E.T.S. de Ingeniería Industrial,
Informática y de Telecomunicación

Desarrollo e implementación de una red inalámbrica de sensores de temperatura y humedad

Grado en Ingeniería
en Tecnologías de Telecomunicación

Trabajo Fin de Grado

Julen Iraceburu González

Javier Goicoechea Fernández

Pamplona, a 18 de Junio del 2014

upna Universidad Pública de Navarra
Nafarroako Unibertsitate Publikoa

Resumen.

Desarrollo e implementación de una red inalámbrica de sensores (WSN) de temperatura y humedad mediante el estudio y la comparativa de las diferentes opciones de mercado para cada uno de los elementos constituyentes de la red. Incluye la realización de montajes y experimentos yendo desde los más simples y básicos hasta la implementación final de la red y la prueba y calibración de los sensores que la conforman mediante una cámara climática.

Lista de palabras clave.

- Red
- Sensores
- Inalámbrica
- Temperatura
- Humedad
- ZigBee
- Arduino

Índice

Sección 1: Introducción y objetivos.

1.1. Motivación.

"Las tecnologías de redes inalámbricas han tenido un rápido desarrollo en los últimos años. Desde las comunicaciones punto a punto por infrarrojos a las redes inalámbricas de área personal de corto alcance (WPAN), pasando por multipuntos como Bluetooth, Wi-Fi para redes de área local, GPRS para telefonía móvil y otras como WIMAX, ANT, etc."[W6]

Es en este ámbito donde se sitúa una de las tecnologías más interesantes desde el punto de vista práctico. Esta son las redes inalámbricas de sensores (Wireless Sensor Networks), una de las opciones de futuro más prometedora debido a sus múltiples aplicaciones en diversos sectores (industria, medio ambiente, seguridad, etc.). Tanto es así que fabricantes como Microsoft, IBM, Texas Instruments o Intel ya desarrollan importantes líneas de investigación orientadas a esta tecnología.

Teniendo en cuenta que se trata de un campo en desarrollo con un futuro prometedor en el que cuanto antes se empiece a trabajar mejor, y que es un proyecto que permite trabajar y repasar muchos de los conceptos vistos a lo largo del grado he decidido desarrollar mi trabajo fin de grado en torno a las WSN.

1.2. Introducción.

"Las redes inalámbricas de sensores se basan en pequeños dispositivos (nodos) que son capaces de obtener información del entorno, procesarla localmente, y enviarla de forma inalámbrica hasta un nodo central coordinador." [W6]

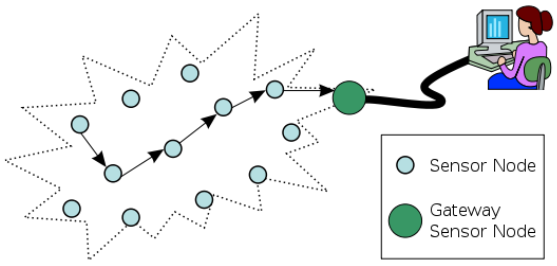


Figura 1.1. Estructura de una red inalámbrica de sensores. [I1]

Estas redes prometen cambiar la forma de obtener información. Se prevé que estarán formadas por miles de sensores diminutos y que no estarán cerradas, es decir, no se centrarán en su aplicación, sino que estarán conectadas a internet.

A parte del patrón de flujo de datos anteriormente descrito en el cual el sensor envía información a un punto coordinador, otra característica principal de las WSN son las restricciones energéticas. Uno de los grandes desafíos de este tipo de redes radica en la

alimentación de los nodos. Estos requieren bajo consumo y ciclos de trabajo cortos, y será de estos factores de los que dependa en gran medida la fiabilidad de la red.

"Cada uno de los nodos o dispositivos se denomina mota, con el objetivo de indicar dos ideas: su tamaño pequeño y el hecho de que pueden estar situadas en cualquier lugar" [W1]. Los elementos de una mota son entonces: un microcontrolador que procesa los datos provenientes de los sensores y los entrega al transceptor; una memoria externa para el almacenamiento de dichos datos; un sensor encargado de generar la señal eléctrica ante un cambio del entorno; una batería como soporte de alimentación y un transceptor que envía y recibe datos de otras motas

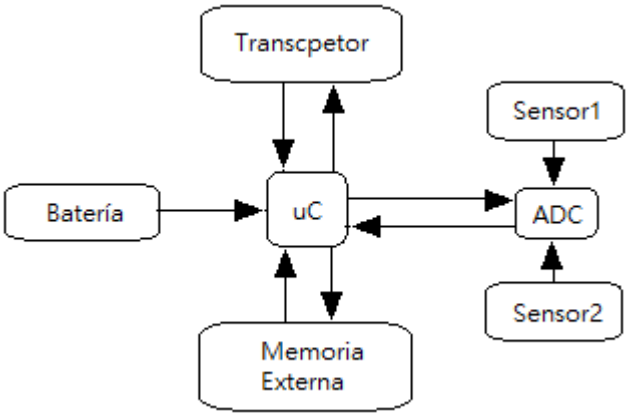


Figura 1.2. Esquema interno de una mota en una WSN.

1.3. Objetivos.

El objetivo del proyecto es entonces el desarrollo de una red inalámbrica de sensores ZigBee mediante el módulo Xbee de la casa Digi. Dicha red tendrá una topología en estrella y estará compuesta por dos sensores que medirán variables del entorno tales como temperatura y/o humedad. Estos sensores irán conectados con un MCU encargado de procesar dicha información, transformarla en datos legibles y transmitirlos al módulo Xbee para su posterior envío al nodo coordinador. Además, como es evidente, al ser una red inalámbrica, cada uno de los nodos sensores estará acompañado de una pequeña batería que permita aislarlo.

Una vez que se hayan recibido los datos en el nodo coordinador mediante la correspondiente antena, dichos datos se transmitirán a un PC que será capaz de representarlos, bien mediante un software ya existente o bien mediante la creación de un pequeño programa que permita esta adquisición.

Quedará así formada una red que permita medir variables del entorno en puntos distintos aislados y representar gráficamente los datos por separado en un PC lejano.

En este punto será interesante un análisis del consumo de las diferentes tecnologías que se puedan emplear para la implementación final puesto que la eficiencia energética es el talón de Aquiles en cualquier red inalámbrica de sensores.

Marcado este objetivo, será necesario dividir el problema en pequeños problemas, más aun teniendo en cuenta que gran parte del hardware y software utilizado en el proyecto no ha sido visto durante el grado.

Así pues, para cada elemento de nuestro sistema se realizarán pequeñas pruebas, analizando pros y contras de la solución elegida con el fin de elegir la óptima.

Sección 2: Estado del arte. Estudio de las diferentes opciones.

El objetivo de esta sección es establecer una relación de ventajas e inconvenientes de las diferentes tecnologías con las que se trabajará durante el proyecto, con el fin de poder establecer cuáles de dichas tecnologías son óptimas para nuestro objetivo y cuáles han de ser desechadas. Para ello se presentarán y estudiarán las diferentes opciones que existen en el mercado para cada uno de los elementos que conforman la red final.

2.1. Hardware. Componentes del sensor inalámbrico.

Como se comentó anteriormente un nodo de una WSN está compuesto por un MCU, una batería, un transceptor y el sensor propiamente dicho.

Figura 2.1. Partes diferenciadas de un sensor inalámbrico con un panel solar. [I2]

2.1.1. Sensor. Sensores de temperatura y humedad.

2.1.1.1. DHT11.

El DHT11 es un sensor que mide temperatura y humedad relativa. Se caracteriza por tener la señal digital calibrada por lo que asegura una alta fiabilidad y estabilidad a largo plazo. Además contiene un microcontrolador de 8 bits integrado ofreciendo excelente calidad, respuesta rápida y gran relación precio-efectividad.

Está compuesto por dos sensores resistivos (humedad y NTC). Puede medir humedad relativa en un rango de entre 20% y 95%, y temperatura entre 0ºC y 50ºC.

El protocolo de comunicación es a través de un único hilo (1-wire), por lo que su integración en proyectos es rápida y sencilla. Es pequeño, presenta un bajo consumo y la señal alcanza hasta 20 metros.

Su principal problema es que tan sólo proporciona medidas enteras, pues tiene una resolución de 1ºC para las temperaturas y del 1% para la humedad relativa.

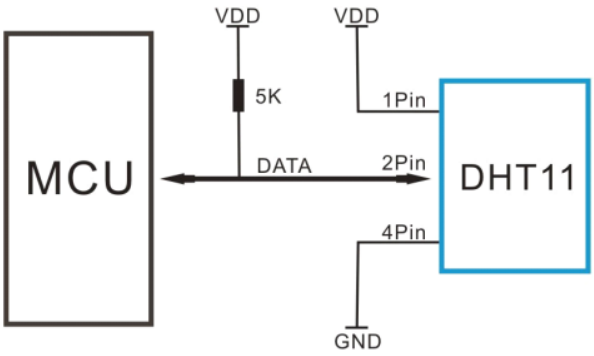


Figura 2.2. Aplicación típica del sensor DHT11. [I3]

La configuración típica requiere de una resistencia de pull-up de 5k para distancias de medida menores de 20 metros.

2.1.1.2. SHT15.

SHT1x (SHT10, SHT11 y SHT15) es la familia de sensores de temperatura y humedad de Sensirion. Los sensores integran el propio sensor más de un procesado de señal que proporciona una salida digital totalmente calibrada. Consta de un sensor capacitivo para medir la humedad mientras que la temperatura se mide por un sensor "band-gap".

Figura 2.3. Sensor SHT1x. [I4]

El empleo de tecnología CMOSens® garantiza excelente fiabilidad y estabilidad a largo plazo. Además, el sensor se acopla a un conversor analógico digital de 14 bits y a un circuito de interfaz serie. Esto se traduce en una calidad superior de la señal, un rápido tiempo de respuesta y falta de sensibilidad ante perturbaciones externas (EMC).

Su pequeño tamaño y bajo consumo hacen de él una elección perfecta para la mayor parte de aplicaciones.

Si bien es un sensor con unas altas prestaciones, posee un inconveniente, y es que no es compatible con el bus I2C presente en gran parte de los microcontroladores actuales. En lugar de ello posee una serie de comandos (códigos de 8 bits) que permiten obtener la información de temperatura y humedad a partir de los pines de reloj y datos. El envío de estos códigos es complejo y requiere de diferentes fases de interacción entre el

microcontrolador y el sensor incluyendo espera de confirmaciones (ACK) y envío de checksum (CRC).

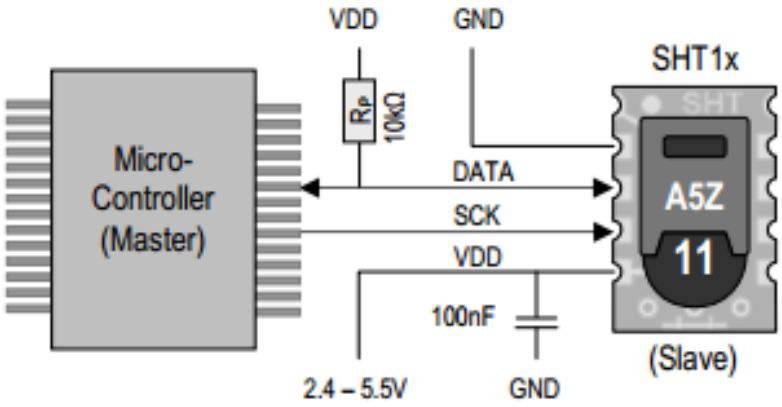


Figura 2.4. Aplicación típica del sensor SHT15. [I4]

2.1.1.3. SHT25.

El SHT25 es el nuevo sensor de temperatura y humedad de Sensirion que promete establecer nuevos estándares en términos de tamaño e inteligencia.

Con un chip CMOSens® completamente rediseñado, un reelaborado sensor de humedad capacitivo y un sensor de temperatura mejorado, ha conseguido superar a la anterior generación (SHT1x).

Figura 2.5. Sensor SHT25. [I5]

Además de los sensores de humedad y de temperatura, el integrado contiene un amplificador, un conversor A/D, memoria OTP y una unidad de procesado digital.

La gran ventaja de este chip con respecto con el SHT15 es que es compatible con I2C, evitando el trabajo de indicar al microcontrolador cómo y cuándo va a recibir los datos mediante el envío de comandos.

2.1.1.4. Elección del sensor.

Parece claro que el sensor DHT11 no ofrece muy buenas prestaciones (de ahí su bajo precio) y aunque la exactitud con la que se reproducen los niveles de temperatura y humedad no es crítica de cara a una implementación final, es deseable elegir el mejor sensor mientras no se salga del presupuesto.

Entre el SHT15 y el SHT25, la elección ideal parece ser el segundo puesto que, como se ha explicado, es compatible con I2C y son de similar precio. Sin embargo existe una limitación que hace que el SHT15 sea el elegido y es que este es el único que se ha podido encontrar ya montado en una placa comercial (breakout board) listo para comprar. El SHT25 no viene acoplado en ninguna placa lo que supone un problema porque la soldadura de este chip es especialmente complicada tal y como se indica en su hoja de características.

Figura 2.6. SHT15 montado en una placa comercial incluyendo el condensador de desacoplo y las resistencias de pull-up y pull-down. [I6]

Así pues se querrá emplear finalmente dos sensores SHT15 en los dos nodos sensores de la red WSN. No obstante para mayor diversidad en el resultado final, y teniendo en cuenta que uno de los objetivos primarios del proyecto es el didáctico, en el otro nodo se utilizará un sensor DHT11.

2.1.2. Microcontrolador.

2.1.2.1. AVR VS PIC.

Existen decenas de empresas fabricantes de microcontroladores (Intel, Motorola, TI, Microchip, Cypress, Atmel, etc.), pero entre todas ellas destacan dos familias de microcontroladores: la familia AVR de Atmel y la familia PIC de Microchip. Ambas son populares por su elevada calidad-precio y es responsabilidad del diseñador elegir entre ambos (u otros) en función de su nivel de integración, su arquitectura, la disponibilidad de recursos o su lenguaje de programación. En [W2] se presenta una comparativa muy interesante de los diferentes aspectos de ambas familias:

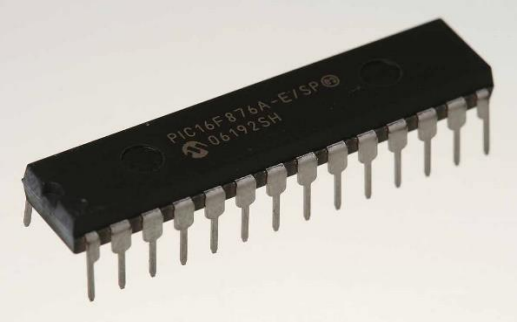


Figura 2.7. PIC16F876A. [I7]

Tanto PIC como AVR son microcontroladores de 8 bits, que cuentan con un CPU RISC y memoria FLASH para almacenar firmware. Además los dos cuentan con periféricos como puertos digitales, conversores A/D, PWM, entre otros.

Figura 2.8. ATMEGA8. [I8]

Visto así se puede pensar que ambos son iguales, y de hecho es así desde un punto de vista de estructura general, sin embargo, es en el ámbito que los rodea donde se encuentran sus diferencias: IDE (entorno de desarrollo), lenguaje de programación, reloj interno, interfaces de programación, alimentación, potencia o precio.

- **IDE y lenguaje de programación**

    - PIC tiene el pequeño inconveniente de que ha de programarse en ASM (lenguaje de bajo nivel), y aunque es cierto que existen IDEs y compiladores para C, C++ y Basic, estos tienen su costo y no son completamente gratis.
    - AVR, por su parte, cuenta con compiladores de C, C++ y Basic completamente gratuitos y descargables desde la propia página web del fabricante. Además existen plataformas de desarrollo basadas en microcontroladores AVR como el ARDUINO, que es un chip con un bootloader que simplifica la programación aún más.

- **Interfaces de programación**

- PIC dispone de un puerto para la programación, pero esta se realiza a alto voltaje (>5Vdc), por lo que se necesitan circuitos externos para la reconversión, complicando el circuito programador.
- Los AVR cuentan con el puerto ISP (In System Programmer) por el que se realiza el grabado en el chip. Ahí hay diferentes opciones de hardware (puerto paralelo, programador serial, programador USB, etc.). Destaca entonces la simpleza y sencillez de estos programadores.

- **Consumo de energía**

  - Los AVR frente a los PIC destacan por su reducido consumo, tanto en voltaje como en corriente, lo que los hacen ventajosos en aplicaciones con baterías.

- **Reloj interno**

  - La familia AVR incluye un bloque oscilador con un circuito RC, mientras que los PIC cuentan con un cristal externo que hace las veces de oscilador.

- **Precio**

  - Los microcontroladores AVR son un poco más baratos en comparación con un PIC, por ejemplo, el ATMEGA8 de ATMEL cuesta en torno a 4$, mientras que uno similar de Microchip, el 16F876 se puede adquirir por unos 7$.

Con todo esto, es hora de tomar una decisión. Si bien es cierto que un AVR es excepcional desde un punto de vista de aprendizaje y PIC es una gran elección para aplicaciones profesionales, los microcontroladores AVR cada vez están siendo más empleados en aplicaciones profesionales de ingeniería. Además cada año la bibliografía y los foros en los que se habla sobre AVR crece exponencialmente puesto que es una tecnología fácil de emplear para usuarios iniciados en electrónica y programación. Si a esto le sumamos hechos como que AVR posee herramientas libres de pago, que su entorno de desarrollo es simple y completo o que su consumo de energía es mínimo (este punto es de especial interés para una aplicación inalámbrica como la que se desarrollará), la elección final será la de un microcontrolador de la familia AVR.

Se empleará la plataforma de desarrollo anteriormente mencionada ARDUINO, en concreto la placa comercial Arduino Uno de la que se hablará a continuación.

2.1.2.2. Arduino Uno

En la página oficial de Arduino [W3] se incluye una descripción de sus diferentes productos. Arduino Uno es una placa basada en el microcontrolador ATMEGA328. Posee 14 pines de entradas/salidas digitales (6 de los cuáles su pueden emplear como salidas PWM y dos de ellas se pueden emplear para la comunicación serie) y 6 entradas analógicas, un resonador de 16MHz, conexión USB, conector de alimentación, un puerto ICSP, y un botón de reset.

Es posible consultar todas las características técnicas de esta placa y de otras en la propia web del fabricante cuyo enlace se deja en la bibliografía.

Arduino dispone de un entorno de desarrollo propio y está pensada para fomentar y facilitar el uso de la electrónica para el público en general, desde escuelas hasta aficionados a la electrónica.

Al tratarse de un microcontrolador con un lenguaje basado en C / C++, existen funciones de entrada/salida analógica y digital, funciones aritméticas, manejo de interrupciones, comunicaciones por el puerto serie, etc.

