que el coordinador se conecta con dispositivos finales y con routers y éstos a su vez sólo con los dispositivos finales.

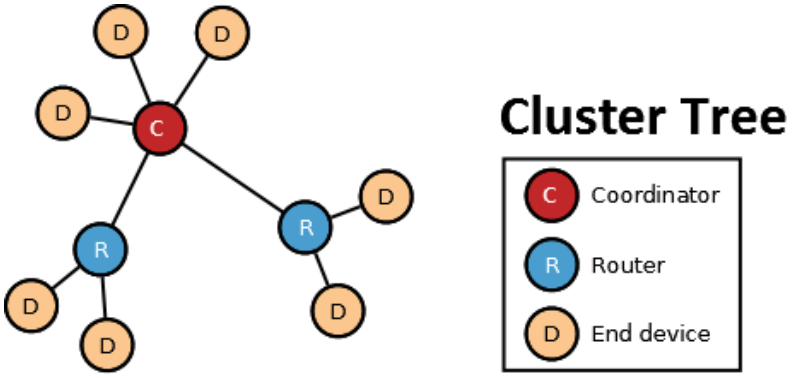


Ilustración 26: Topología de red "Cluster Tree" (ZigBee)

De las cuatro topologías mostradas, la más versátil y ventajosa es la Mesh, ya que tiene una alta fiabilidad y estabilidad.

3.2.3.2.1.3. CAPAS

Las capas del protocolo Zigbee están basadas en el modelo OSI (Open Systems Interconnection). Mientras que el modelo OSI especifica 7 capas, Zigbee utiliza sólo 4 capas. La ventaja de dividir un protocolo en capas es que si el protocolo cambia, es más

fácil cambiar una capa que el protocolo entero. La siguiente ilustración muestra un esquema de las capas del protocolo, mostrando cómo se comunican entre ellas:

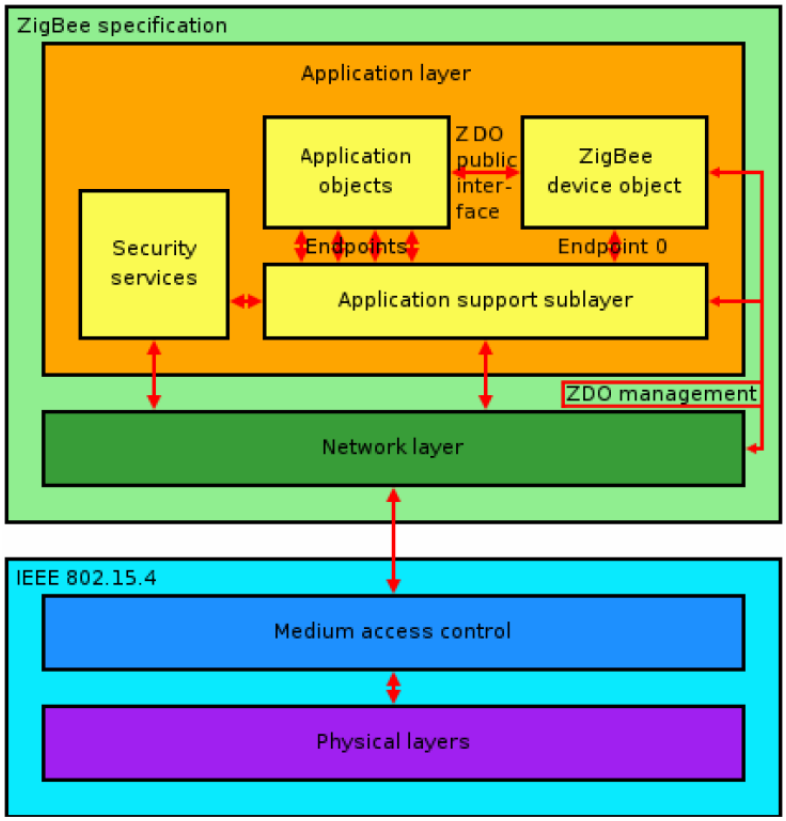


Ilustración 27: Capas del protocolo ZigBee

Como se puede apreciar en la imagen, el estándar Zigbee define sólo las capas de red y aplicación y adopta del estándar IEEE 802.15.4 las capas PHY y MAC.

3.2.3.2.1.3.1. CAPA PHY

Es la capa más cercana al hardware y es donde tiene lugar el control y las comunicaciones del transceptor. Esta capa es la responsable de activar la transmisión y recepción de paquetes del sistema. Además, también selecciona el canal de frecuencia y se asegura que éste no es usado por otros dispositivos de la red.

3.2.3.2.1.3.2. CAPA MAC

Esta capa tiene la misión de proveer servicios a las capas superiores para que éstas se encarguen tanto del manejo de los datos que son transferidos en una red WSN, como de las primitivas para que un sistema operativo administre estas dos capas (capa física y de enlace de datos).

3.2.3.2.1.3.3. CAPA NWK

La capa NWK hace de interfaz entre la capa MAC y la capa APL y es la responsable de gestionar la formación de redes y del routing. El routing es el proceso de seleccionar el camino a través del cual viajará el mensaje hasta el dispositivo al que va dirigido.

En esta capa se brindan los métodos necesarios para: iniciar la red, unirse a la red, enrutar paquetes dirigidos a otros nodos y proporcionar los medios para garantizar la entrega del paquete al destinatario final.

El Zigbee Coordinator y los routers son los responsables de este proceso. El coordinador Zigbee también asigna en la capa NWK las direcciones a los dispositivos en su red.

3.2.3.2.1.3.4. CAPA APL

Es la capa más alta del protocolo y alberga los objetos de la aplicación y la que hace a los dispositivos versátiles. Además, puede haber hasta 240 objetos de aplicación en un único dispositivo.

El estándar Zigbee ofrece la opción de utilizar perfiles en el desarrollo de la aplicación. Un perfil de aplicación permite la interoperabilidad entre productos desarrollados por diferentes fabricantes. Por ejemplo, si dos fabricantes utilizan el mismo perfil de aplicación, sus productos serán capaces de interactuar entre sí como si hubieran sido fabricados por el mismo fabricante.

3.2.3.2.1.4. SEGURIDAD DE RED

La seguridad en las comunicaciones inalámbricas es una preocupación normal por la naturaleza de la propia comunicación. Hay dos aspectos importantes respecto a la seguridad: la confidencialidad y la autenticación de datos.

Para resolver el problema de la confidencialidad, el estándar IEEE 802.15.4 soporta el uso del Advanced Encryption System [15](AES), para codificar el mensaje. El AES en un algoritmo de cifrado que modifica un mensaje con una cadena de bits conocida como la clave de seguridad, y sólo el destinatario será capaz de recuperar el mensaje original.

Por lo que respecta a la autenticación de datos, en criptografía, un código de autenticación de mensajes (MIC), es una pequeña pieza de información que se utiliza para autenticar un mensaje.

3.2.3.2.1.5. PROFUNDIDAD DE RED

La profundidad de un dispositivo en una red se define como el número mínimo de saltos necesarios para alcanzar el coordinador ZigBee. Por lo tanto, la profundidad máxima de la red es el número de saltos desde Coordinador hasta dispositivo más alejado.

Para clarificar el concepto, se muestra la siguiente ilustración en la que se muestra una red ZigBee en la que cada nivel está definido por la distancia entre nodos:

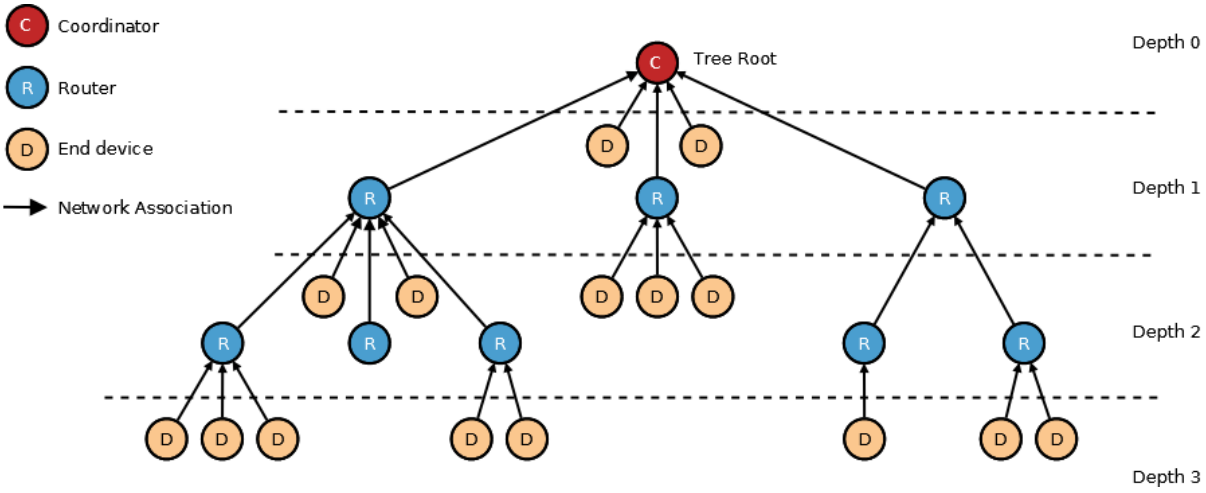


Ilustración 28: Profundidad de una red ZigBee

Lo que caracteriza una red se determina por tres parámetros:

- La profundidad de red.
- El número máximo de "hijos" que un "padre" puede aceptar.
- La profundidad que ocupa el dispositivo en la propia red.

3.2.3.2.2. DIGIMESH

DigiMesh es un protocolo de red propietario de la marca Digi basado en ZigBee. Se usa en soluciones que requieren conectividad inalámbrica.

Soporta múltiples topologías como punto a punto, punto a multipunto y malla. También proporciona modos de ahorro de energía, por lo que es ideal para aplicaciones en las que el consumo es algo crucial.

DigiMesh está disponible en plataformas con mayor rango de RF y posee más opciones de velocidad de datos.

DigiMesh®

Ilustración 29: Logo DigiMesh

La carga útil del paquete de datos es generalmente mayor, lo que puede mejorar el rendimiento de las aplicaciones que envían bloques de datos más grandes. Además, DigiMesh utiliza un método de direccionamiento simplificado, que mejora la configuración de la red y la resolución de problemas.

Sus principales características son:

- ***Self-healing***: referido a la capacidad de añadir o quitar un nodo de la red en cualquier momento sin que ésta falle.
- **Arquitectura punto a punto**: no existen jerarquías o relaciones padre-hijo entre los nodos.

- **Fácil de usar**: debido a que no hay jerarquías, una red de malla está más simplificada.
- ***Route Discovery***: las rutas se descubren y se crean sólo cuando es necesario, eliminando la necesidad de mantener un mapa de la red.
- ***Selective acknowledgments***: sólo los nodos de destino responderán a las peticiones de rutas.
- **Fiable**: las confirmaciones permiten el envío correcto de los datos.
- **Modo dormido**: soporta modos de bajo consumo con tiempos de encendido sincronizados, así como tiempos de hibernación variables.

3.2.3.2.2.1. NODOS

DigiMesh también diferencia entre 3 tipos de nodos, pero normalmente se suele usar el tipo Router. La ventaja es que se pueden construir redes homogéneas, es decir, al ser todos routers, los nodos pueden encaminar paquetes. Además, no existen relaciones padre-hijo, por lo que la red es más simple. Los nodos se pueden configurar para su uso en dispositivos que necesiten ahorrar energía. En la siguiente ilustración se muestra un ejemplo de red DigiMesh homogénea:

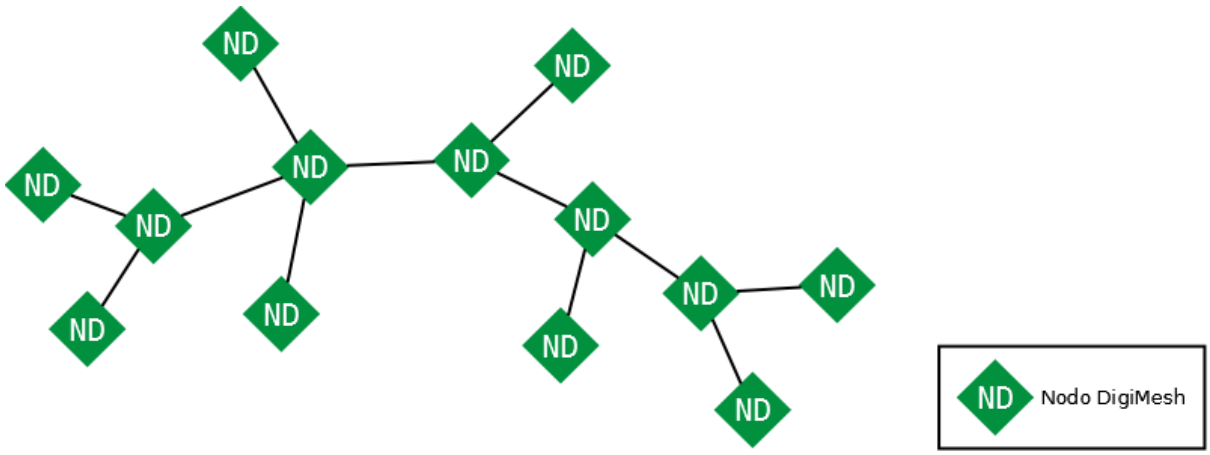


Ilustración 30: Red DigiMesh homogénea

Las principales ventajas de este tipo de nodos consisten en:

- La configuración de la red es muy sencilla.
- Permite expandir la red con mayor flexibilidad.
- Más fiable en entornos en los que las comunicaciones puedan ser interrumpidas o los nodos dañados.
- Todos los nodos pueden ponerse en estado de hibernación para reducir el consumo. La hibernación no es sincronizada, sino que se hace mediante un proceso de elección, lo que permite que la red opere de manera autónoma.

3.2.3.3. WI-FI VS BLUETOOTH VS ZIGBEE

Una vez explicados los diferentes protocolos y haber repasado las características de los mismos, se puede hacer una comparativa entre ellos. En la siguiente table se muestra un cuadro resumen que compara los aspectos más relevantes de las tres tecnologías:

	Wi-Fi	Bluetooth	ZigBee
Velocidad	<50 Mbps	1 Mbps	< 250 kbps
Número de nodos	32	8	25 / 65535
Duración de batería	Horas	Días	Años
Consumo transmitiendo	400 mA	40 mA	30 mA
Consumo en reposo	20 mA	0.2 mA	3 µA
Precio	Caro	Medio	Barato
Configuración	Compleja	Compleja	Simple
Aplicaciones	Internet en edificios	Informática y móviles	Domótica y monitorización

Tabla 3: Tabla comparativa entre Wi-Fi, Bluetooth y Zigbee

3.2.3.4. ZIGBEE VS DIGIMESH

Después de explicar ambos protocolos, se pueden discernir una serie de diferencias entre ellos. Cada uno tiene sus ventajas e inconvenientes y se adaptarán mejor a ciertos escenarios, por lo que es importante decidir si lo que ofrece cada protocolo se ajusta a nuestras necesidades.

A continuación se muestra una tabla comparativa de ambos protocolos:

	ZigBee® Mesh	DigiMesh
Tipos de nodos y beneficios	Coordinadores, Routers, End Devices. Los End Devices son mucho más baratos por su reducida funcionalidad	Un solo tipo de nodo. Más flexible a la hora de expandir la red. Simplifica la configuración de la red. Incrementa la fiabilidad en entornos en los que los routers pueden tener interferencias o resultar dañados
Modo dormido, duración de la batería	Sólo los End Devices pueden hibernar	Todos los nodos pueden hibernar. No hay puntos de fallos relacionados con coordinadores que mantengan la sincronización
Actualizaciones de Firmware Over-the-Air	Sí	No
Opciones de largo alcance	La mayoría de dispositivos ZigBee tienen un rango menor a 3.2 Km entre saltos	Rango de hasta 64 Km entre saltos (según el dispositivo)
Carga útil, Rendimiento	Up to 80 bytes	Hasta 256 bytes. Se mejora mediante aplicaciones que envían bloques de datos mayores
Tamaño del código	Más largo. Menos espacio para que crezcan las prestaciones	Más pequeño (alrededor de la mitad). Más espacio disponible
Frecuencias soportadas y tasas	Predomina 2.4 GHz (250 kbps). 900 MHz (40 Kbps) y 868 MHz	900 MHz (10, 125, 150 Kbps). 2.4 GHz (250 Kbps)

de datos RF	(20 Kbps) no están disponibles globalmente	
Seguridad	Encriptación AES. La red se puede bloquear y así prevenir que otros nodos se unan a la misma	Encriptación AES
Interoperabilidad	Gran potencial para interoperabilidad entre fabricantes.	Propietario
Tolerancia a interferencias	Direct-Sequence Spread Spectrum (DSSS)	900 MHz: Frequency-Hopping Spread Spectrum (FHSS). 2.4 GHz: Direct-Sequence Spread Spectrum (DSSS).
Direccionamiento	2 capas. MAC address (64 bit) y Network address (16 bit)	MAC address (64 bit)
Mantenimiento	Más herramientas disponibles en el Mercado	El direccionamiento más simple ayuda en la diagnosis y la configuración de la red

Tabla 4: Comparativa entre ZigBee y DigiMesh

Las conclusiones que se desprenden de la tabla y que sirven para elegir un protocolo u otro, son las siguientes:

- Si se necesita un estándar abierto, utilizar dispositivos de diferentes fabricantes y tener actualizaciones automáticas over-the-air, ZigBee es el protocolo adecuado.
- Por otro lado, si se necesita ahorrar energía, simplificar el proceso de configuración y expansión de la red, más velocidad y mayor rango entre nodos, DigiMesh es la opción a elegir.

4. NORMATIVA Y LEGISLACIÓN

A continuación se expone la legislación vigente que afecta a este Trabajo Fin de Grado.

4.1. GNU GENERAL PUBLIC LICENSE

La Licencia Pública General GNU[16] o más conocida por su nombre en inglés GNU GPL, es una licencia que garantiza al usuario final el libre uso, estudio, distribución y modificación del software.

Ilustración 31: Logotipos de los diferentes tipos de licencias GPL

La GPL tiene como mayor característica el hacer hincapié en el copyleft, es decir, sólo permite que las copias y derivados de una obra bajo GPL usen a su vez la misma licencia.

Para el desarrollo de este proyecto, se utiliza Arduino IDE, que posee licencia GNU.

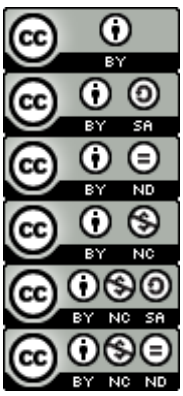
4.2. CREATIVE COMMONS

Creative Commons[17] es una organización que permite usar y compartir creaciones a través de licencias de derechos de autor simples y estandarizadas. Inspiradas en la licencia GPL, permite la distribución y el uso de contenidos.

Ilustración 32: Logotipo de Creative Commons

Cada autor elige la manera en la que quiere compartir su trabajo, eligiendo los términos y condiciones que mejor satisfagan sus necesidades. Así, es posible elegir entre seis licencias diferentes:

- Reconocimiento (CC BY)
- Reconocimiento-CompartirIgual (CC BY-SA)
- Reconocimiento-NoDerivadas (CC BY-ND)
- Reconocimiento-NoComercial (CC BY-NC)
- Reconocimiento-NoComercial-CompartirIgual (CC BY-NC-SA)
- Reconocimiento-NoComercial-NoDerivadas (CC BY-NC-ND)



En este proyecto se utiliza una placa hardware llamada Arduino UNO, que posee licencia Creative Commons-Compartir Igual.

4.3. LICENCIA ZIGBEE[18]

Para usos no comerciales, la especificación de ZigBee está disponible para el público general. Sin embargo, para acceder a especificaciones no publicadas y crear productos para el mercado usando estas especificaciones, se necesita una membresía anual.

Esta membresía anual causa conflictos con la GNU GPL y otras licencias de software libre.

4.4. SOFTWARE PROPIETARIO

El software propietario[19] es aquel al que no se puede acceder libremente a su código fuente, pues sólo se encuentra a disposición del desarrollador. Por tanto, tampoco se permite su modificación o adaptación.

El poseedor de los derechos de autor sobre este tipo de software controla y restringe los derechos del usuario sobre el programa, por lo que normalmente éste último sólo podrá ejecutarlo bajo ciertas condiciones.

4.4.1. DIGI

Digi es una empresa americana de comunicaciones y tecnología, fundada en 1985. Se dedica principalmente a soluciones de comunicación embebidas externas (tanto cableadas como inalámbricas), así como a productos empresariales.

Ilustración 33: Logo Digi

La empresa Digi tiene copyright sobre todos sus productos, tanto software como hardware, según se menciona en el acuerdo de uso de licencia de sus productos[20], tal y como se desprende del siguiente fragmento:

> Toda la propiedad intelectual del hardware Digi pertenece a la marca Digi. El software Digi está protegido por las leyes de copyright y por tratados internacionales. El software no es open source a menos que se especifique. Por lo tanto, se debe tratar el software de Digi como cualquier otro material con copyright.

Por tanto, los módulos XBee mencionados, el protocolo DigiMesh y el software

XCTU poseen copyright y deben usarse bajo las condiciones del propietario.

5. APORTACIONES AL ENTORNO SOCIO-ECONÓMICO

El estudio y posterior desarrollo del sistema implementado en el proyecto Dunas ha dotado al sector topográfico de una herramienta capaz de tomar mediciones exactas sobre cómo afecta en las dunas costeras la velocidad y la dirección del viento.

Además, dicha herramienta puede ser aplicada en otros ecosistemas para obtener distintas tomas de datos, tales como terrenos montañosos. Esto facilitaría por ejemplo, al sector de la topografía, de un estudio completo sobre la erosión de cara a futuros proyectos urbanísticos.

No obstante, la mejora de este sistema dotaría una mayor independencia en cuanto a la sincronización de los sensores de viento y una mayor robustez. Así pues, de tener éxito, este proyecto proporcionaría a distintos sectores profesionales una herramienta actualizada capaz de adaptarse en función del número de sensores de viento que se posean y además, el sistema sería lo suficientemente inteligente a la hora de configurar las rutas más óptimas para hacer llegar a la estación base los datos obtenidos.

En este aspecto, este proyecto puede a su vez suponer un interesante avance, pues serviría de ejemplo de transferencia tecnológica a sectores profesionales como la AEMET, Institutos de Geografía y Física, etcétera.

6. ANÁLISIS

En esta fase de análisis, se describirá el escenario actual sobre el que se va a trabajar, se estudiará el protocolo de red inalámbrica DigiMesh, pues el que se utilizará con los módulos XBee disponibles y además se estudiarán las librerías libxbee3 y XBee para Arduino, para C++ y Arduino respectivamente.

6.1. ESCENARIOS

En el escenario descrito en puntos anteriores, referente a la medición de la dinámica eólica de la zona de Playa del Inglés y Maspalomas, es necesario describir el estado actual del mismo y la evolución que se plantea en este TFG.

6.1.1. ESTADO ACTUAL

El sistema actual consiste en 11 sensores de viento y una estación base, donde cada sensor envía datos periódicamente a la estación.

Para poder entender el sistema en su totalidad, es necesario describir los componentes que lo forman:

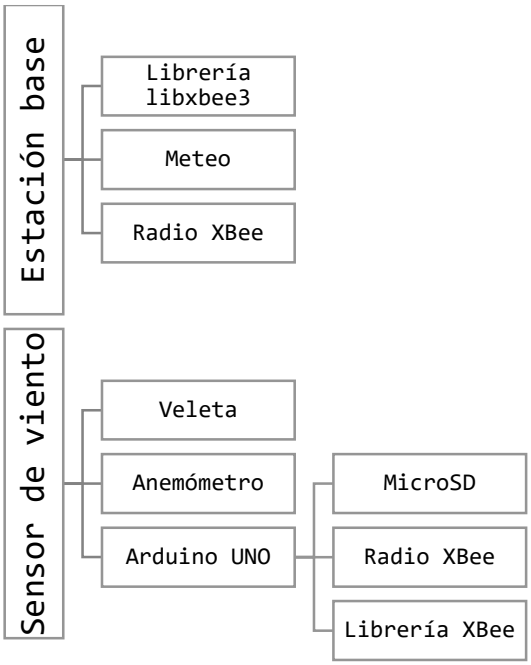


Ilustración 34: Esquema de componentes del sistema

En el esquema anterior se muestran los diferentes componentes que forman el sistema, a saber:

- **Estación base**, normalmente un ordenador portátil, que tiene:
  - > Una librería llamada libxbee3 que permite interactuar con radios XBee mediante programas escritos en C++.
  - > Un software hecho a medida llamado meteo que permite monitorizar todo el sistema.

  - > Módulo de radio XBee, con el que se enviarán y recibirán paquetes.
- Los **sensores de viento**, compuestos por:
  - > **Veleta** y **anemómetro.**
  - > Una placa **Arduino**, que posee:
    - MicroSD
    - Módulo de radio XBee, que va conectado al Arduino. Permite la comunicación inalámbrica entre los sensores y la estación base.
    - **Librería** XBee para Arduino que permite programar en Arduino UNO para poder interactuar con el módulo de radio y los paquetes enviados y recibidos.

Una vez que se conocen los tipos de elementos que componen la red de sensores, es momento de hablar de la topología de la red. La topología actual es punto a punto, donde todos los sensores se comunican directamente con la estación base, tal y como se puede apreciar en la siguiente ilustración:

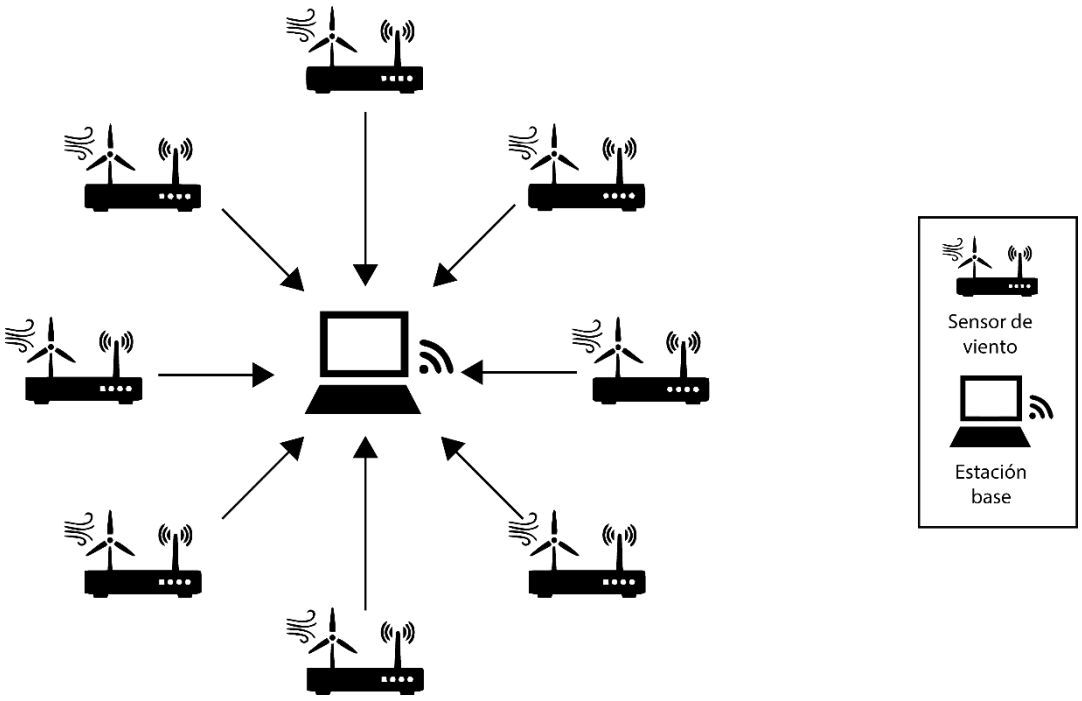


Ilustración 35: Estado actual del Proyecto Dunas

Esta topología tiene varias limitaciones: por un lado, la colocación de cada sensor debe realizarse con una estación total de topografía; por otro lado, el sistema se basa en un sistema de sincronización mediante envío de paquetes, que impide añadir nodos una vez que el sistema está funcionando, lo cual obliga a reiniciarlo todo; y lo más importante, todas las estaciones deben comunicarse directamente con la estación base, acotando el área efectiva de funcionamiento.

El funcionamiento del sistema, a grandes rasgos, consiste en:

1. Encender la estación base.
2. Encender y calibrar cada sensor.
3. Cada sensor envía un paquete de inicialización a la base.
4. En la base se visualiza el estado del sensor.
5. La base envía un paquete de vuelta a ese sensor indicando la dirección MAC con la que éste deberá comunicarse de ahora en adelante.
6. Una vez añadidos todos los sensores, desde la base se envía un paquete de sincronización para que todos tengan la misma hora, con una precisión de milisegundos.
7. Los sensores empiezan a enviar datos periódicamente (periodo configurable en el sistema actual).

En el siguiente diagrama se puede observar el proceso descrito anteriormente:

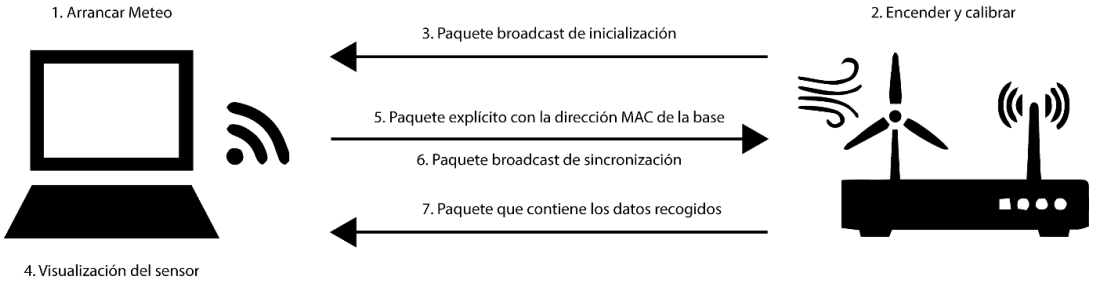


Ilustración 36: Proceso de inicialización del sistema de sensores de viento

El hecho de que los sensores estén limitados a un área en el que todos deben ver la estación base, ha obligado a que los experimentos realizados hasta ahora se hayan realizado en parcelas de unos 50 m. de lado, en la zona de Playa del Inglés y Maspalomas, tal y como se muestra en la siguiente imagen satélite:

Ilustración 37: Ubicación de la parcela de estudio

Por tanto, el siguiente paso a seguir es extender el área de trabajo con el que poder utilizar la red de sensores de viento.