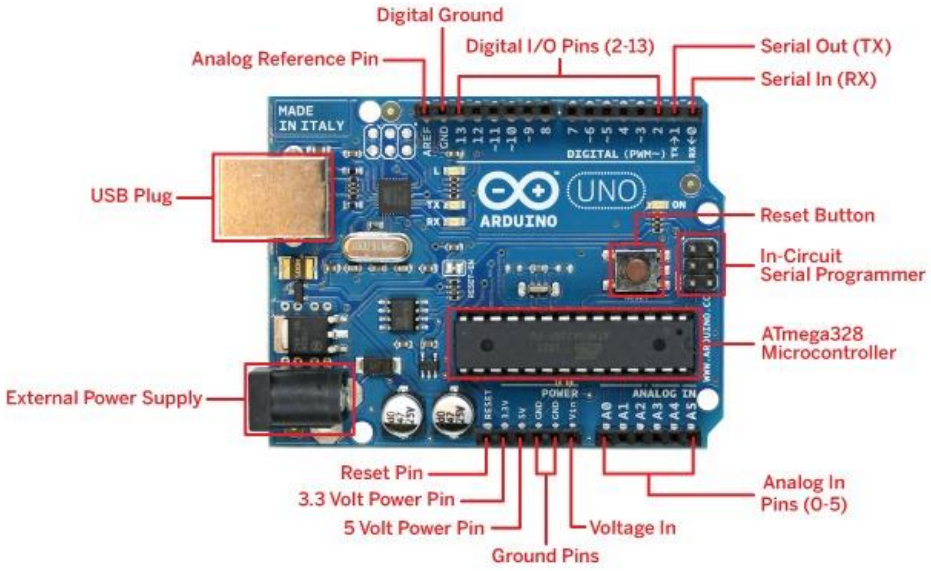


Figura 2.9. Disposición de los elementos que conforman la placa Arduino Uno. [I9]

Las aplicaciones de la Arduino son incontables y de hecho el presente proyecto no explota, ni de lejos, todas las posibilidades de Arduino.

2.1.3. Módulo inalámbrico.

2.1.3.1. Tecnologías inalámbricas.

2.1.3.1.1. Bluetooth.

"Bluetooth es un estándar de tecnología inalámbrica para el intercambio de datos a cortas distancias (empleando un enlace por radiofrecuencia en la banda ISM de 2,4 GHz) desde dispositivos fijos y móviles construyendo redes de área personal (PAN)."[W4]

Utiliza canales RF (79 canales) de f = 2402 + k MHz siendo k = 0.78. El espacio entre los mismos es de 1 MHz más unas bandas de guarda.

La distancia nominal del enlace está comprendida entre 10cm y 10m, pero aumentando la potencia de transmisión se puede llegar a 100m.

La modulación empleada por Bluetooth es GFSK (Gaussian Frequency Shift Keying) con un producto ancho de banda por tiempo BT = 0.5 y un índice de modulación entre 0.28 y 0.35.

Estos dispositivos se clasifican en clases: clase 1, clase 2 y clase 3, en referencia a su potencia de transmisión.

Clase	Potencia máxima permitida (mW)	Potencia máxima permitida (dBm)	Alcance (aproximado)
Clase 1	100 mW	20 dBm	~30 metros
Clase 2	2.5 mW	4 dBm	~10-5 metros
Clase 3	1 mW	0 dBm	~1 metro

Figura 2.10. Potencia y alcance de bluetooth según la clase. [I10]

La tecnología Bluetooth ha evolucionado enormemente en los últimos años, y son ya varias las generaciones de bluetooth que han existido, suponiendo mejoras en ancho de banda, alcance o seguridad. Así, los dispositivos también se pueden clasificar por su ancho de banda.

Versión	Ancho de banda
Versión 1.2	1 Mbit/s
Versión 2.0 + EDR	3 Mbit/s
Versión 3.0 + HS	24 Mbit/s
Versión 4.0	24 Mbit/s

Figura 2.11. Ancho de banda en las diferentes generaciones Bluetooth. [I10]

La generación Bluetooth 4.0 incluye un subconjunto denominado Bluetooth de baja energía (Bluetooth Low Energy o BLE) con una pila de protocolos completamente nueva para desarrollar rápidamente enlaces sencillos.

2.1.3.1.2. Bluetooth Low Energy.

"También denominada como Bluetooth LE, Bluetooth ULP y Bluetooth Smart está diseñada para reducir drásticamente el consumo de energía manteniendo un rango de

comunicación similar a Bluetooth. Esto hace pensar que se pueda estar ante una nueva tecnología emergente dentro de las redes de sensores inalámbricas."[W4]

BLE es una tecnología todavía en desarrollo pero hoy en día es compatible con casi todos los sistemas operativos (iOS, Android, Windows Phone, Blackberry, OS X, Windows 8), y se prevé que en 2018 más del 90% de los smartphones incluirán BLE.

2.1.3.1.3. ZigBee.

"ZigBee surge de la necesidad de desarrollar una tecnología inalámbrica de no muy alta transferencia de datos. Así, en 1998, un conjunto de empresas, conocidas como la ZigBee Alliance se juntaron para crear un estándar de comunicaciones que complementara a Wi-fi y Bluetooth. Surge así ZigBee, publicado por el IEEE en Mayo de 2003." [W8]

ZigBee se establece como principal objetivo el de comunicar aplicaciones que requieren una comunicación segura, con baja tasa de envío y bajo consumo. Se basa en dispositivos inalámbricos operando en la banda ISM para usos industriales, científicos y médicos (868 MHz, 915 MHz y 2.4 GHz) con una modulación en espectro ensanchado por secuencia directa (DSSS) también conocida como acceso múltiple por división de código en secuencia directa (DS-CDMA). En el rango de frecuencias de 2.4 GHz (banda más extendida) se definen 16 canales con un ancho de banda de 5 MHz.

IEEE 802.15.4 y ZigBee definen diferentes tipos de dispositivos:

- Tipos de dispositivos (según ZigBee):
  - **Coordinador**: Controla el ruteado y administra la red. Existe uno por red.
  - **Router**: Interconecta diferentes nodos mediante direccionamiento.
  - **Dispositivo final**: Elemento pasivo que responde ante peticiones de otros dispositivos. Se pasa la mayor parte del tiempo dormido.
- Tipos de dispositivos (según IEEE 802.15.4):
  - **FFD** (Full Function Device)
  - **RFD** (Reduced Function Device)

ZigBee está diseñado para ser una solución que permita crear redes e interconectar dispositivos remotos. De esta forma existen diversas topologías de red que se pueden formar con los dispositivos previamente mencionados:

- **Punto a punto**
  - Topología más sencilla.
  - Uno de los dispositivos es el coordinador y el otro puede ser un router o un dispositivo final.
- **Estrella**
  - Todos los dispositivos de la red se pueden comunicar con el coordinador pero no entre sí.
- **Malla**
  - Se trata de una comunicación punto a punto pero existiendo restricciones en la intercomunicación de dispositivos.

    - Cualquier dispositivo puede comunicarse con otro.
    - El coordinador se encarga de la gestión de caminos.
- **Árbol**
    - El coordinador establece la red inicial.
    - Los routers forman ramas y retransmiten los mensajes.
    - Los dispositivos finales son las hojas del árbol.

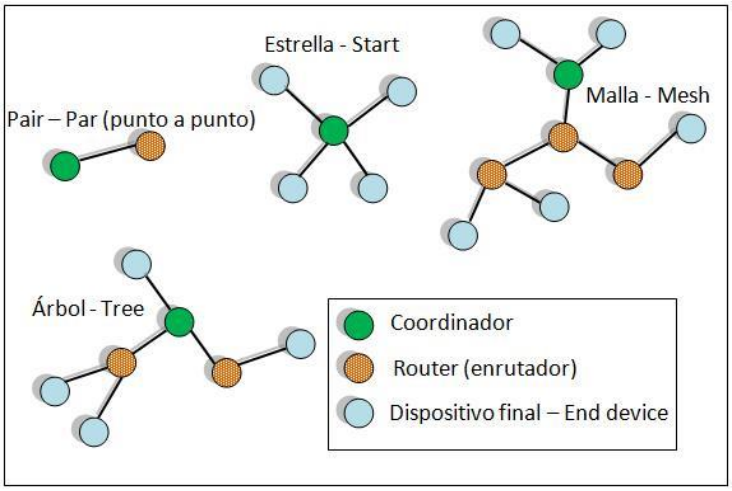


Figura 2.12. Topologías ZigBee. [I11]

En su concepción ZigBee se ideó para que tuviese una estructura por capas, lo que simplifica el problema y permite dividir el problema en otros más pequeños. Además una estructura por capas es más flexible ante cambios.

ZigBee sigue el modelo OSI (Interconexión de Sistemas Abiertos) pero reduce las 7 capas de la pila de protocolos a únicamente 4. El estándar 802.15.4 define las dos primeras capas, física y de enlace de datos, mientras que ZigBee define la capa de red y la capa de aplicación.

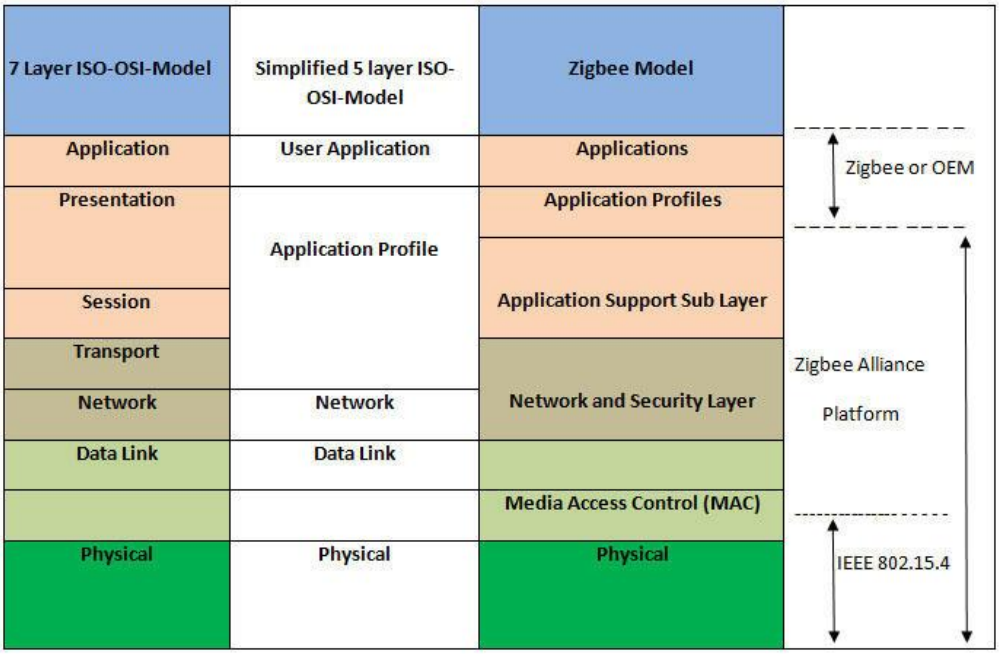


Figura 2.13. Pila de protocolos ZigBee. [I12]

- **Capa Física**: Define las funciones con la capa de enlace de datos, los niveles de potencia del transmisor y la sensibilidad del receptor, transfiriendo los datos por medio de un entero de 8 bits. Sus funciones son: canales, numeración de canales, detección de energía, medición de portadora, indicador de calidad del enlace, evaluación de canal libre, cliente, servidor entre capas e interfaz/área de datos capa física, enlace de datos.
- **Capa de enlace de datos:** Provee la interfaz entre la capa física y la capa de red. Maneja los servicios y los datos. Sus funciones son: operación de la red usando balizas, espaciado entre tramas, CSMA-CA, nodo oculto y nodo expuesto, formato de la trama.
- **Capa de red ZigBee**: Otorga las funcionalidades para el armado y manejo de redes y una interfaz simple para las aplicaciones de usuarios. Se encarga de: establece la red, configuración de dispositivos, routeado, y seguridad.
- **Capa de aplicación**: Nivel más alto de la especificación. Es la interfaz efectiva entre el nodo y el usuario. En él se ubican la mayor parte de los componentes definidos por la especificación como la funcionalidad requerida para los dispositivos. El estándar ZigBee ofrece la opción de emplear perfiles en el desarrollo de aplicaciones. Un perfil de aplicación permite la interoperabilidad entre productos de diferentes fabricantes como si fuesen del mismo. La ZigBee Alliance define varios identificadores de perfil, un número de 16 bits de la capa de aplicación que define el perfil.

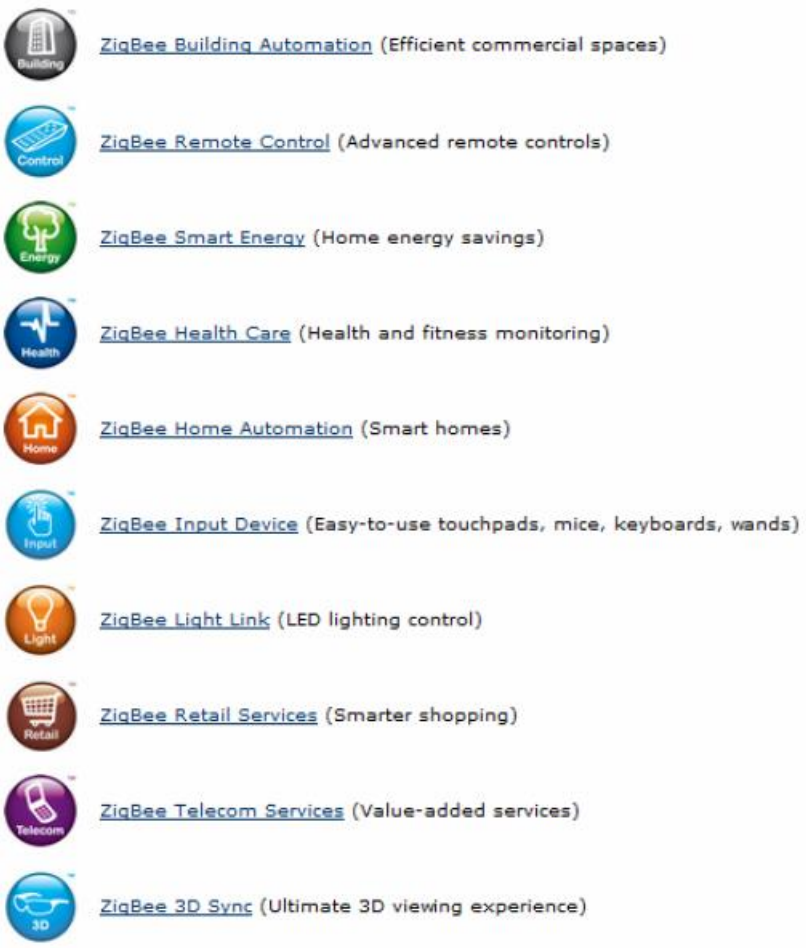


Figura 2.14. Perfiles definidos por la ZigBee Alliance. [I13]

2.1.3.1.4. ANT.

ANT es una tecnología propietaria de libre acceso para redes inalámbricas de sensores multicast que ofrece una pila de protocolos de comunicación inalámbrica para dispositivos que operan en la banda de 2,4 GHz (aplicaciones industriales, científicas y médicas) permitiendo a estos comunicarse estableciendo reglas estándar de coexistencia, representación de datos, señalización, autentificación y detección de errores.

El protocolo ANT está disponible en transceptores de baja potencia de fabricantes como Nordic Semiconductor y Texas Instruments.

ANT se caracteriza por una baja carga computacional y una baja eficiencia, lo que se traduce en un consumo de energía mínimo.

Las aplicaciones para las que ANT está pensado se caracterizan por una transferencia periódica de pequeñas cantidades de información entre varios dispositivos interconectados de punto a punto o con una topología de red en estrella, en malla o en árbol.

Las aplicaciones más conocidas y difundidas de ANT se encuentran en el sector del deporte, más en concreto en fitness y ciclismo. En estas, los transceptores se integran en elementos como relojes o cinturones que miden parámetros tales como velocidad y distancia permitiendo al usuario una monitorización de su rendimiento.

Figura 2.15. Reloj que emplea la tecnología ANT para medir parámetros durante el entrenamiento. [I14]

2.1.3.1.5. Bluetooth vs ZigBee vs ANT.

Vistas las características técnicas de estas tres tecnologías inalámbricas, es la hora de escoger una de ellas para poder emplear en nuestro proyecto. Cada una de ellas tiene sus puntos fuertes y de hecho cada una acapara diferentes sectores del mercado.

Bluetooth está pensado para aplicaciones como los teléfonos móviles o la informática casera y tiene un consumo eléctrico relativamente alto en comparación con ZigBee o ANT. De hecho, en términos exactos, Bluetooth tiene un consumo de 40mA transmitiendo y 0.2mA en reposo, frente a los 30mA transmitiendo y 3µA en reposo de ZigBee. Es por esta razón que bluetooth no es una elección óptima a la hora de montar nuestro proyecto, pues este requiere de un bajo consumo puesto que va a depender de una pequeña batería.

Existe la posibilidad aquí de emplear BLE, sin embargo, debido al bajo conocimiento de este tipo de tecnología y a que aún se encuentra en desarrollo, se decide desechar Bluetooth por completo para la implementación final. Sin embargo, es importante notar que antes de llegar a la solución final se realizarán pruebas con módulos JY-MCU con tecnología Bluetooth para ver el funcionamiento de esta y familiarizarse con la misma.

Así, la elección se centra en ZigBee y ANT. ZigBee parece una tecnología ideal puesto que presenta un bajo consumo, a una tasa de envío de datos reducida (aunque más que suficiente para nuestros propósitos), con un alcance de más de 100 metros y lo más importante, es una tecnología creada para la interconexión de pequeños dispositivos con batería en una red de múltiples nodos, que es exactamente como se habían definido las redes inalámbricas de sensores.

ANT es una tecnología más desconocida que ZigBee y su mercado se reduce a dispositivos deportivos como los mencionados con anterioridad. A pesar de que el alcance es menor en ANT que en ZigBee, ANT presenta una ventaja muy importante, la de un consumo de energía mínimo.

Sin embargo, en [W7] podemos ver que para un escenario cíclico en el que los sensores inalámbricos envían datos cada cierto intervalo de tiempo y el resto de tiempo permanecen en modo dormido ("sleep"), ANT proporciona un consumo mayor que ZigBee. Esto es debido a que en modo "sleep" un dispositivo ZigBee no consume prácticamente energía. Claro que el experimento propuesto no se puede generalizar, pero lo que es cierto es que nuestra aplicación va a presentar un escenario similar en el que los nodos sensores van a permanecer dormidos largos períodos de tiempo.

Con esto en mente y con el añadido de que ZigBee es una tecnología estándar y no propietaria como ANT, permitiendo mayor libertad y compatibilidad, se decide optar por realizar la aplicación final en base a la tecnología ZigBee.

2.1.3.2. Elección del módulo inalámbrico.

Una vez determinado que la tecnología a emplear será ZigBee, es hora de tomar otra importante decisión, la elección del módulo ZigBee apropiado. La elección debe tener en cuenta aspectos tales como el precio, la sencillez de la programación o la extensión en el mercado.

Existe una amplia variedad de dispositivos ZigBee (transceptor + micro) en el mercado que proporcionan un listado de fabricantes. A continuación se resumen los más interesantes y sus características.

- **Xbee ZNet 2.5 RF Module**
  - Bajo coste y bajo consumo típico de WSN.
  - Fácil uso y entrega fiable de datos entre dispositivos.
  - Pequeño tamaño.
  - No está configurado para ningún nivel de aplicación específico.
  - Muy extendido entre el público.
  - No dispone de microcontrolador.
  - Posee un software de programación gratuito y se puede descargar de la propia web del fabricante.
  - Existe una versión superior que proporciona mayor potencia y cobertura a cambio de mayor tamaño y mayor consumo.

Figura 2.16. Módulo Xbee ZNet 2.5 RF Series2 de Digi. [I15]

- **EasyBee ZigBee Transceiver Module.**
  - Contiene toda la circuitería de RF, una antena integrada, el regulador y el enchufe de unión.
  - Posee un SPI de 4 cables que hacen de interfaz con el microcontrolador.
  - La capa PHY incluye una impedancia al igual que la antena.
  - La capa MAC incluye la generación CRC-16, la evaluación de canal clara, la detección de señal de energía, la seguridad, encriptación y autentificación.
  - No precisa de una capa de red. Si se precisa ZigBee, se pueden encontrar pilas ZigBee libres.

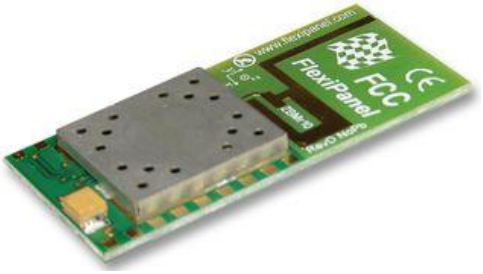


Figura 2.17. Módulo EasyBee ZigBee Transceiver. [I16]

- **ZB-21 ZigBee OEM Module.**
  - Permite gran flexibilidad en las conexiones.
  - Incluye transceptor y microcontrolador en un mismo PCB.
  - Consumo cuidado con detalle: 23µA de consumo en estado dormido.
  - Uno de los módulos con mejores prestaciones del mercado.

Figura 2.18. Módulo ZB-21 ZigBee OEM. [I17]

- **Cirronet ZMN2400.**
    - Provee flexibilidad y versatilidad para servir aplicaciones que van desde el reemplazo de cable en redes de sensores.
    - Es fácil de integrar y aporta una comunicación inalámbrica robusta incluyendo la operación en una topología de tipo malla.
    - Incluye el potente perfil de aplicación CSM de Cirronet, el cual elimina la necesidad del cliente de programar el firmware.

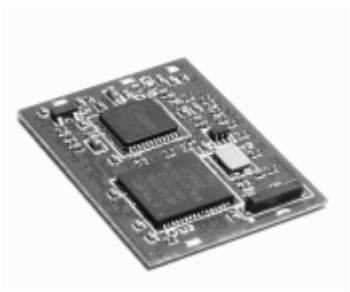


Figura 2.19. Módulo Cirronet ZMN2400. [I18]

Los presentados anteriormente son algunos de los módulos compatibles con ZigBee más importantes del mercado. La elección final ha sido la de realizar el proyecto con las antenas Xbee de Digi, las cuales a pesar de no contar con microcontrolador, son mucho más económicas ya que no necesitan de un kit de desarrollo para programarlas, y cuentan con una gran cantidad de bibliografía y foros de Internet acerca de ellas.

2.1.3.3. Módulos Xbee de Digi.

2.1.3.3.1. Operación.

Este apartado se apoya en lo descrito en [L2] sobre el modo de trabajo de los módulos Xbee de Digi.

Existen dos tipos de módulos Xbee; los módulos Xbee Serie2 y los módulos Xbee Serie1. La principal diferencia existente entre ambos es que los de la Serie2 sí permiten hacer redes mesh. Además existen los mencionados Xbee Pro, que permiten mayor alcance y potencia de señal. En lo sucesivo describiremos el funcionamiento y configuración de todos estos módulos (pues es muy similar), sin embargo, nosotros emplearemos módulos Xbee Serie2.

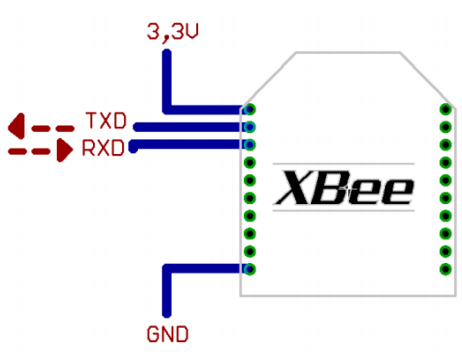


Figura 2.20. Conexiones mínimas requeridas para el Xbee.

El módulo requiere una alimentación de 2.8 a 3.4 V, la conexión a tierra y las líneas de transmisión de datos por medio del UART (TXD y RXD) para comunicarse con un microcontrolador o un puerto serial.

Los modos de operación del módulo Xbee son los siguientes:

- **Modo recepción/transmisión.**

  Se encuentra en uno de estos modos cuando llega al módulo algún paquete RF por la antena (modo recepción) o cuando se manda información serial al buffer del pin 3 que luego será transmitida (modo transmisión).

- **Modo de bajo consumo.**

  Este modo hace posible que el módulo se encuentre en un estado de bajo consumo cuando no está en uso. La configuración de los ciclos de sueño se realiza mediante comandos. Por defecto, los ciclos de sueño están deshabilitados (módulo en estado de reposos/recepción).

  Mediante el pin de hibernación Sleep_RQ (pin 9) en estado alto, el módulo termina cualquier transmisión, recepción o procedimiento de asociación y entra en modo reposo y luego en modo sueño. En este estado no responde a comandos entrantes de ningún tipo. Sólo cuando se baje el estado lógico de Sleep_RQ el módulo saldrá del estado de sueño y podrá volver a enviar y recibir datos.

  Existe otro modo de sueño a parte del de controlado por pin, este es el modo de sueño cíclico que se basa en despertar al módulo cada cierto período para que compruebe si existen datos para enviar.

- **Modo de comando.**

  En este modo es posible configurar parámetros del módulo mediante comandos AT. Para ello se debe emplear un software como el Hyperterminal de Windows o el X-CTU de Digi.

  Se entra en modo de comando escribiendo por el terminal el string “+++”. Tras la devolución de un OK, es posible ingresar comandos mediante la sintaxis siguiente.

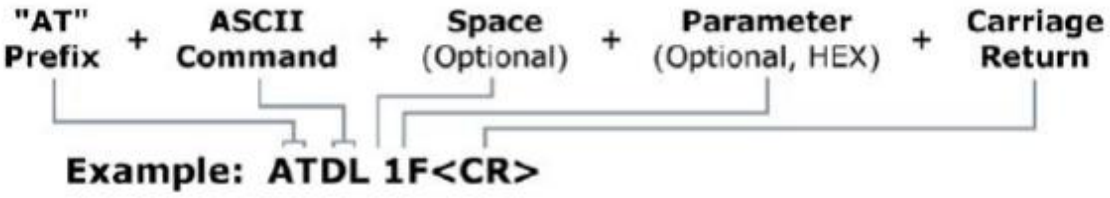


Figura 2.21. Ejemplo comando AT.

Figura 2.22. Entrada en el modo de comandos mediante el software X-CTU.

- **Modo transparente.**

  En modo transparente todo lo que ingresa por el pin 3 (DIN) se guarda en buffer de entrada y se transmite y aquello que ingresa como paquete RF se guarda en buffer de salida y se envía por el pin 2 (DOUT) inmediatamente o tras el paso de un tiempo (parámetro RO). Este modo se destina a comunicación punto a punto, donde no es necesario ningún tipo de control.

- **Modo de operación API.**

  Cuando se encuentra en este modo, el módulo empaqueta en tramas toda la información que entra y sale. Estos definen operaciones y eventos del módulo. Entre las opciones que permite la API, se definen:

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

- **Modo Idle.**

  Se dice que si el módulo no está en ninguno de los estados anteriormente mencionados se encuentra en estado IDLE.

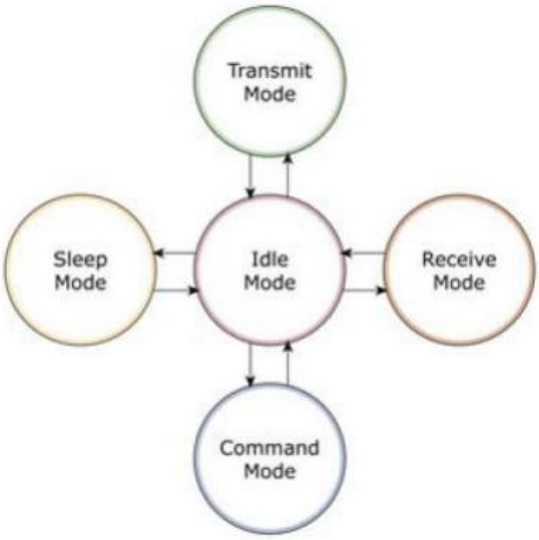


Figura 2.23. Modos de funcionamiento del módulo Xbee.

Con la descripción dada ya se puede empezar a trabajar sobre el módulo y crear las primeras comunicaciones.

2.1.3.3.2. Breakout board para el módulo Xbee.

Únicamente se trata de una placa comercial para colocar el módulo Xbee. La función que cumple es la de convertir el molesto espaciado de pines del módulo Xbee a uno estándar de 0.1 pulgadas ideal para el montaje en protobards.

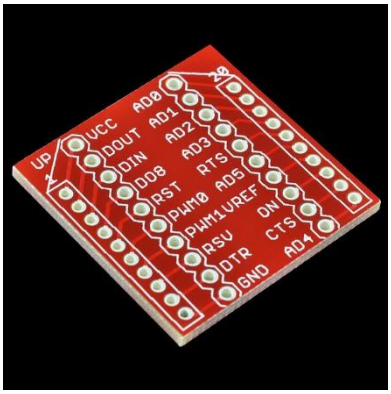


Figura 2.24. Breakout Board de Sparkfun para Xbee. [I19]

2.1.3.3.3. Placa Xbee Explorer USB.

Esta pequeña placa impresa contiene los pines de conexión para un módulo Xbee así como un controlador que sirve para programarlos y tomar sus datos. Sirve para todo tipo de módulos Xbee, tanto serie1 como serie2 y también para la versión Pro.

Está equipada con una entrada mini USB permitiendo conectar la placa a un ordenador y leer y escribir datos a partir del puerto serie.

Figura 2.25. Xbee Explorer USB. [I19]

Figura 2.26. Conexión del móudlo Xbee al Explorer USB. [I19]

En el caso que nos ocupa será el módulo Xbee que actúe como coordinador el que esté sobre el Xbee Explorer USB recibiendo los datos de los demás sensores inalámbricos y enviándoselos a un ordenador para que este pueda recogerlos, interpretarlos y mostrarlos por pantalla.

2.1.4. Batería.

A menudo, a la hora de diseñar una red inalámbrica de sensores, se invierte mucho tiempo en resolver desafíos de conectividad y en pensar en las diferentes opciones de tecnología inalámbrica existentes para emplear la mejor de todas. Sin embargo, la alimentación es un elemento del sistema tan importante como el resto. La elección de una batería adecuada puede determinar el éxito o fracaso de un proyecto de WSN.

La duración de la batería va a ser un parámetro crucial en la vida útil del sensor. La pregunta que uno se debería hacer en lugar de ¿qué tecnología inalámbrica voy a emplear? o ¿en qué rango voy a recibir datos?, es ¿cuánto va a durar la batería?

La capacidad total de una batería se mide en mili-amperios-hora (mAh) o amperios-hora (Ah). Una capacidad de 3Ah significa que la batería puede suministrar 3A durante una hora o 1A durante 3 horas. También es importante especificar cuándo una batería está vacía, esto es, cuando la tensión cae por debajo de cierto nivel necesario en el exterior. Por ejemplo en una celda de 1.5V como puede ser una pila AA, se considera vacía la batería si la tensión cae a 0.9V.

Normalmente existen dos criterios importantes a la hora de elegir una batería. Tal y como se expone en [W5]:

- El primero tiene que ver con la temperatura. Las baterías frías conservan bien la electricidad y las baterías calientes suministran bien la electricidad, por tanto es bueno conservar la batería en un lugar a baja temperatura y calentarla cuando se vaya a suministrar energía. Este hecho cobra importancia en una aplicación como la nuestra pues tratamos con sensores de temperatura. En principio no expondremos los sensores a temperaturas extremas y normalmente se tomarán datos a temperatura ambiente, por lo que se pasará por alto el diferente comportamiento de las baterías según la temperatura, sin embargo es importante no olvidarlo para futuros trabajos.
- La segunda regla se refiere a la no linealidad de las baterías. La capacidad de una batería está relacionada con la corriente que se le pide suministrar y esta no es líneas. Un alto consumo de corriente hace disminuir la capacidad de la batería. Por ejemplo, una batería podría tener una capacidad de 1000mAh para un consumo de corriente a 5mA, sin embargo, la misma batería podría tener una capacidad de 500mAh para un consumo de corriente de 200mA. Esto no es muy intuitivo pero es importante. Suponer que una batería es lineal para toda corriente es peligroso, y solo es cierto cuando los cambios de temperatura y de corriente son muy pequeños.

Además habrá que tener en cuenta aspectos como los voltajes de alimentación de los módulos Xbee (2.8 – 3.4 V) y el tamaño de la batería.

Con todo esto en mente, se puede echar un vistazo a diferentes tipos de baterías y escoger la más apropiada.

2.1.4.1. Baterías alcalinas.

Las baterías alcalinas son las más comunes en hogares y son muy buenas para una gran variedad de aplicaciones electrónicas, sobre todo porque las empresas de baterías han mejorado la química de estas.

Las pilas alcalinas son de bajo costo, ampliamente disponibles y son ideales para aplicaciones de baja corriente a temperatura ambiente, pero tienen dos defectos: son muy malas en condiciones frías y no tienden a funcionar correctamente en condiciones de alta corriente. En la siguiente figura se puede apreciar, cómo la capacidad de una pila alcalina cualquiera es excelente a bajas corrientes y temperaturas altas.

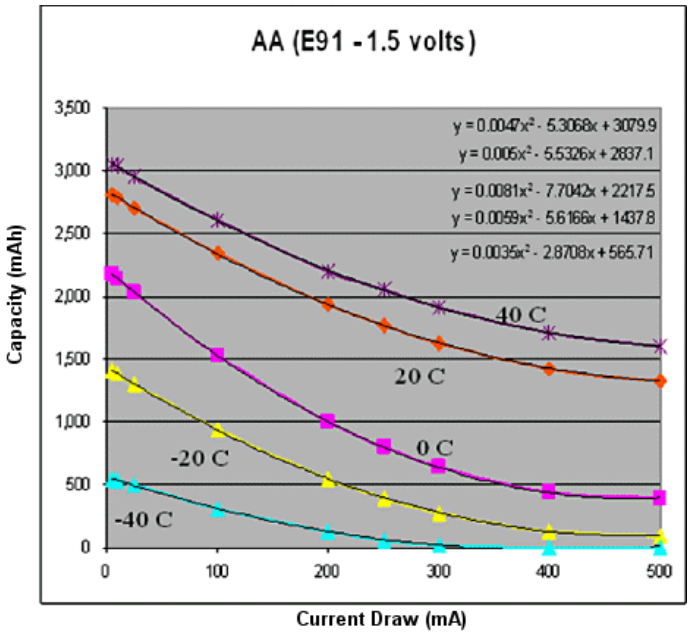


Figura 2.27. Capacidad en una pila alcalina. [I20]

2.1.4.2. Baterías de litio.

Las variedades de baterías de litio son muchas. Es complicado clasificar a todas en un mismo grupo que siga un modelo común, pero sí tienen unas características comunes. Son baterías desechables y son mejores que las alcalinas a bajas temperaturas. Se puede modelar su comportamiento en capacidad usando una función logarítmica.

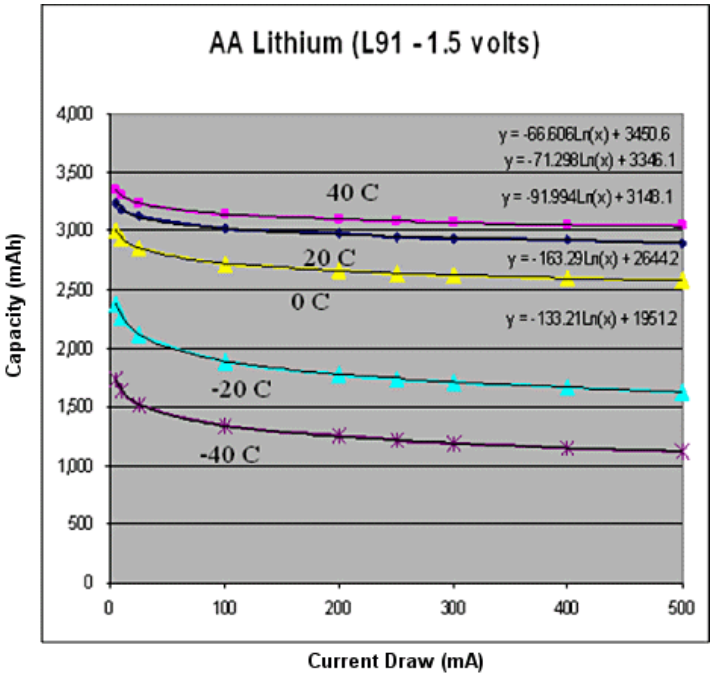


Figura 2.28. Capacidad en una pila de litio. [I20]

2.1.4.3. Baterías de níquel-metal hidruro (NiMH).

Es una de las mejores baterías recargables. A diferencia de su prima, la batería de níquel-cadmio (NiCad), ésta tiende a mantener sus características tras muchas recargas. Es la batería alcalina de las recargables y su comportamiento en capacidad sigue un modelo similar descrito por un polinomio de segundo orden, con un mejor proceder a bajas temperaturas y no disminuyendo tan rápidamente la capacidad a medida que aumenta la demanda de corriente.