6.1.1. EVOLUCIÓN DEL SISTEMA

Lo que se pretende hacer en base al sistema actual, es implementar el uso del protocolo DigiMesh, que permitirá crear una red malla y así ampliar el área efectiva de trabajo de todo el sistema.

DigiMesh es un protocolo de red inalámbrica que deriva de ZigBee, pero perteneciente a la marca Digi, que es el fabricante de los módulos que se usan actualmente en los sensores de viento. Este protocolo permite crear redes de nodos en los que éstos se interconectan entre sí, permitiendo reenviar paquetes entre un origen y un destino usando nodos intermedios.

El que cada nodo actúe como enrutador de paquetes, permitiría establecer redes que abarquen grandes distancias, siempre y cuando cada radio vea al menos otras 2, para así enlazar unas con otras y que al menos una de ellas pueda comunicarse con la estación base para la recepción de datos. Un ejemplo de lo descrito se muestra a continuación:

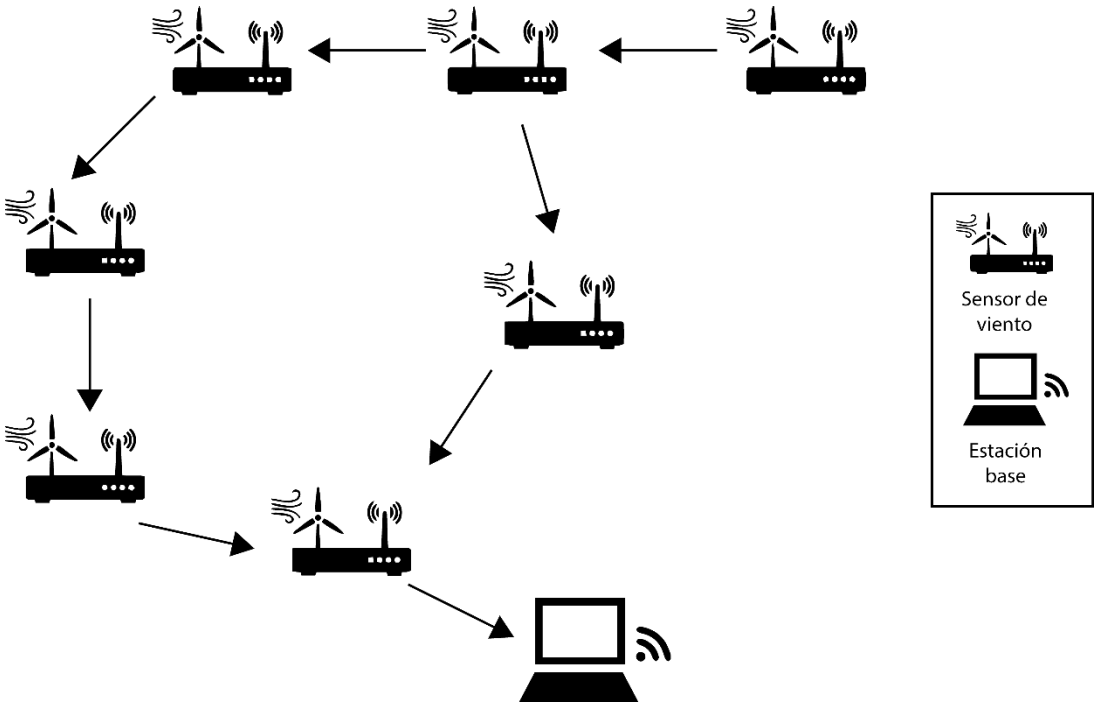


Ilustración 38: Ejemplo de red DigiMesh

Sabiendo esto, sería posible desplegar un número indefinido de estaciones a lo largo de la costa de Playa del Inglés y Maspalomas. Con la implementación de DigiMesh y las ventajas que ofrece el protocolo, el establecimiento de la red debería ser relativamente sencillo, al ofrecer funciones propias para el descubrimiento de nodos y caminos de enrutado.

Ilustración 39: Línea de costa que se desea cubrir con los sensores de viento

En la imagen anterior se muestra la zona de costa que se quiere abarcar, que cubriría aproximadamente desde el Faro de Maspalomas hasta la zona del Anexo 2 de Playa del Inglés.

La única restricción a tener en cuenta es la distancia máxima que pueden cubrir las radios XBee, de las que dispone el sistema actual, que actualmente se establece en unos 1.6 Km ideales, sin obstáculos de por medio. En la práctica, el rango efectivo de las radios está en torno a unos 800 m., dato recogido en una de las pruebas realizadas en la citada zona del sur de la isla.

Sin embargo, uno de los problemas que podría surgir a partir de esta implementación, es la carga de trabajo que tendrán los sensores que estén más cerca de la estación base, pues tendrán que procesar los paquetes de todos los sensores de la red. Un ejemplo de carga se puede observar en la siguiente imagen, donde los nodos más cercanos procesan más paquetes:

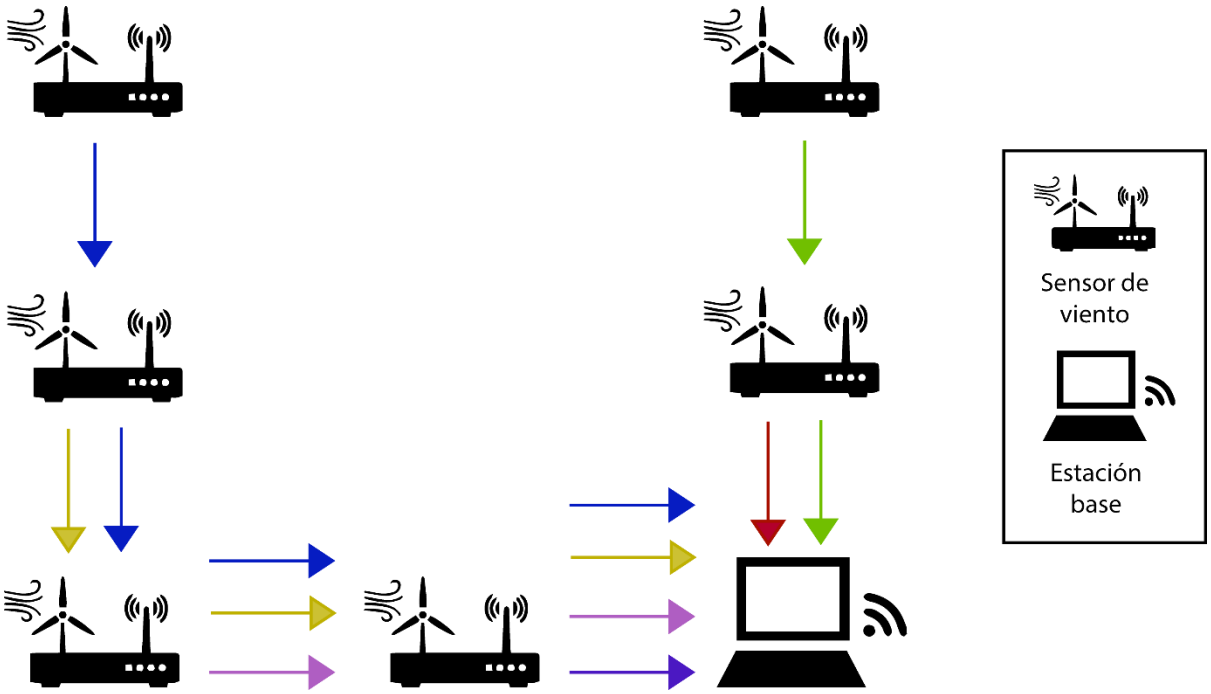


Ilustración 40: Carga de trabajo de los sensores de viento

Además, esta carga adicional podría tener un efecto colateral en el uso de la batería de los sensores de viento, pues el uso de las radios en estos nodos sería prácticamente continuo, dejando poca posibilidad para dejar la radio en modo dormido.

Otro de los problemas que surge de esta implementación y de las grandes distancias, es la sincronización de los nodos. Si se siguiera con la filosofía actual de sincronización a través de envío de paquetes, habrá retardo en los envíos de los paquetes entre nodo y nodo. Esto provocaría que todos los sensores estén retrasados con respecto al primero. Para solucionar esto, es posible dotar a cada sensor de un GPS, que se manejaría a través del Arduino y con el que se le proporcionaría el tiempo UTC. Esta solución trae consigo dos ventajas: la primera es que se eliminaría la fase de envío de paquetes de sincronización y la segunda, que sería posible añadir nodos una vez que el sistema está funcionando, sin interrumpir el funcionamiento del resto de dispositivos de una misma sesión de medida.

6.2. REQUISITOS

Los requisitos necesarios para llevar a cabo la implementación completa de DigiMesh en el sistema actual, son los siguientes:

- Librería libxbee3 para C++.
- Estación base con software específico para monitorización, llamado meteo.
- Librería XBee para Arduino para Arduino.
- Sensores de viento basados en Arduino.
- Módulo de radio XBee con firmware con soporte DigiMesh.

6.2.1. FUNCIONES DEL SISTEMA

En este apartado se explicarán las funciones del sistema, que se enumeran a continuación:

- Detección y configuración de la red DigiMesh
- Sincronización y adición de sensores de viento
- Envío/recepción de paquetes a través de saltos
- Recálculo de rutas en caso de caída de nodos

6.2.1.1. DETECCIÓN Y CONFIGURACIÓN DE LA RED DIGIMESH

La principal ventaja de DigiMesh, es que el propio protocolo permite descubrir las radios cercanas y que éstas, automáticamente, descubran a su vez a las radios vecinas.

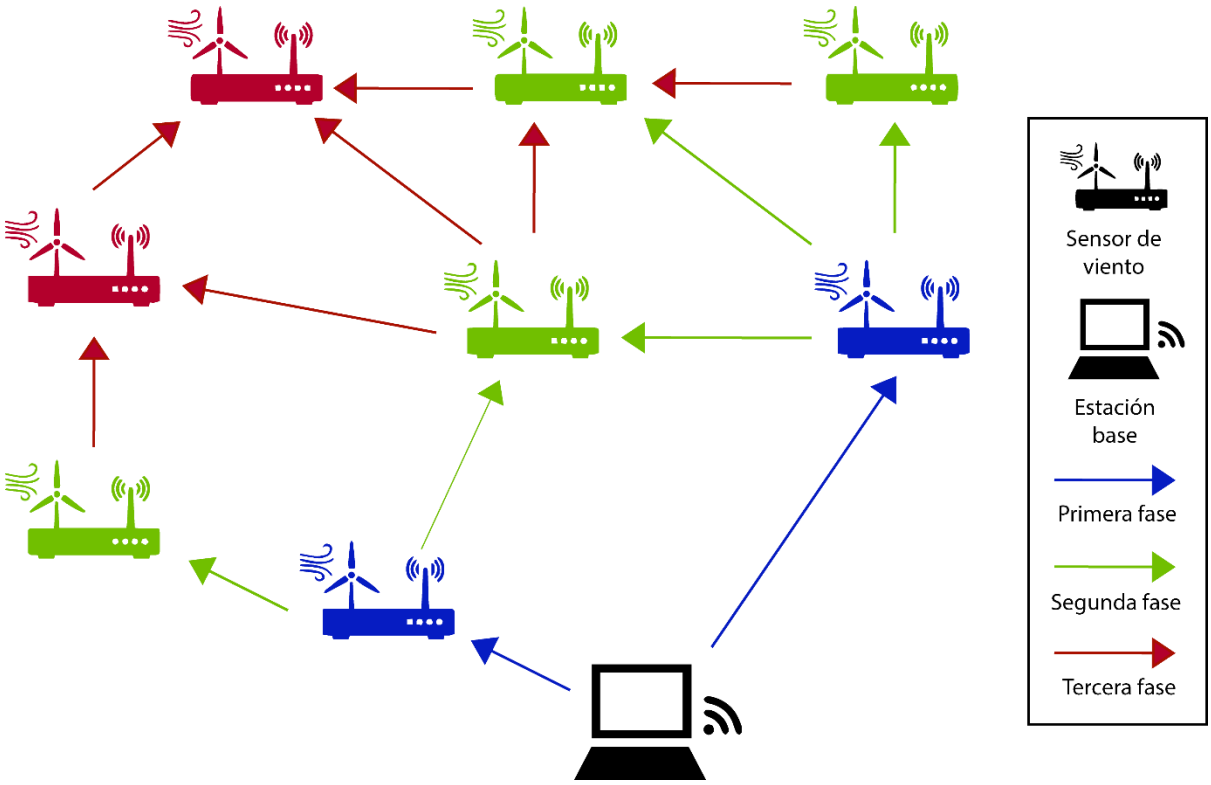


Ilustración 41: Proceso de descubrimiento de la red DigiMesh

En la ilustración anterior se muestra un ejemplo de cómo sería el descubrimiento de una red de varios sensores. Primero se descubrirían los más cercanos a la estación base, y luego éstos, a su vez, irán descubriendo a los más cercanos y así sucesivamente hasta que todos los nodos sean encontrados.

Respecto a la configuración de la red, el protocolo DigiMesh permite, una vez establecida la misma, indicar a cada nodo con qué nodo comunicarse, y es el protocolo el que se encargará de calcular la ruta más óptima hacia el destino, que será aquella en la que haya que pasar por el menor número de nodos posible, o la calidad de las conexiones sea mejor. En la siguiente ilustración se muestran dos caminos: el camino más óptimo desde uno de los sensores hasta la estación base y otro más largo:

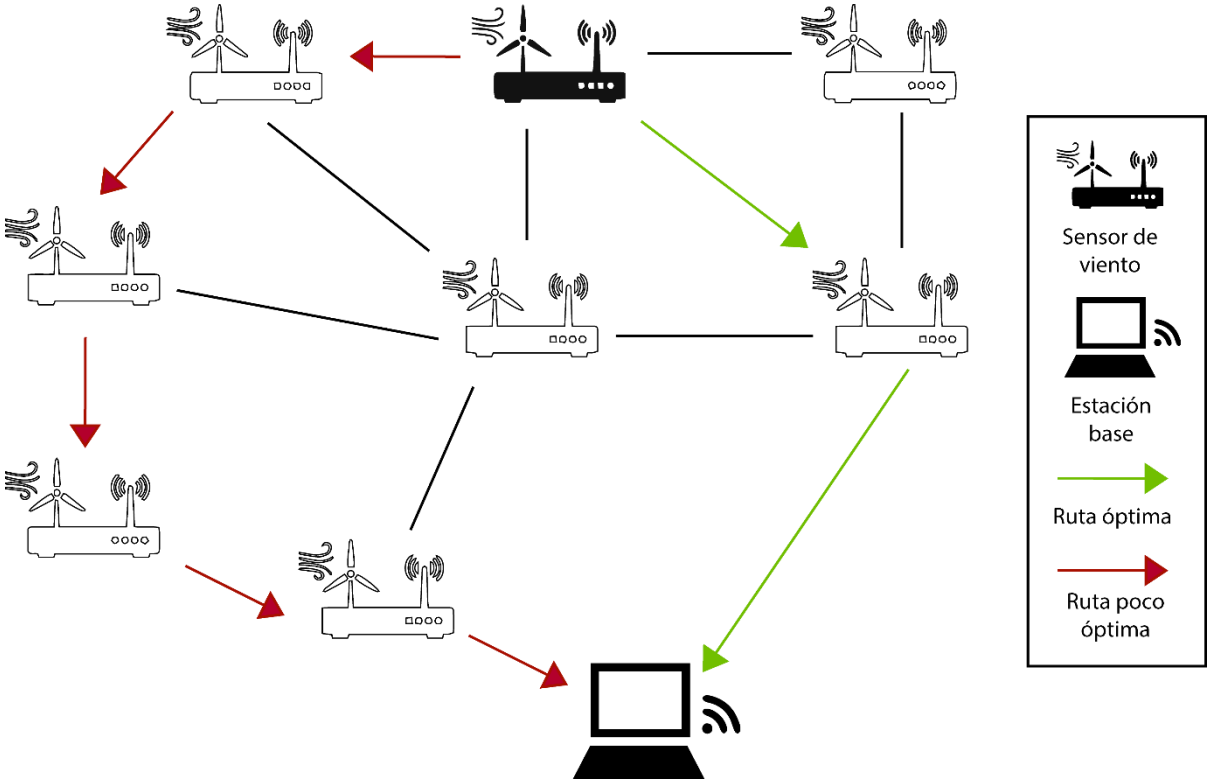


Ilustración 42: Tipos de rutas entre nodos DigiMesh

6.2.1.2. SINCRONIZACIÓN Y ADICIÓN DE LOS SENSORES

Como se comentó en puntos anteriores, la implementación de DigiMesh permitirá ampliar el área en el que utilizar la red de sensores. Al tener los nodos muy separados, el retardo de los paquetes entre nodo y nodo se verá incrementado con la distancia.

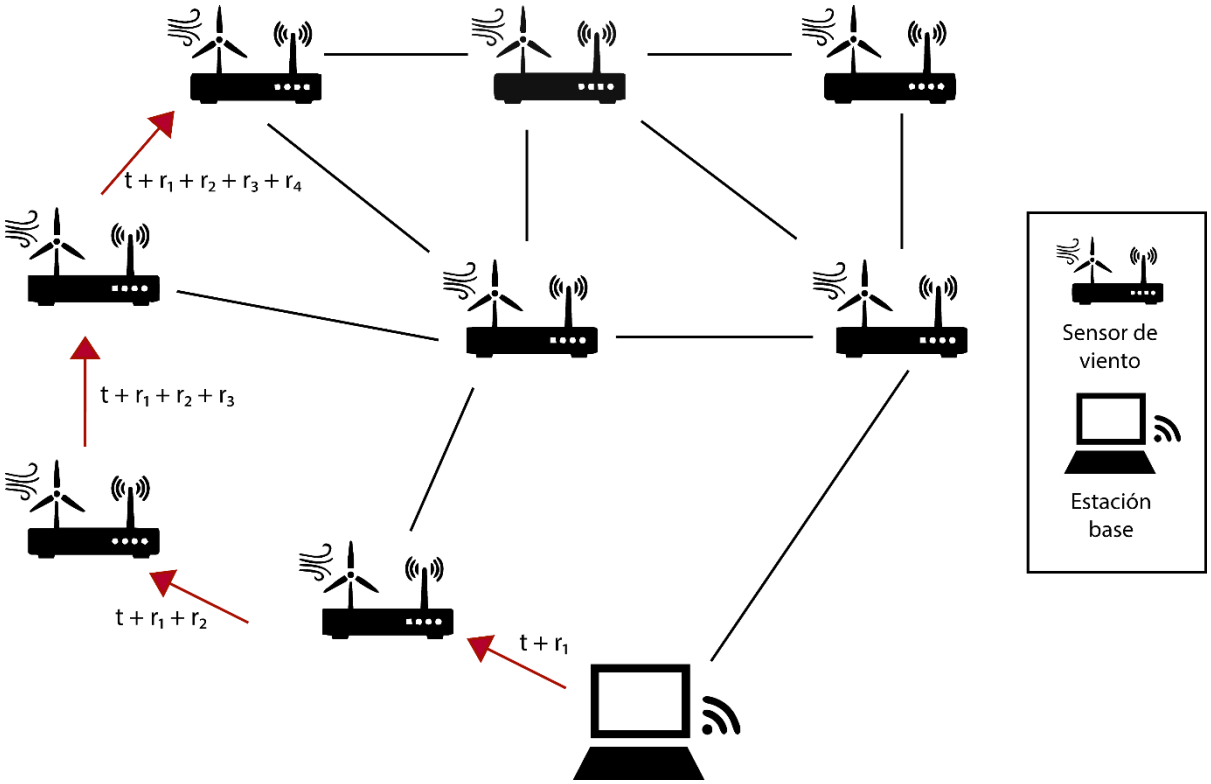


Ilustración 43: Acumulación de retardos en la sincronización

Como se observa en la imagen anterior, al sincronizar desde la estación base, la sincronización mediante paquetes enviando una marca de tiempo se quedaría obsoleta, pues el envío entre nodos añadiría un retardo en cada salto, que se iría acumulando a medida que se vaya avanzando por la red, provocando que, cuanto más lejos se encuentre un nodo, más retrasado estará con respecto a los más cercanos a la estación base.

Este problema se puede solventar incluyendo un GPS en cada sensor de viento. Los sensores podrían unirse a la red con el sistema en funcionamiento, ya que podrían solicitar el tiempo UTC que proporcionan los GPS en cualquier momento. Este tiempo estaría coordinado en todo momento, pues lo obtendrían de la propia red GPS. Un ejemplo de este comportamiento se puede observar en el siguiente esquema:

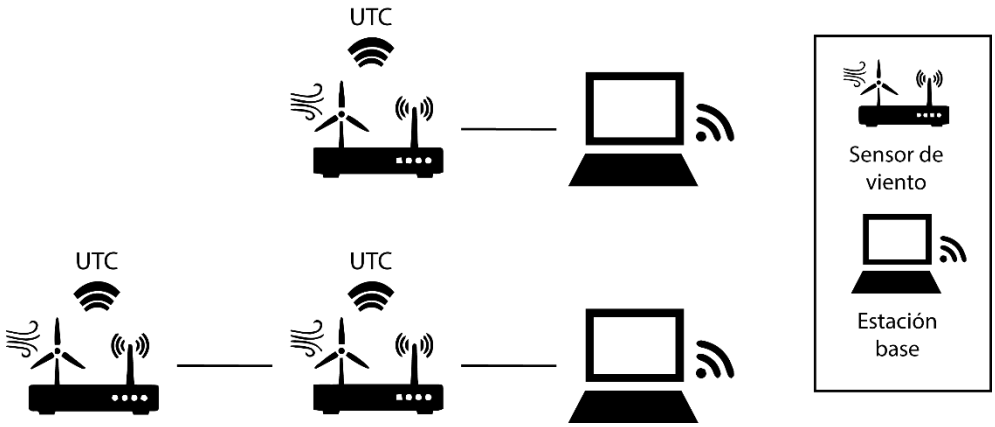


Ilustración 44: Sincronización de nodos mediante GPS

Estando la red en funcionamiento, los nodos poseen el tiempo UTC que les proporciona el GPS. Cuando se añade uno nuevo, éste solicita el tiempo UTC, que será igual al de todos los nodos ya añdidos, pues lo proporciona la red de satélites.

Esto garantiza que todos tengan la misma marca de tiempo y además eliminaría la necesidad de estar enviando paquetes de sincronización, por lo que la configuración ganaría en simplicidad.

6.2.1.3. ENVÍO/RECEPCIÓN DE PAQUETES A TRAVÉS DE SALTOS

La característica más importante de la red malla es la capacidad de enviar paquetes desde un origen a un destino a través de nodos intermedios, como se ve en la siguiente ilustración:

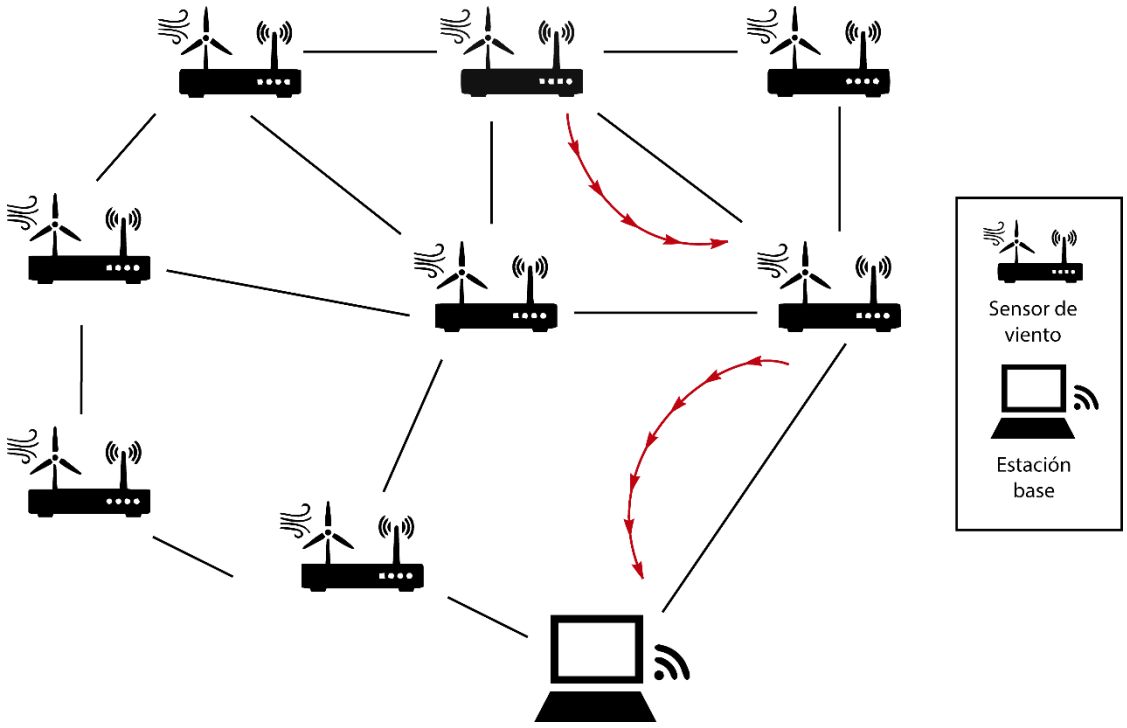


Ilustración 45: Envío de paquetes entre saltos en red DigiMesh

Esta capacidad de la red DigiMesh permitirá ampliar la misma dependiendo de las necesidades. Además, el protocolo permite calcular caminos entre origen y destino automáticamente, siendo éste el más óptimo de entre todos los caminos posibles.

6.2.1.4. RECÁLCULO DE RUTAS EN CASO DE CAÍDA DE NODOS

El protocolo DigiMesh es capaz de recalcular una ruta entre un origen y un destino en caso de que alguno de los nodos intermedios falle, ya sea por tener obstáculos que impidan la comunicación o porque el nodo ha fallado irremediablemente.

En los siguientes diagramas se muestra el proceso de recálculo de ruta en caso del fallo de un nodo:

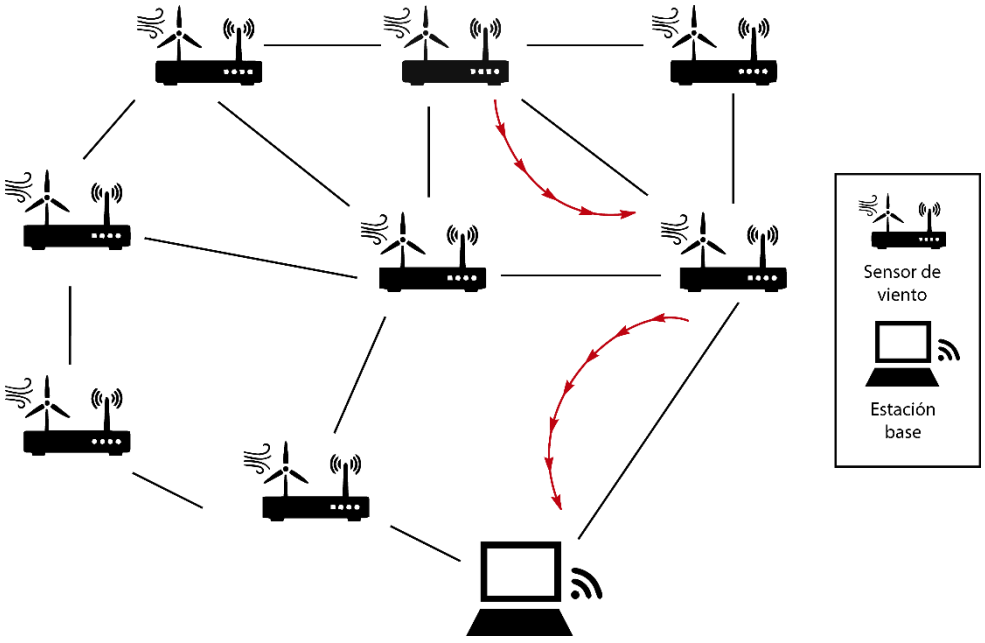


Ilustración 46: Camino entre origen y destino

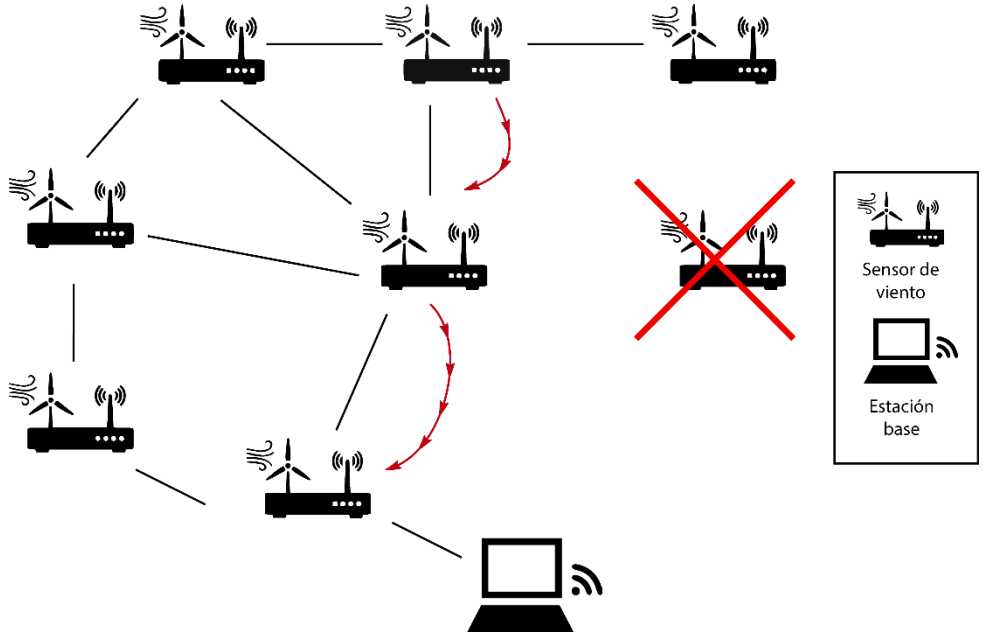


Ilustración 47: Recálculo de ruta al fallar un nodo

6.3. DIGIMESH

Como se ha mencionado varias veces a lo largo de este documento, DigiMesh es un protocolo basado en ZigBee y por lo tanto su similitud es más que notable.

6.3.1. AT COMMANDS

Los comandos AT permiten leer y escribir datos en las radios XBee, para configurarlos según las necesidades. Los comandos más importantes son:

Comando AT	Descripción
CH	Lee/escribe el canal que se va a utilizar
ID	Lee/escribe el identificador de la red
BH	Número máximo de saltos en un broadcast
DM	Una máscara para habilitar opciones de DigiMesh
DH	Parte superior de la dirección de 64 bits del destino
DL	Parte inferior de la dirección de 64 bits del destino
AG	Envía un broadcast a través de la red, de manera que si la dirección especificada coincide con los módulos, éstos cambiarán los valores de sus parámetros DH y DL a los SH y SL del origen
ND	Descubre la red mediante broadcast. Reporta todas las radios que descubre.
FN	Permite que una radio encuentre a otras en un radio de 1 salto.

Tabla 5: Comandos AT de DigiMesh que admite la radio XBee

6.3.2. API FRAMES

Por ejemplo, muchos de los frames (paquetes) que posee DigiMesh coinciden en IDs con respecto a ZigBee.

A continuación se muestran los diferentes APIs existentes:

API frame name	API ID
AT Command	0x08
AT Command-Queue Parameter Value	0x09
Transmit Request	0x10
Explicit Addressing Command Frame	0x11
Remote Command Request	0x17
AT Command Response	0x88
Modem Status	0x8A
Transmit Status	0x8B
Route Information Packet	0x8D
Aggregate Addressing Update	0x8E
Receive Packet (AO=0)	0x90
Explicit Rx Indicator (AO=1)	0x91
I/O Data Sample Rx Indicator	0x92
Node Identification Indicator (AO=0)	0x95
Remote Command Response	0x97

Tabla 6: API frames de DigiMesh

Los paquetes que más se van a utilizar en este trabajo, se enumeran a continuación:

- **Transmit Request (0x10)**: este frame envía datos a una dirección específica
- **Transmit Status (0x8B)**: cuando se envía un frame del tipo Transmit Request, el dispositivo que lo recibió genera este frame, indicando si llegó satisfactoriamente o no.
- **Receive Packet (0x90)**: se genera cuando una radio recibe un paquete.

La principal diferencia entre los distintos frames, es su estructura interna. Por ejemplo, el frame Transmit Request (`0x10`) tiene la siguiente estructura:

Frame data fields	Offset	Descripción
Frame type	3	0x10
Frame ID	4	Identifica el paquete para relacionarlo con el ACK. Si vale 0, no envía respuesta de vuelta
64-bit destination address	5-12	Primero el MSB, luego el LSB. Especifica la dirección de destino. Si es broadcast, es 0x000000000000FFFF
Reserved	13-14	0xFFFE
Broadcast radius	15	Especifica el máximo número de saltos en un broadcast. Si es 0, se usa el valor máximo
Transmit options	16	0x01 = ACK deshabilitado 0x02 = Descubrimiento de la red deshabilitado
RF data	17-n	Hasta NP (parámetro de la radio) bytes por paquete

Tabla 7: Estructura del frame 0x10

El frame Transmit Status (`0x8B`) tiene esta estructura:

Frame data fields	Offset	Descripción
Frame type	3	0x8B
Frame ID	4	Identifica la interfaz serie. Si es 0, no se envía paquete de respuesta
16-bit destination address	5	La dirección de red de 16 bits hacia la que se envió el paquete. Si no hubo éxito al entregar el paquete, estos bytes toman el valor 0XFFFD
	6	
Transmit retry count	7	Número de reintentos realizados
Delivery status	8	0x00 = Correcto 0x01 = Fallo en ACK de la MAC 0x02 = Colisión 0x21 = Fallo en ACK de la red 0x25 = Ruta no encontrada 0x31 = Fallo de recurso 0x32 = Fallo interno
Discovery status	9	0x00 = Sin descubrimiento de ruta 0x02 = Descubrimiento de ruta

Tabla 8: Estructura del frame 0x8B

El frame Receive Packet (`0x90`) tiene esta estructura:

Frame data fields	Offset	Descripción
Frame type	3	0x90
64-bit source address	4-11	La dirección de la radio que envía el paquete
Reserved	12-13	Reservado
Receive options	14	0x01 = Paquete recibido 0x02 = Paquete de broadcast
Received data	15-n	Los datos enviados (payload)

Tabla 9: Estructura del frame 0x90

7. DESARROLLO E IMPLEMENTACIÓN

En las fases de desarrollo e implementación se cubrirán aspectos técnicos de las acciones realizadas para llevar a cabo la elaboración de los objetivos de este Trabajo de Fin de Grado.

Primero se describirán los requisitos, tanto hardware como software, que son necesarios para una correcta realización de la implementación.

Luego se describirán una serie de pruebas con el sistema ya en funcionamiento y se analizarán los resultados de las mismas.

7.1. REQUISITOS HARDWARE

Para el correcto desarrollo del código y la ejecución de las herramientas necesarias, es necesario un computador de prestaciones medias, por ejemplo uno con un procesador Intel Core i3 y 2 GB de RAM.

En este caso se ha utilizado un portátil con procesador Intel Core i7 a 2.4 GHz y 12 GB de RAM, pues es el que había disponible.

Ilustración 48: Portátil utilizado

7.1.1. ARDUINO UNO

Arduino/Genuino UNO[21] es una placa que posee un Microcontrolador Atmel AVR ATmega328P. Su atractivo es que es hardware Open Source, es decir, todos los archivos originales de diseño del hardware están disponibles para su consulta y están distribuidos bajo licencia Creative Commons - Compartir Igual[22], tanto para uso personal como comercial.

Posee su propio entorno de desarrollo llamado Arduino IDE, lo que permite programar funcionalidades nuevas y usar la placa para múltiples propósitos.

Todo el sistema, tanto la placa como el entorno de desarrollo, están pensados para fomentar y facilitar el uso de la electrónica para el público en general.

7.1.1.1. ESPECIFICACIONES TÉCNICAS

Respecto a sus características físicas, tiene 14 pines digitales de entrada/salida, 6 entradas analógicas, un cristal de cuarzo de 16 MHz, una conexión USB, un conector de alimentación y un botón de reinicio. En la siguiente imagen se pueden ver los componentes más relevantes de la placa:

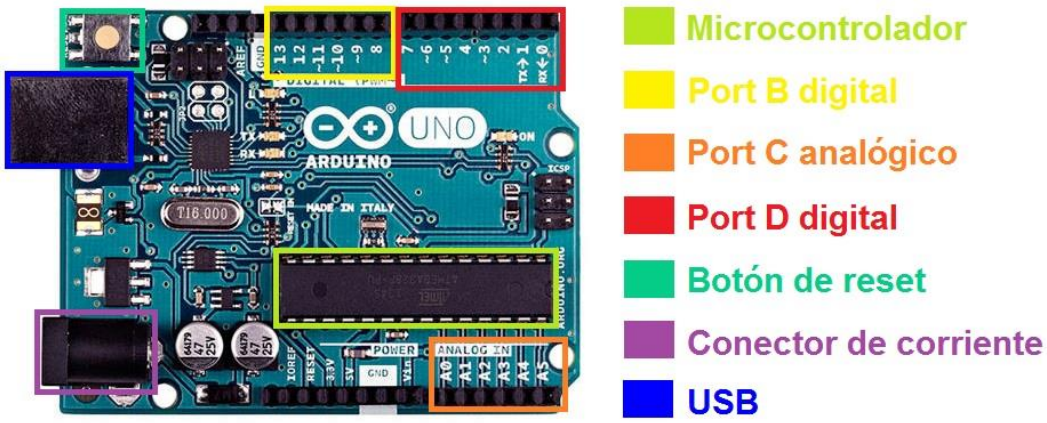


Ilustración 49: Partes más relevantes de la placa Arduino UNO

Para ir más al detalle, en el siguiente cuadro se muestran las características más importantes de la placa:

Voltaje de operación	5V
Voltaje de entrada (recomendado)	7-12V
Voltaje de entrada (límite)	6-20V
Piines digitales de E/S	14 (6 proporcionan PWM)
Pines de entrada analógica	6
Corriente por cada pin E/S	20 mA
Corriente para pin de 3.3V	50 mA
Memoria Flash	32 KB (ATmega328P) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Velocidad del reloj	16 MHz
LED	13
Largo	68.6 mm
Ancho	53.4 mm
Peso	25 g

Tabla 10: Especificaciones técnicas de la placa Arduino UNO

7.1.2. MICROCONTROLADOR ATMEL

La placa Arduino posee un microcontrolador del fabricante ATMEL que se basa en la versión 328P[23] que proviene de la familia de microcontroladores de bajo consumo de 8-bits y alto rendimiento.

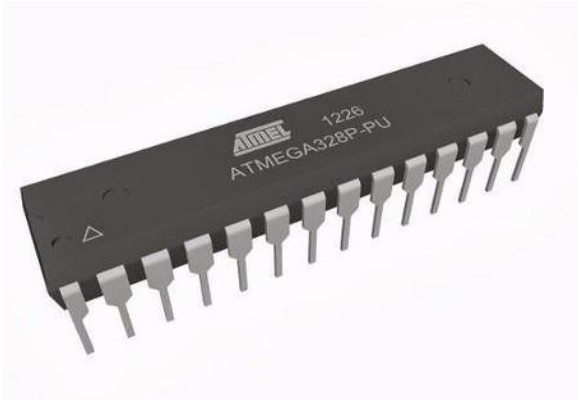


Ilustración 50: Microcontrolador ATMEL ATMega 328P

Su arquitectura es RISC (Reduced Instruction Set Computing) avanzada que posee las siguientes características principales:

- 131 instrucciones.
- Ejecución de ciclo de reloj más simple.
- 32 x 8 registros de trabajo de propósito general.
- Operación completamente estática.
- Hasta 20 MIPS de rendimiento a 20MHz.
- Multiplicador On-Chip de 2 ciclos.

A continuación se muestra una tabla con el resto de características relevantes del microcontrolador:

Microcontrolador ATMEL ATmega 328P
8-bit AVR CPU
RISC instruction set (assembly)
32KB of flash memory
1024B EEPROM
2KB SRAM
20 MHz max. freq. (Usually 16 MHz)
Peripherals
Timers, Counters, Watch dog
6 PWM channels
Serial USART
SPI/I2C communication
Analog-to-Digital Converter (ADC)

Tabla 11: Características técnicas del microcontrolador ATmega328P

7.1.3. XBEE

El módulo utilizado es el modelo XBee Pro DigiMesh 2.4 de la marca Digi, más concretamente, el módulo `XBP24-DMWIT-250`.

Soporta modos de bajo consumo y redes punto a punto o de malla.

Ilustración 51: Módulo Bee Pro utilizado

7.1.3.1. MODOS DE OPERACIÓN

El módulo XBee puede operar de 4 formas diferentes: `Transmit, Receive, Command` y `Sleep`. En la siguiente ilustración se muestran los distintos modos de operación:

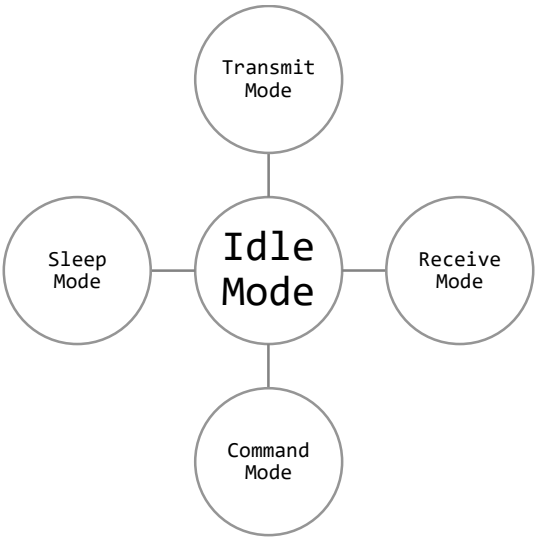


Ilustración 52: Modos de operacíon del módulo XBee

7.1.3.1.1. IDLE

Cuando no se están recibiendo o enviando datos, el dispositivo se encuentra en modo `idle`, escuchando en la radio y en los puertos serie.

El dispositivo cambia de estado bajo alguna de estas condiciones:

- ***Transmit mode**:* se reciben datos por buffer.
- ***Receive mode**:* se reciben datos validos por la antena.
- ***Sleep mode**:* cuando se cumple la condición para modo dormido.
- ***Command mode**:* se ejecuta un comando.

7.1.3.1.2. TRANSMIT

Este estado es cuando el dispositivo recibe datos y está preparado para empaquetarlos. Hay dos tipos de transmisión:

- **Transmisión directa**: los datos se transmiten inmediatamente hacia la dirección de destino.
- **Transmisión indirecta**: los paquetes se retienen durante un periodo de tiempo y sólo se transmiten al destino cuando éste pide los datos. Este tipo de transmisiones sólo las realiza un Coordinador. Si todos los nodos de la red son End Devices, sólo pueden operar con transmisión directa.
  Es útil para asegurarse de que un paquete se envió a un nodo dormido.

7.1.3.1.3. RECEIVE

Si un nodo destino recibe un paquete RF válido, éste transfiere el dato a su buffer de transmisión. Para que la interfaz serial indique que ha recibido datos de una red RF, ciertos datos deben coincidir: el ID, el canal y la dirección.

7.1.3.1.4. COMMAND

Es el estado en el que el firmware interpreta caracteres entrantes como comandos. Éstos permiten modificar el firmware usando parámetros establecidos mediantes comandos AT. Si se desea leer o establecer cualquier opción del dispositivo, se debe hacer mediante estos comandos.

7.1.3.1.5. SLEEP

Los modos dormido o `Sleep` permiten al dispositivo entrar en un estado de bajo consumo cuando éste no está realizando ninguna actividad. Para entrar en modo dormido, se debe dar al menos una de estas condiciones:

- Se fuerza el modo dormido a través de un pin.
- El dispositivo está inactivo durante un tiempo determinado, que se puede configurar con el parámetro `ST` (Time before Sleep).
- Tiene ciclos de ahorro de energía de tiempo prefijado.

A su vez, existen diferentes modos de ahorro de energía:

- **Normal**: es el modo por defecto. En este modo, el dispositivo no se duerme nunca.
- **Asíncrono por pin**: permite dormir y despertar el dispositivo en base al estado del pin `Sleep_RQ`.
- **Ciclo asíncrono**: permite dormir y despertar el dispositivo durante pequeños intervalos de tiempo.
- **Ciclo asíncrono con despertar por pin**: se puede despertar el dispositivo prematuramente usando el pin `Sleep_RQ`.
- **Modo dormido síncrono**: el nodo se sincroniza con una red dormida, pero no se duerme él mismo. En cualquier momento, el nodo responde a otros nodos que intenten unirse a la red dormida usando un mensaje de sincronización.

- **Ciclo síncrono**: el dispositivo duerme durante un tiempo programado y se despierta al unísono junto con otros nodos, intercambia datos y sincroniza mensajes y después retorna a modo dormido.

7.1.3.2. MODOS DE COMUNICACIÓN

7.1.3.2.1. MODO TRANSPARENTE

Es el modo por defecto y está destinado principalmente a la comunicación punto a punto, donde no es necesario ningún punto de control. También se usa para reemplazar alguna conexión serie por cable, ya que es la configuración más sencilla posible y no requiere una mayor configuración.

Existen 4 tipos de conexión transparente. La diferencia principal radica en el número de nodos o puntos de acceso, y la forma en cómo éstos interactúan entre sí.

7.1.3.2.2. MODO API

Este modo es más complejo, pero permite el uso de tramas con cabeceras que aseguran la entrega de los datos, al estilo TCP. Extiende el nivel en el cual la aplicación del cliente, puede interactuar con las capacidades de red del módulo.

Cuando el módulo XBee se encuentra en este modo, toda la información que entra y sale, es empaquetada en tramas, que definen operaciones y eventos dentro del módulo.

Así, una trama de Transmisión de Información (información recibida por el pin 3 o DIN) incluye:

- Trama de información RF transmitida.
- Trama de comandos (equivalente a comandos AT).

Mientras que una trama de Recepción de Información incluye:

- Trama de información RF recibida.
- Comando de respuesta.
- Notificaciones de eventos como Reset, Associate, Disassociate, etc.

El modo API permite formas alternativas de configurar los modulos y enrutar los datos en la capa de aplicación. Este modo facilita las siguientes operaciones:

- Transmitir información a múltiples destinatarios, sin entrar al modo de Comandos.
- Recibir estado de éxito/falla de cada paquete RF transmitido.
- Identificar la dirección de origen de cada paquete recibido.

Start delimiter	Length		Frame data								Checksum
			API identifier	Identifier-specific Data							
1	2	3	4	5	6	7	8	9	…	n	n+1
0x7E	MSB	LSB	cmdID	cmdData							Single byte

Tabla 12: Ejemplo de comando AT

El primer byte `0x7E` indica el comienzo de la trama. Los dos bytes siguientes indican solamente la longitud de la trama de Datos (Data Frame). La estructura API que viene después se compone según vemos en la siguiente ilustración:

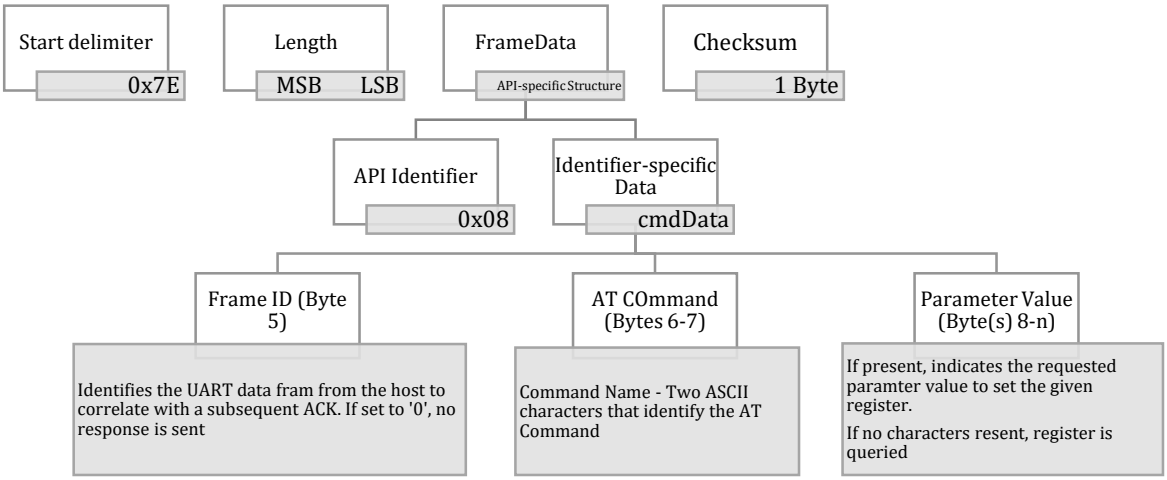


Ilustración 53: Trama XBee

En el modo API, podemos enviar y recibir desde el Coordinador o cualquier elemento de red. La información que viene en cada trama es mucho más detallada. En la práctica, este modo permite tener más control de la red, ya que por ejemplo si la comunicación falla obtendremos un código de error.

Entre las opciones que permite API se tienen:

- Transmitir información a múltiples destinatarios, sin entrar al modo de Comandos.
- Recibir estado de éxito/falla de cada paquete RF transmitido.
- Identificar la dirección de origen de cada paquete recibido.

7.1.3.2.3. Modo Comando

Este modo permite ingresar comandos AT al módulo XBee, para configurar, ajustar o modificar parámetros. Permite ajustar parámetros como la dirección propia o la de destino, así como su modo de operación, entre otros aspectos.

Para poder ingresar los comandos AT es necesario utilizar el Hyperterminal de Windows, el programa XCTU o alguna aplicación que acceda directamente a la UART. En la siguiente imagen se muestra un ejemplo de comando AT con las diferentes partes que lo componen:

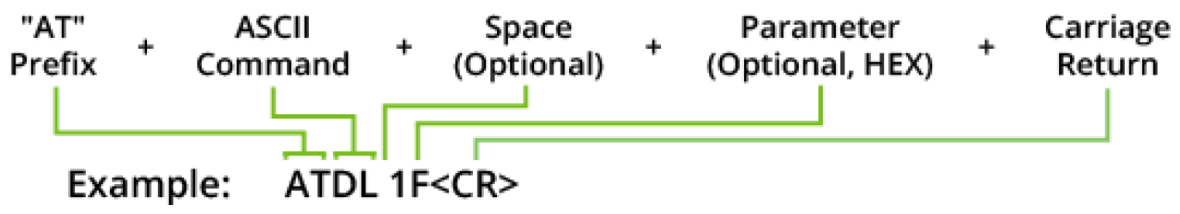


Ilustración 54: Ejemplo de comando AT

7.1.3.3. ESPECIFICACIONES TÉCNICAS

Las especificaciones técnicas del módulo de radio se muestran en la siguiente tabla:

Referencia	XBP24-DMWIT-250
RENDIMIENTO	
Velocidad de datos RF	250 Kbps
Distancia en entorno urbano	300 ft (90 m)
Distancia en entornos al aire libre	1 mile (1.6 km)
Transmit power	63 mW (+18 dBm)
Sensibilidad del receptor (1% per)	-100 dBm
CARACTERÍSTICAS	
Interfaz Serial	3.3V CMOS serial UART
Método de configuración	AT & API
Banda de frecuencia	2.4 GHz ISM
Inmunidad frente a interferencias	DSSS (Direct Sequence Spread Spectrum)
Velocidad de datos Serial	Up to 115.2 Kbps
Adc inputs	(6) 10-bit ADC inputs
E/S Digital	13
Opciones de antena	Chip, Wire Whip, U.FL, RPSMA
RED Y SEGURIDAD	
Encriptación	128-bit AES
Entrega de paquetes fiable	Retries/Acknowledgments
Opciones de direccionamiento	address PAN ID, channel, 64-bit address
Canales	12
REQUERIMIENTOS DE ENERGÍA	
Voltaje	2.8 – 3.4VDC
Corriente transmitida	250 mA (Wire, Chip, U.FL), 340 mA (RPSMA
Corriente recibida	55 mA
Corriente de desconexión	<50 uA

Tabla 13: Especificaciones técnicas del módulo XBee XBP24-DMWIT-250

7.1.3.4. VISTAS/DIAGRAMAS

7.1.3.4.1. LADO

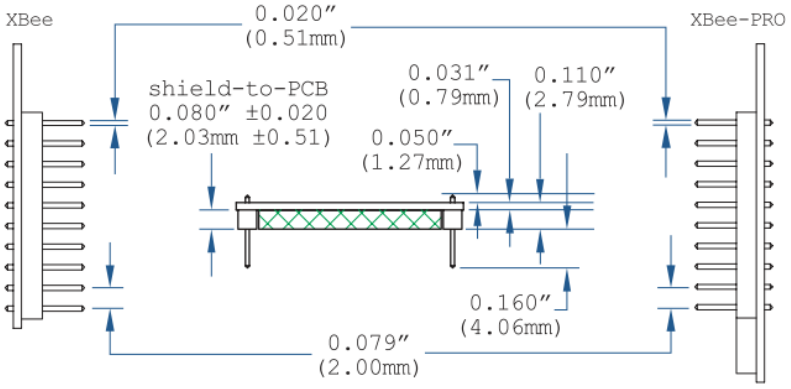


Ilustración 55: Vista de lado del módulo XBee

7.1.3.4.2. PLANTA

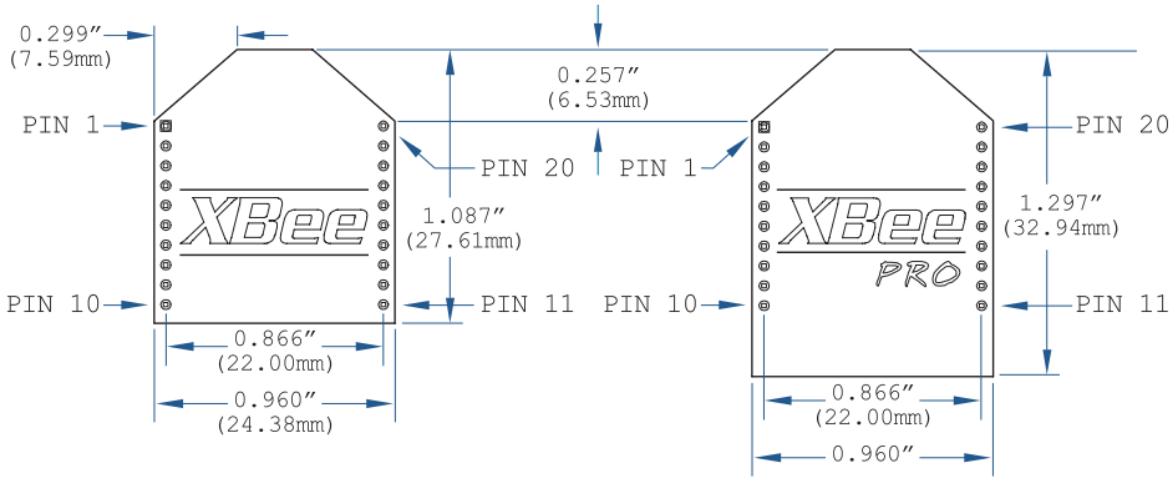


Ilustración 56: Vista de planta del módulo XBee

7.1.4. SENSOR DE VIENTO

El sensor de viento es un dispositivo inalámbrico encargado de medir en tiempo real la dirección y velocidad del viento mediante una veleta y un anemómetro. Este dispositivo almacena la información de las mediciones y se comunica con la estación base para que desde ésta se pueda controlar su funcionamiento.

En la siguiente imagen se muestra el conjunto estación base / sensor de viento interconectado:

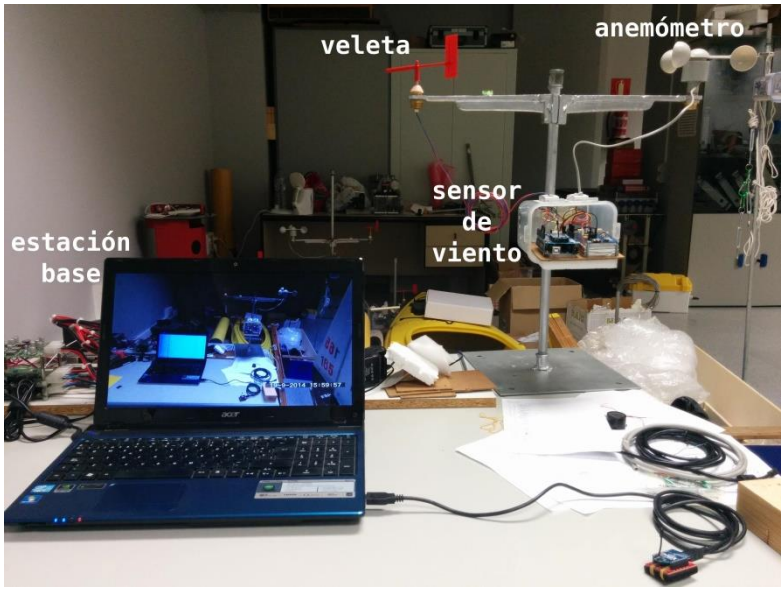


Ilustración 57: Estación base y sensor de viento con anemómetro y veleta

El módulo de procesamiento está compuesto de dos placas: una placa de prototipado Arduino UNO[21] y un módulo inalámbrico Wireless SD Shield[24] en la que se conecta un módulo de comunicaciones XBee, así como un soporte para una tarjeta MicroSD para almacenar la información recogida. También posee un zumbador o "beeper", para

notificar situaciones de error y para dirigir el calibrado inicial. En la siguiente imagen se pueden apreciar los dos módulos y la interconexión cableada entre ellos:

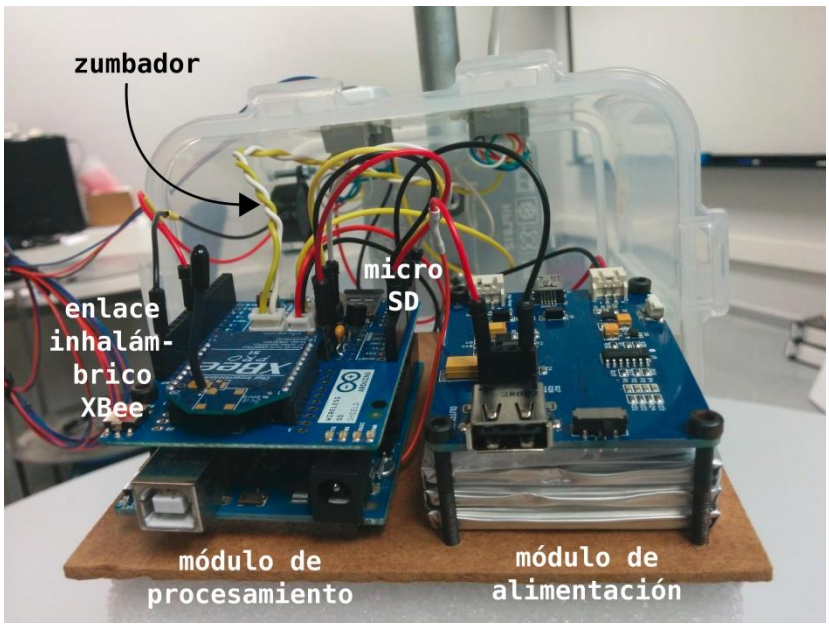


Ilustración 58: Módulo de recogida de datos

En cuanto a sus características técnicas, el sensor posee un módulo de alimentación compuesto por 3 baterías de 3.7 v. y 6000 mAh, y una placa Lipo Rider Pro para el control de la carga y descarga de las mismas.