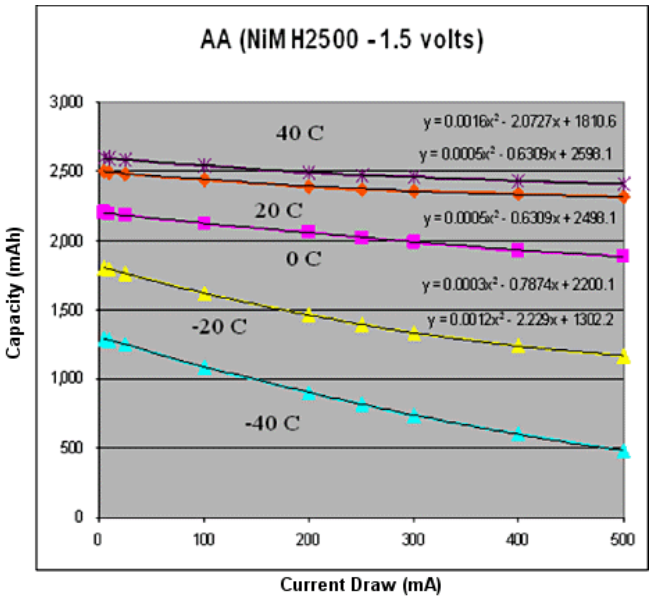


Figura 2.29. Capacidad en una pila NiMH. [I20]

2.1.4.4. Elección de la batería.

Vistas ventajas y desventajas de cada tipo de batería se puede establecer que para una aplicación como la nuestra en la cual las temperaturas no van a ser extremas, la mejor elección es una batería alcalina típica de 9V, pues ofrece una muy buena calidad precio.

Al tratarse de un sistema de prueba el tamaño de la pila no debe importar demasiado, sin embargo, de cara a un posible sistema embebido, habría que pensar en una pila alcalina de botón que se pueda integrar fácilmente.

Por último, es importante notar que las alimentaciones de nuestro sistema son de 5V (SHT15) y 3.3V (Xbee). El que empleemos una pila de 9V no va a suponer un problema pues Arduino cuenta con una entrada (Vin) de alimentación a un regulador (AMS1117) que otorga salidas de 5V y 3.3V.

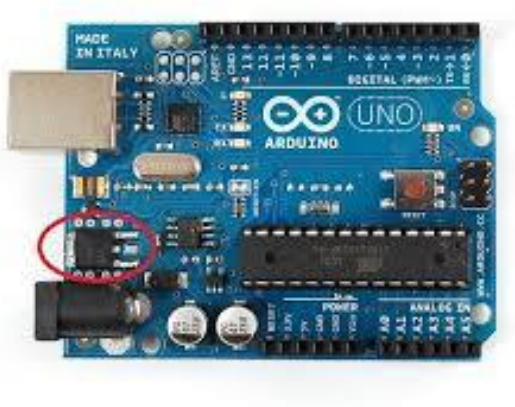


Figura 2.30. Localización del regulador en la placa Arduino Uno. [I9]

2.2. Software.

2.2.1. IDE de Arduino.

Arduino ofrece un entorno de desarrollo gratuito en el que realizar programas para que sean volcados al microcontrolador posteriormente vía USB. Arduino se programa mediante el uso de un lenguaje propio basado en el lenguaje de programación de alto nivel Processing. Sin embargo, se pueden emplear otros lenguajes y aplicaciones en Arduino, ya que emplea la transmisión serial de datos soportada por la mayoría de lenguajes. Incluso es posible emplear software intermediario para aquellos lenguajes no compatibles con el formato serie.

No se entrará a explicar el lenguaje de programación aquí (hay una buena referencia en la página web de Arduino) pues en la Sección3 se desglosan los programas empleados en nuestra aplicación.

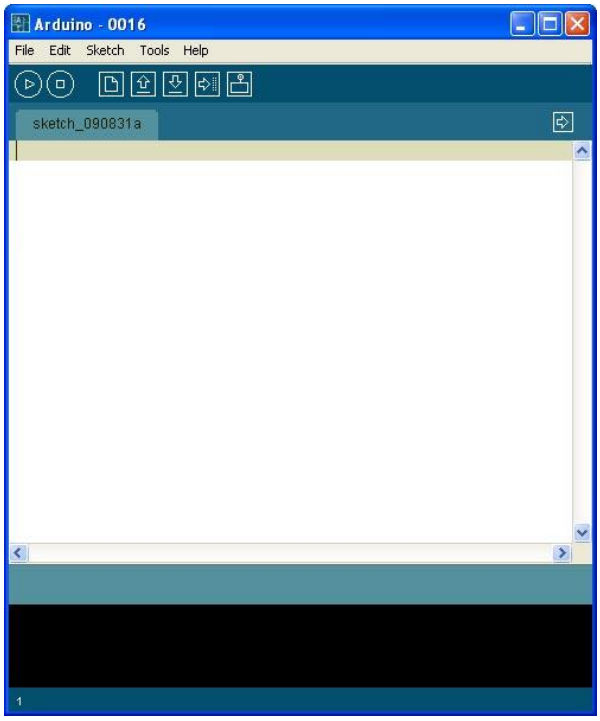


Figura 2.31. IDE Arduino.

Este entorno de desarrollo además habilita un monitor serie que permite ver los datos que se encuentran en los diferentes puertos serie del ordenador. En él simplemente hay que ajustar la velocidad (baudios) de dicho puerto.

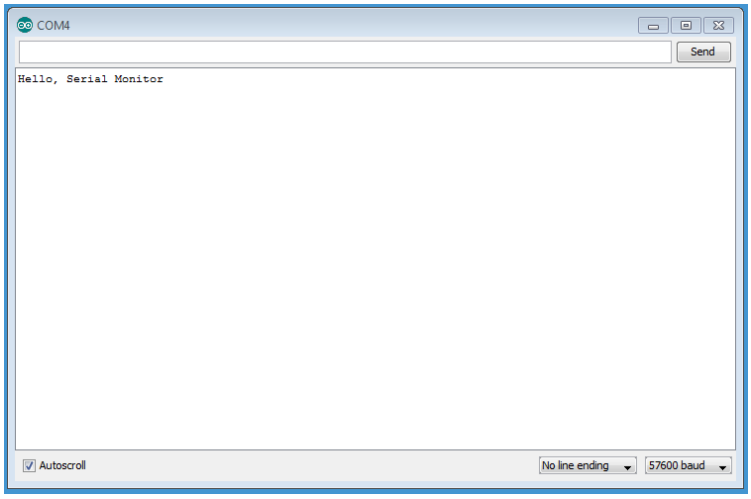


Figura 2.32. Monitor Serie de Arduino.

Con esta información, ya es posible comenzar a volcar desde un ordenador sobre la placa Arduino Uno los programas más básicos.

2.2.2. Programación de los módulos Xbee: X-CTU.

Si bien es cierto que es posible emplear el Hyperterminal de Windows para configurar un módulo Xbee, existe un programa llamado X-CTU, el cual permite realizar estas operaciones de manera más natural, fácil y rápida.

El programa cuenta con cuatro pestañas muy útiles:

- PC-Settings: Sirve para seleccionar el puerto COM serie conectado al módulo Xbee y realizar ajustes en parámetros como bitrate, control de flujo, bits de datos, bits de paridad, etc. Permite habilitar el modo API.

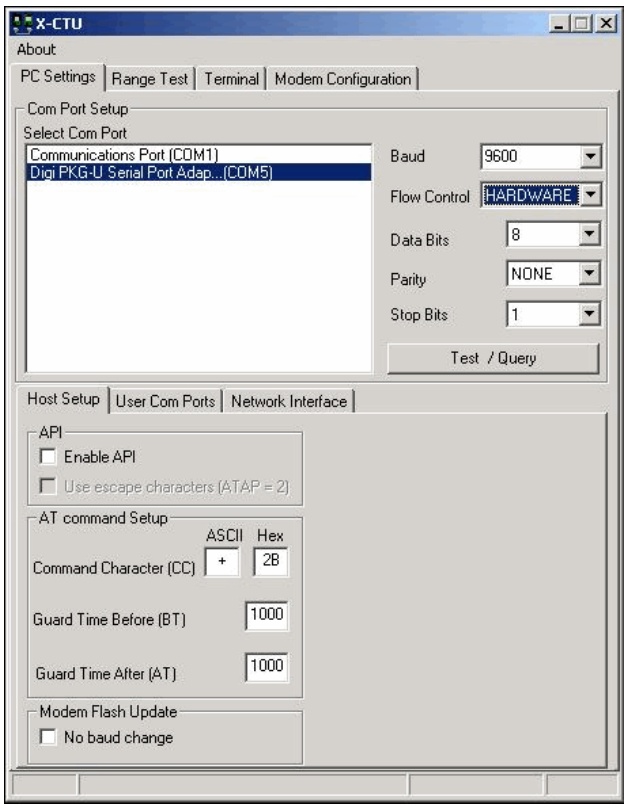


Figura 2.33. Pestaña PC-Settings de X-CTU.

- Range Test: En esta pestaña es posible enviar una cadena de datos para probar el rango de alcance de la señal.

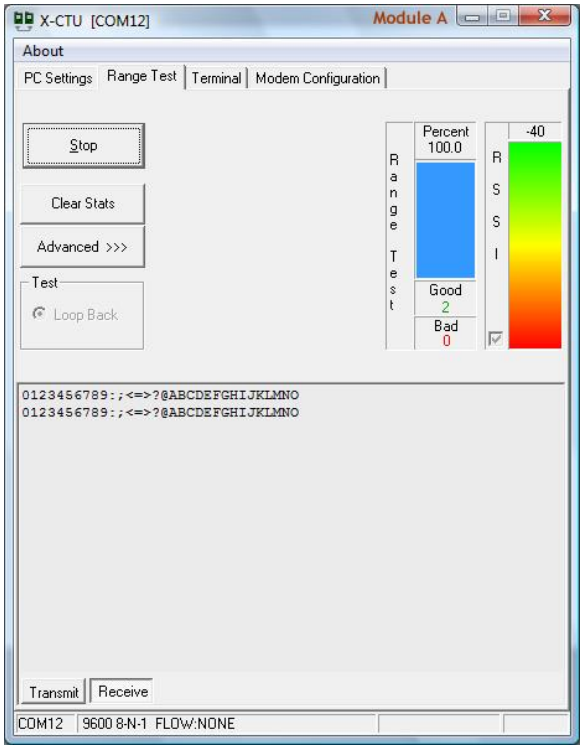


Figura 2.34. Pestaña Range Test de X-CTU.

- Terminal: Lo escrito aquí entra al módulo como si estuviera en el modo de comandos. También permite ver lo que llega al módulo.

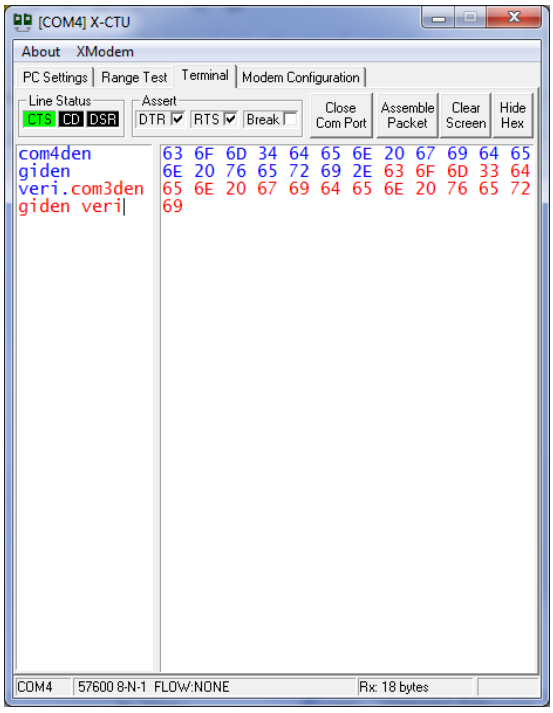


Figura 2.35. Pestaña Terminal de X-CTU.

- Modem Configuration: En esta pestaña es posible leer, guardar o cargar cierta configuración predeterminada. Se puede ver cómo está configurado el módem, cambiar un parámetro y guardarlo.

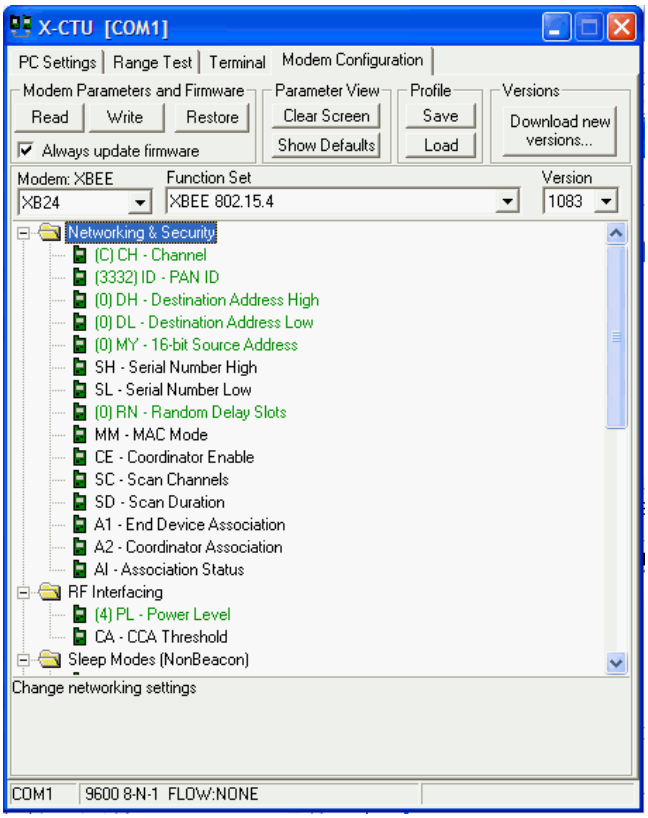


Figura 2.36. Pestaña Modem Configuration de X-CTU.

2.2.3. Interfaz de usuario.

Ya se comentó anteriormente la idea de programar una pequeña interfaz que permitiese al usuario observar en tiempo real los datos que llegasen al nodo coordinador.

Así pues para la realización de dicha interfaz se ha pensado en emplear el lenguaje de programación Processing puesto que Arduino está basada en él y tiene la ventaja de ser familiar, sin embargo, la creación de gráficos en tiempo real de Processing no es demasiado buena (lo cual es sorprendente pues se dedica a eso), y por ello la combinaremos con la herramienta Matlab, conocida de sobra y que es capaz de presentar unos gráficos más precisos y elegantes.

Tanto uno como otro lenguaje permiten comunicación con el puerto serie al que está conectado el nodo coordinador por lo que no habrá ningún problema durante la adquisición. En la sección 3 se explicará detalladamente la adquisición y se expondrán los resultados obtenidos (códigos completos en el Anexo I).

2.3. Resumen de los materiales empleados.

Con el objetivo de recordar todo lo escrito en esta sección, se expondrán aquí, a modo de resumen, los materiales que se emplearán en la implementación final.

Sensor de temperatura y humedad – SHT15

Microcontrolador – ATMEGA328 en placa Arduino Uno

Módulo inalámbrico – Xbee ZNet 2.5 RF Series2 de Digi programados mediante el software X-CTU

Batería – Pila alcalina 9V

A continuación se muestra un esquema de los nodos sensores.

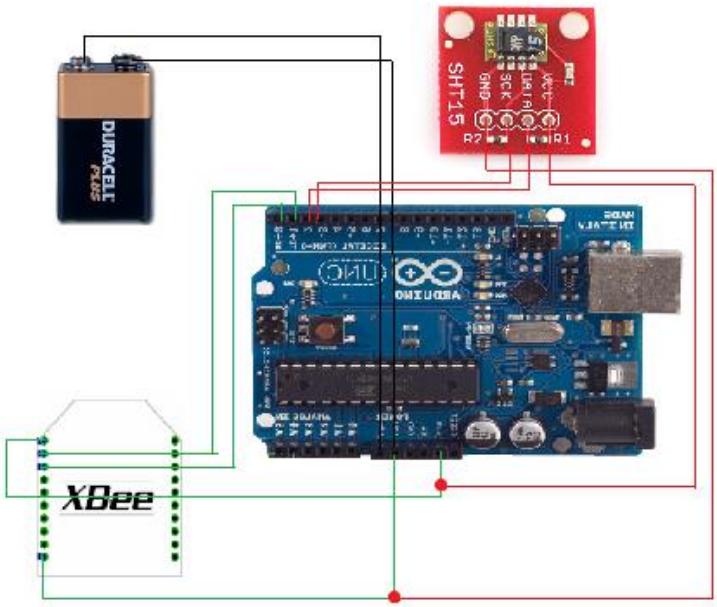


Figura 2.37. Esquema de los nodos sensores.

Se conectará vía USB otro módulo Xbee al ordenador (mediante el USB Explorer), que actuará como coordinador, terminando la red ZigBee.

Sección 3: Desarrollo experimental.

A lo largo de esta sección se describirán detalladamente una a una todas las pruebas y montajes realizados a lo largo del trabajo, desde las más básicas y puramente didácticas hasta las que forman en sí mismas la implementación final.

Para ello la sección se dividirá en diferentes subsecciones, cada una correspondiente a cada caso práctico, y en ellas se expondrán las configuraciones previas a los montajes, la explicación de los códigos pertinentes (con diagramas de bloques) y las conclusiones y utilidades de la prueba realizada.

En el Anexo I se adjuntan los códigos completos empleados en cada experimento junto a sus correspondientes diagramas de flujo.

3.1. Configuración 1: NTC + Bluetooth.

Ya se comentó anteriormente que Bluetooth no es la tecnología ideal para una aplicación como la que nos ocupa, sin embargo, a forma de prueba inicial se realizará un pequeño montaje con módulos Bluetooth. Esto servirá además para iniciarse en la programación y el entorno de desarrollo de Arduino.

Esta prueba pertenece a los inicios del trabajo y durante su desarrollo no se disponía todavía de los sensores de temperatura y humedad SHT15 y DHT11. Es por esto que se utilizará una NTC (NJ28RA0104F) a modo de sensor de temperatura.

Una NTC es un tipo de termistor, un sensor resistivo de temperatura. Su funcionamiento se basa en el cambio de resistividad que se produce en un semiconductor cuando varía la temperatura. En una NTC la resistencia disminuye cuando la temperatura aumenta. Su curva de respuesta se puede aproximar por una exponencial del tipo:

$$R_T = R_0 e^{B\left(\frac{1}{T}-\frac{1}{T_0}\right)}$$

donde R_o es la resistencia a la temperatura de referencia T_0 (K), R_T es la resistencia a la temperatura T (K) y B es la temperatura característica del material.

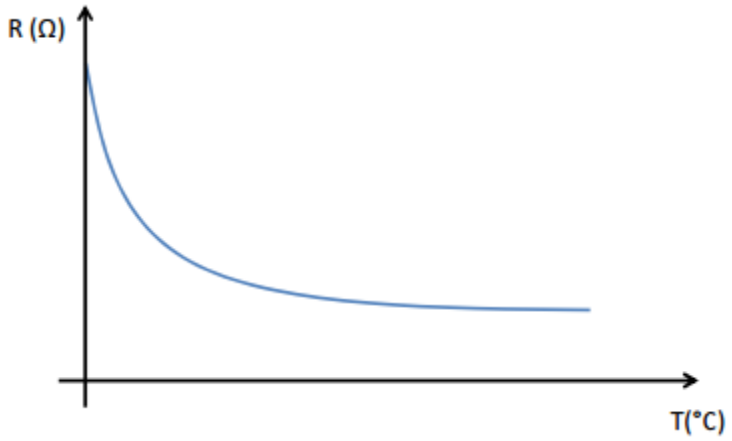


Figura 3.1. Respuesta exponencial de un termistor NTC.

Como se puede ver, presenta el inconveniente de no ser lineal, sin embargo, mediante un divisor de tensión resistivo se puede linearizar una salida en tensión.

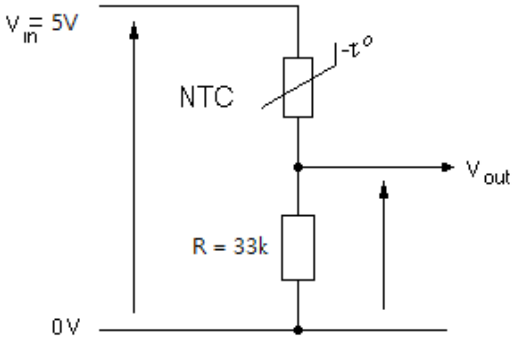


Figura 3.2. Linearización de NTC mediante divisor de tensión resistivo.

Según la hoja de características el termistor NJ28RA0104F tiene una resistencia de referencia de 100kΩ y una temperatura característica de 4380K a una temperatura de referencia de 25ºC. Con estos parámetros se tiene la siguiente relación temperatura-resistencia:

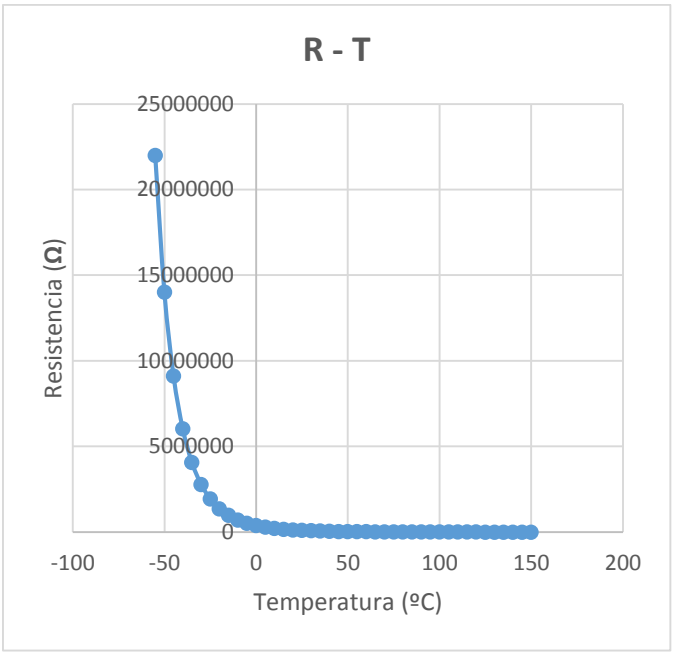


Figura 3.3. Temperatura frente a resistencia de la NTC empleada. Observar alta no linealidad.

Si se emplea el anterior divisor de tensión con una resistencia de carga de 33kΩ, y una alimentación de 5V, se obtiene una tensión de salida mucho más lineal. Además la zona más lineal se puede modificar para que se encuentre en el rango de medida sin más que variando la resistencia de carga.

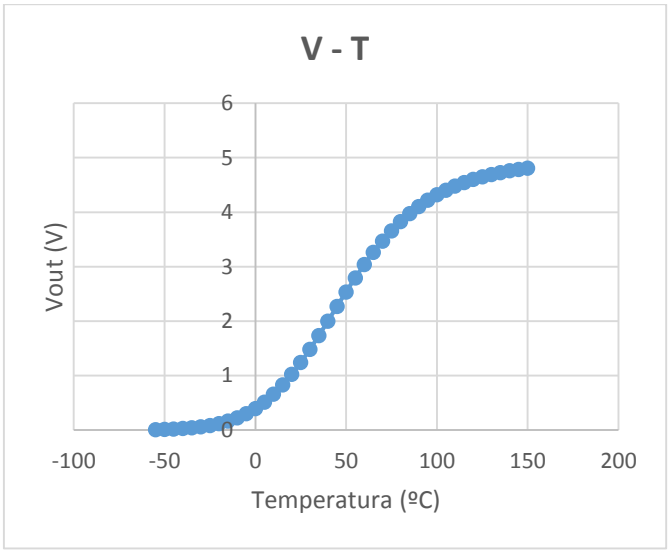


Figura 3.4. Tensión de salida frente a temperatura en NTC NJ28RA0104F.

Tras este paréntesis acerca de nuestro sensor es hora de explicar la configuración de los módulos Bluetooth para que funcionen correctamente.

Se emplearán dispositivos JY-MCU Bluetooth, son muy típicos y económicos. Su configuración se lleva a cabo mediante comandos AT (comandos Hayes). Para ello se conectan los pines de transmisión y recepción de los módulos bluetooth a los pines serial de la placa Arduino (0-RX, 1-TX) y tras alimentar, se vuelca un pequeño código Arduino sobre la placa. Existen multitud de comandos AT, sin embargo en ese código se establecen solamente los necesarios: el nombre del módulo, la velocidad del puerto y la contraseña de interconexión con otros dispositivos.

AT+NAMExxx

AT+BAUDy (y representa una velocidad en baudios del puerto serie)

AT+PINzzzz

Una vez configurados los JY-MCU y establecido un circuito que proporcione una salida lineal para cambios en temperatura ya se puede realizar el montaje. El esquema del mismo se muestra a continuación. Como se puede observar la salida lineal del divisor resistivo se conecta al pin analógico A0 e internamente (ver código) se copia ese valor en los terminales serie de Arduino. De ahí los recoge el dispositivo Bluetooth y los envía a posibles receptores.

Mediante la aplicación para móviles BlueTerm podemos conectarnos al JY-MCU con el pin de interconexión establecido y ver por pantalla los datos que este está enviando.

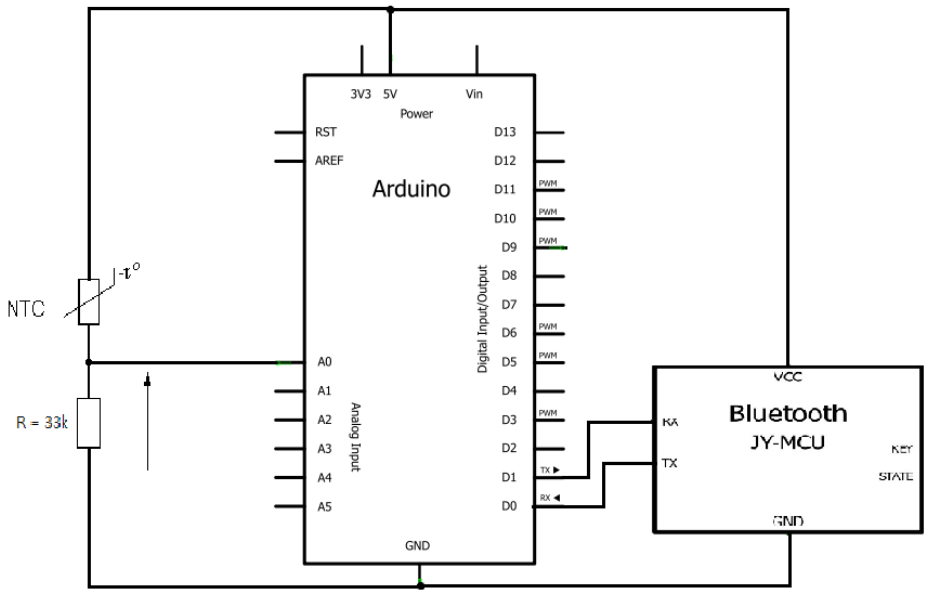


Figura 3.5. Esquemático configuración 1.

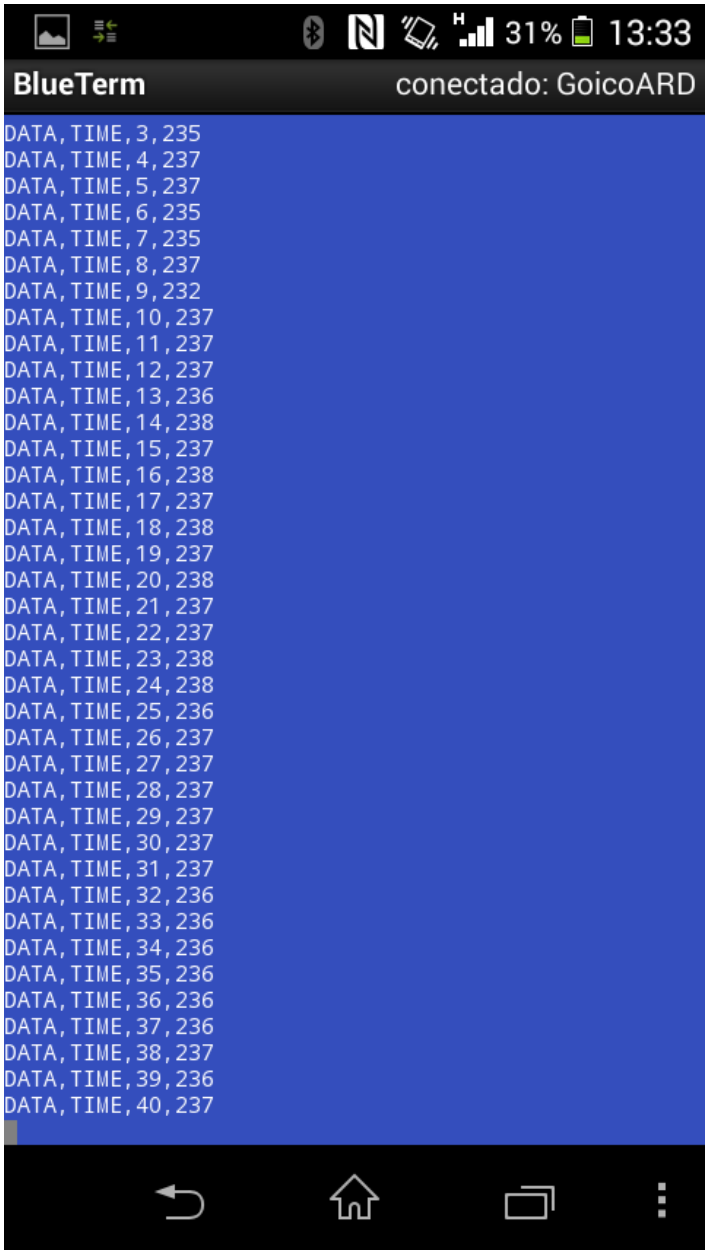


Figura 3.6. Datos enviados por el módulo Bluetooth vistos en BlueTerm.

Los datos recogidos pueden ir de 0 a 1023 debido al conversor analógico-digital de 10 bits de Arduino. Haciendo una equivalencia donde 0 es 0V y 1023 es 5V, vemos que estamos recogiendo una temperatura aproximada de 26ºC (237), lo que se corresponde a la sensación térmica del lugar de medición.

3.2. Configuración 2: NTC + ZigBee.

Este montaje tiene como objetivo explicar y comprender la programación de los módulos ZigBee mediante el software X-CTU presentado anteriormente ya que tanto el código de programa como el esquema de montaje son iguales (sustituyendo el módulo Bluetooth por un Xbee).

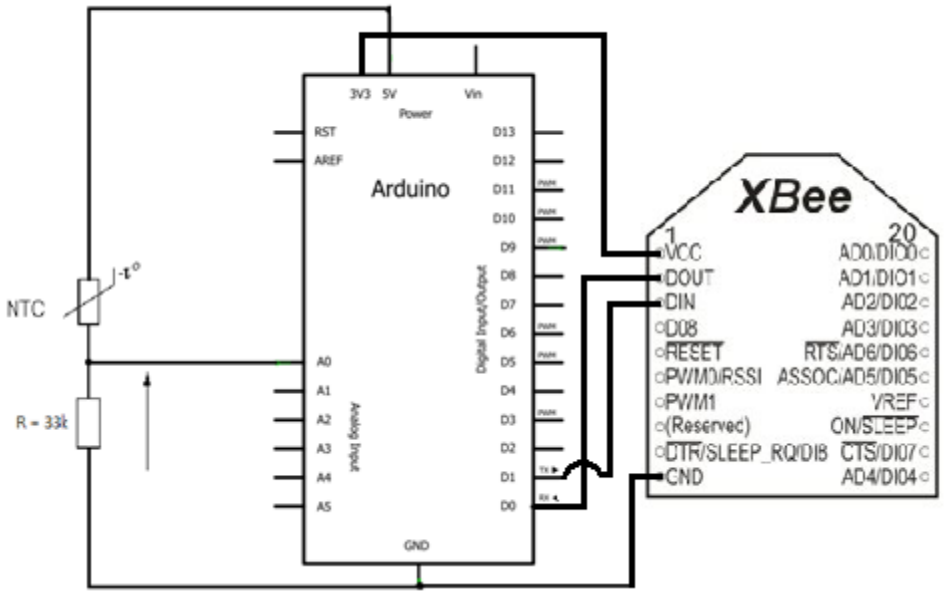


Figura 3.7. Esquemático configuración 2 Nodo sensor.

La comunicación será punto a punto, por lo que tan solo se necesitará un nodo que actúe como coordinador y otro que lo haga como Router o End Device. En estos momentos el consumo es un problema que carece de importancia ya que al estar realizando pruebas experimentales se prescinde de baterías y se emplea el propio cable USB como sistema de alimentación. Además, para la recogida de los datos en el PC a partir del nodo coordinador emplearemos todavía el propio monitor serie de Arduino en lugar de nuestro script Matlab.

Los módulos Xbee ofrecen la posibilidad de ser programados mediante comandos AT de una forma similar a los JY-MCU Bluetooth, pero como se dijo, existe la posibilidad de configurarlos de una forma más sencilla y completa desde la pestaña Modem Configuration de X-CTU.

El Xbee perteneciente al nodo coordinador será configurado con los siguientes parámetros:

> *Operating Channel (CH): E (Los canales disponibles vienen dados por el estándar 802.15.4. No importa cuál sea mientras sea el mismo para toda la red ZigBee).*
>
> *PAN ID (ID): 7 (El identificador de red de área personal ha de ser igual para todos los nodos de la red).*
>
> *Destination Address High (DH): 0013A200 (Dirección de arriba que aparece en la parte trasera del módulo que actuará como dispositivo final).*
>
> *Destination Address Low (DL): 40AFCEAA (Dirección de abajo que aparece en la parte trasera del módulo que actuará como dispositivo final).*

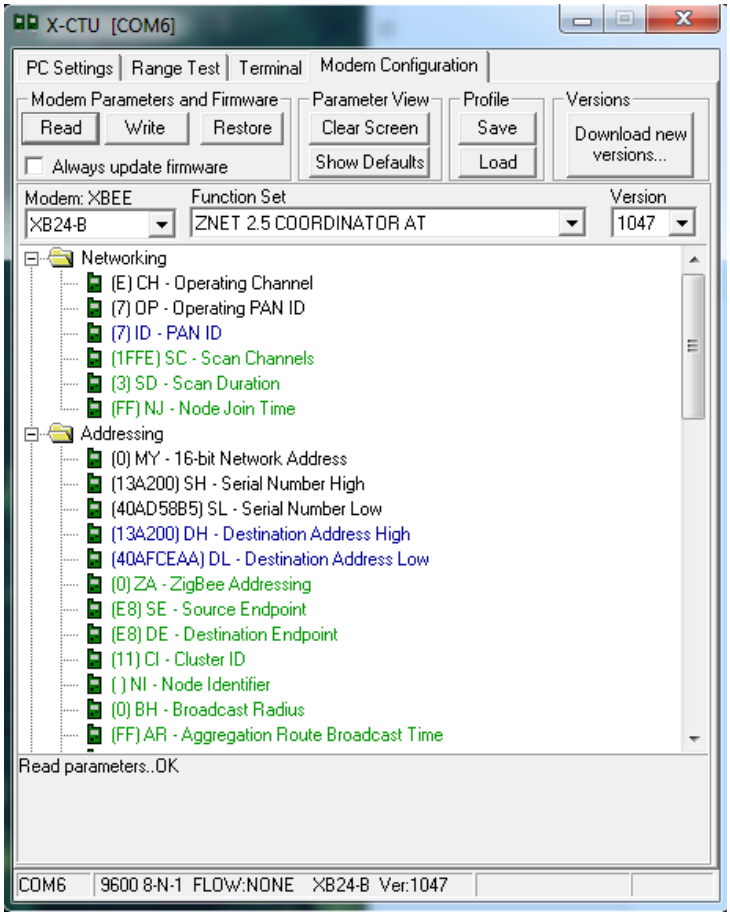


Figura 3.8. Configuración del Xbee coordinador.

y el perteneciente al nodo sensor dispositivo final:

Operating Channel (CH): E.

PAN ID (ID): 7.

Destination Address High (DH): 0013A200 (Dirección de arriba que aparece en la parte trasera del módulo que actuará como coordinador).

Destination Address Low (DL): 40AD58B5 (Dirección de abajo que aparece en la parte trasera del módulo que actuará como coordinador).

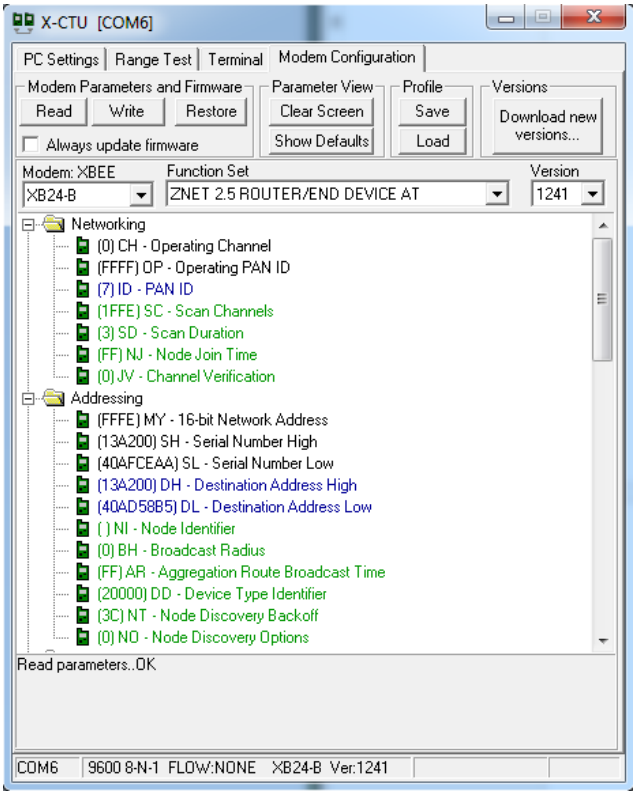


Figura 3.9. Configuración del Xbee end device.