Componentes	Descripción
Módulo de procesamiento	
Microcontrolador	Microcontrolador Atmel ATmega328P en el que se ejecuta el software de recogida de datos de viento. Almacenamiento de información en la tarjeta MicroSD. Comunicaciones con la estación base. Una vez inicializado adquiere información de viento (dirección y velocidad) de manera periódica.
MicroSD	4 GB de capacidad. Es un almacenamiento secundario donde se guardan durante una sesión de medida todos los datos de viento, así como información de estado durante la ejecución del software empotrado en el sensor de viento.
Enlace inalámbrico XBee	Módulo XBee PRO 60 mW bajo protocolo 802.15.4 con un alcance de 1'6 Km.
Módulo de alimentación	
Baterías	Paquete de baterías LiPo de 3'7 v de 6.000 mAh que proporciona una autonomía de 24 horas aproximadamente.
Placa de recarga de baterías	Permite la alimentación del módulo de procesamiento mediante el paquete de baterías, así como la recarga de las mismas mediante conexión directa a la corriente a través de un conector USB, o bien mediante un panel solar.

Anemómetro	
	Anemómetro de copa (sensibilidad: 2’4 Km/h)
Veleta	
	Veleta basada en un sensor angular analógico US Digital MA3[25]. Al desplegar cada sensor en el campo, y al inicio de cada sesión de medida, es preciso calibrar la veleta con el norte (dicho proceso es dirigido al encender el sensor mediante el zumbador del módulo de procesamiento). Una vez calibrado se aplica a las muestras de dirección del viento un filtro de mediana con una ventana de 7 muestras en el software de adquisición de datos.

Tabla 14: Detalle de los componentes utilizados en las estaciones

7.1.5. ESTACIÓN BASE

La estación base es un ordenador portátil provisto de un enlace inalámbrico XBee que permite comunicarse con los sensores de viento.

La función principal de la estación es ejecutar un software específico para monitorizar y controlar los sensores de viento, la inicialización de los mismos y almacenar la información recogida enviada por todos los sensores.

7.2. REQUISITOS SOFTWARE

En cuanto a los requisitos software, ha sido necesario disponer de una serie de herramientas para el desarrollo del proyecto, las cuales se detallan a continuación.

7.2.1. ARDUINO IDE

Como entorno de desarrollo se ha utilizado Arduino IDE en su versión 1.8.2. Es un software de código libre bajo licencia GPL que permite programar en las distintas placas basadas en Arduino. Está disponible para distintas plataformas, como Windows, Mac y Linux. En el caso que nos ocupa, se ha elegido la versión para Linux por compartir entorno con el resto de herramientas utilizadas.

Ilustración 59: Entorno Arduino IDE

7.2.2. XBEE PARA ARDUINO

La librería XBee para Arduino sirve para comunicarse con radios XBee en modo API, que soporta tanto la Serie 1 como la Serie 2 (la que soporta DigiMesh). Incluye soporte para la mayoría de los paquetes.

A continuación se muestran las clases más importantes para poder comunicarse con una radio XBee que tiene cargado un firmware DigiMesh:

`AtCommandRequest`	`Representa un paquete AT Command (0x08)`
`AtCommandResponse`	`Representa un paquete AT Command Response (0x88)`
`ModemStatusResponse`	`Representa un paquete Modem Status (0x8a)`
`RemoteAtCommandRequest`	`Representa un paquete Remote Command Request (0x17)`
`RemoteAtCommandResponse`	`Representa un paquete Remote Command Response (0x88)`
`XBee`	`Interfaz para comunicarse con una radio XBee`
`XBeeAddress64`	`Representa una dirección de 64 bits`
`ZBRxResponse`	`Representa un paquete Receive Packet (0x90)`
`ZBTxRequest`	`Representa un paquete Transmit Request (0x10)`
`ZBTxStatusResponse`	`Representa un paquete Status (0x8B)`

Tabla 15: Elementos principales de XBee para Arduino

7.2.3. C/C++

El lenguaje usado en Arduino está basado en Wiring[26] que a su vez está basado en C/C++[27]. La sintaxis es muy similar y utiliza las estructuras, operadores y tipos de datos más básicos de estos lenguajes. Además, usa varias librerías [28] para ampliar su funcionalidad, de las cuales la más importante es avr-lib [29], específica para microcontroladores Atmel AVR de 8-bits.

Ilustración 60: Logo de C

Ilustración 61: Logo de C++

7.2.4. XCTU

XCTU[30] es un software propietario[31] de la marca Digi, que proporciona una interfaz gráfica para la configuración de módulos inalámbricos, concretamente para los adaptadores XBee. También permite al usuario la posibilidad de instalar nuevos firmwares en los módulos de radio, que están disponibles en la propia página de Digi.

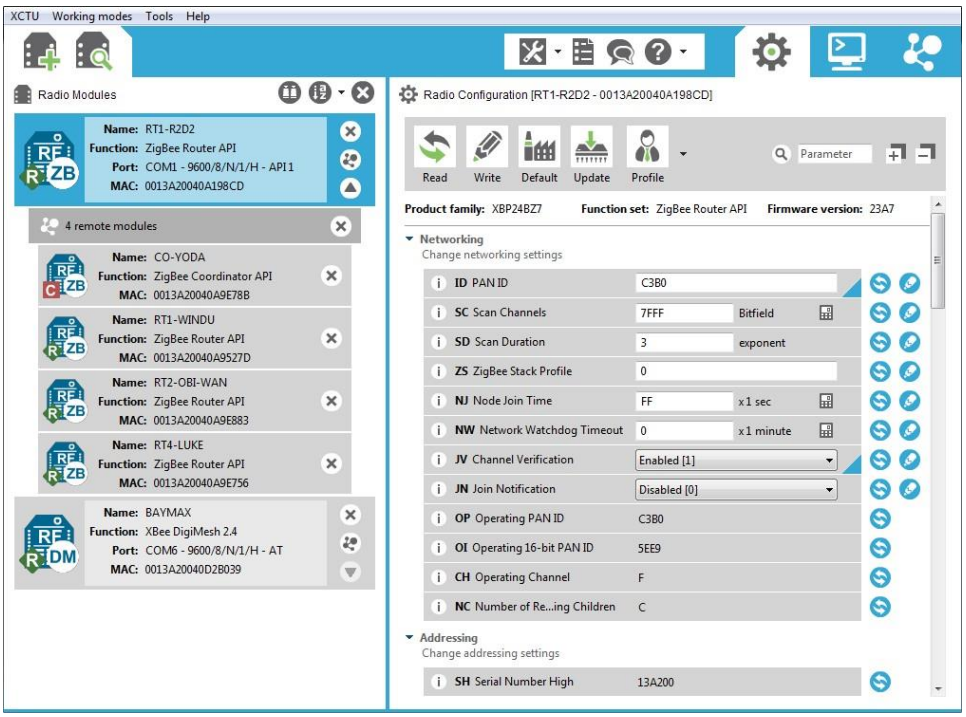


Ilustración 62: Entorno gráfico XCTU

En la imagen anterior se puede apreciar que el programa XCTU se divide en 2 zonas principales:

- A la izquierda, el listado de módulos conectados o descubiertos en la red inalámbrica.
  - > Ofrece la posibilidad de seleccionar el puerto COM a través del que irá conectada nuestra radio, así como los ajustes típicos de la comunicación serie (Baudrate, Flow Control, Data Bits, Parity y Stop Bits). Ofrece también la posibilidad de habilitar el modo API y el response timeout.
- A la derecha, el contenido cambia según la pestaña elegida, que puede ser una de las siguientes:
  - >  ***Configuration***: Esta es la interfaz para leer / escribir parámetros de un XBee. Desde esta pestaña también es posible actualizar el firmware disponible desde la página de Digi.

>  ***Consoles***: Esta pestaña permite leer y escribir las tramas que llegan al módulo XBee. Permite leerlas en modo hexadecimal para poderlas interpretar.

>  ***Network***: Esta pestaña permite descubrir otros módulos que haya al alcance, mostrando un gráfico de todas ellas, con datos relevantes como la potencia de la señal entre nodos.

7.2.5. METEO

En la estación base se ejecuta un software específico desarrollado como parte del proyecto Dunas, que permite inicializar los sensores de forma sincronizada para que todos empiecen la recogida de datos en el mismo instante de tiempo.

Este software, mediante la librería libxbee3, permite interactuar con radios XBee conectadas directamente al computador, permitiendo así gestionar paquetes y actuar en base a ciertos criterios.

En este caso, el software está diseñado para monitorizar los sensores de viento de la red del Proyecto Dunas. Permite verificar si alguno ha dejado de transmitir (indicador S), si la temperatura es demasiado alta (indicador T) o si la señal recibida no tiene cambios (indicador D). También contiene un apartado en el que es posible visualizar la información recibida en tiempo real, tal y como se muestra en la siguiente imagen:

Ilustración 63: Software de monitorización y control en la estación base

7.2.6. LIBXBEE3

Libxbee3 es una librería para C/C++ que permite comunicarse con radios XBee en modo API o API2. Está diseñada para tener gran flexibilidad, y además hace gran parte del trabajo, haciendo que sea transparente para el programador.

Los elementos principales de la librería se dividen en structs, funciones instancia, funciones de conexión y funciones de paquetes. A continuación se muestran los más importantes:

Estructuras	
xbee_pkt	Representa un paquete
xbee_conAddress	Representa una dirección
xbee_conSettings	Representa las opciones de una conexión
xbee_con	Representa una conexión
Funciones	
xbee_setup	Configura una conexión a la radio
xbee_conNew	Crea una nueva conexión configurada previamente
xbee_conEnd	Cierra una conexión
xbee_conTx	Transmite un mensaje
xbee_conRx	Recibe un mensaje
xbee_pktFree	Libera la memoria asociada a un paquete después de recibirlo

Tabla 16: Elementos principales de libxbee3

En líneas generales, un programa debe seguir el siguiente orden para poder crear una conexión satisfactoria a una radio:

1. Se configura la conexión a la radio con xbee_setup. En el caso de una radio con DigiMesh, debe especificarse el modo "xbee2" tal que:

   ```
   struct xbee *xbee;
   ret = xbee_setup(&xbee, "xbee2", "/dev/ttyUSB0", 9600);
   ```

   Donde:

   - &xbee es una estructura xbee
   - "xbee2" es el modo de conexión
   - "/dev/ttyUSB0" es el puerto USB (en Linux) donde se ha conectado la radio
   - 9600 la velocidad en baudios a la que funciona la radio XBee.

2. Se configura la dirección de la radio con la que se va a conectar, de la siguiente manera:

   ```
   struct xbee_conAddress address;

   memset(&address, 0, sizeof(address));
   address.addr64_enabled = 1;

   address.addr64[0] = 0x00;
   address.addr64[1] = 0x13;
   address.addr64[2] = 0xA2;
   ```

```
address.addr64[3] = 0x00;
address.addr64[4] = 0x40;
address.addr64[5] = 0xB0;
address.addr64[6] = 0xAA;
address.addr64[7] = 0xD1;
```

3. Se conecta con la radio:

```
struct xbee_con *con;
ret = xbee_conNew(xbee, &con, "Data", &address)
```

Donde:

- xbee es la estructura xbee
- con es una estructura de conexión
- "Data" es el modo en que se establecerá la conexión. Este valor depende del modo indicado en la función xbee_setup.
- &address es la dirección de la radio a la que se desea conectar.

4. Se envía un paquete:

```
ret = xbee_conTx(con, NULL, "Hola");
```

Donde:

- con es la estructura de conexión
- NULL es un parámetro que para recoger el valor del ACK
- "Hola" es la información que se desea enviar

5. Por último, se espera a recibir un paquete y una vez recibido, se imprime por la salida estándar:

```
do
{
  ret = xbee_conRx(con, &pkt, NULL);

  if ( (ret != XBEE_ENONE) && (ret != XBEE_ENOTEXISTS) )
          throw pack_error("xbee_conRx()", ret, __FILE__, __LINE__);

  usleep(1000000);

} while (ret == XBEE_ENOTEXISTS);

data = (char*) pkt->data;

cout << "  << " << data << endl;
```

7.3. RESULTADOS

Todas las pruebas y resultados mostrados a continuación se realizaron con:

- Un PC con la radio BASE, dentro del laboratorio.
- Arduino con la radio E006 en un pasillo cercano.
- Arduino con la radio E008 al fondo del pasillo, detrás de una pared.

Las direcciones MAC de las radios utilizadas se muestran a continuación:

Nombre del sensor	Radio XBee	SH	SL
-	BASE	0013A200	40A1CE39
E006	E006	0013A200	40B0AAD1
E008	E008	0013A200	40B0AB2F

Ilustración 64: Datos de las radios XBee con las que se realizan las pruebas

El esquema general de pruebas implica conectar una radio a la estación base y colocar dos radios XBee lo suficientemente separadas como para que al menos dos de ellas no se puedan comunicar directamente:

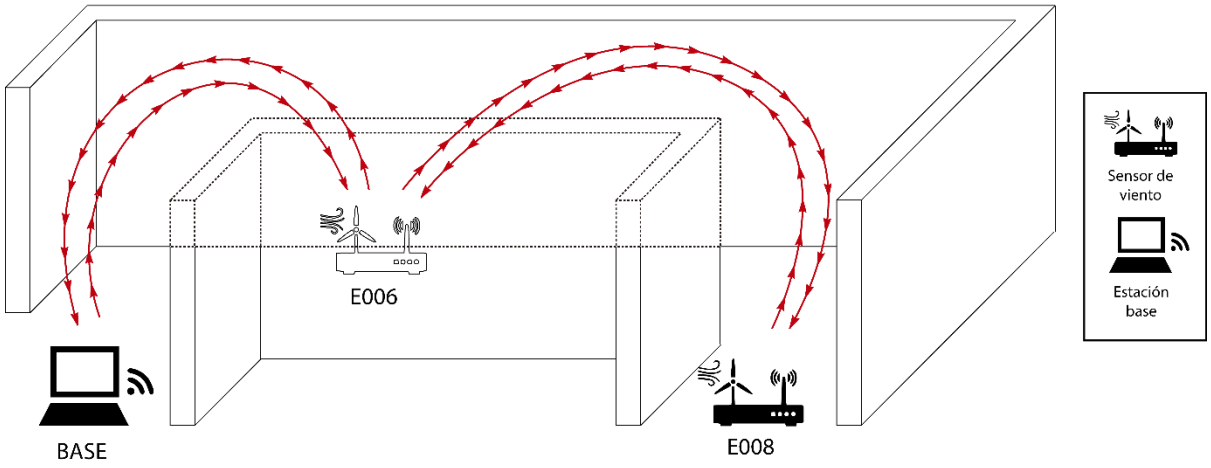


Ilustración 65: Esquema general de las pruebas realizadas

En las sucesivas pruebas, se irán cambiando ciertos aspectos del escenario planteado y se indicarán consecuentemente.

7.3.1. LATENCIA ENTRE NODOS CON XCTU

Esta prueba se realiza a modo de introducción para detectar las distintas radios XBee desde XCTU.

Para descubrir todas las radios, hay que realizar una serie de pasos:

1. Detectar la radio conectada al PC (BASE) −proceso se explica en el apartado Detección de radios en XCTU del Anexo−.
2. Ir a la pestaña **Consoles** y pulsar sobre el botón **Open** para ir monitorizando todos los paquetes que se envían y se reciben en la radio BASE.
3. Luego, se procede a ir a la pestaña **Network** y se pulsa sobre el botón **Scan**, tal y como se indica en la siguiente imagen:

Ilustración 66: Descubrimiento de la red en XCTU

Una vez pulsado, se empieza a descubrir la red mediante la ejecución de una serie de comandos DigiMesh encargados de esta tarea, como el FN. Para ver los paquetes enviados y recibidos, se vuelve a la pestaña **Consoles**, donde se muestra el proceso de descubrimiento de la red:

	ID	Time	Length	Frame
→	0	11:22:16.262	4	AT Command
←	1	11:22:16.325	7	AT Command Response
→	2	11:22:16.465	4	AT Command
←	3	11:22:16.512	6	AT Command Response
→	4	11:22:16.528	4	AT Command
←	5	11:22:16.575	6	AT Command Response
→	6	11:22:17.387	4	AT Command
←	7	11:22:27.390	29	AT Command Response
→	8	11:22:53.634	15	Remote AT Command Request
←	9	11:23:00.931	39	Remote Command Response
←	10	11:23:05.806	39	Remote Command Response

Ilustración 67: Proceso del descubrimiento de una red en XCTU

En la imagen se puede observar lo siguiente:

1. Los paquetes con ID0, ID2, e ID4 consultan los campos ID, CH y NO de la radio BASE, que sirven para obtener los datos básicos de la red. Cada uno de estos comandos devuelve un paquete con el valor de cada parámetro consultado.

   El contenido del paquete de envío ID0 es el siguiente, donde se manda el comando ID:

```
7E 00 04 08 52 49 44 18

Start delimiter: 7E
Length: 00 04 (4)
Frame type: 08 (AT Command)
Frame ID: 52 (82)
AT Command: 49 44 (ID)
Checksum: 18
```

   El contenido del paquete de respuesta ID1 es el siguiente, donde se recibe el valor del parámetro ID:

```
7E 00 07 88 01 49 44 00 7F FF 6B

Start delimiter: 7E
Length: 00 07 (7)
Frame type: 88 (AT Command Response)
Frame ID: 53 (83)
AT Command: 49 44 (ID)
Command status: 00 (OK)
Command data: 7F FF
Checksum: 6B
```

   El contenido del paquete de envío ID2 es el siguiente, donde se envía el comando CH:

```
7E 00 04 08 54 43 48 6B

Start delimiter: 7E
Length: 00 04 (4)
Frame type: 08 (AT Command)
Frame ID: 54 (84)
AT Command: 43 48 (CH)
Checksum: 6B
```

   El contenido del paquete de respuesta ID3 es el siguiente, donde se recibe el valor del parámetro CH:

```
7E 00 07 88 01 43 48 00 0C DF

Start delimiter: 7E
Length: 00 06 (6)
Frame type: 88 (AT Command Response)
Frame ID: 55 (85)
AT Command: 43 48 (CH)
Command status: 00 (OK)
Command data: 0C
```

```
Checksum: DF
```

El contenido del paquete de envío ID4 es el siguiente, donde se envía el comando NO:

```
7E 00 04 08 56 4E 4F 06

Start delimiter: 7E
Length: 00 04 (4)
Frame type: 08 (AT Command)
Frame ID: 56 (86)
AT Command: 4E 4F (NO)
Checksum: 06
```

El contenido del paquete de respuesta ID5 es el siguiente, donde se recibe el valor del parámetro NO:

```
7E 00 07 88 01 4E 4F 00 00 D9

Start delimiter: 7E
Length: 00 06 (6)
Frame type: 88 (AT Command Response)
Frame ID: 55 (85)
AT Command: 4E 4F (NO)
Command status: 00 (OK)
Command data: 00
Checksum: D9
```

Estos datos sirven para detectar todas aquellas radios que tengan los mismos valores en ID y CH, teniendo en cuenta las opciones devueltas en NO.

2. El paquete con ID6 es el comando FN (Find Neighbours) que localiza automáticamente las radios que pueda "ver" directamente. En este caso localiza la radio E006.
   Se recibe la respuesta en el paquete ID7, con los datos de la radio E006.

3. El paquete con ID8 ejecuta remotamente el comando FN en E006.
   Los paquetes con ID9 e ID10 son las respuestas de los vecinos de E006, que son BASE y E008.

Una vez terminado el proceso, se puede observar la latencia existente entre radios donde, a mayor valor, peor calidad de la señal:

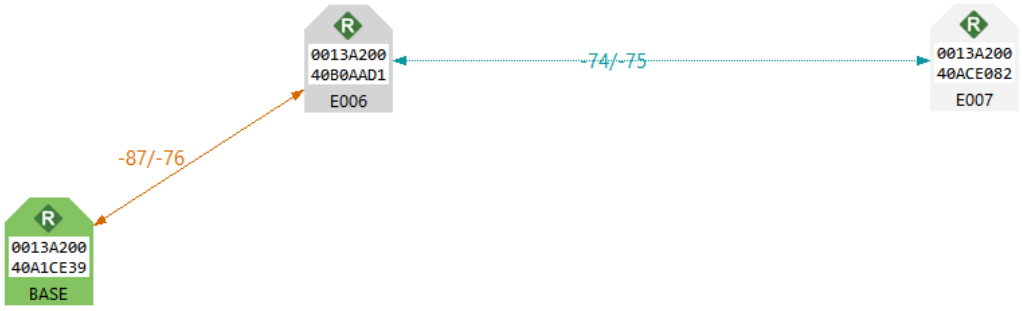


Ilustración 68: Latencia entre nodos

7.3.2. PRUEBAS ENTRE XCTU Y RADIOS XBEE

Para ir avanzando en complejidad de las pruebas, se va a proceder a cambiar el escenario, que consiste en tener:

- La radio BASE conectada a un ordenador con XCTU.
- La radio E006 conectada a un ordenador con XCTU.
- La radio E008 conectada a ordenador con XCTU.

Un esquema general se puede observar en la siguiente imagen:

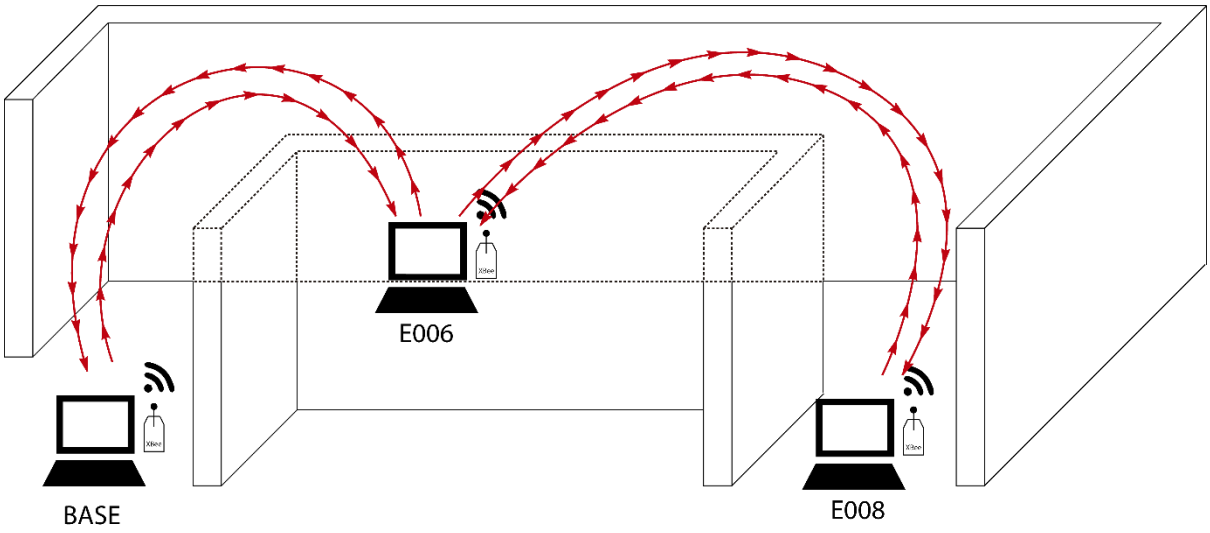


Ilustración 69: Esquema de pruebas con XCTU y radios XBee

Las pruebas consistirán en ejecutar manualmente comandos locales en la radio BASE, comandos remotos a las radios E006 y E008 y por último se realizará un descubrimiento de la red desde la radio BASE, que actuará como concentrador.

7.3.2.1. COMANDOS LOCALES

El primero paso para enviar comandos locales a una radio conectada a un PC con XCTU es detectar la radio con este software, proceso que se describe en el apartado Detección de radios en XCTU.

Para ejecutar un comando en la radio conectada localmente a XCTU, basta con crear un nuevo frame en la parte **Send frames** de la interfaz de XCTU y pulsar sobre el botón  :

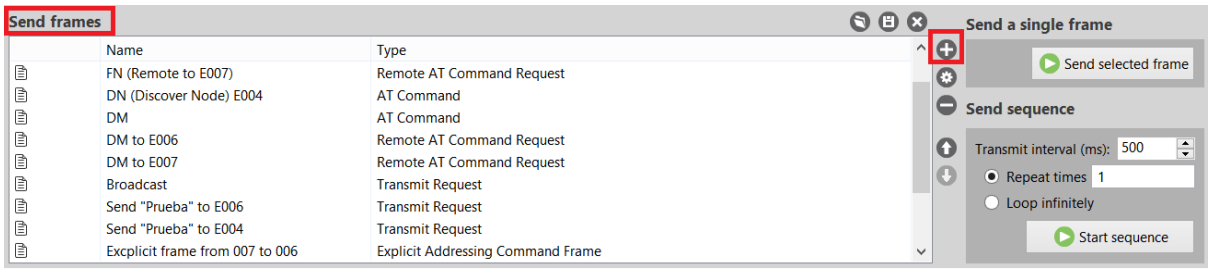


Ilustración 70: Generación de un frame con XCTU

Mediante este método se puede enviar cualquier tipo de comando a la radio XBee. En el ejemplo actual, el comando que queremos enviar es el **ID**, que devuelve el valor del campo **ID** que posee la radio.

Como crear un frame a mano puede ser muy tedioso, ya que habría que calcular el checksum a mano, XCTU posee un asistente que facilita la tarea en gran medida. Este asistente gráfico permite tener una idea general de la estructura de los frames más clara y más intuitiva.

En la imagen siguiente se muestra cómo debe generarse un comando del tipo **AT command** (`0x08`), para mandar el comando **ID**:

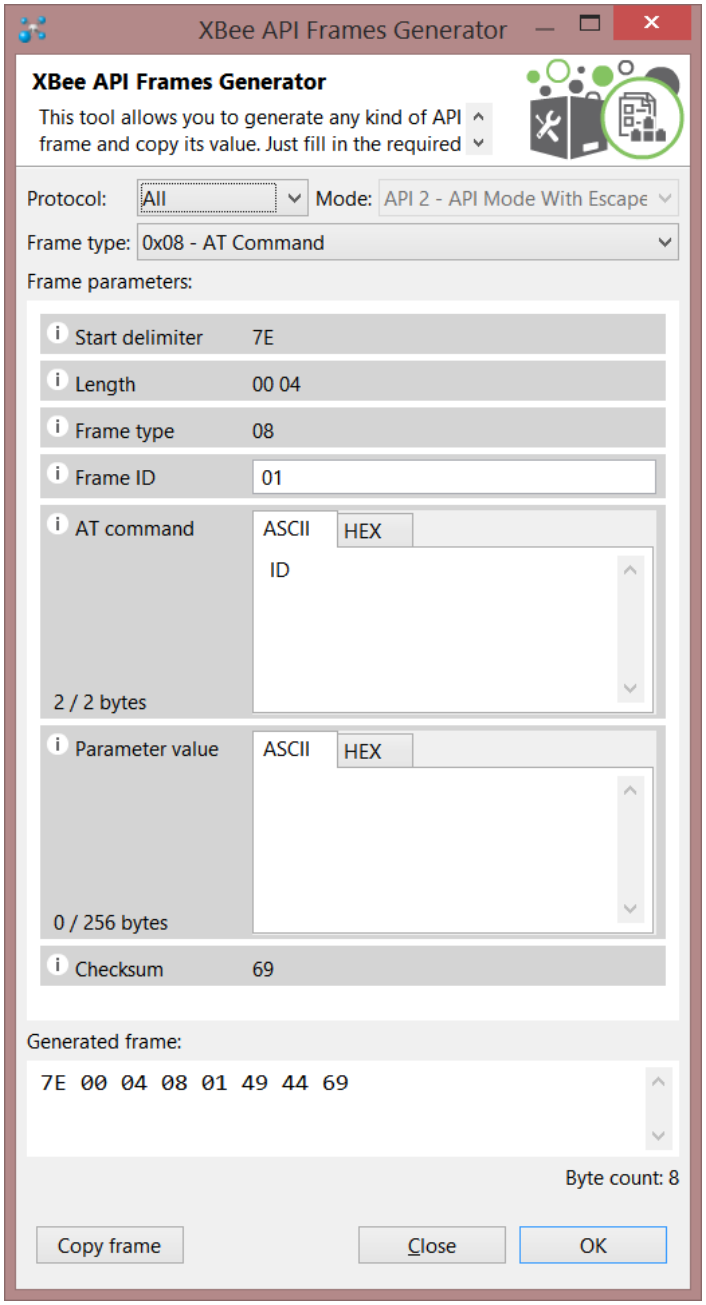


Ilustración 71: Generación de frames en XCTU – AT command (0x08)

Una vez creado, sólo hace falta enviar ese frame. Desde la interfaz de XCTU, se realiza mediante el botón **Send selected frame**, tal y como se ve en la siguiente imagen:

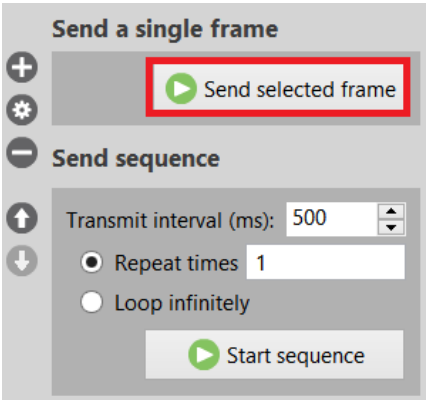


Ilustración 72: Envío de un paquete mediante XCTU

Una vez que se pulsa el botón, se observa en la pestaña **Consoles** la traza de envío y recepción del comando:

Frames log

	ID	Time	Length	Frame
→	0	11:36:30.283	4	AT Command
←	1	11:36:30.330	7	AT Command Response

Ilustración 73: Traza de envío de un comando local

Para comprobar que efectivamente el paquete devolvió el valor buscado, se pueden ver los datos del paquete con ID1:

```
7E 00 07 88 11 49 44 00 7F FF 6B

Start delimiter: 7E
Length: 00 07 (7)
Frame type: 88 (AT Command Response)
Frame ID: 01 (1)
AT Command: 49 44 (ID)
Status: 00
Response: 7F FF
Checksum: 6B
```

7.3.2.2. COMANDOS REMOTOS

Para comprobar la conectividad entre radios, una prueba sencilla consiste en enviar comandos remotos a una radio con la que no se tiene conexión directa. En este caso, se enviarán comandos remotos a las radios E006 y E008.