Los datos recibidos a través del monitor serial son los mismos que en el caso Bluetooth pues nos encontramos ante la misma configuración con idéntico código de programa:

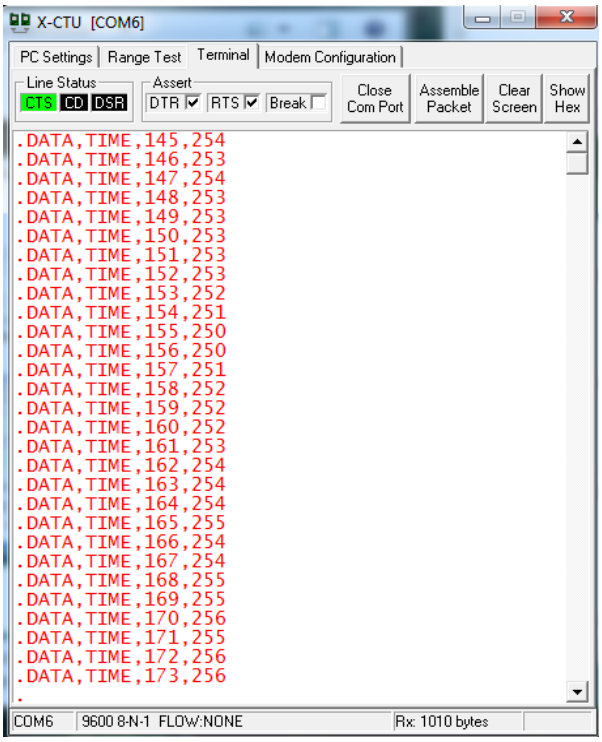


Figura 3.10. Datos recibidos en el nodo coordinador.

3.3. Configuración 3: Comunicación punto a punto ZigBee + SHT15.

Con la configuración 2 ya ha quedado realizada la comunicación punto a punto en una red ZigBee, sin embargo, este montaje no incluía un sensor comercial como el SHT15 o el DHT11, sino que hacía uso de un simple termistor para medir temperatura.

En este apartado se empleará el sensor SHT15 de Sensirion, un sensor de temperatura y humedad relativa cuyas características fueron descritas en la sección 2. Como la configuración de los módulos Xbee ya está hecha, sólo queda preocuparse de cómo extraer los datos relativos a temperatura y humedad del sensor.

Se recuerda que el SHT15 no cuenta con un bus de comunicación compatible con I2C con el cual conectarse directamente con Arduino, por lo que la extracción de la información va ser más compleja de lo habitual. En su lugar cuenta con una serie de códigos (ver datasheet), cada uno asociado con un comando, que al enviarlos al pin de DATA del sensor, este realiza la acción requerida.

Command	Code
Reserved	0000x
Measure Temperature	**00011**
Measure Relative Humidity	**00101**
Read Status Register	00111
Write Status Register	00110
Reserved	0101x-1110x
Soft reset, resets the interface, clears the status register to default values. Wait minimum 11 ms before next command	**11110**

Figura 3.11. Tabla de códigos de SHT15. [I4]

Luego como se puede observar si se introducen los códigos 00011 y 00101, el sensor devolverá los valores de temperatura y humedad relativa respectivamente. Sin embargo, esta no es una operación inmediata, y como en muchos sensores, este tipo de acciones requieren un protocolo de intercambio de datos entre los pines DATA y CLOCK que incluye: comienzo de la transmisión, transmisión de códigos, tiempo de medida, espera de confirmaciones (ACK), envío de datos y CRC.

Al no ser un proceso trivial, es necesario acudir a la hoja de características del módulo. En ella se entiende que la comunicación con el sensor tiene las siguientes etapas.

- **Puesta en marcha del sensor**: Primeramente se alimenta el sensor a la tensión de alimentación VDD, en este caso 5V. La velocidad de subida durante el encendido debe ser de por lo menos 1V/ms. Después de encender el sensor, necesita 11ms para llegar al estado de reposo. Durante este tiempo no debe enviarse nada.

- **Envío de comandos**: Para iniciar la transmisión se envía una secuencia de inicio. Consiste en una puesta a nivel bajo de la línea de datos mientras la línea de reloj permanece en alto seguida de un pulso a nivel bajo en la línea de reloj y luego un nuevo flanco ascendente en la línea de reloj y una subida en la de datos mientras la de reloj permanece en estado alto. Esto se puede ver mejor en la siguiente figura:

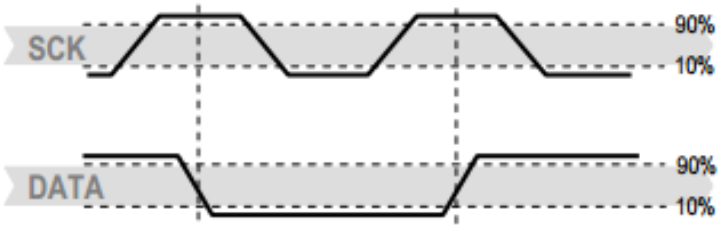


Figura 3.12. Secuencia de transmisión inicial. [I4]

La siguiente secuencia es ya el envío del comando para medir temperatura y/o humedad. Esta secuencia está formada por 3 bits de dirección (siempre son 000) y 5 bits del comando en sí mismo (los vistos en la anterior tabla). El SHT15 indica la correcta recepción de un comando poniendo el pin de datos a nivel bajo (bit de ACK) tras el flanco descendente del octavo pulso de reloj.

- **Medida de temperatura y humedad relativa**: Tras el envío de comandos el controlador debe esperar a que se complete la medida entre 20 y 320ms. El tiempo puede variar con la velocidad del oscilador interno. Para señalizar la conclusión de la medida, el SHT15 pone la línea de datos a nivel bajo y entra en modo inactivo. El controlador tiene que esperar a esta señal para reiniciar la línea de reloj y poder leer los datos.

  Se envían a partir de entonces 2 bytes de datos y un byte de CRC, este último es opcional. El controlador tiene que confirmar cada uno de estos bytes poniendo la línea de datos a nivel bajo. Todos los valores pasan primero el MSB, es decir, de izquierda a derecha.

  La comunicación termina tras la confirmación del CRC. Como en nuestro caso no va a ser necesario el empleo de CRC (porque si no se lee el valor no se enviará ningún dato y habrá una medida no tomada que no es crítica pues los valores de temperatura y humedad cambian muy lentamente), el controlador puede terminar la comunicación tras la recepción del último bit de medida sin más que poniendo el ACK a nivel alto en lugar de bajo. De esta forma el dispositivo vuelve automáticamente al modo “sleep” y se da por concluida la comunicación.

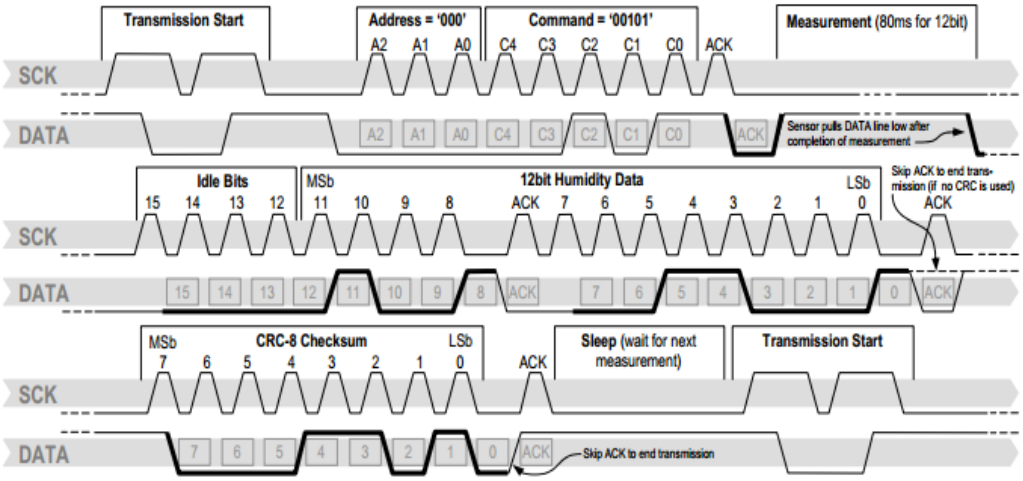


Figura 3.13. Ejemplo de medida de RH para un valor de 0000'1001"0011'001" = 2353 = 75.79% [I4]

Como se puede observar en la anterior figura, el sensor devuelve un valor que representa una humedad pero no devuelve directamente la humedad relativa en porcentaje o la temperatura en grados. Los valores reales de temperatura y humedad han de ser calculados a partir de las fórmulas que aporta el fabricante:

$$RH_{linear} = c_1 + c_2 \cdot SO_{RH} + c_3 \cdot SO_{RH}^2 (\%RH)$$

, donde para una medida de 12 bits como la nuestra $c_1 = -4$, $c_2 = 0.0405$ y $c_3 = -0.0000028$. Aquí hay que tener en cuenta que, al no estar midiendo a temperaturas significativamente distintas a 25ºC, no se realiza compensación de la humedad.

Para la temperatura, se procede de forma similar para obtener los grados centígrados según la siguiente ecuación:

$$T = d_1 + d_2 \cdot SO_T$$

, donde para una alimentación de 5V $d_1 = -40$ y $d_1 = 0.01$ (sin tener en cuenta el punto de rocío).

Tras obtener estos valores, se envían al puerto serie de Arduino, los recoge el módulo Xbee y los envía al nodo coordinador como en la configuración 2.

En el Anexo I se puede ver el código en el que se realiza todo el proceso de medición explicado paso a paso para mejor comprensión.

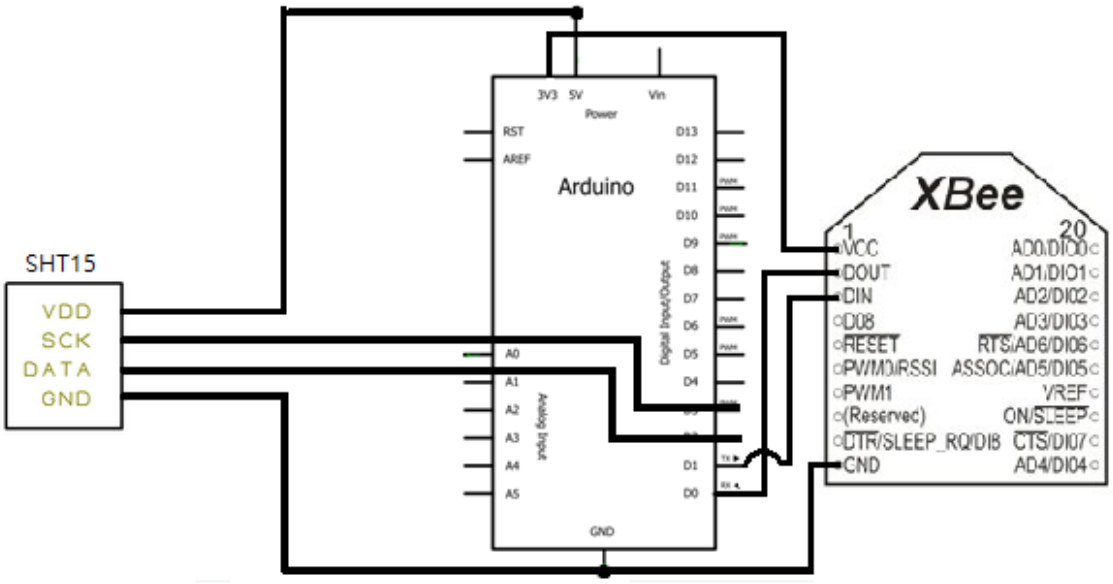


Figura 3.14. Esquema de la configuración 3 en el nodo sensor con el SHT15.

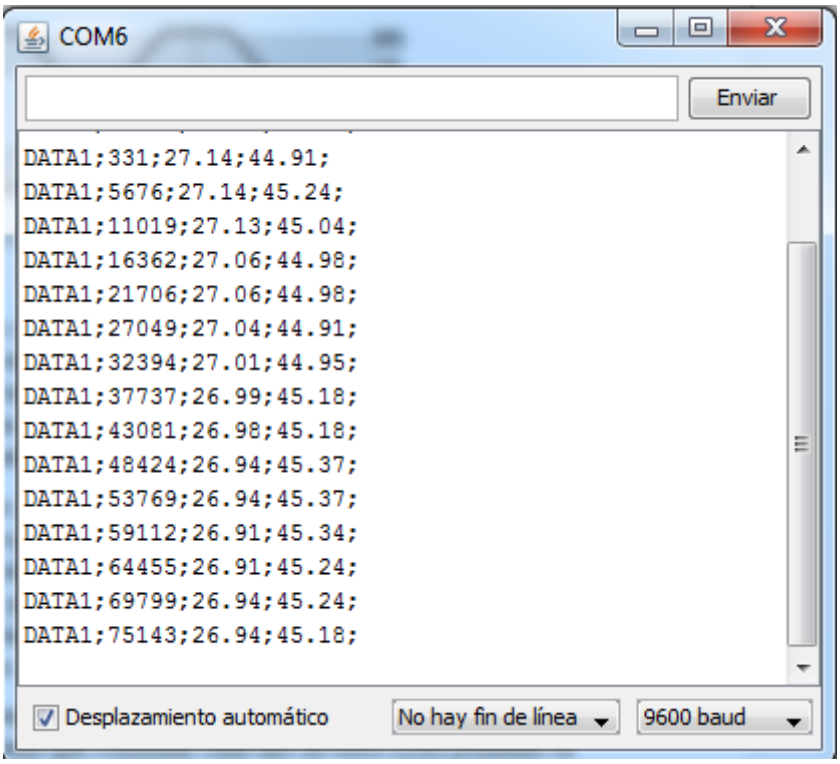


Figura 3.15. Datos recibidos en el nodo coordinador.

Como se puede ver, se recibe “DATA1” para distinguir que se trata del sensor SHT15 y posteriormente se recibe tiempo desde el inicio (ms), temperatura y humedad relativa. El envío es cada 5 ms.

3.4. Configuración 4: Multipunto ZigBee.

Esta es, por fin, la implementación a la que se quería llegar desde un principio y de la que se hablado una y otra vez a lo largo de la memoria del proyecto. Una vez vistas las 3 configuraciones previas a esta, la actual configuración se antoja necesaria. Y es que tras concluir con éxito la comunicación punto a punto entre dos nodos ZigBee, crear una red en estrella en la que existan 2 sensores a modo de dispositivo final y un nodo coordinador encargado de recolectar la información de ambos sensores, parece un paso natural.

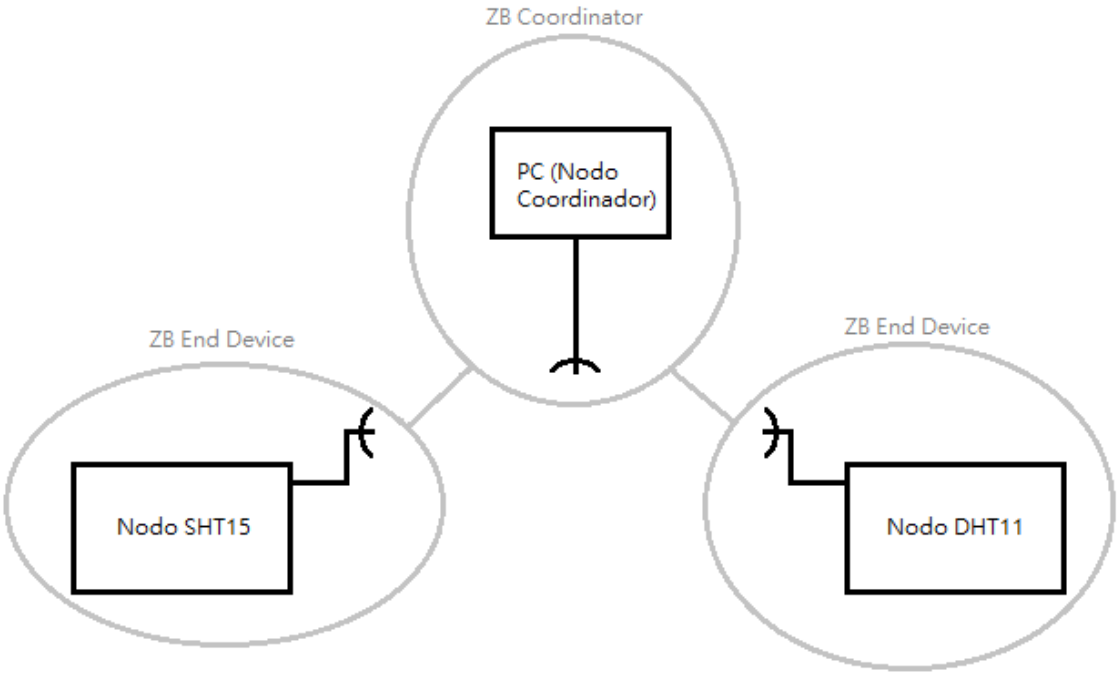


Figura 3.16. Red ZigBee final en estrella.

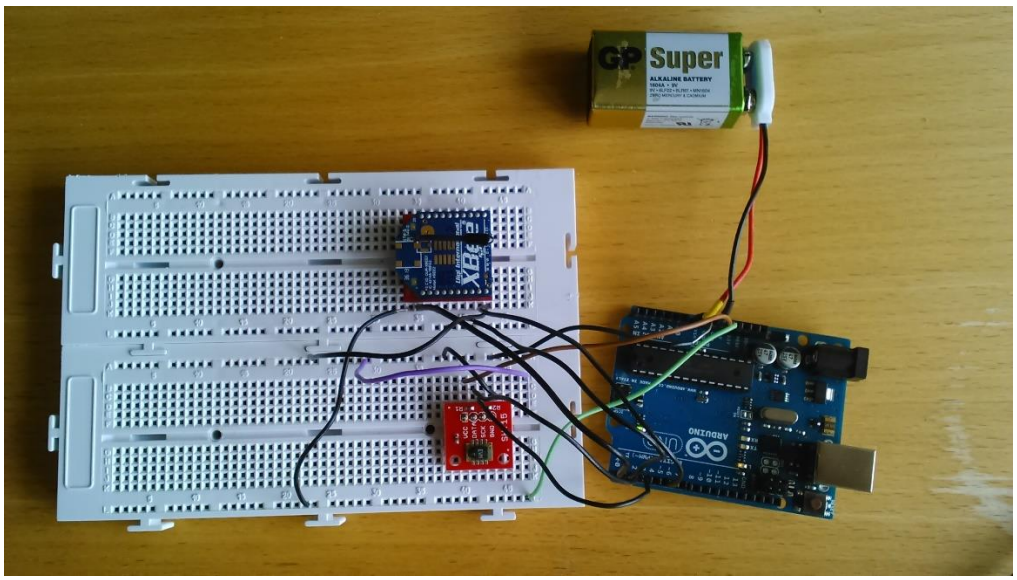


Figura 3.17. Nodo sensor.

Así pues, a partir de ahora se trabajará de forma inalámbrica total. En realidad hasta ahora también se estaba haciendo, pero la alimentación se realizaba vía USB, lo cual va en contra de lo que es un sistema completamente inalámbrico. Es por esta razón que se comienzan a utilizar las baterías de las que disponemos.

Lo primero es configurar el nuevo Xbee como dispositivo final añadiéndole el mismo canal e identificador de área personal que a los demás y estableciendo sus direcciones destino (alta y baja) como las del nodo coordinador.

Si a estas alturas hay algo claro es que el consumo de energía es el punto crítico de cualquier sistema inalámbrico como el que nos ocupa. Al retirar la alimentación cableada nos encontramos con este problema.

En nuestro sistema hay tres elementos que consumen energía, la placa Arduino, el sensor y el módulo Xbee, sin embargo el dispositivo que más consume con diferencia es el Xbee. Necesitamos reducir el consumo del transceptor. Ya se dijo en la sección 2 que los módulos Xbee poseían un modo de bajo consumo o "sleep" que le permitían reducir el consumo drásticamente cuando no tenía nada que enviar. Esto es positivo puesto que la temperatura y humedad son variables que varían lentamente y no se desea enviar información en todo momento.

Los módulos Xbee tienen desactivados por defecto los ciclos de sueño. Para ponerlos en modo "sleep" (mediante el pin de hibernación) en la práctica hay que hacer lo siguiente:

- En X-CTU, en el panel de Modem Configuration, se pone el parámetro Sleep Mode (SM) en "1 - PIN HIBERNATE" para indicarle al dispositivo que cuando el pin 9 está en estado alto debe terminar cualquier transmisión y entrar en modo reposo y que cuando el pin 9 está en estado bajo se sale del modo "sleep" y se puede enviar y recibir datos de nuevo.

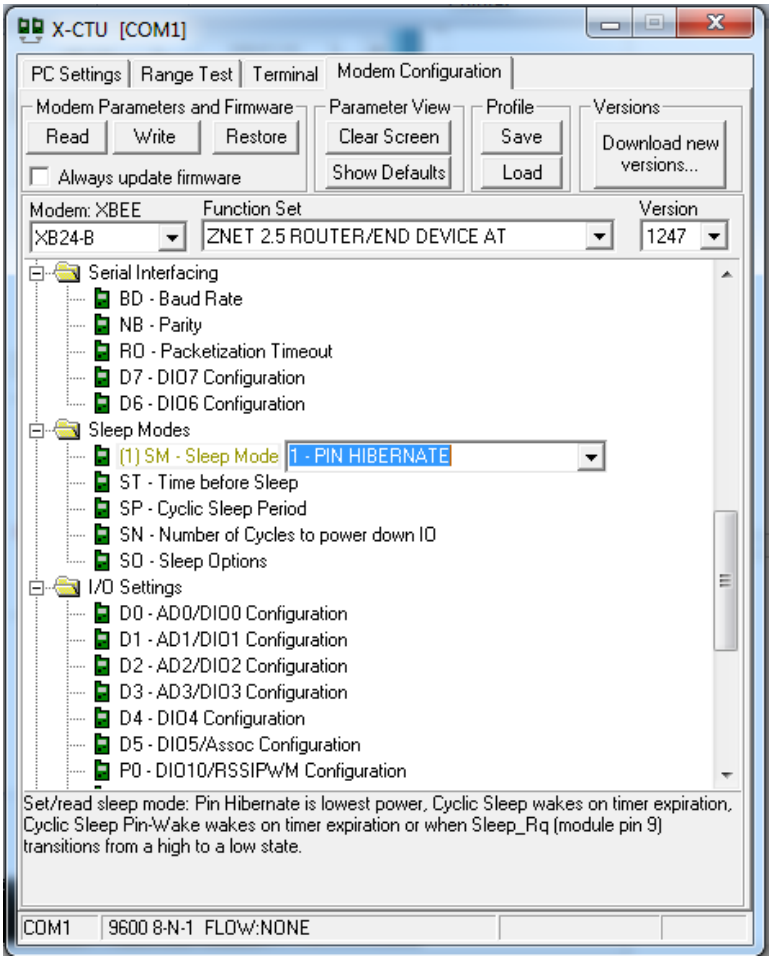


Figura 3.18. Xbee en modo “pin-hibernate”.

- En el código de programa Arduino, justo tras escribir los datos de temperatura o humedad relativa en el puerto serial, poner el pin de la placa Arduino conectado al pin 9 del Xbee en estado alto y justo antes de enviarlos, ponerlo en estado bajo. Se ha comprobado que resulta necesario contar con tiempos de guarda (10ms) para realizar dichas acciones ya que la lectura no es inmediata.

Con esto nos aseguramos que el consumo queda reducido de una forma importante y la red puede empezar a ser utilizada para realizar medidas.

Por último, notar que en la figura de la red ZigBee no existen dos sensores SHT15. Las razones de este cambio de sensor ya se dieron en la sección 2, lo que nos ocupa ahora es describir el funcionamiento del nuevo sensor y su comunicación con el microcontrolador.

Este nuevo sensor cuenta con un protocolo de comunicación a través de un hilo (1-wire), lo que facilita la integración al montaje mucho más que con el SHT15. Para la comunicación con el DHT11, en el código Arduino se emplea la librería disponible DHT.h y mediante las funciones de dicha librería “*dht.readHumidity()”* y “*dht.readTemperature()”*, se obtienen directamente la temperatura y la humedad relativa. Como vemos el proceso es mucho más simple, pero contamos con el inconveniente de

que este sensor sólo devuelve valores enteros y además su precisión no es muy buena tal y como se comprobará posteriormente.

Los datos recibidos en el nodo coordinador, realizando las medidas en la misma ubicación se pueden ver en la siguiente figura:

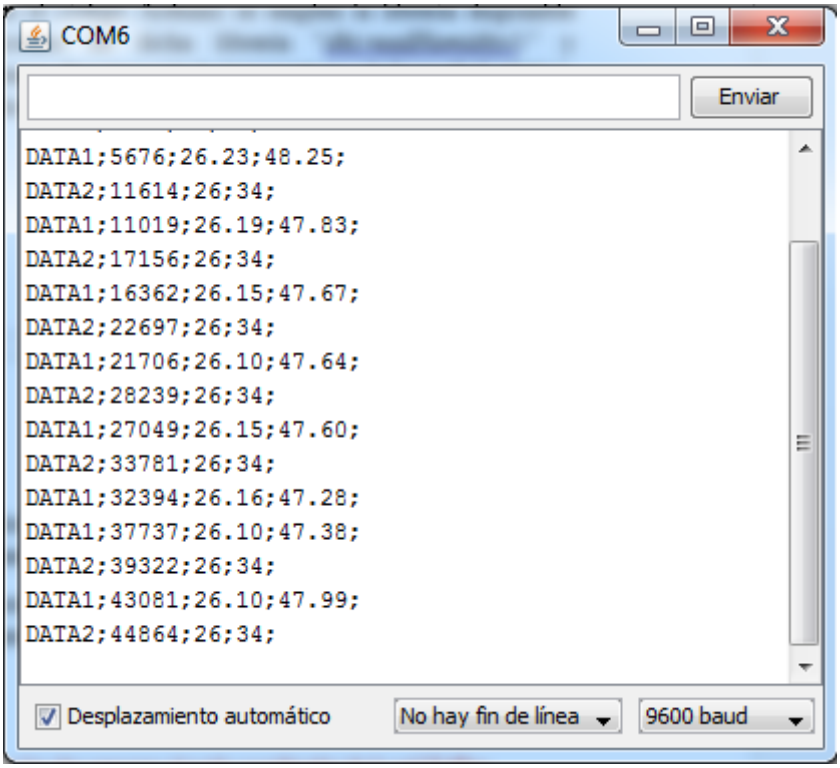


Figura 3.19. Datos enviados por los dos sensores al nodo coordinador de la red ZigBee.

Se puede ver cómo el SHT15 (“DATA1”) continúa enviando igual que antes y el sensor DHT11 (“DATA2”) comienza a enviar y da medidas enteras tal como se espera.

Parece necesario separar las medidas de ambos sensores y mostrarlas de una forma más limpia e intuitiva, con un gráfico por ejemplo. Esto se llevará a cabo en la siguiente configuración.

3.5. Configuración 5: Calibración de los sensores en una cámara climática.

La red en estrella está totalmente configurada, sin embargo, todavía no conocemos como van a responder los sensores a cambios en temperatura y/o humedad relativa. Este es un punto de máxima importancia de cara a una implementación final, ya que si el sensor no responde como se espera ante dichas variaciones, no sirve de mucho que tenga un consumo reducido o un gran alcance, pues no se debe olvidar que en última instancia lo más importante en cualquier aplicación de sensores es la medida que estos aportan de la

variable de interés. Luego entonces la gran pregunta a hacerse es: ¿me puedo fiar de mis sensores?

Según los fabricantes de sensores sus productos siempre son excelentes y aportan gran fiabilidad, pero se ha de tener en cuenta que el objetivo de dichos fabricantes es vender y es posible que sus sensores no respondan a nuestras expectativas.

En nuestra red hay ahora mismo dos sensores de temperatura y humedad. El sensor SHT15 presenta un mayor rango y una mejor precisión de medida tanto en humedad como en temperatura aunque también es cierto que la diferencia de precio entre ambos es muy significativa. Mientras que el sensor DHT11 cuesta en torno a 7€, el SHT15 se eleva hasta los 40€. Por tanto podemos esperar que la medida que proporcione el sensor SHT15 será mejor en todos los aspectos que la que nos dé el DHT11. La única duda es cómo comprobarlo experimentalmente con certeza.

La Universidad Pública de Navarra cuenta en su haber con una cámara climática para realizar pruebas de temperatura y humedad relativa con sensores. Por suerte, se ha conseguido acceso a ella y se van a poder calibrar los sensores de nuestra red.

La cámara de pruebas de humedad y temperatura del grupo Angelantoni permite establecer en su interior condiciones climáticas a nuestra elección. Para ello cuenta con un interfaz de usuario formado por un PC y el software WinKratos, que permite el control local y alejado de la cámara.

Así pues, el experimento que se va realizar funciona como sigue:

- Se montan fuera de la cámara climática los dos nodos sensores, muy cercanos el uno del otro y con la misma alimentación. Esto es importante dado que al estar alimentados conjuntamente, se reinician de forma simultánea (sin más que soltando y conectando el cable de alimentación de nuevo) y por tanto el contador de tiempo es el mismo para los dos.
- La alimentación se realiza mediante cable USB pues no se puede correr el riesgo de que en medio del proceso de medición se pierda la conexión.
- Se introducen ambos nodos en la cámara climática con cuidado de no producir cortocircuitos (se ponen encima de una superficie aislante) y se saca el cable de alimentación por fuera.

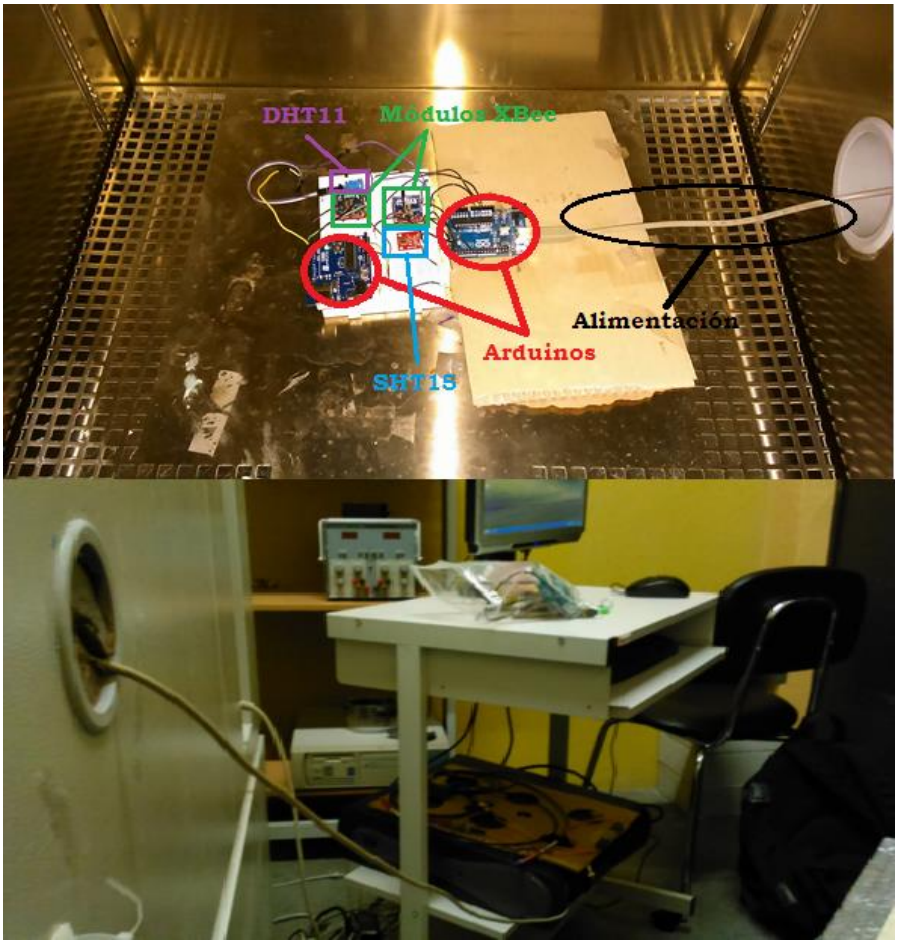


Figura 3.20. Montaje experimental.

- Se conecta el Xbee USB Explorer con el módulo Xbee configurado como coordinador a un PC.
- Se inicia el software WinKratos y se carga un perfil de 3 ciclos suaves de cambios de temperatura y humedad de 8 horas de duración.

Figura 3.21. WinKratos [I21]

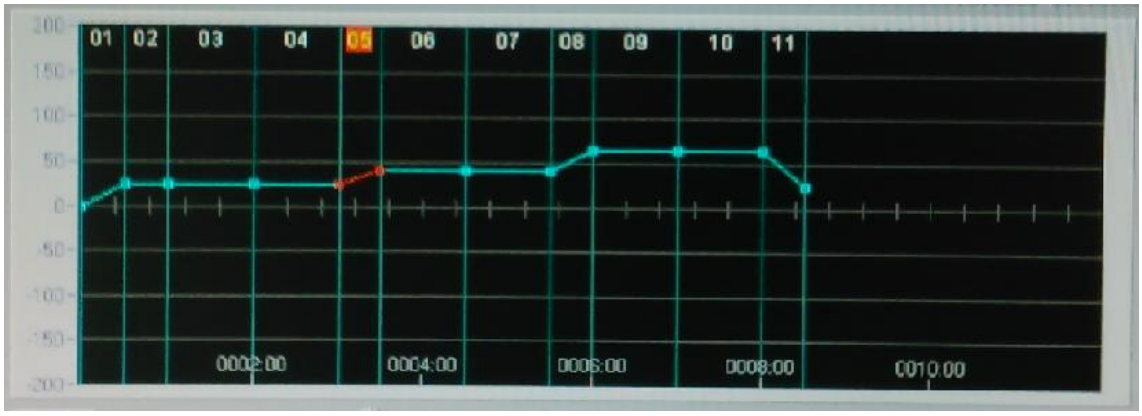
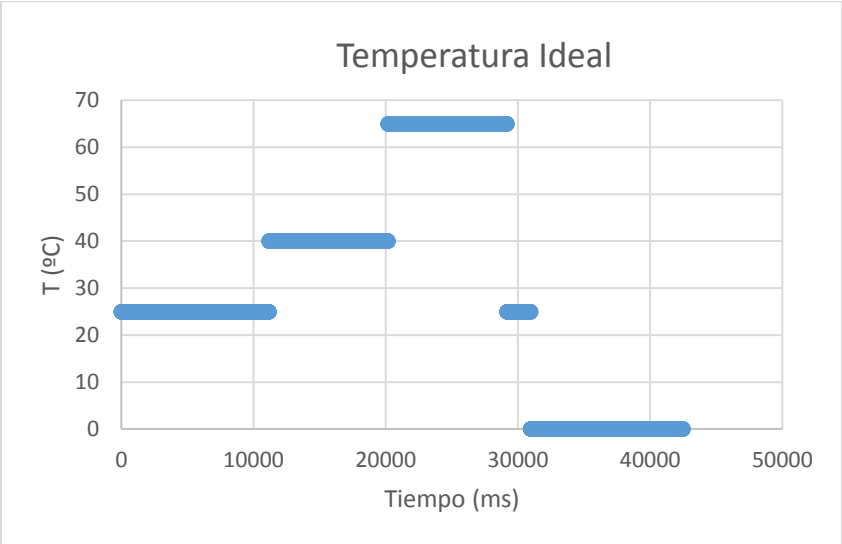


Figura 3.22. Perfil cargado en la cámara climática (sólo se ve temperatura, similar para HR).

- De forma inmediata se corre un sketch de Processing encargado de recoger los datos provenientes del puerto serie del Xbee USB Explorer y guardarlos en un archivo de texto (código completo explicado en el Anexo I).
- Se desconecta y vuelve a conectar la alimentación para reiniciar el contador de tiempo.

Una vez concluidas las 8 horas se coge el archivo de texto donde se han guardado por filas cada uno de los eventos producidos en el puerto serie y se pasa al programa de Matlab encargado de interpretarlo y dibujarlo (código completo explicado en el Anexo I).

Tras esto se tendrán dos gráficas, una generada por Matlab y otra la de evolución de la cámara climática. Es interesante notar que el perfil cargado en la cámara describe un comportamiento lineal ideal y que el resultado final no es exactamente así pues se producen oscilaciones y variaciones del modelo ideal. Lo importante es que la evolución de cada sensor sea la misma que la evolución real de la cámara.