Para enviar un comando remoto, se sigue el mismo proceso que en el apartado anterior, lo que en este caso, habrá que enviar un frame del tipo **Remote AT Command** (`0x17`):

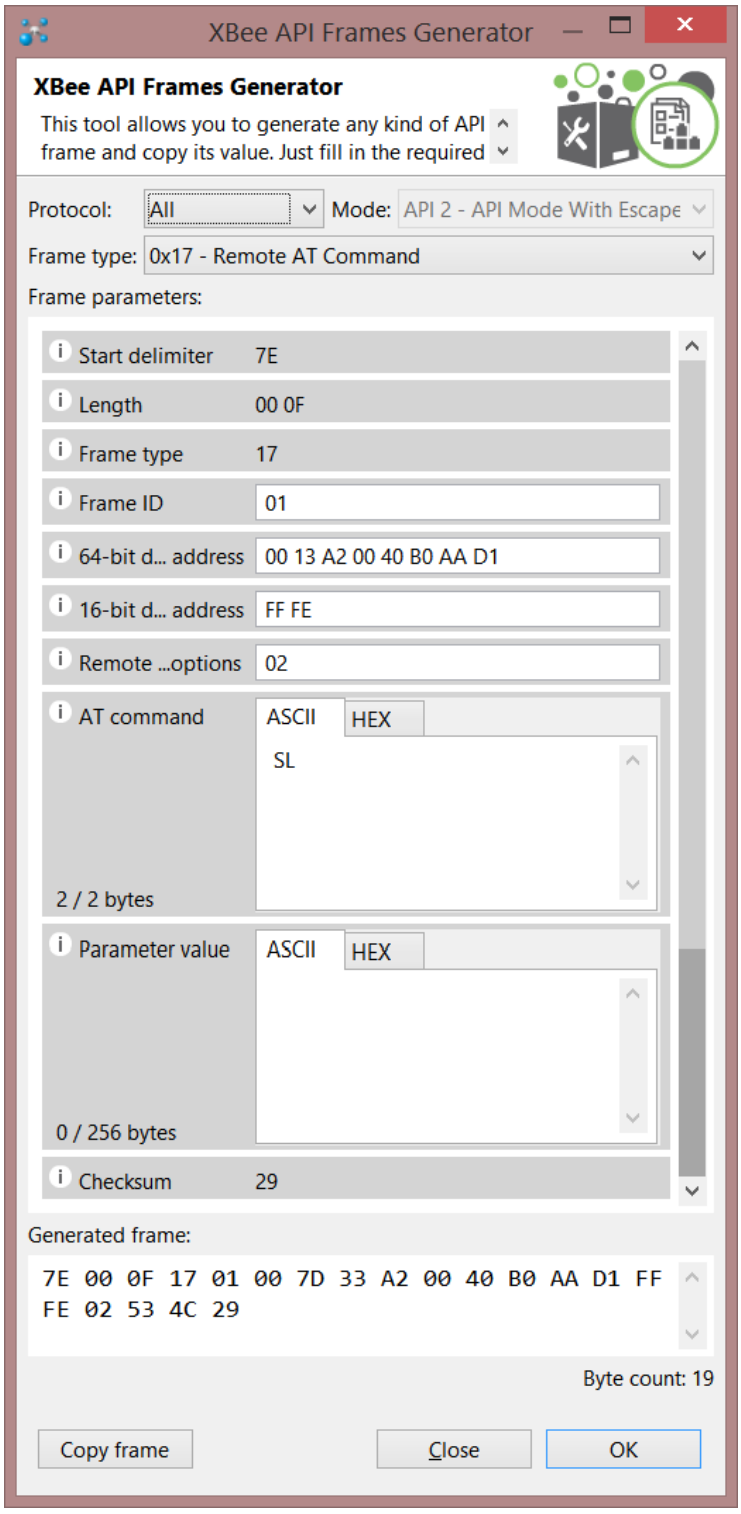
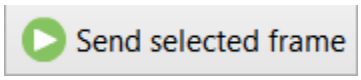


Ilustración 74: Generación de frames en XCTU – Remote AT command (0x17)

En la imagen anterior se está generando un frame del tipo **Remote AT command**, en el que se pretende ejecutar el comando **SL** en la radio remota E006, cuya dirección MAC es `0x0013A200, 0x40B0AAD1`.

El comando **SL** lo que hace es devolver la parte baja de la dirección MAC de la radio XBee.

Para enviar el comando, se procede igual que en el caso anterior, pulsando sobre el botón



Una vez que se pulsa el botón, se vuelve a la pestaña **Consoles** y se observa la traza de envío y recepción del comando:

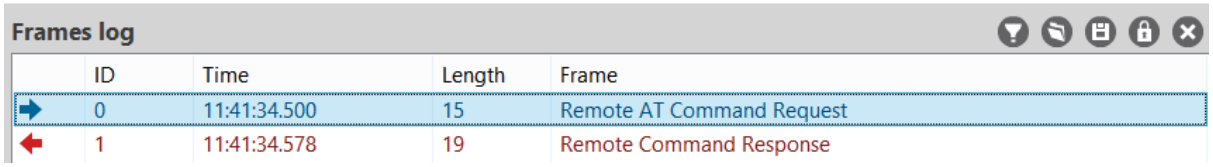


Ilustración 75: Traza de envío de un comando remoto a la radio E006

Para comprobar que efectivamente el paquete devolvió el valor buscado, se pueden ver los datos del paquete con ID1:

```
7E 00 7D 33 97 01 00 7D 33 A2 00 40 B0 AA D1 FF FE 53 4C 00 40 B0 AA D1 40

Start delimiter: 7E
Length: 00 13 (19)
Frame type: 97 (Remote Command Response)
Frame ID: 01 (1)
64-bit dest. address: 00 13 A2 00 40 B0 AA D1
16-bit dest. address: FF FE
AT Command: 53 4C (SL)
Status: 00
Response: 40 B0 AA D1
Checksum: 40
```

Para la radio E008 se realiza el mismo proceso, obteniendo lo siguiente:

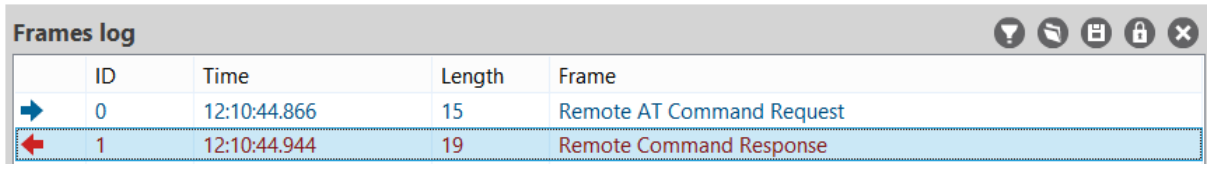


Ilustración 76: Traza de envío de un comando remoto a la radio E006

Se comprueba que el paquete con ID1 contiene lo esperado:

```
7E 00 7D 33 97 01 00 7D 33 A2 00 40 B0 AB 2F FF FE 53 4C 00 40 B0 AB 2F 7A

Start delimiter: 7E
Length: 00 13 (19)
Frame type: 97 (Remote Command Response)
Frame ID: 01 (1)
64-bit source address: 00 13 A2 00 40 B0 AB 2F
16-bit source address: FF FE
AT Command: 53 4C (SL)
Status: 00 (Status OK)
Response: 40 B0 AB 2F
Checksum: 7A
```

7.3.2.3. COMANDOS ND, AG Y FN

En este experimento se va a proceder a descubrir la red de manera manual, con comandos enviados explícitamente a las radios XBee. El proceso es similar al que se realiza en el apartado Latencia entre nodos con XCTU.

Cuando se despliegue la red de sensores, este sería el primer paso en ejecutarse, tanto para comprobar que todos los sensores están activos como para su posterior configuración.

En este experimento, la radio BASE actuará como agregador y todas las radios deberán tener como destino de comunicaciones la radio BASE, enviando los paquetes por la ruta más óptima para ello.

Las fases de configuración de la red consisten en:

1. Configurar todas las radios de la red con sus parámetros DH a 0x0 y DL a 0xFFFF. En la siguiente imagen se muestra la configuración inicial de la radio E006:

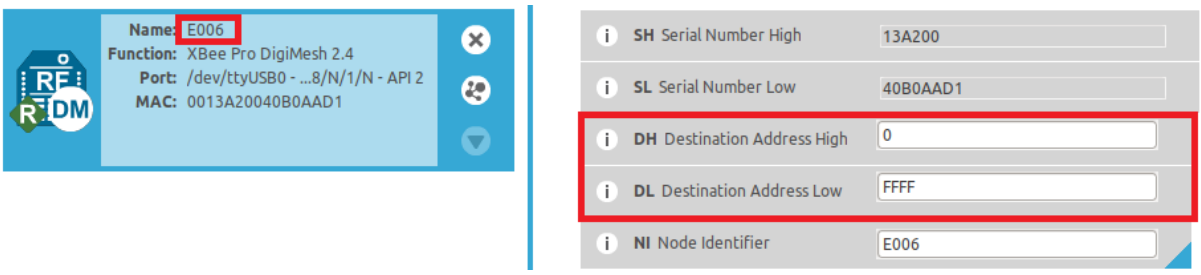


Ilustración 77: Configuración inicial de las radios XBee

2. Encender todas las radios.
3. Lanzar un comando ND, que se encarga de descubrir todos los nodos de la red. El protocolo DigiMesh se encarga de gestionar todo el proceso. Cada nodo descubierto devuelve un **Remote Command Response** como confirmación.
4. Lanzar comandos FN en cada radio, para que encuentre los nodos adyacentes a esa radio.
   Por ejemplo, para enviar este comando a la radio E006 y que descubra a sus vecinas BASE y E008, se le enviaría el siguiente paquete:

```
7E 00 0F 17 01 00 7D 33 A2 00 40 B0 AA D1 FF FE 02 46 4E 34

Start delimiter: 7E
Length: 00 0F (15)
Frame type: 17 (Remote AT Command Request)
Frame ID: 01 (1)
64-bit dest. address: 00 13 A2 00 40 B0 AA D1
16-bit dest. address: FF FE
Command options: 02
AT Command: 46 4E (FN)
Checksum: 34
```

El efecto del envío de este paquete, es que se recibirán tantos paquetes de tipo **Remote Command Response** como radios descubra a su alrededor, tal y como se observa en la siguiente traza:

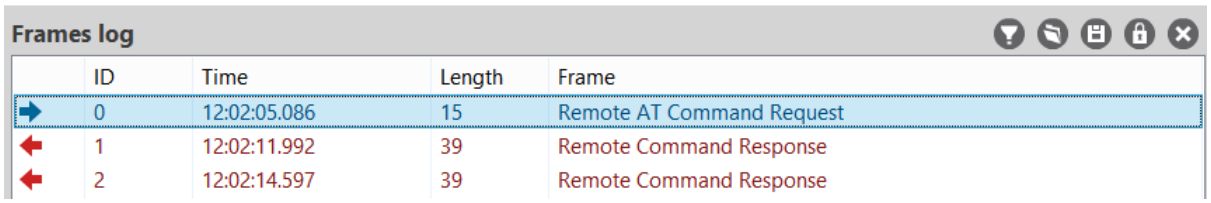


Ilustración 78: Traza del comando FN

Donde el paquete ID1 es una respuesta desde la radio BASE:

```
7E 00 27 97 01 00 7D 33 A2 00 40 A1 CE 39 FF FE 46 4E 00 FF FE 00 7D 33 A2
00 40 A1 CE 39 42 41 53 45 00 FF FE 01 00 C1 05 10 1E 41 51

Start delimiter: 7E
Length: 00 27 (39)
Frame type: 97 (Remote Command Response)
Frame ID: 01 (1)
64-bit source address: 00 13 A2 00 40 A1 CE 39
16-bit source address: FF FE
AT Command: 46 4E (FN)
Status: 00 (Status OK)
Response: FF FE 00 13 A2 00 40 A1 CE 39 42 41 53 45 00 FF FE 01 00 C1 05 10
1E 41
Checksum: 51
```

Y el paquete ID2 es una respuesta desde la radio E008:

```
7E 00 27 97 01 00 7D 33 A2 00 40 B0 AB 2F FF FE 46 4E 00 FF FE 00 7D 33 A2
00 40 B0 AB 2F 45 30 30 37 00 FF FE 01 00 C1 05 10 1E 4A BB

Start delimiter: 7E
Length: 00 27 (39)
Frame type: 97 (Remote Command Response)
Frame ID: 01 (1)
64-bit source address: 00 13 A2 00 40 B0 AB 2F
16-bit source address: FF FE
AT Command: 46 4E (FN)
Status: 00 (Status OK)
Response: FF FE 00 13 A2 00 40 B0 AB 2F 45 30 30 37 00 FF FE 01 00 C1 05 10
1E 4A
Checksum: BB
```

Los pasos 3 y 4 permiten hacer un mapa de toda la red de sensores.

5. Seguidamente, se configurarían las radios para que se comuniquen con la radio BASE. Esto se realiza lanzando el comando AG en la radio BASE, que se enviará en modo broadcast a todas las radios, que una vez reciban el paquete, configurarán sus parámetros DH y DL a los valores de SH y SL de la radio BASE. Para que la configuración de los parámetros tenga efecto, hay que enviar el comando AG con el parámetro FFFF, para que coincida con la configuración descrita en el paso 1.

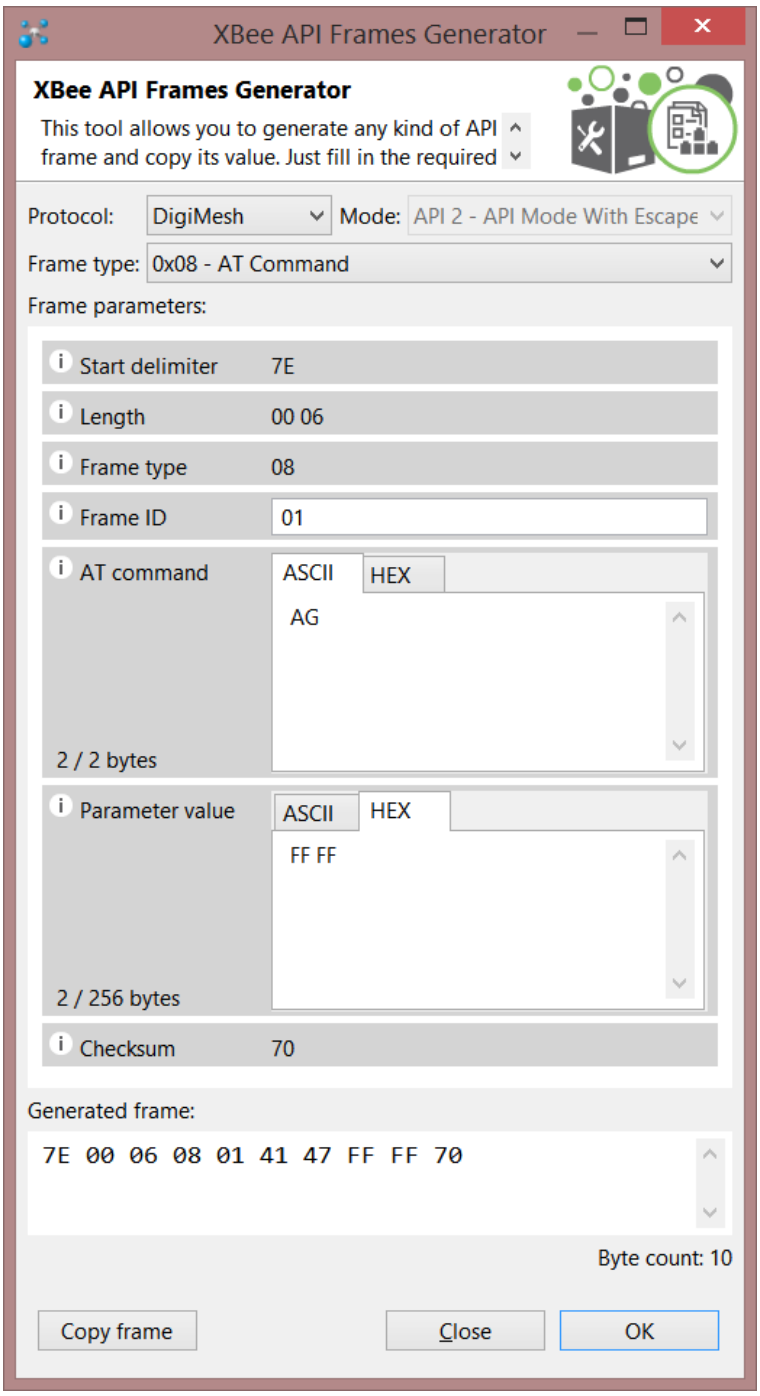


Ilustración 79: Generación del comando AG

Hay que tener en cuenta que el parámetro FFFF se debe escribir en la parte hexadecimal y no en la ASCII para que lo interprete correctamente.

6. Cuando cada radio haya recibido el comando AG, reconfigurará sus parámetros SH y SL, indicándolo con un paquete del tipo **Aggregate Addressing Update** (0x8E). En la siguiente imagen se puede apreciar el efecto que se produce en la radio E006:

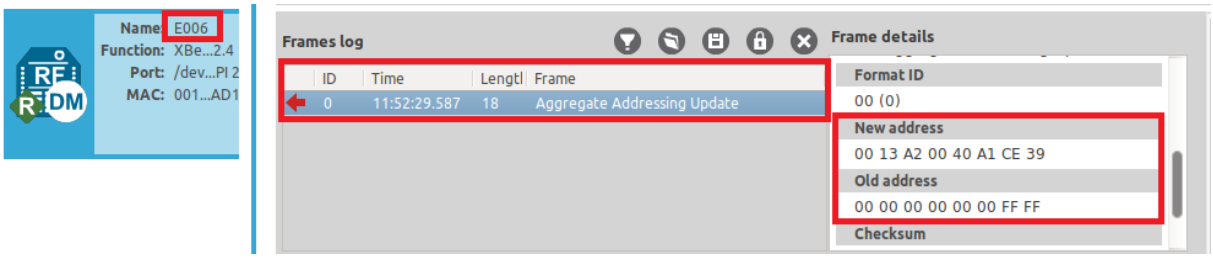


Ilustración 80: Paquete de respuesta a un comando AG

Además, si se desea realmente comprobar que la nueva dirección se ha escrito correctamente, se puede conectar la radio a XCTU y consultar sus valores:

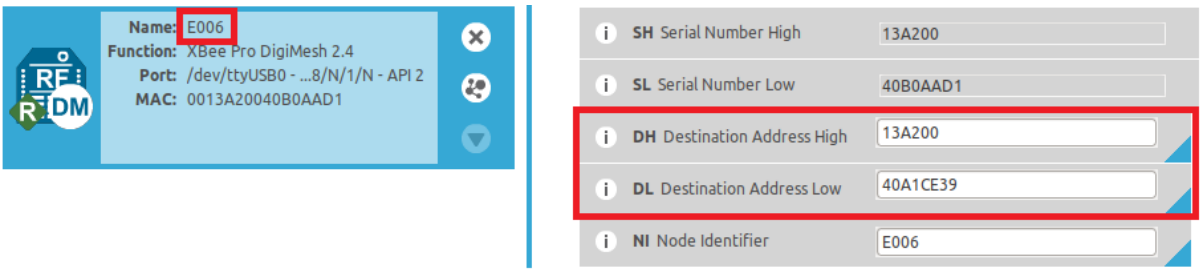


Ilustración 81: Valores de DH y DL después de un comando AG

7. Finalmente, como la red ya está establecida de antemano, el protocolo se encargará de descubrir la ruta más óptima desde cada radio hasta la radio BASE, ruta que se escribirá en las tablas de rutas de cada radio.

7.3.3. PRUEBAS ENTRE XCTU Y ARDUINO

Hasta ahora, las pruebas se han realizado con las radios conectadas a otros ordenadores y monitorizándolas con XCTU. En estas nuevas pruebas, el escenario cambia ligeramente, de tal manera que:

- La radio BASE está conectada a un PC con XCTU.
- Las radios E006 y E008 están conectadas a las estaciones de viento con Arduino.
- El Arduino utiliza la librería XBee Arduino para interactuar con la radio XBee. Tiene cargado un programa "echo", que devuelve los mismos datos que se le envían.

En la siguiente imagen se muestra un esquema del escenario de pruebas:

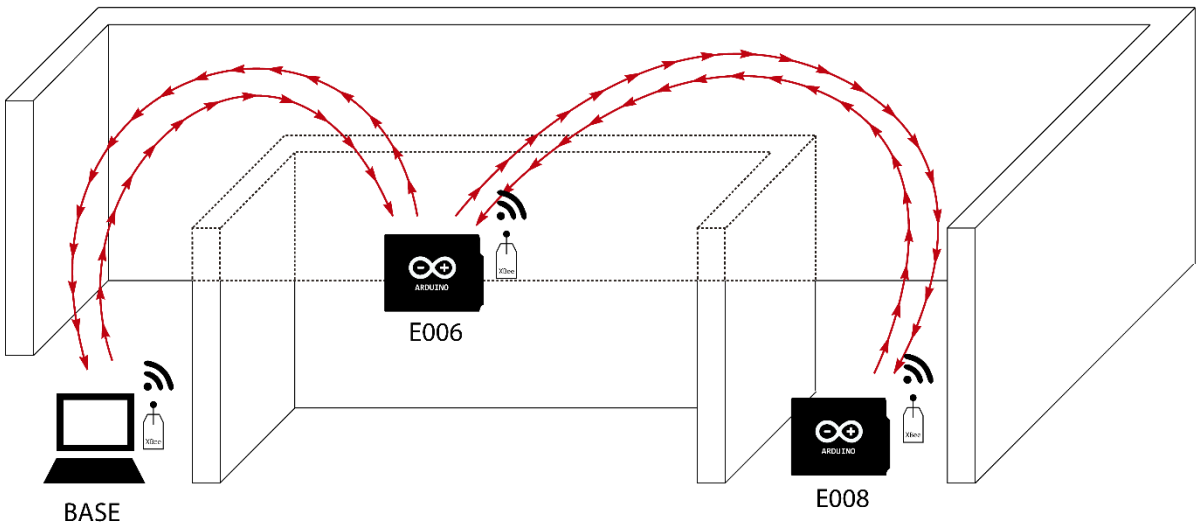


Ilustración 82: Escenario de pruebas entre XCTU y Arduino

7.3.3.1. ENVÍO DE MENSAJES CON SALTOS

Las pruebas que se van a realizar se enviarán siempre a la estación E008 para comprobar que la comunicación entre esta radio y la radio BASE se realiza correctamente en ambos sentidos.

7.3.3.1.1. ENVÍO DE MENSAJES DIRECTOS

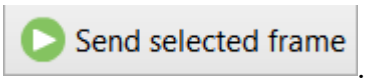
El primer caso de prueba consiste en enviar un mensaje directo a la radio, es decir, indicando la dirección MAC de la radio E008.

1. Desde el XCTU, se genera un frame de tipo **Transmit Request**, como se ha venido haciendo hasta ahora. El dato que se va a enviar es una cadena de caracteres con la palabra "Prueba".
   Los datos del paquete se muestran a continuación:

```
7E 00 14 10 01 00 7D 33 A2 00 40 B0 AB 2F FF FE 00 00 50 72 75 65 62 61 7D
33

Start delimiter: 7E
Length: 00 14 (20)
Frame type: 10 (Transmit Request)
```

```
Frame ID: 01 (1)
64-bit dest. address: 00 13 A2 00 40 B0 AB 2F
16-bit dest. address: FF FE
Broadcast radius: 00 (0)
Options: 00
RF data: 50 72 75 65 62 61 (Prueba)
Checksum: 13
```

2. Se envía el frame desde la BASE con el botón 

. La traza se puede observar en la siguiente imagen:

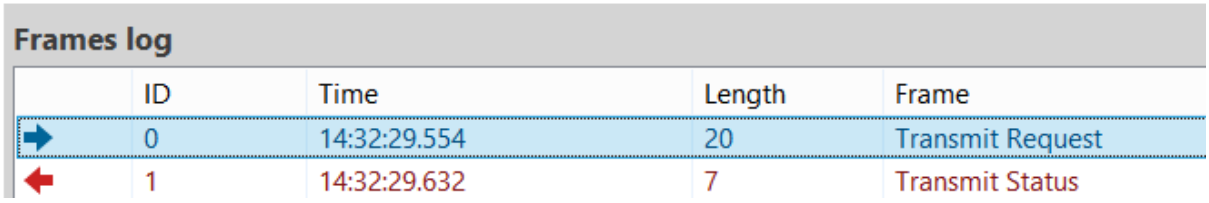


Ilustración 83: Envío de datos a la estación E008

3. Se recibe el "echo" desde el Arduino E008:

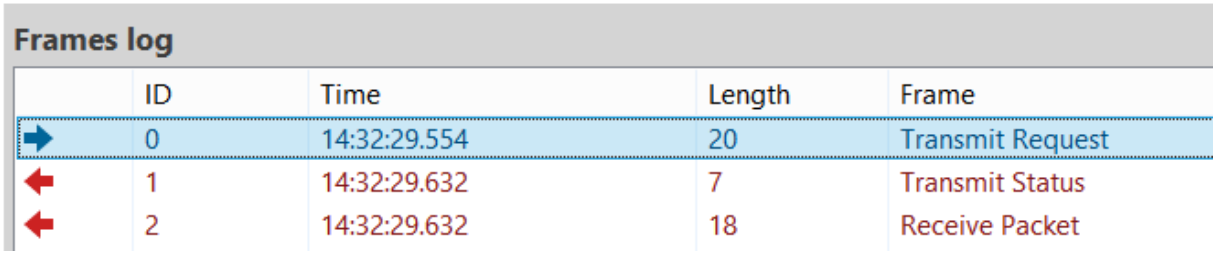


Ilustración 84: Recepción del "echo" desde la estación E008

Para comprobar que se ha devuelto la cadena enviada, se mira el contenido del paquete ID2, que es el siguiente:

```
7E 00 12 90 00 7D 33 A2 00 40 B0 AB 2F FF FE C1 50 72 75 65 62 61 D3

Start delimiter: 7E
Length: 00 12 (18)
Frame type: 90 (Receive Packet)
64-bit source address: 00 13 A2 00 40 B0 AB 2F
16-bit source address: FF FE
Receive options: C1
RF data: 50 72 75 65 62 61 (Prueba)
Checksum: D3
```

Como se puede observar, los datos del campo **RF data** corresponden a los mismos del paquete descrito en el paso 1.

7.3.3.1.2. BROADCAST

En la siguiente prueba se comprobará que los mensajes de broadcast llegan a todas las radios conectadas a Arduino, independientemente de si se ven directamente con la radio BASE o a través de otras radios.

El proceso de envío del broadcast se realiza como en pruebas anteriores:

1. Desde el XCTU, se genera un frame de tipo **Transmit Request**. El dato que se va a enviar es una cadena de caracteres con el texto "Esto es un broadcast".
   Los datos del paquete se muestran a continuación:

```
7E 00 22 10 01 00 00 00 00 00 00 FF FF FF FE 00 00 45 73 74 6F 20 65 73 20
75 6E 20 62 72 6F 61 64 63 61 73 74 8A

Start delimiter: 7E
Length: 00 22 (34)
Frame type: 10 (Transmit Request)
Frame ID: 01 (1)
64-bit dest. address: 00 00 00 00 00 00 FF FF
16-bit dest. address: FF FE
Broadcast radius: 00 (0)
Options: 00
RF data: 45 73 74 6F 20 65 73 20 75 6E 20 62 72 6F 61 64 63 61 73 74 (Esto
es un broadcast)
Checksum: 8A
```

2. Se envía el frame desde la BASE. La traza se observa a continuación:

Frames log

	ID	Time	Length	Frame
→	0	15:06:27.228	34	Transmit Request
←	1	15:06:27.337	7	Transmit Status

Ilustración 85: Envío de datos en modo broadcast

3. Se recibe el "echo" de la estación E006:

Frames log

	ID	Time	Length	Frame
→	0	15:06:27.228	34	Transmit Request
←	1	15:06:27.337	7	Transmit Status
←	2	15:06:27.337	32	Receive Packet

Ilustración 86: Recepción del "echo" desde la estación E006

El contenido del paquete con ID2 se puede ver a continuación:

```
7E 00 20 90 00 7D 33 A2 00 40 B0 AA D1 FF FE C1 45 73 74 6F 20 65 73 20 75
6E 20 62 72 6F 61 64 63 61 73 74 28

Start delimiter: 7E
Length: 00 20 (32)
Frame type: 90 (Receive Packet)
64-bit source address: 00 13 A2 00 40 B0 AA D1
16-bit source address: FF FE
Receive options: C1
RF data: 45 73 74 6F 20 65 73 20 75 6E 20 62 72 6F 61 64 63 61 73 74 (Esto
es un broadcast)
Checksum: 28
```

Donde se puede apreciar la dirección de la estación E006 y que el contenido del campo RF data se corresponde con el dato descrito en el paso 1.
Además, a modo informativo, se muestra la traza por el puerto serie del Arduino en la estación E006:

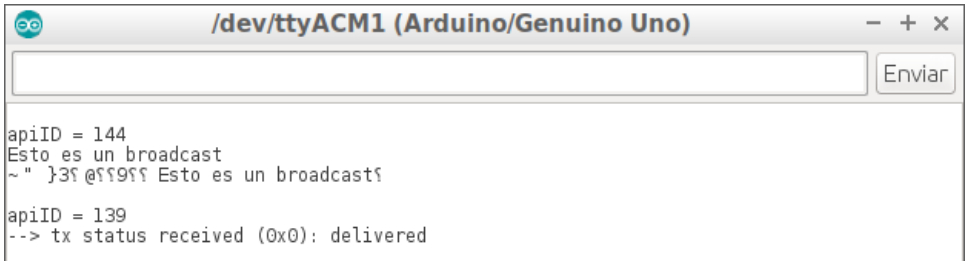


Ilustración 87: Traza del Arduino E006 mediante puerto serie

4. A continuación se recibe el "echo" de la estación E008:

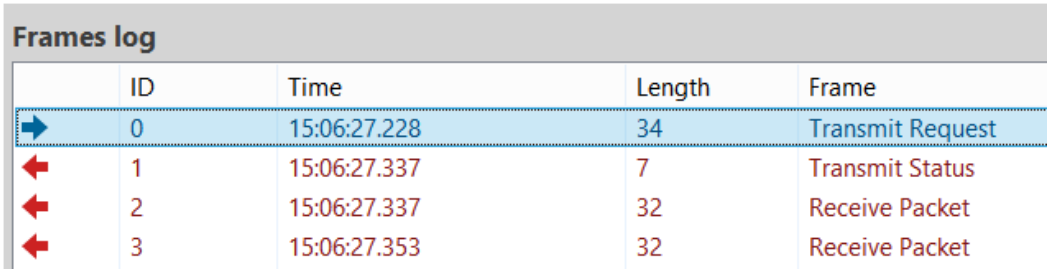


Ilustración 88: Recepción del "echo" desde la estación E008

El contenido del paquete con ID3 es el siguiente:

```
7E 00 20 90 00 7D 33 A2 00 40 B0 AB 2F FF FE C1 45 73 74 6F 20 65 73 20 75
6E 20 62 72 6F 61 64 63 61 73 74 C9

Start delimiter: 7E
Length: 00 20 (32)
Frame type: 90 (Receive Packet)
64-bit source address: 00 13 A2 00 40 B0 AB 2F
16-bit source address: FF FE
Receive options: C1
RF data: 45 73 74 6F 20 65 73 20 75 6E 20 62 72 6F 61 64 63 61 73 74
Checksum: C9
```

Al igual que con el otro paquete, se observa la dirección de la radio correspondiente, en este caso la E008 y que el contenido de los datos es el mismo que el enviado en el paso 1.
También se muestra, como en el caso anterior, la traza recogida por el puerto serie del Arduino E008:

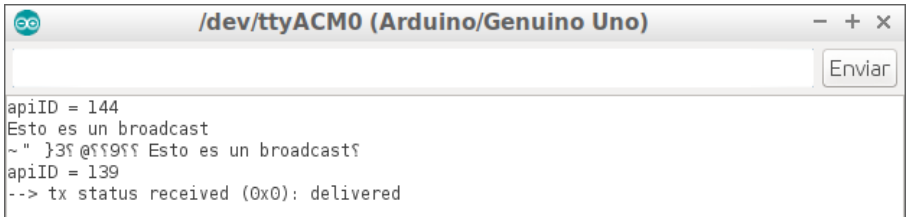


Ilustración 89: Traza del Arduino E008 mediante puerto serie

7.3.3.2. PRUEBAS SUPER_XBEE Y LIBXBEE3

En esta batería de pruebas se cambia el escenario ligeramente, donde ya se deja de lado el XCTU para sustituirlo por un Programa de pruebas en C++, que utiliza la librería libxbee3 para interactuar con la radio BASE, por lo tanto, la configuración del escenario queda como sigue:

- La radio BASE está conectada a un PC con un Programa de pruebas en C++.
- Las radios E006 y E008 están conectadas a las estaciones de viento con Arduino.
- Las placas Arduino tienen cargadas un programa "echo", que devuelve los mismos datos que se le envían.

Un esquema del escenario se muestra a continuación:

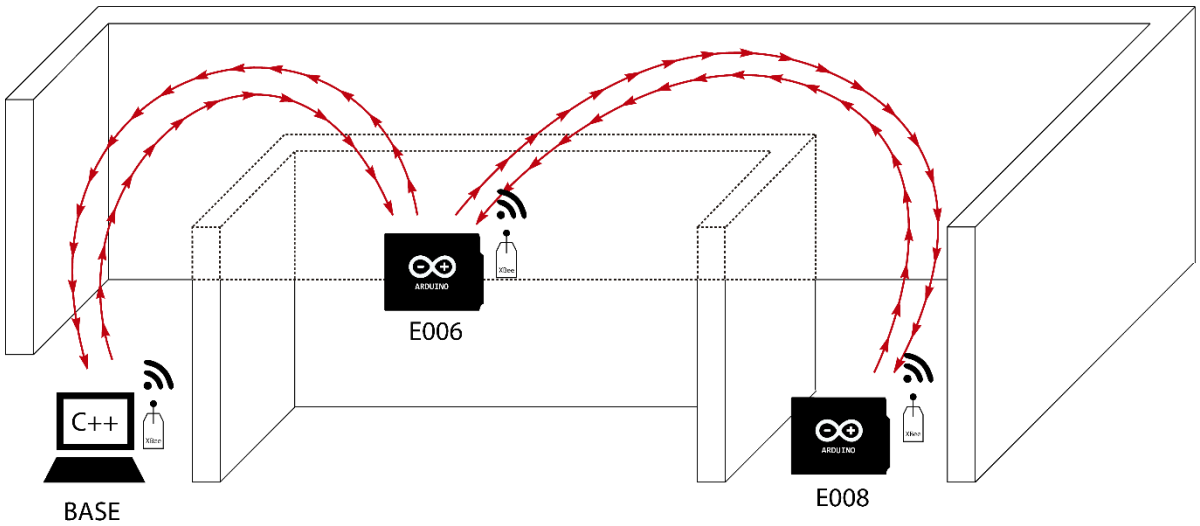


Ilustración 90: Escenario entre C++ y los sensores

7.3.3.2.1. MENSAJES DIRECTOS

La primera prueba que se va a realizar es enviar un mensaje directo desde la radio BASE a la radio E006, que se comunican directamente. El proceso para realizar la prueba se describe a continuación:

1. Desde Linux, se lanza el programa de pruebas en modo explícito.
2. Se escribe la frase "Mensaje explícito a la radio" y se pulsa enter:

```
yasypol@lubuone:~/super_xbee/build$ sudo ./super_xbee /dev/ttyUSB0
type 0: Modem Status
type 1: Transmit Status
type 2: Local AT
type 3: Remote AT
type 4: Data
type 5: Data (explicit)
type 6: I/O
type 7: Sensor
type 8: Identify
>>? Mensaje explícito a la radio
```

Ilustración 91: Envío de mensaje directo a la radio E006

   El programa se quedará a la espera de que el Arduino envíe un paquete con el "echo".
3. La estación recibe el paquete, como se puede comprobar en la siguiente imagen:

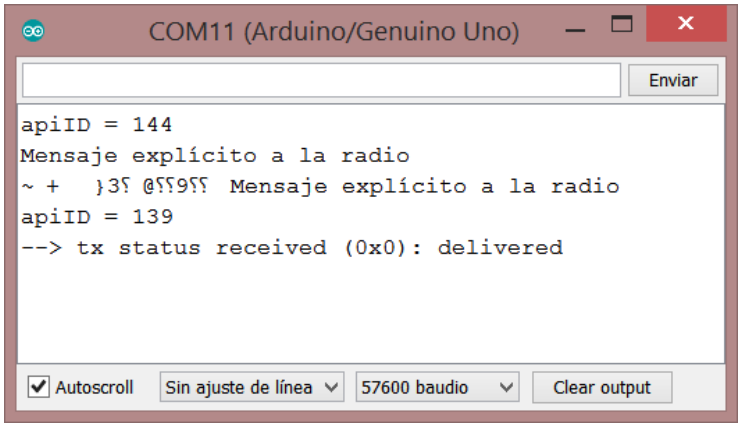


Ilustración 92: Recepción de paquete directo desde C++ en Arduino E008

La conclusión de esta prueba es que el envío de paquetes directos funciona correctamente.

7.3.3.2.2. BROADCAST ENTRE BASE Y ARDUINO

En este caso, se enviará un broadcast a todas las estaciones y se comprobará que todas lo reciben correctamente. El proceso llevado a cabo es el siguiente:

1. Desde Linux, se lanza el programa de pruebas en modo broadcast.
2. Se escribe la frase "Broadcast desde C++ usando libxbee3" y se pulsa enter:

```
yasypol@lubuone:~/super_xbee/build$ sudo ./super_xbee /dev/ttyUSB0
type 0: Modem Status
type 1: Transmit Status
type 2: Local AT
type 3: Remote AT
type 4: Data
type 5: Data (explicit)
type 6: I/O
type 7: Sensor
type 8: Identify
>>? Broadcast desde C++ usando libxbee3
```

Ilustración 93: Envío de broadcast mediante C++

El mensaje se enviará en bucle hasta que se pulse Ctrl+C

3. Las estaciones empiezan a recibir datos, como se puede comprobar en las siguientes imágenes:

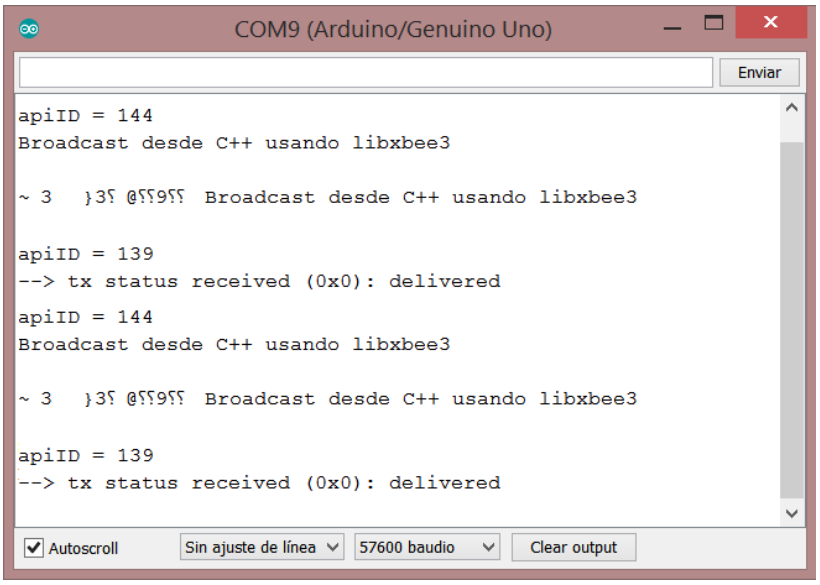


Ilustración 94: Recepción de broadcast desde C++ en Arduino E006

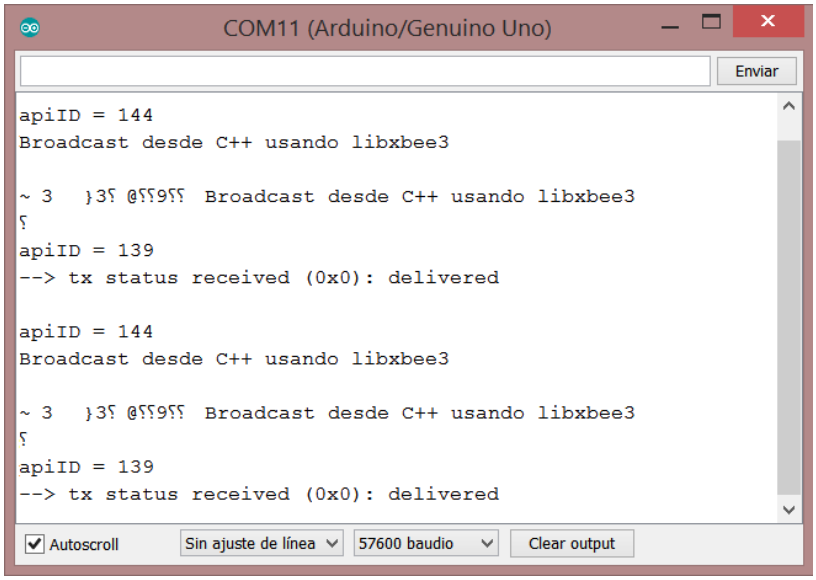


Ilustración 95: Recepción de broadcast desde C++ en Arduino E008

Se puede comprobar que los mensajes tipo broadcast llegan a todas las radios, independientemente del número de saltos que las separen de la radio BASE.

7.3.3.3. PRUEBAS CON METEO

En estos escenarios de prueba se comprobará la comunicación de las estaciones directamente y mediante saltos.

7.3.3.3.1. COMUNICACIÓN CON LA BASE DIRECTAMENTE

En este caso de prueba, el escenario cambia de la siguiente manera:

- La radio BASE está conectada a un PC con el meteo (C++ y libxbee3).
- Las radios E006 y E008 están conectadas a las estaciones de viento con Arduino.
- La placa Arduino del sensor E006 tiene cargada el programa de medición de viento.

El esquema de pruebas se puede ver a continuación:

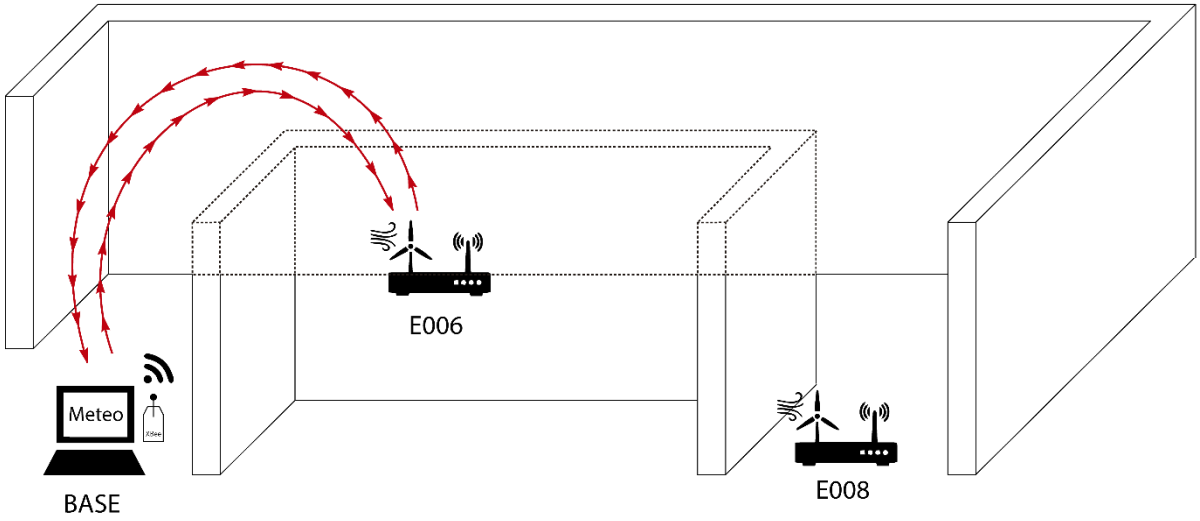


Ilustración 96: Escenario de pruebas con comunicación directa con la estación base

Para ir haciendo pruebas incrementales, primero se hará una conexión entre uno de los sensores de vientos y la estación base y se comprobará que se reciben datos correctamente desde ese sensor.

Para ello, se siguen los siguientes pasos:

1. Desde el terminal de Linux, se lanza el programa meteo, cuya interfaz aparece vacía nada más arrancarlo:

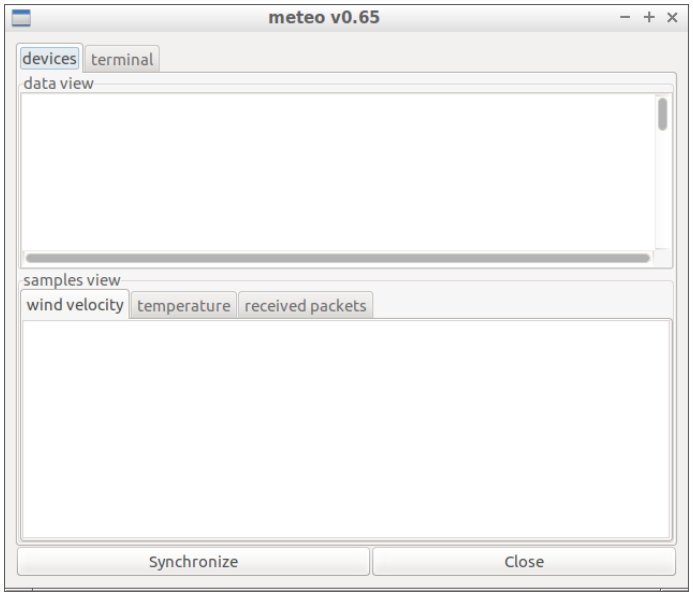


Ilustración 97: Interfaz de meteo al iniciar la aplicación

2. Luego, se procede a encender el sensor de viento E006 y calibrarlo, cuyo proceso se puede ver en la siguiente imagen:

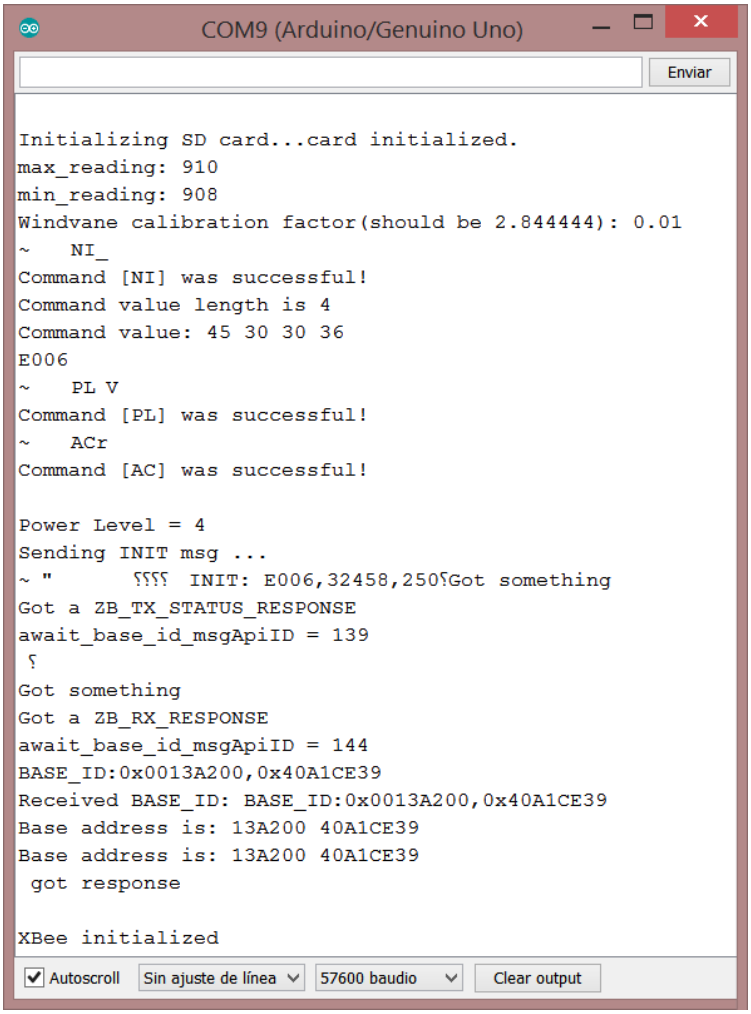


Ilustración 98: Traza de inicialización del sensor de viento

Como se puede comprobar, se calibra la veleta, se envía un mensaje INIT a la estación base y ésta contesta enviando su dirección MAC, así la estación sabrá con quién comunicarse de ahora en adelante.

3. Una vez que el sensor ha enviado el mensaje INIT, aparece en la interfaz de meteo:

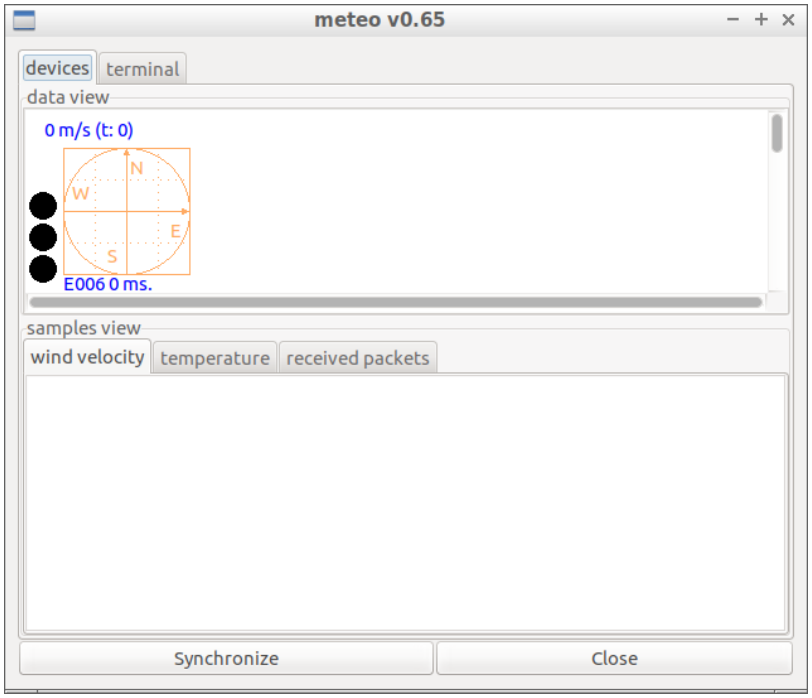


Ilustración 99: Inclusión del sensor E006 en la interfaz meteo

4. Para que el sensor empiece a recoger datos de la veleta y el anemómetro, se debe pulsar sobre el botón **Synchronize**, que se encargará de enviar un mensaje broadcast con una marca de tiempo con precisión de milisegundos a todas las estaciones conectadas (en este caso la E006).
5. Cuando el sensor de viento recibe el mensaje SYNC, comienza a recoger datos y a guardarlos en su propia MicroSD y a mandarlos a la estación base:

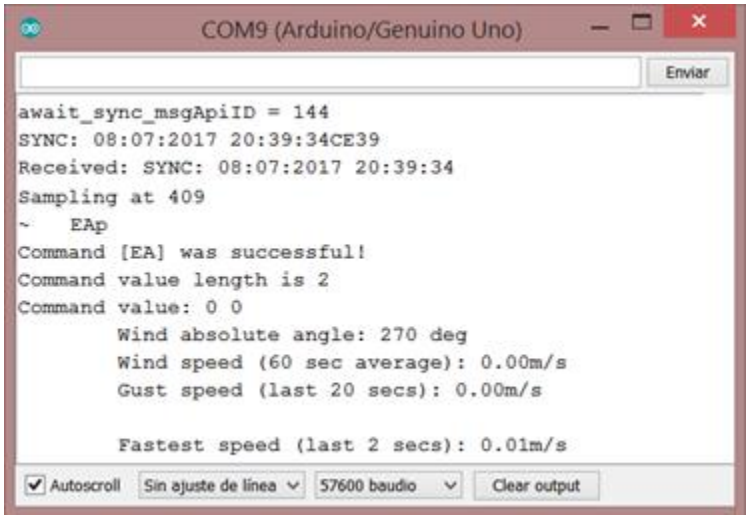


Ilustración 100: Traza de repción del mensaje SYNC y envío de datos

6. Los datos enviados a la estación base deben verse en la interfaz de meteo, tal y como se muestra a continuación:

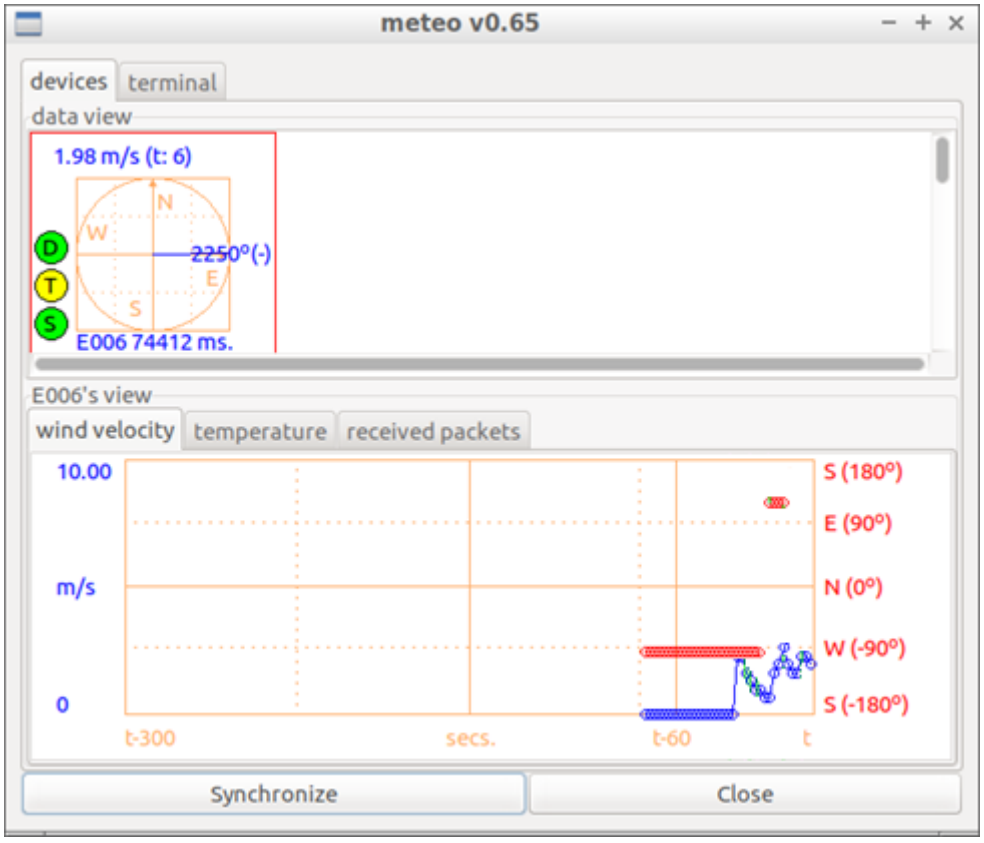


Ilustración 101: Visualización de los datos de los sensores en meteo

Lo que se puede deducir de esta prueba, es que la comunicación entre meteo y un sensor de viento se realiza de manera correcta.

7.3.3.3.2. COMUNICACIÓN CON LA BASE CON SALTOS

Estas pruebas se van a dividir en 2 escenarios diferentes, para ir avanzando poco a poco en los cambio de los componentes.

7.3.3.3.2.1. ESCENARIO 1

El primer escenario se compone de los siguientes elementos:

- La radio BASE está conectada a un PC con el programa meteo (C++ y libxbee3).
- Las radios E006 y E008 están conectadas a las estaciones de viento con Arduino.
- La placa Arduino del sensor E006 tiene el programa "echo" de pruebas anteriores.
- La placa Arduino del sensor E008 tiene cargada el programa de medición de viento.

Para una mayor comprensión del escenario, se muestra el esquema a continuación:

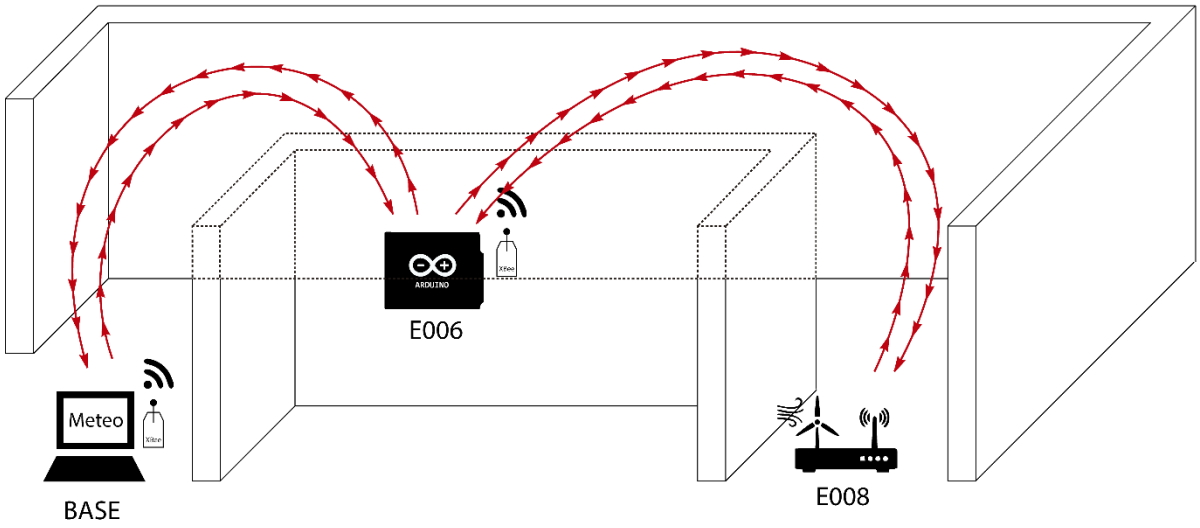


Ilustración 102: Escenario de pruebas con meteo, "echo" y sensor de viento

Las fases del experimento son similares al caso anterior:

1. Desde el terminal de Linux, se lanza el programa meteo.
2. Luego, se procede a encender el sensor de viento E008 y calibrarlo, cuyo proceso se puede ver en la siguiente imagen:

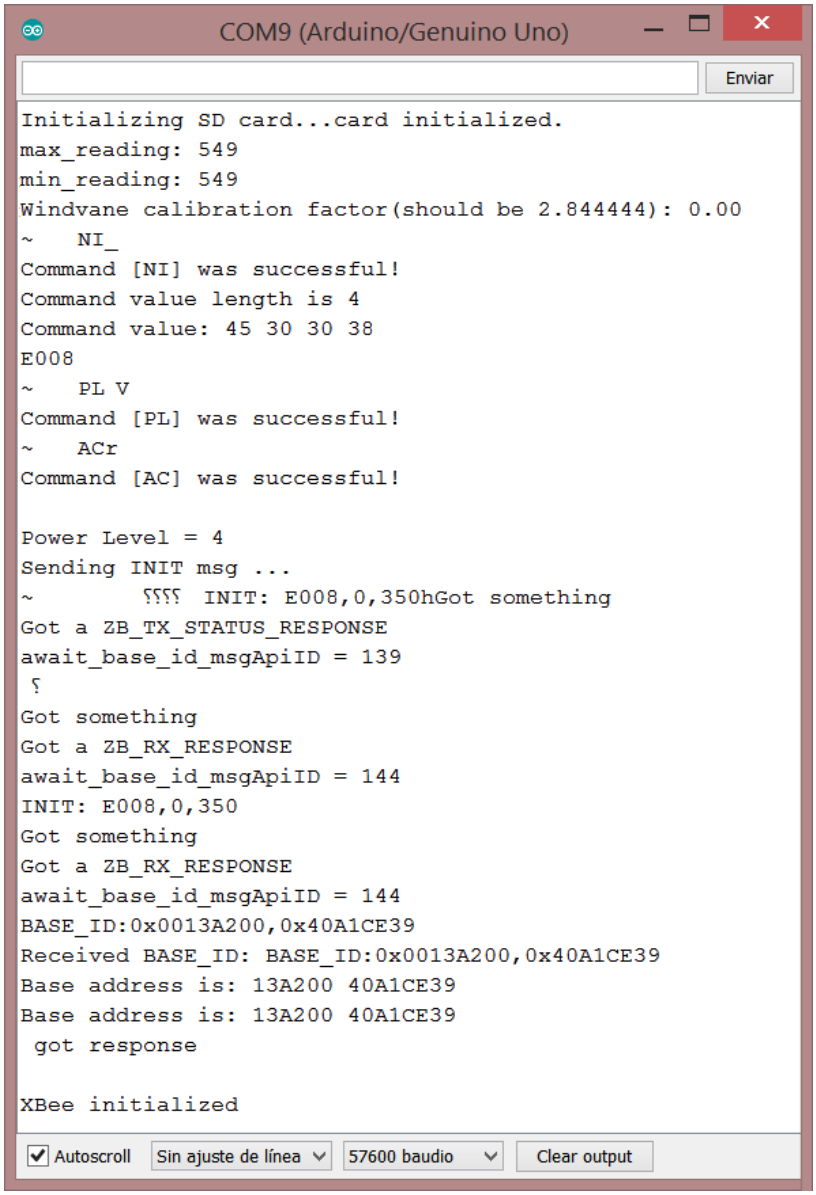


Ilustración 103: Traza de inicialización del sensor de viento E008

Al igual que en el caso anterior, se calibra la veleta, se envía un mensaje INIT a la estación base y ésta contesta enviando su dirección MAC.

3. Una vez que el sensor ha enviado el mensaje INIT, aparece en la interfaz de meteo:

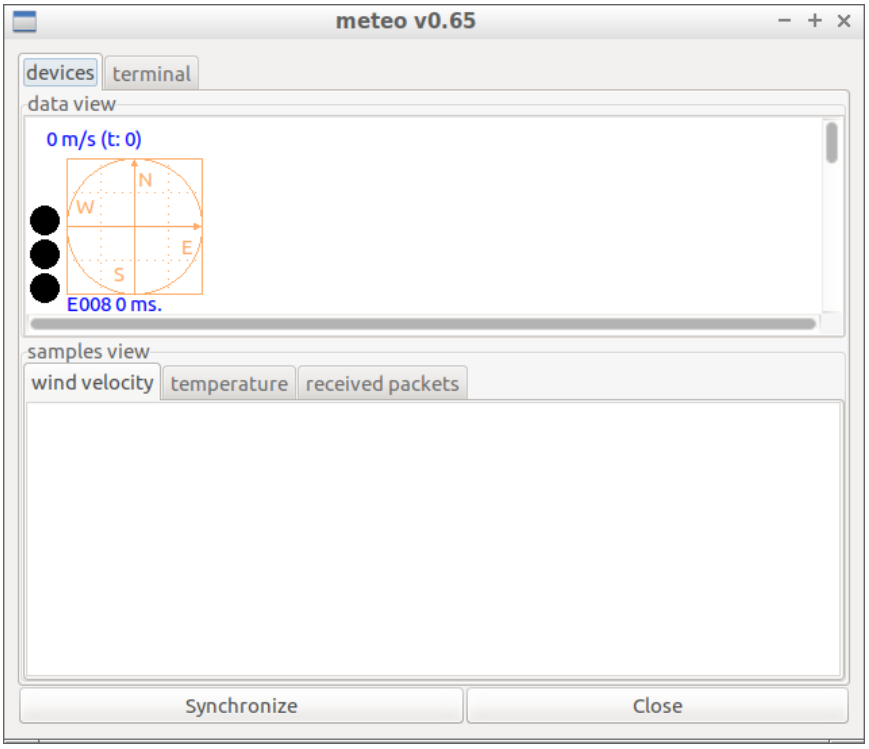


Ilustración 104: Inclusión del sensor E008 en la interfaz meteo

4. Se pulsa sobre el botón **Synchronize**, que se enviará en modo broadcast, llegando al sensor E008.
5. Cuando el sensor de viento recibe el mensaje SYNC, comienza a recoger datos y a guardarlos en su propia MicroSD y a mandarlos a la estación base:

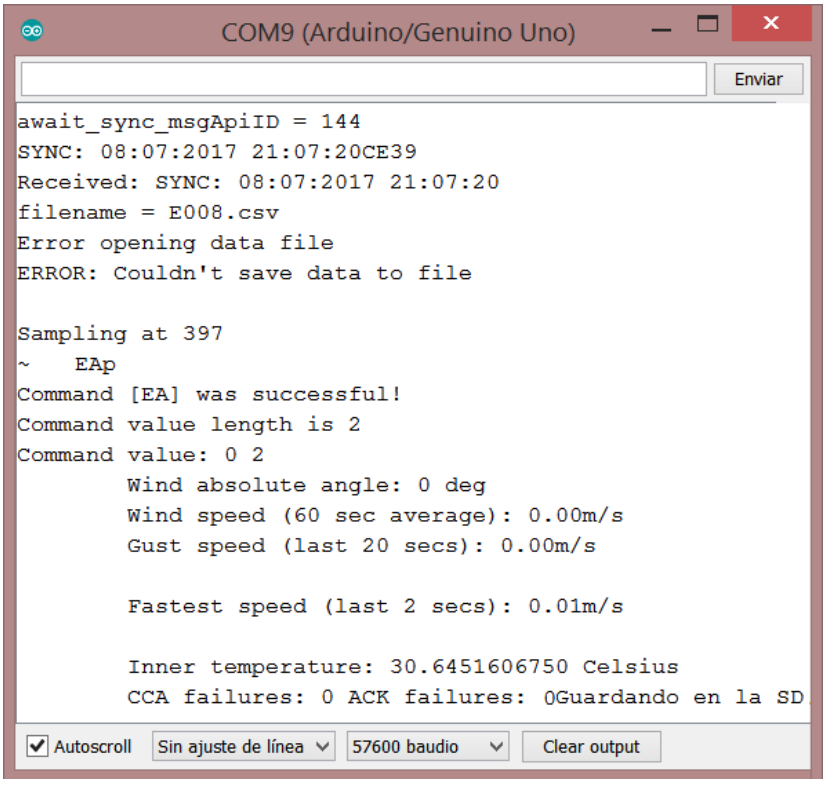


Ilustración 105: Traza de repción del mensaje SYNC y envío de datos

6. Los datos enviados a la estación base deben verse en la interfaz de meteo, tal y como se muestra a continuación:

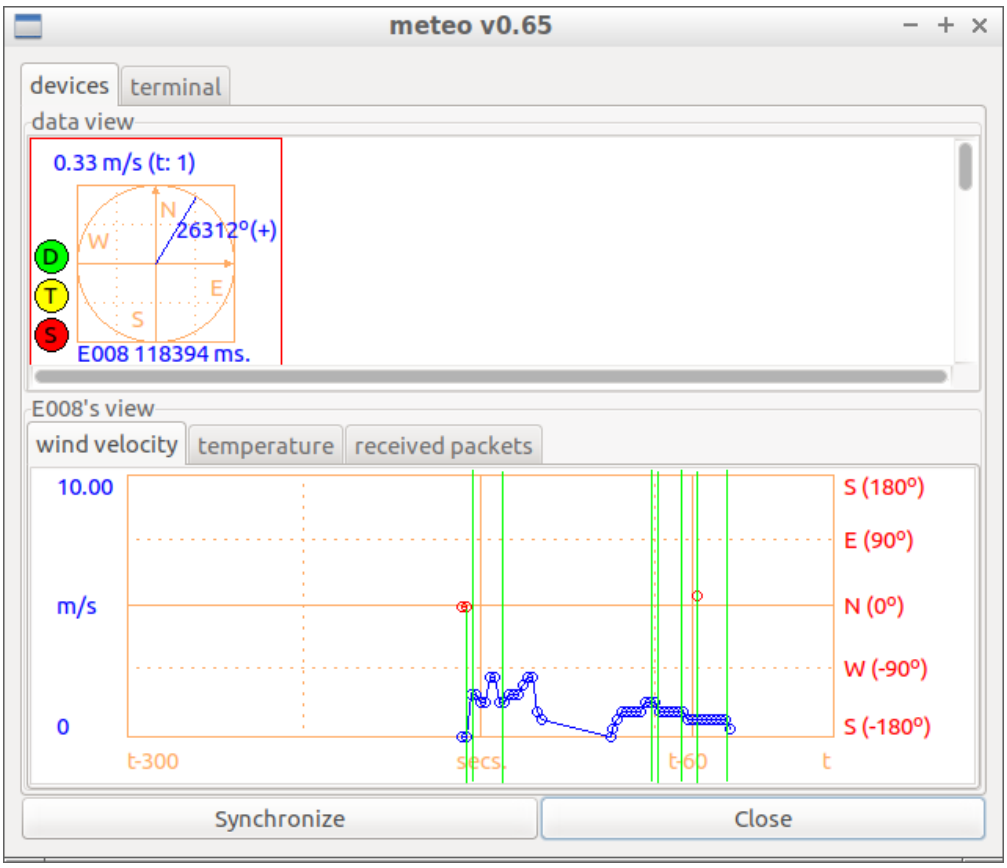


Ilustración 106: Visualización de los datos del sensor E008 en meteo

Por lo tanto, de esta prueba se puede deducir que la comunicación entre un sensor de viento y la base con saltos por medio, con un Arduino ejecutando la librería para XBee, se realiza de manera correcta.

7.3.3.3.2.2. ESCENARIO 2

En el segundo escenario, se comprobará la comunicación del sensor E008 con meteo teniendo como nodo intermedio el sensor E006. Por lo tanto, ambos sensores incluirán el software de medición de viento.

Los componentes de este escenario son los siguientes:

- La radio BASE está conectada a un PC con el programa meteo (C++ y libxbee3).
- Las radios E006 y E008 están conectadas a las estaciones de viento con Arduino.
- Las placas Arduino tienen cargadas el programa de medición de viento.

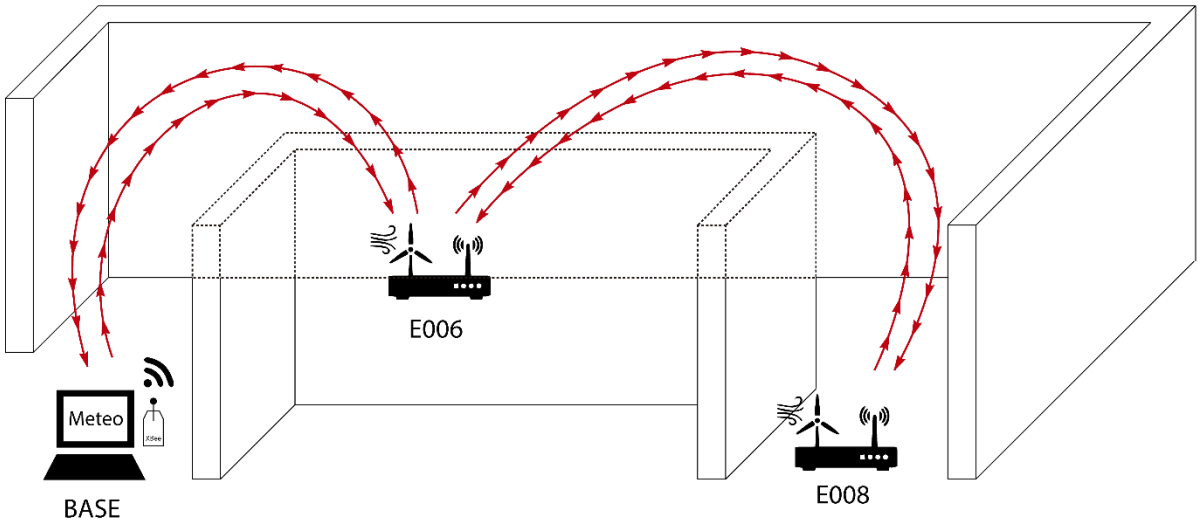


Ilustración 107: Escenario de pruebas con meteo y 2 sensores de viento

Las fases de este escenario de prueba son muy similares al caso anterior:

1. Desde el terminal de Linux, se lanza el programa meteo.
2. Luego, se procede a encender los sensores de viento E006 y E008 y se calibran.
3. Al igual que en el caso anterior, se calibra la veleta. Los sensores envían un mensaje INIT a la estación base y ésta contesta a cada estación enviando su dirección MAC.
4. Una vez que los sensores han enviado el mensaje INIT, aparece en la interfaz de meteo:

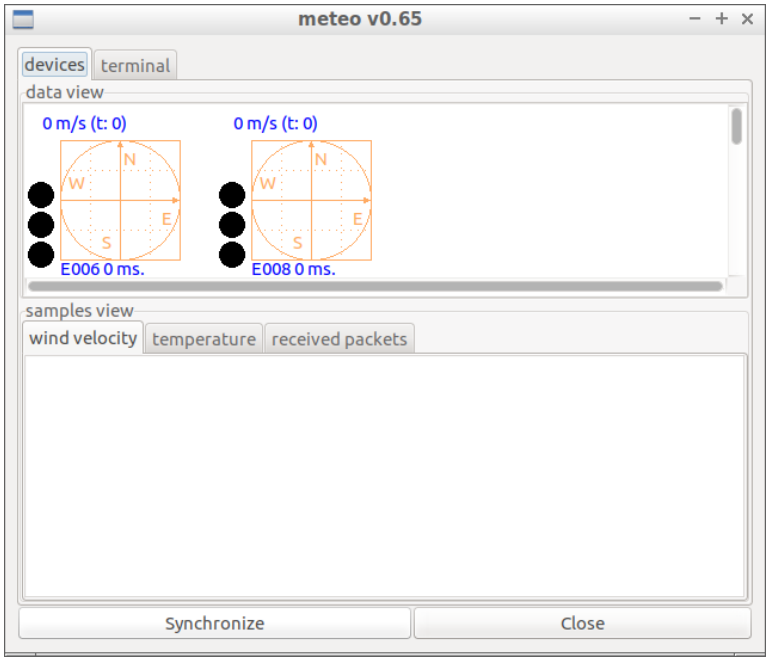


Ilustración 108: Inclusión de los sensores en la interfaz meteo

5. La estación E006 recibe el mensaje con la MAC de la radio BASE y queda a la espera del mensaje SYNC, como en pruebas anteriores.
6. Sin embargo, el sensor de viento E008 se queda a la espera del mensaje con la MAC de la radio BASE, que no llega a recibir nunca:

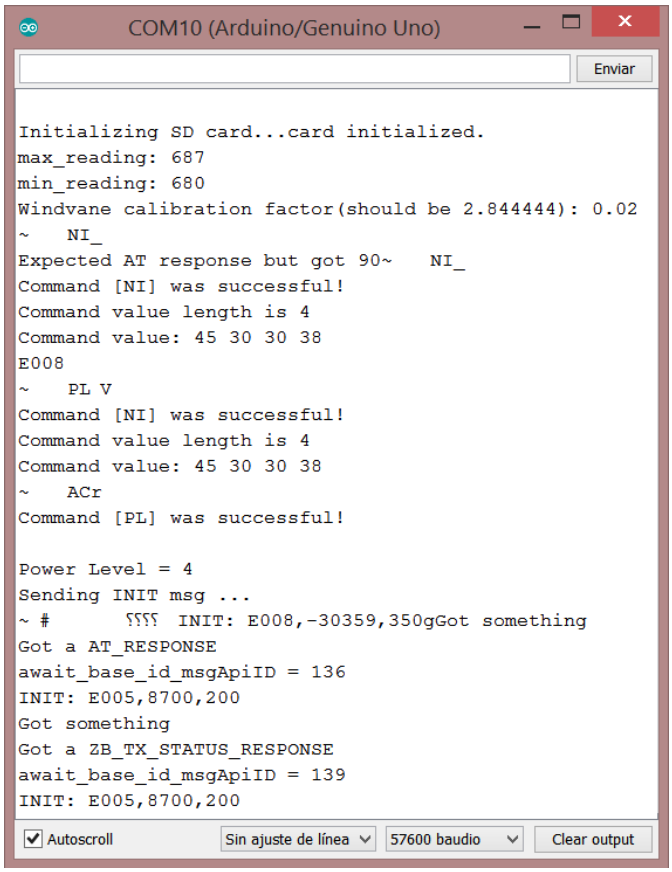


Ilustración 109: Sensor de viento E008 a la espera del mensaje BASE

7. En este momento se pulsa sobre el botón **Synchronize**, que sincronizará todos los sensores.
    a. El sensor E008 no se sincronizará porque está a la espera del mensaje BASE.
    b. El sensor de viento E006 recoge el mensaje de sincronización y comienza a enviar datos.
8. En este caso, en la interfaz de meteo se verá la estación E006 activa, pero la E008 no:

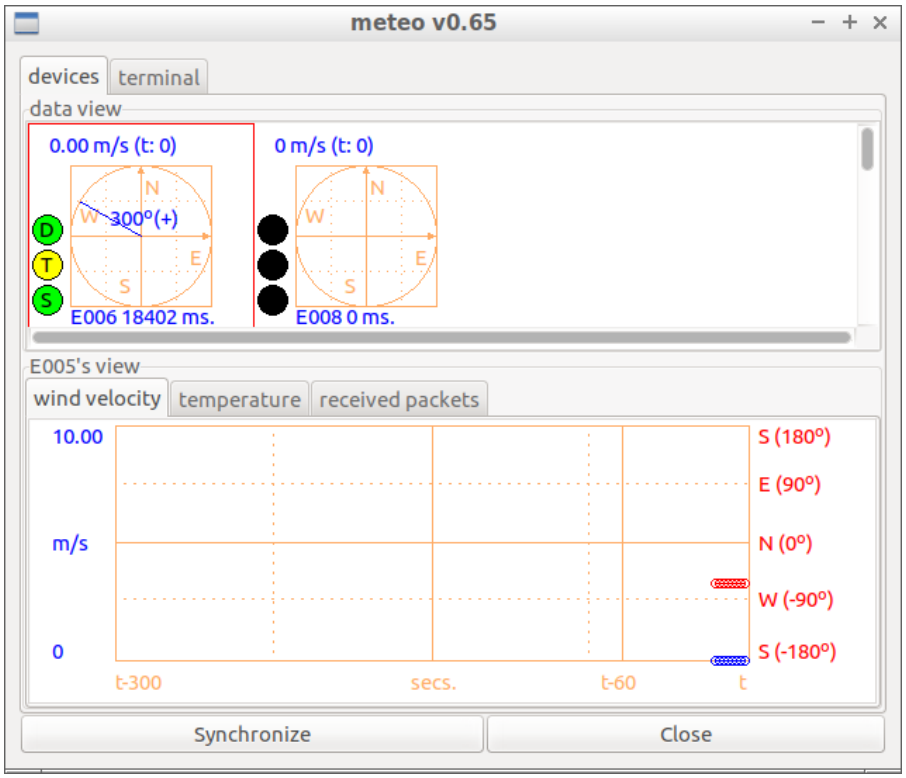


Ilustración 110: Visualización de los datos de los sensores E006 y E008 en meteo

Las conclusiones que se pueden sacar de este experimento es que hay algún aspecto del código de meteo o de Arduino que está entorpeciendo el envío o recepción de los paquetes enviados desde otros sensores de viento.

8. CONCLUSIONES

El desarrollo de este Trabajo de Fin de Grado, que tenía como objetivo reconfigurar las radios XBee para que use el protocolo DigiMesh, ha llevado a diversas conclusiones como:

- Las librerías libxbee3 y XBee Arduino en principio cubren todas las necesidades para adaptar el sistema desde ZigBee a DigiMesh.
- La adaptación del código meteo y de Arduino debe extenderse más allá de la gestión de los paquetes que usa DigiMesh, es decir, hay que reestructurar el código para que añada funcionalidades como el descubrimiento de la red de sensores, la configuración remota de los mismos, las rutas hasta la radio BASE y la fase de sincronización.
- Con la nueva topología, la frecuencia de transmisión de datos, va a estar condicionada por la retransmisión de paquetes entre radios. En especial, las radios más cercanas a la estación base sufrirán una tasa de transmisión alta, pues tendrán que procesar sus propios paquetes y todos aquellos que les lleguen desde otras radios XBee.
  Esto implica un impacto en la duración de la batería de los sensores de viento y aún más importante, en la frecuencia máxima de comunicación de medidas que la red es capaz de enviar a la estación base. Básicamente, podrían provocarse cuellos de botella en estos sensores tan cercanos o incluso colisiones entre paquetes.

8.1. LÍNEAS FUTURAS

8.1.1. INCORPORACIÓN DE UN RECEPTOR GPS

Como en el escenario en el que se usarían estas estaciones puede abarcar kilómetros, una manera de simplificar el proceso de posicionamiento de los sensores de viento consistiría en utilizar receptores GPS en cada uno de ellos. A grandes rasgos, el procedimiento sería el siguiente:

- Se tomaría el tiempo UTC, que se recibe con los mensajes tipo NMEA que utiliza el GPS, de manera que así se sincronizarían los relojes de cada Arduino.
- Tal y como está concebido el sistema, la recuperación o incorporación de nodos es una cuestión que entraña complicaciones, pues al estar todas las estaciones ya en funcionamiento, la re-sincronización de los relojes implicaría reiniciar el sistema. Si se usara el tiempo UTC que se obtendría del GPS, el problema se solucionaría o al menos lo simplificaría en gran manera.
- Actualmente, el despliegue de estaciones se realiza mediante una estación total de topografía, lo cual puede resultar tedioso en extensiones de terreno grandes. Al incorporar el GPS, cada estación podría obtener su latitud y longitud periódicamente, valores que se irían promediando a lo largo de la sesión para obtener un margen de error relativamente pequeño.

Si se implementara esta solución, podría eliminarse la fase de sincronización mediante paquetes.

Contras o posibles complicaciones de esta línea de investigación:

- Aumento del uso de la batería de los sensores de viento. El consumo podría reducirse si el GPS se pusiera en un modo de bajo consumo durante la mayor parte del tiempo y que sólo se despertara cuando fuera necesario.
- La inclusión de una librería para el GPS podría exceder el tamaño máximo de la memoria del Arduino UNO, que actualmente está ocupada al 85%.

8.1.2. REESCRITURA DEL CÓDIGO DE METEO

Como se comentaba en las conclusiones, el código actual de la aplicación meteo necesita implementar ciertas mejoras que podrían ayudar en el despliegue de la red de sensores, a saber:

- Implementación de un apartado de configuración de la red, para automatizar o proporcionar herramientas que permitan mandar comandos de descubrimiento de la red, como son los comandos de DigiMesh ND, el FN y el AG.
  - > Como consecuencia de esta mejora, también debería implementarse una especie de "visor" de la red de sensores, donde se muestren qué radios se comunican entre sí.
  - > Otra consecuencia es la manera en la que se enviaría el mensaje con la MAC de la radio BASE, paso que podría omitirse si se implementa la gestión del comando AG y todo lo que ello conlleva.
- Cambios en la fase de sincronización y envío del paquete SYNC. Si se implementa el GPS citado anteriormente, esta fase se reestructuraría o desaparecería, dependiendo del enfoque.

8.1.3. REESCRITURA DEL CÓDIGO DE ARDUINO

Como el código de Arduino y el comportamiento de éste va ligado a meteo, habrá que reprogramar ciertas partes del código, acordes a los cambios que se realicen finalmente en meteo.

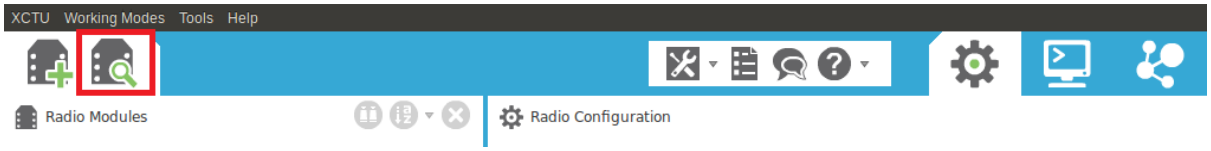
9. ANEXOS

9.1. DETECCIÓN DE RADIOS EN XCTU

Para empezar a trabajar con XCTU y una radio XBee, el primer paso es detectar la misma con el software. Esto se realiza conectando la radio a un explorer, que es un adaptador USB que irá conectado al computador:

Ilustración 111: Radio conectada a un explorer XBee

La forma más sencilla de descubrir un módulo de radio es entrar pulsar sobre el botón  de la barra superior de XCTU:



Luego, en el diálogo que aparece, seleccionaremos el puerto en el que está conectado el USB:

Ilustración 112: Selección del puerto donde está conectado el explorer USB

Luego, se configuran los parámetros de búsqueda del módulo:

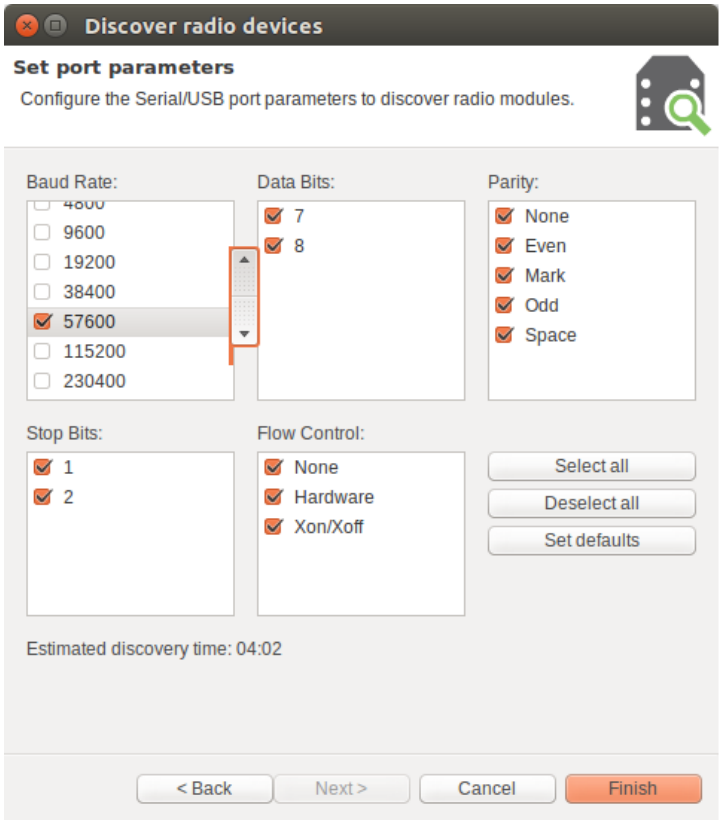


Ilustración 113: Parámetros de búsqueda de un módulo XBee

Saldrá una nueva ventana donde se pondrá a escanear el puerto elegido. Cuando se encuentre la radio, se mostrará en la lista de dispositivos encontrados. Para añadir el módulo, se pulsa sobre **Add selected devices**:

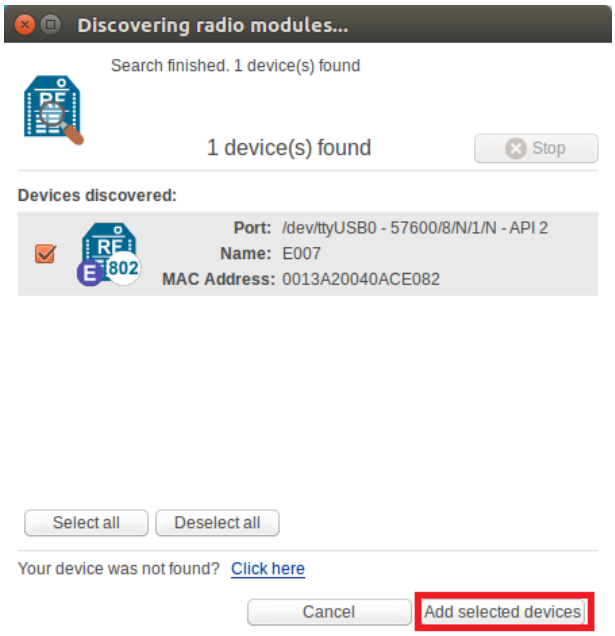


Ilustración 114: Adición de radio XBee

Una vez añadida, se verá la radio en la parte izquierda de la interfaz. Si se pulsa sobre ella, se cargarán en el lado derecho las propiedades de la misma:

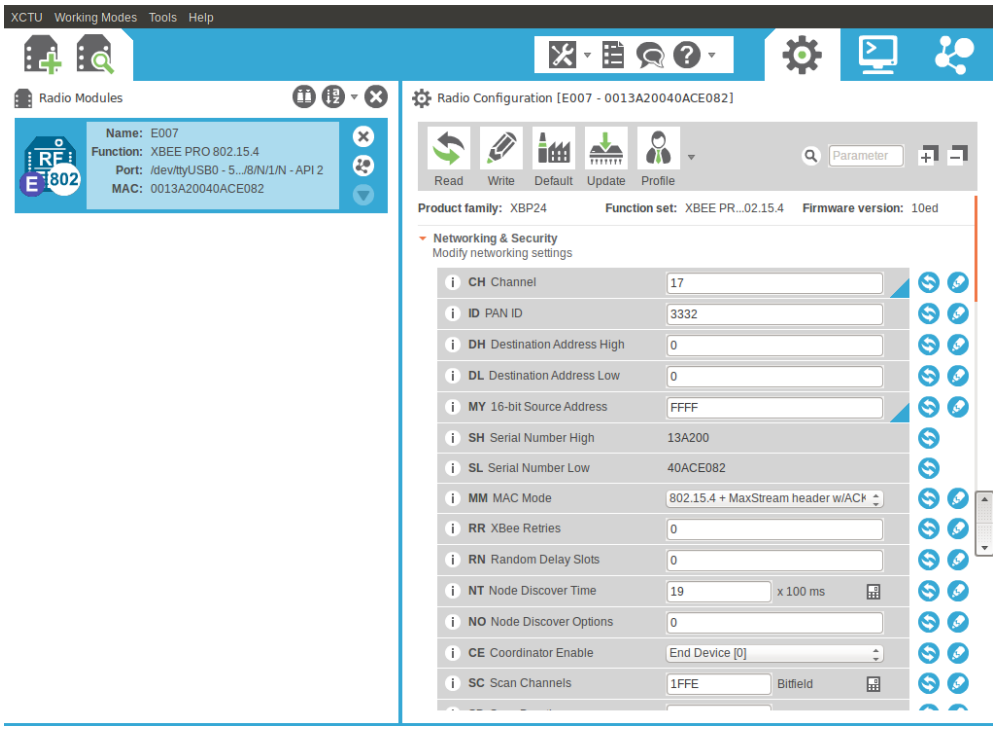


Ilustración 115: Propiedades de un módulo XBee

A partir de este momento, se puede trabajar con la radio conectada.

9.2. ACTUALIZACIÓN DEL FIRMWARE DE LAS RADIOS PARA USAR DIGIMESH

El modelo de radio XBee utilizado en este trabajo es el XBP24-DMWIT-250, que inicialmente trabaja con el protocolo 802.15.4. Digi ha puesto a disposición del usuario un firmware para poder utilizar DigiMesh, por lo que se aprovechará esta circunstancia para reutilizar dichos módulos.

Para que los módulos de radio utilicen DigiMesh, es necesario actualizar el firmware, hecho que se realizará con el programa XCTU.

Una vez se haya realizado el apartado Detección de radios en XCTU, sería conveniente apuntar los siguientes datos para un futuro, por si es necesario volver a los valores anteriores:

Product family	XBP24
Function set	XBEE PRO 802.15.4
Firmware versión	10ed

Seguidamente, se realiza una copia de seguridad de la configuración de la radio. Esta acción se realiza en el panel superior, en la opción Profile → Save configuration profile:

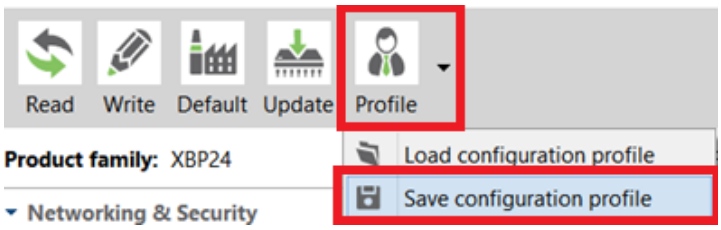


Ilustración 116: Copia de seguridad de la configuración de la radio XBee

Ahora se procede a actualizar el firmware, pulsando sobe el botón Update:

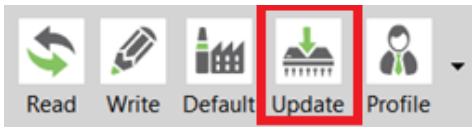


Ilustración 117: Actualización del firmware DigiMesh

Aparecerá una nueva ventana donde se deberá elegir el firmware que se desea instalar. En este caso, deberemos elegir el producto XBP24-DM y otra serie de valores, como se muestra en la siguiente ilustración:

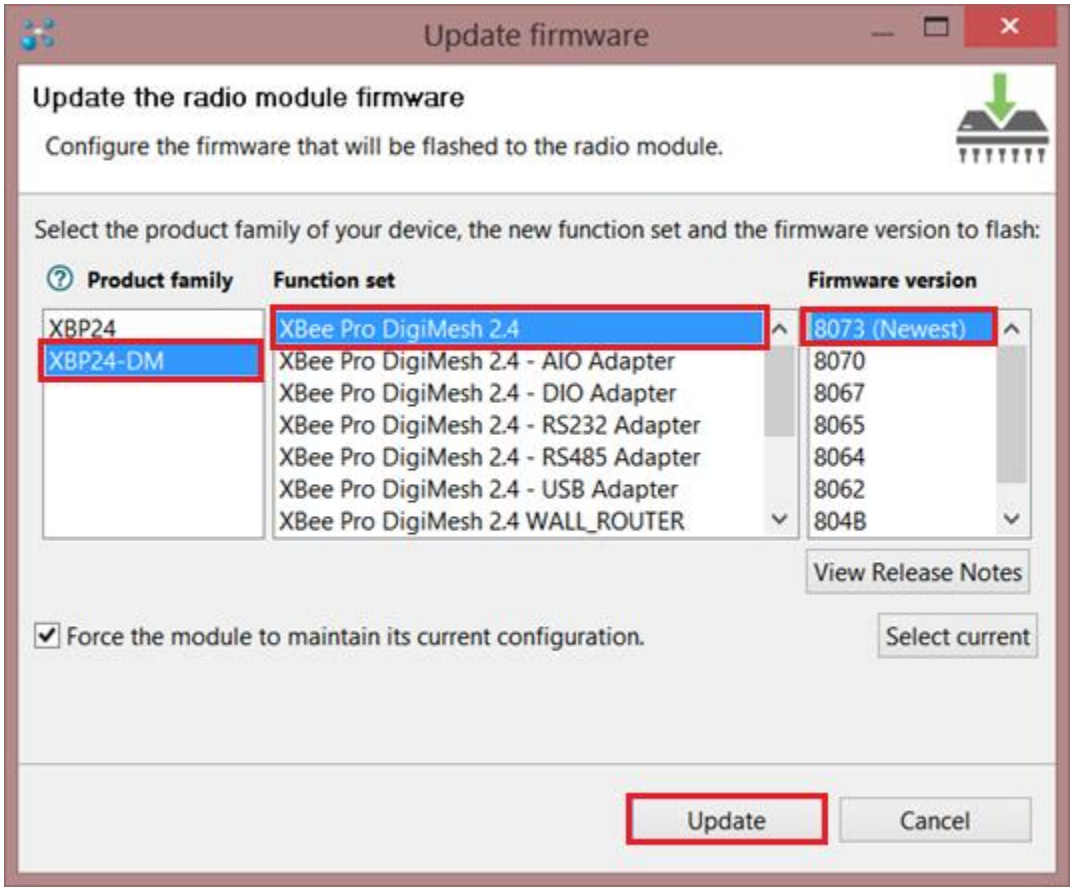


Ilustración 118: Selección del firmware DigiMesh

El proceso puede tardar unos minutos. Al terminar la instalación del firmware aparecerá un mensaje indicando que está todo correcto:

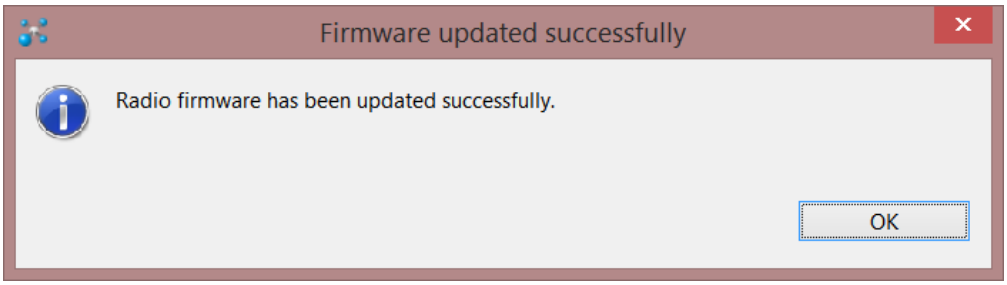


Ilustración 119: Confirmación de instalación del firmware DigiMesh

Para comprobar que el firmware está realmente instalado, se vuelven a cargar los datos de configuración del módulo con el botón Read y debería mostrar los siguientes valores en la parte inferior de la botonera:

Product family: XBP24-DM **Function set:** XBee Pro DigiMesh 2.4 **Firmware version:** 8073

9.3. CONFIGURACIÓN DE LAS RADIOS XBEE PARA UTILIZAR DIGIMESH

La configuración básica para que las radios formen una misma red, se basa en los valores de los parámetros CH e ID. Con esto y las opciones por defecto, DigiMesh considerará que todas las radios que tengan estos valores idénticos, estarán en la misma red.

Para poder usar las radios con la librería libxbee3, es necesario tener la radio XBee en modo Api2, que permite un uso más avanzado pero a la vez más complejo. Esto se indica en la radio con el parámetro AP, cuyo valor debe ser 2.

Otro parámetro a tener en cuenta es el A0, que deberá estar a 0 para enviar datos de manera normal y se transmitan paquetes del tipo `0x10` y se reciban del tipo `0x90`.

Particularmente, cada radio deberá tener su propio identificador, indicado en el parámetro NI.

A continuación se muestra la configuración de uno de los módulos de radio, para tomarlo como referencia. La única diferencia entre los distintos módulos (incluyendo la radio BASE) será el parámetro NI, mencionado anteriormente.

```
<?xml version="1.0" encoding="UTF-8"?>
<data>
  <profile>
    <description_file>xbp24-dm_8073.xml</description_file>
    <settings>
      <setting command="CH">C</setting>
      <setting command="ID">7FFF</setting>
      <setting command="MT">3</setting>
      <setting command="PL">4</setting>
      <setting command="RR">A</setting>
      <setting command="CA">0</setting>
      <setting command="CE">0</setting>
      <setting command="BH">0</setting>
      <setting command="NH">7</setting>
      <setting command="MR">1</setting>
      <setting command="NN">3</setting>
      <setting command="DH">0</setting>
      <setting command="DL">FFFF</setting>
      <setting command="NI">E007</setting>
      <setting command="NT">82</setting>
      <setting command="NO">0</setting>
      <setting command="CI">11</setting>
      <setting command="EE">0</setting>
      <setting command="KY"></setting>
      <setting command="BD">3</setting>
      <setting command="NB">0</setting>
      <setting command="RO">3</setting>
      <setting command="FT">BE</setting>
      <setting command="AP">2</setting>
      <setting command="AO">0</setting>
      <setting command="D0">1</setting>
      <setting command="D1">0</setting>
      <setting command="D2">0</setting>
      <setting command="D3">0</setting>
```

```
      <setting command="D4">0</setting>
      <setting command="D5">1</setting>
      <setting command="D6">0</setting>
      <setting command="D7">1</setting>
      <setting command="D8">1</setting>
      <setting command="D9">1</setting>
      <setting command="P0">1</setting>
      <setting command="P1">0</setting>
      <setting command="P2">0</setting>
      <setting command="PR">1FFF</setting>
      <setting command="M0">0</setting>
      <setting command="M1">0</setting>
      <setting command="LT">0</setting>
      <setting command="RP">28</setting>
      <setting command="IC">0</setting>
      <setting command="IF">1</setting>
      <setting command="IR">0</setting>
      <setting command="SM">0</setting>
      <setting command="SO">2</setting>
      <setting command="SN">1</setting>
      <setting command="SP">C8</setting>
      <setting command="ST">7D0</setting>
      <setting command="WH">0</setting>
      <setting command="CC">2B</setting>
      <setting command="CT">64</setting>
      <setting command="GT">3E8</setting>
      <setting command="DD">50000</setting>
    </settings>
  </profile>
</data>
```

Configuración 1: Configuración por defecto de los módulos XBee

9.4. PROGRAMA DE PRUEBAS EN C++ (SUPER_XBEE)

Este programa prueba las radios XBee que ya contienen un firmware de DigiMesh. Se ha creado expresamente como una fase inicial de familiarización con la libxbee3, los modos de conexión, la configuración, etc.

Las pruebas son muy sencillas y consisten en:

- Envío de paquetes explícitamente a una radio:

  En este modo será necesario introducir la MAC en el código y volver a compilar.

- Envío de paquetes broadcast:

  En este modo, el mensaje broadcast se envía en bucle hasta que se pare la ejecución del programa.

- Envío de comandos locales a la radio conectada por cable al computador:

  En este modo, se pueden enviar comandos a la radio para consultar o escribir valores en sus registros.

Para que el programa funcione en alguno de estos tres modos, habrá que dejar sin comentar la macro correspondiente y comentar el resto y después compilar. Como ejemplo, si se quiere usar el programa en modo BROADCAST, se hará lo siguiente:

```
//#define EXPLICIT
//#define REMOTE_AT_COMMAND
#define BROADCAST
```

9.5. METEO

meteo es un software que se creó específicamente para monitorizar el sistema de sensores de viento y que sirvió de apoyo en la elaboración del proyecto Dunas. Está escrito en C++ y utiliza la librería libxbee3. Se compone de un único archivo, llamado `meteo.cpp`.

9.5.1. CAMBIOS EN EL CÓDIGO

Es importante mencionar que ha habido que realizar cambios para que pueda funcionar con las radios que usan el firmware de DigiMesh. Los cambios realizados en el código son mínimos, pero a la vez importantes para una correcta conexión con el módulo de radio. A continuación se describen los detalles de los cambios realizados:

a. Para configurar el módulo con el firmware de DigiMesh, hay que indicar que el tipo es "`xbee2`":

```
// XBee initialization: begin
if((ret = xbee_setup(&(g_ctx.xbee),
                     xbee2",
                     usb_explorer_port.c_str(),
                     SERIAL_BAUD_RATE)) != XBEE_ENONE)
{
  cout<<"[ERROR] "<<ret<<" ("<<xbee_errorToStr(ret)<<")"<<endl;
  return ret;
}
```

b. Para comunicarse en el modo correcto, habrá que cambiar, cuando se vaya a crear la conexión, el parámetro correspondiente a "`Data`":

```
if ((ret = xbee_conNew(g_ctx.xbee,
                       &(g_ctx.con),
                       "Data",
                       &address)) != XBEE_ENONE) {
  xbee_log(g_ctx.xbee,
           -1,
           "xbee_conNew() returned: %d (%s)",
           ret,
           xbee_errorToStr(ret));

  return ret;
}
```

9.5.2. COMPILACIÓN

Una vez realizados los cambios necesarios, se procede a compilar el programa. Los pasos se describen a continuación, para ejecutar bajo Linux:

1. Instalar la librería `GTK`:

```
$ apt install libgtk2.0-dev
```

2. Clonar la librería libxbee3:

```
$ cd /home/usuario/Documents/
$ git clone https://github.com/attie/libxbee3
$ cd /home/usuario/Documents/libxbee3
$ make configure
```

3. Modificar el archivo config.mk que se acaba de crear, y descomentar las líneas:

```
OPTIONS+= XBEE_NO_RTSCTS
OPTIONS+= XBEE_API2
```

4. Compilar

```
$ make all
$ sudo make install
```

5. Clonar el repositorio del proyecto Dunas:

```
$ cd /home/usuario/Documents/
$ git clone http://bizet.dis.ulpgc.es/antodom/project_dunas.git
```

   Se creará la carpeta `/home/usuario/Documents/project_dunas`

6. Copiar la carpeta `project_dunas/base_station` a `/home/usuario/Documents/`

```
$ cd /home/usuario/Documents/
$ cp -R base_station ../
$ cd ../base_station
```

7. Compilar y ejecutar el programa:

```
$ make
$ ./meteo &
```

9.6. ARDUINO

El código de Arduino existente, se realizó para que los sensores de viento se comunicaran con la estación base y recogieran los datos de la veleta y el anemómetro.

El proyecto de Arduino se compone de diferentes archivos, enumerados a continuación:

- `dunas.h`
- `dunas_beep.ino`
- `dunas_median.ino`
- `dunas_memory.ino`
- `dunas_sd.ino`
- `dunas_sensors.ino`
- `dunas_v14.ino`
- `dunas_xbee.ino`

9.6.1. CAMBIOS EN EL CÓDIGO

En este proyecto, se ha tenido que modificar el código para que pueda utilizar las radios con DigiMesh.

9.6.1.1. DUNAS.H

Se ha cambiado el tipo de las variables `tx` y `txStatus`, pasan a ser de tipo `ZBTxRequest` y `ZBTxStatusResponse` respectivamente:

```
ZBTxRequest tx = ZBTxRequest(gw_addr64, payload, sizeof(payload));
ZBTxStatusResponse txStatus = ZBTxStatusResponse();
```

Se añade una variable "response" de tipo `ZBRxResponse`:

```
ZBRxResponse rx = ZBRxResponse();
```

9.6.1.2. DUNAS_XBEE.INO

9.6.1.2.1. AWAIT_BASE_ID_MSG()

Se modifica la condición de espera del paquete que envía la `MAC` de la radio `BASE`:

```
if ( (xbee.getResponse().getApiId() == ZB_RX_RESPONSE) &&
     (!strncmp((char*)rx.getData(), "BASE_ID:", 8))
   ) break;
```

9.6.1.2.2. AWAIT_SYNC_MSG()

Se modifica la condición de espera del paquete que envía el mensaje de sincronización `SYNC`:

```
if ( (xbee.getResponse().getApiId() == ZB_RX_RESPONSE) &&
     (!strncmp((char*)rx.getData(), "SYNC:", 5))
   ) break;
```

9.6.1.2.3. RCV_PACKET()

Se han añadido los casos correspondientes a los Tabla 5: Comandos AT de DigiMesh que admite la radio XBee

API frames utilizados por DigiMesh:

```
...
    else if (xbee.getResponse().getApiId() == AT_RESPONSE )
    {
      #if defined(XBEE_DEBUG)
        Serial.println(F("Got a AT_RESPONSE"));
      #endif
      xbee.getResponse().getAtCommandResponse(rx);
      return 0;
    }
    else if (xbee.getResponse().getApiId() == ZB_TX_STATUS_RESPONSE )
    {
      #if defined(XBEE_DEBUG)
        Serial.println(F("Got a ZB_TX_STATUS_RESPONSE"));
      #endif
      return 0;
    }
    else if (xbee.getResponse().getApiId() == ZB_RX_RESPONSE )
    {
      #if defined(XBEE_DEBUG)
        Serial.println(F("Got a ZB_RX_RESPONSE"));
      #endif
      xbee.getResponse().getZBRxResponse(rx);
      return 0;
    }

...
```

9.6.1.2.4. TRANSMIT()

En este caso, se controla que el API ID del paquete es el que corresponde al `0x8B` (`ZB_TX_STATUS_RESPONSE`):

```
if (xbee.getResponse().getApiId() == ZB_TX_STATUS_RESPONSE) { ... }
```

9.6.2. COMPILACIÓN

Para poder compilar, es necesario añadir varias librerías al entorno de desarrollo Arduino IDE:

- TimerOne
- XBee Arduino

Para ello se usa el gestor de librerías del entorno, situado en `Sketch → Include Library → Manage libraries`. Una vez ahí se abre una nueva ventana donde se puede buscar por algún término.

En la siguiente ilustración se muestra la búsqueda de la librería `TimerOne`:

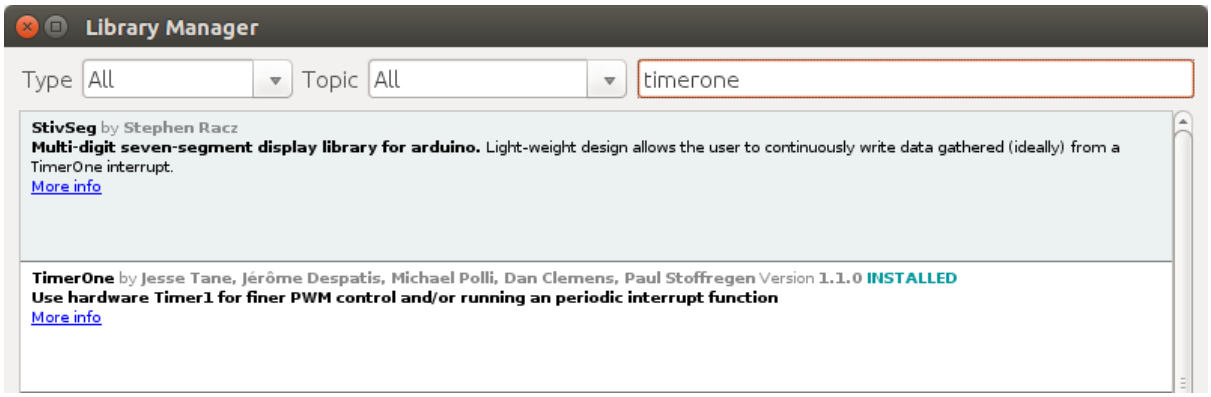


Ilustración 120: Búsqueda de la librería TimerOne para Arduino

En la siguiente ilustración se muestra la búsqueda de la librería `XBee`:

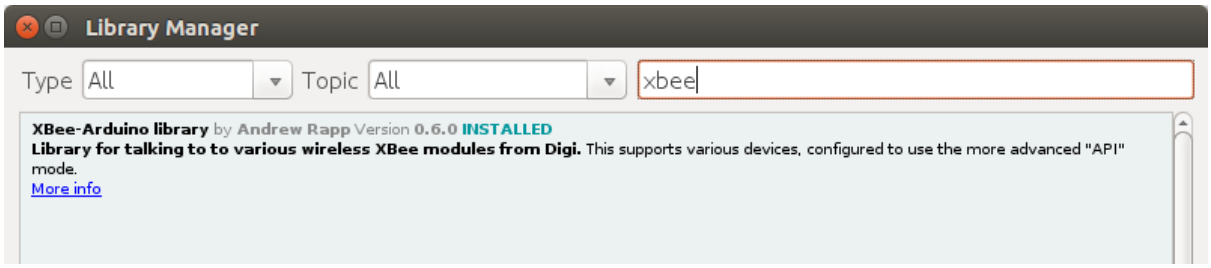


Ilustración 121: Búsqueda de la librería XBee para Arduino

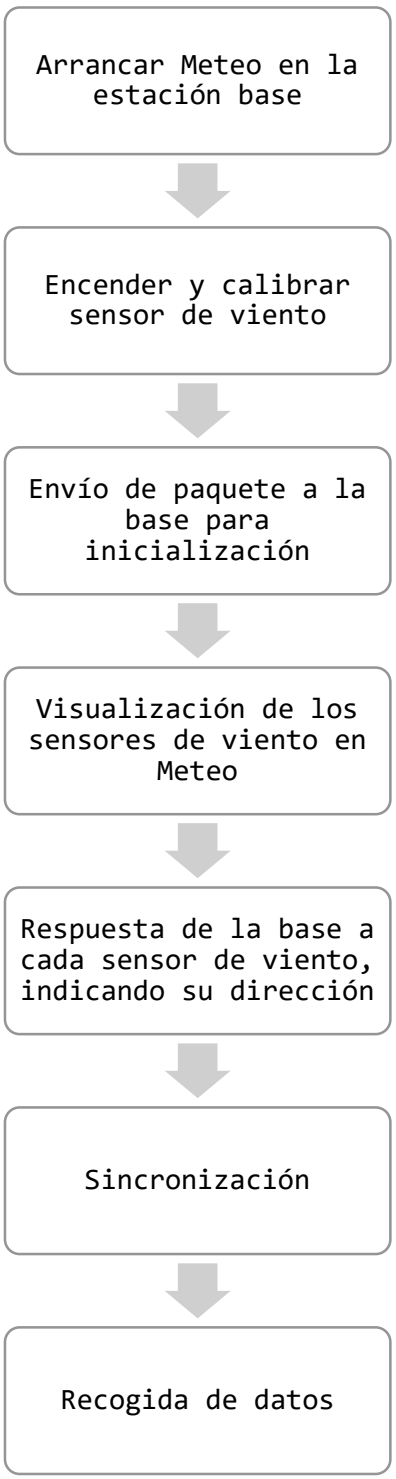
Una vez cargado el código del proyecto, sólo hace falta darle al botón `Enviar`:

Ilustración 122: Compilación del código en Arduino IDE

Es importante tener en cuenta, que cuando se vaya a compilar el código, hay que desconectar la radio XBee de la placa Arduino UNO, pues usa el puerto serie e interfiere en la comunicación con el IDE.

9.7. PUESTA EN MARCHA DEL SISTEMA

Para utilizar todo el sistema de medición de viento, es necesario seguir una serie de pasos en un orden concreto para una correcta visualización y detección de los sensores. Este proceso se describe a continuación:



1. Primero se ejecuta la aplicación meteo en el ordenador portátil, que quedará a la espera de recibir paquetes desde los sensores de viento.

2. Luego se encenderá cada sensor de viento y se calibrará. El proceso de calibración consiste en girar la veleta 360º y colocarla mirando hacia el norte, usando como referencia una brújula.

3. Una vez calibrado, el sensor de viento envía un paquete broadcast de inicalización, que se recoge en la estación base.

4. El paquete recibido en meteo permite visualizar el sensor de viento en la interfaz, donde se muestra el nombre del sensor.

5. Seguidamente, la estación base envía un mensaje explícitamente al sensor de viento del que ha recibido el paquete de inicalización, para indicarle a éste la dirección MAC de la estación base, con la que deberá comunicarse de ahora en adelante.

6. Cuando se hayan realizado los pasos 2 a 5 para todos los sensores y éstos se visualicen en meteo, se procede a sincronizar todos los dispositivos. Este proceso consiste en enviar un paquete broadcast −a todas las estaciones− con la hora actual, hasta una precisión de milisegundos.

7. Cuando los sensores reciben el paquete de sincronización, éstos se ponen inmediatamente a recoger muestras de datos cada 2 segundos, guardándolas en la SD interna y además enviándolos a la estación base, que permite visualizarlos mediante la interfaz de meteo.

10. REFERENCIAS

[1] Qué es XBee

http://xbee.cl/que-es-xbee/

[2] ZigBee

https://es.wikipedia.org/wiki/ZigBee

[3] IEEE 802.15.4

http://standards.ieee.org/findstds/standard/802.15.4-2011.html

[4] Digi

https://www.digi.com/

[5] Smart cities by Digi

https://www.digi.com/pdf/smart-cities-is-your-city-keeping-up.pdf

[6] Sistema de control de riego

http://www10.ujaen.es/sites/default/files/users/biblio/Revistas%20pdf/Riegos%20y%20drenajes/N%20182.pdf

[7] Proyecto ADI-BIDEA

https://www.bizkailab.deusto.es/wp-content/uploads/2013/06/ADI-BIDEA-Pilotos.pdf

[8] Libelium

http://www.libelium.com/

[9] Monitoring Residential Wind Turbines

https://xbee.wikispaces.com/Uses+of+Xbee

[10] Imagen WSN

https://upload.wikimedia.org/wikipedia/commons/5/5d/Appppp.png

[11] Red punto a punto

https://es.wikipedia.org/wiki/Red_punto_a_punto

[12] Wi-Fi

http://standards.ieee.org/findstds/standard/802.11-2016.html

[13] Bluetooth

https://es.wikipedia.org/wiki/Bluetooth

[14] Direct Sequence Spread Spectrum (DSSS)

https://es.wikipedia.org/wiki/Espectro_ensanchado_por_secuencia_directa

[15] Advanced Encryption Standard

https://es.wikipedia.org/wiki/Advanced_Encryption_Standard

[16] GNU

https://www.gnu.org/licenses/gpl-3.0.en.html

[17] Licencias Creative Commons

https://es.creativecommons.org/

[18] Licencia ZigBee

http://www.zigbee.org/zigbeealliance/join/

[19] Software propietario

https://es.wikipedia.org/wiki/Software_propietario

[20] End-user license agreement Digi Development Kit

http://ftp1.digi.com/support/utilities/93009461_A.txt

[21] Arduino UNO

https://www.*Arduino*.cc/en/Main/*Arduino*BoardUno

[22] Licencias *Arduino*

https://www.*Arduino*.cc/en/Main/FAQ

[23] Microcontrolador ATMEL ATmega 328P

http://www.mouser.com/ds/2/268/atmel-8271-8-bit-avr-microcontroller-atmega48a-48p-1065900.pdf

[24] Arduino Wireless SD Shield

https://store.arduino.cc/arduino-wirelss-sd-shield

[25] Sensor angular analógico US Digital MA3

https://www.usdigital.com/products/ma3

[26] Lenguaje basado en Wiring

https://www.*Arduino*.cc/en/Reference/Comparison

[27] Wiring

https://en.wikipedia.org/wiki/Wiring_(development_platform)

[28] Librerías de *Arduino*

https://www.arduino.cc/en/Reference/Libraries

[29] avr-lib

http://www.nongnu.org/avr-libc/user-manual/modules.html

[30] XCTU

https://www.digi.com/products/xbee-rf-solutions/xctu-software/xctu

[31] Licencia XCTU

http://knowledge.digi.com/articles/Knowledge_Base_Article/End-User-License-Agreement-for-XCTU