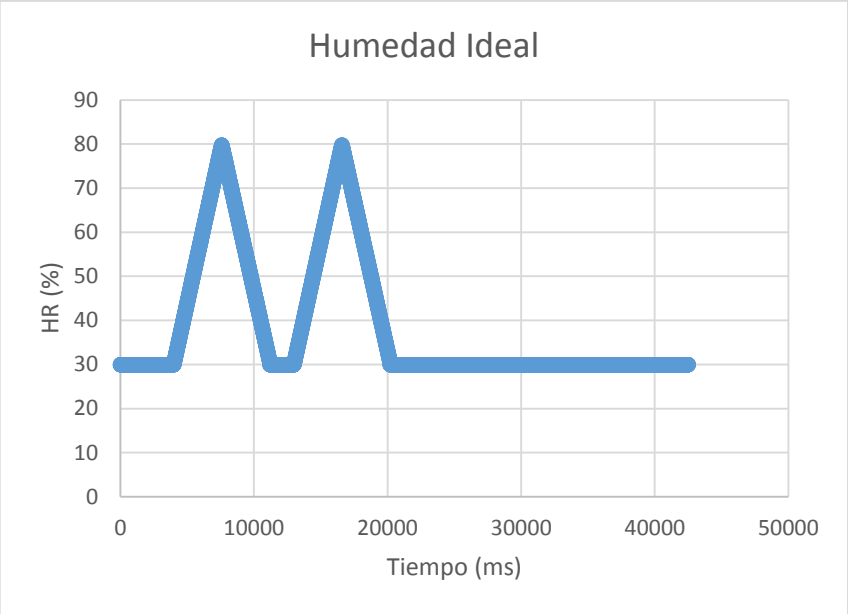
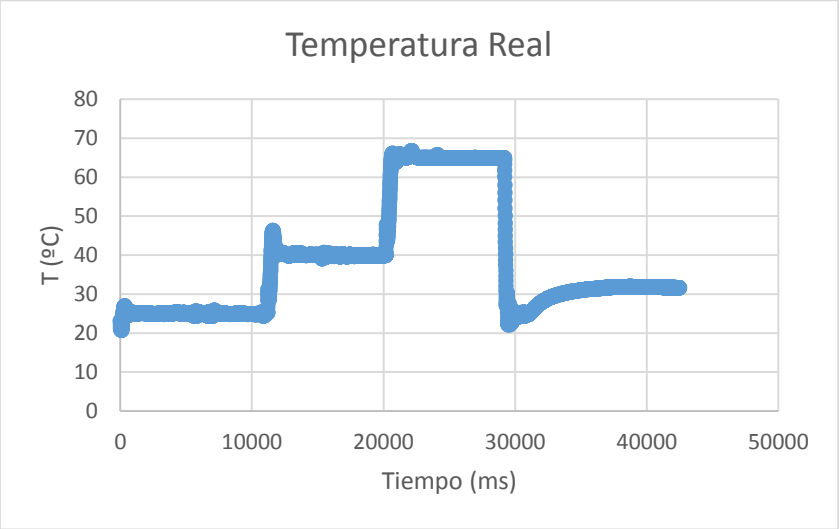


Figura 3.23. Evolución ideal de temperatura y humedad en la cámara climática.



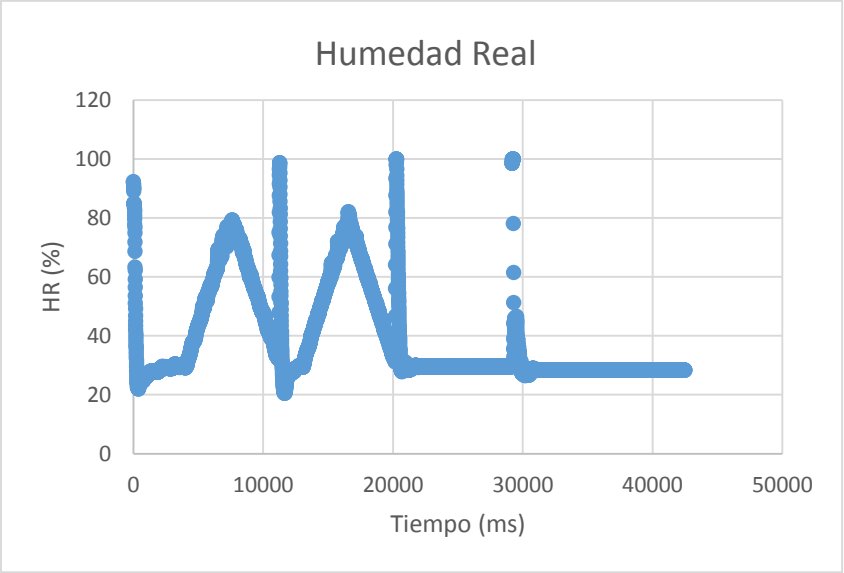


Figura 3.24. Evolución real de temperatura y humedad en la cámara climática.

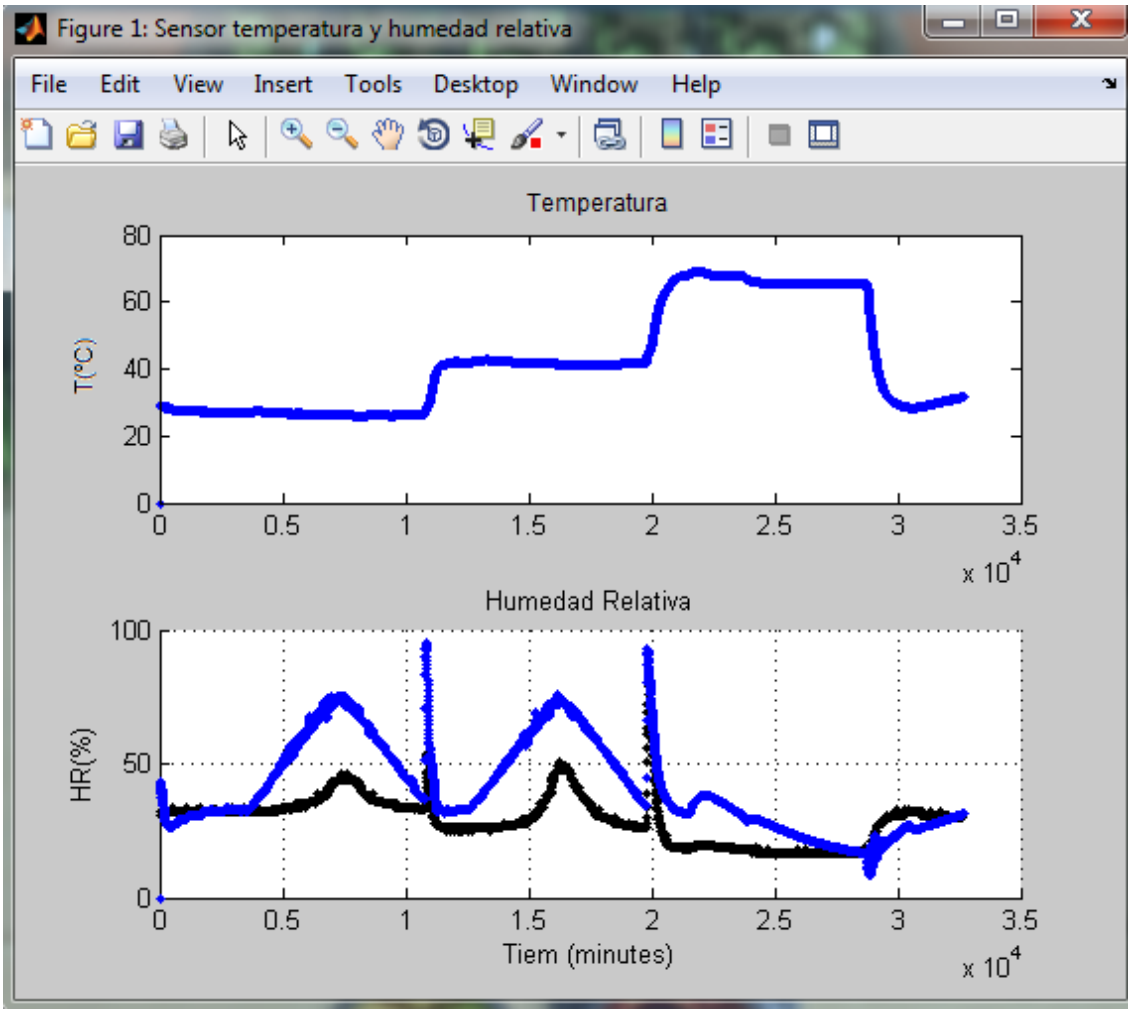


Figura 3.25. Evolución de temperatura y humedad de los sensores (Negro-DHT11; Azul-SHT15)

En la figura 3.23 se puede observar el perfil lineal cargado en la cámara climática. Como ya se ha dicho, éste va a distar mucho del auténtico comportamiento de la cámara, el cuál puede ser apreciado en la figura 3.24. Obsérvense aquí dos comportamientos interesantes.

- Cuando la temperatura cambia dentro de la cámara climática lo hace como toda planta real, es decir, sufre oscilaciones pues un control automático perfecto es imposible. Estas oscilaciones son rápidamente corregidas y el sistema enseguida entra en un ciclo de temperatura constante tal y como cabía esperar.
- En el momento en que la humedad relativa se encuentra en descenso y llega a un ciclo de pendiente nula donde la humedad debe mantenerse constante, se producen unos picos en la humedad. Esto se debe a las variaciones de temperatura que se producen en esos instantes, no al comportamiento de la humedad relativa y es un proceso inherente al funcionamiento de la propia cámara climática.

  En el momento en que se desea subir la temperatura, la cámara introduce una gran cantidad de vapor de agua procedente de un radiador a la misma para calentarla y conseguir el aumento de temperatura requerido. Este proceso es controlado inmediatamente pero durante el instante en que entra todo el vapor se producen máximos de humedad y por eso aparecen los picos en el gráfico de humedad relativa.

Aunque estos comportamientos están presentes durante todo el funcionamiento, la calibración del sensor no depende de ellos. Mientras el sensor en cuestión se comporte de la misma forma que la cámara climática (en su modelo real), se puede afirmar que es fiable.

Así pues en la figura 3.25 se aprecian las evoluciones de temperatura y humedad relativa del sensor SHT15 (azul) y del sensor DHT11 (negro). Como era de esperar, el seguimiento por parte del módulo SHT15 es prácticamente perfecto, tanto en humedad relativa como en temperatura, por lo que no se ve necesario ajustar la fórmula que devuelve dichos valores y se considera calibrado.

No así el módulo DHT11, el cual a pesar de generar cifras enteras, realiza un seguimiento de temperatura perfecto (se solapa con el SHT15), pero desarrolla una evolución en humedad relativa muy lejos de la deseada. Como el procedimiento de medida de humedad relativa se ha realizado mediante el empleo de una librería y es igual que para temperatura (que sí funciona correctamente), se desconoce el motivo del mal funcionamiento del sensor y se achaca al menor precio con respecto al sensor SHT15.

3.6. Creación de una placa de circuito impreso.

Con la finalización de cualquier proyecto se abre una serie de preguntas importantes: ¿y ahora qué?, ¿qué empleo se le puede dar al proyecto en el futuro?, ¿cuáles son los siguientes pasos a dar (si los hay)?, etc.

En nuestro caso, un proyecto electrónico, llega un punto en el cual se quiere crear algo funcional y definitivo tras las pruebas iniciales realizadas en las placas de montaje rápido. Una de las mejores formas de hacer esto es mediante una placa PCB (Printed Circuit Board), con la cual se pueden generar archivos Gerber que se pueden enviar a un servicio de fabricación PCB y obtener una placa profesional.

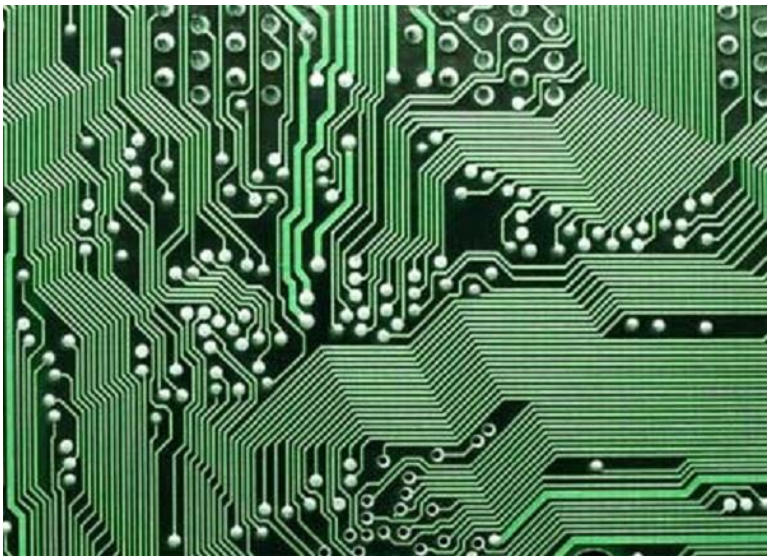


Figura 3.26. PCB. [I22]

Las alternativas en este caso son muchas y variadas y es responsabilidad del diseñador elegir cuál es óptima para el diseño. En nuestro caso se opta entre dos soluciones: emplear una placa Arduino Mini junto a un pequeño PCB con el transceptor, el sensor y la batería o bien realizar una shield para Arduino Uno. Se ha elegido esta última pues ya se dispone de la placa y no es necesario comprar una nueva. Aun así no se va a crear la placa sino únicamente se va a proceder a su diseño.

Las shields son placas que pueden ser conectadas encima de la placa Arduino extendiendo sus capacidades. Las diferentes shields siguen la misma filosofía que el conjunto original: son fáciles de montar y baratas de producir.

El software elegido para el diseño del PCB es Eagle, un programa de diseño de diagramas y PCBs con autoenrutador. Su gran ventaja es que tiene una versión gratuita y con gran cantidad de bibliotecas de componentes disponibles en Internet.

A continuación se puede ver el esquemático correspondiente a la shield Arduino y la placa con sus pistas ya enrutadas. La disposición de los elementos se realiza con libertad pero

teniendo en cuenta la cercanía de los pines. Así, para mayor comodidad durante el enrutado, se ha hecho que el pin que controla el modo pin hibernate sea el pin 2 en lugar del pin 6 y que los pines de comunicación con el sensor SHT15 sean el 3 y el 4 en lugar del 2 y el 3.

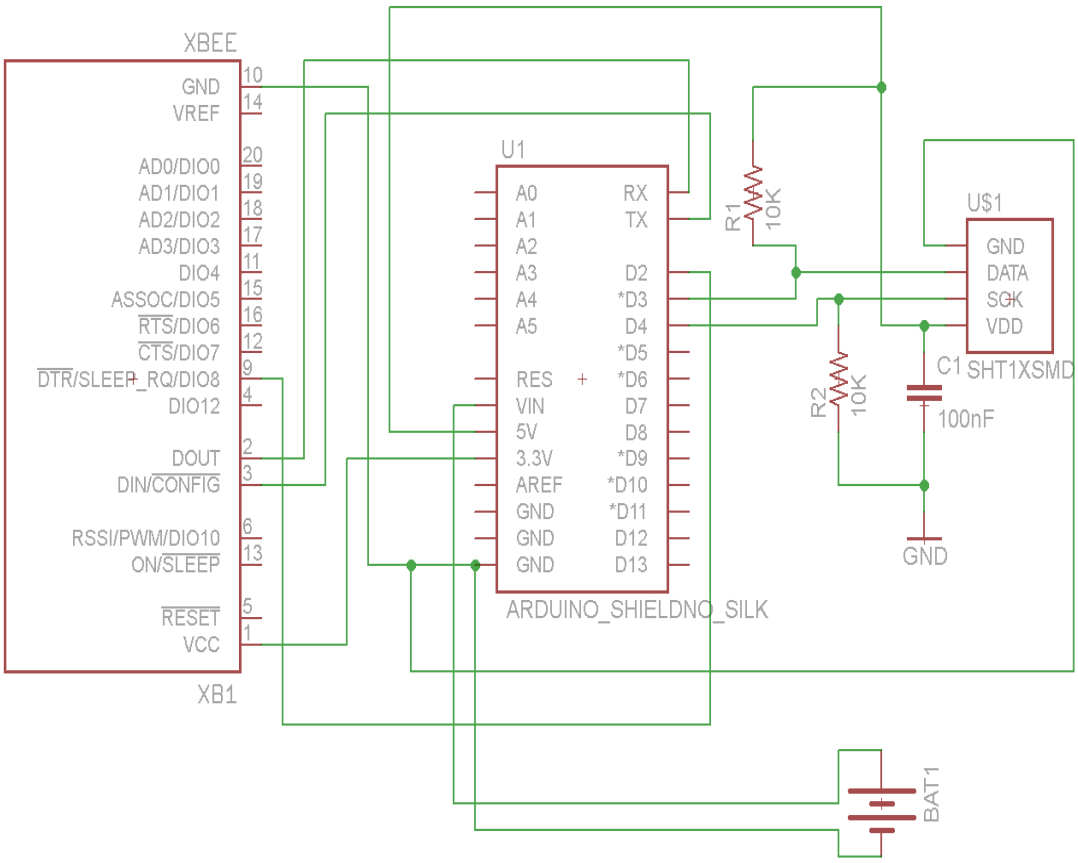


Figura 3.27. Esquemático del PCB.

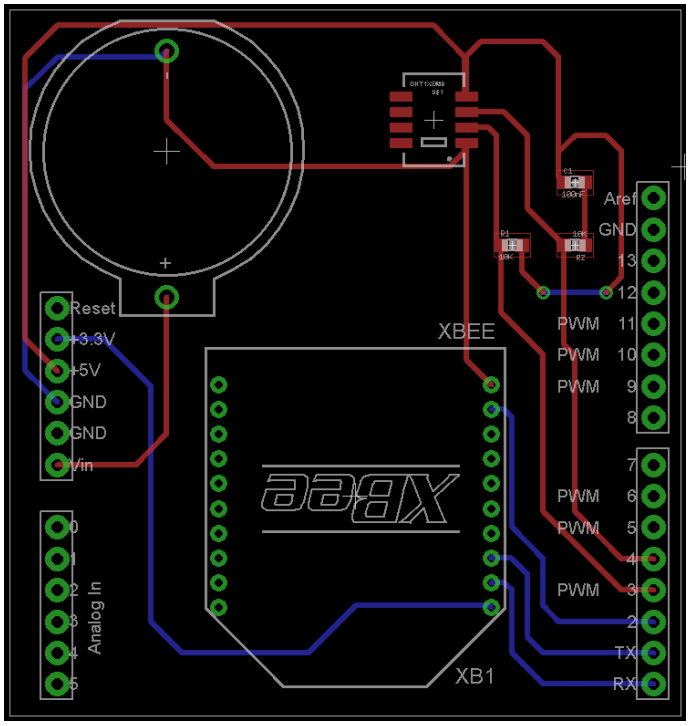


Figura 3.28. PCB enrutado sin plano de masa.

Notar que la pila empleada va a ser una pila de botón que minimiza el tamaño y que el sensor cuenta con las resistencias de pull-up y pull-down y con el condensador de desacoplo. El enrutado de las pistas se ha hecho en dos capas por necesidad y el plano de masa se ha colocado en la capa superior.

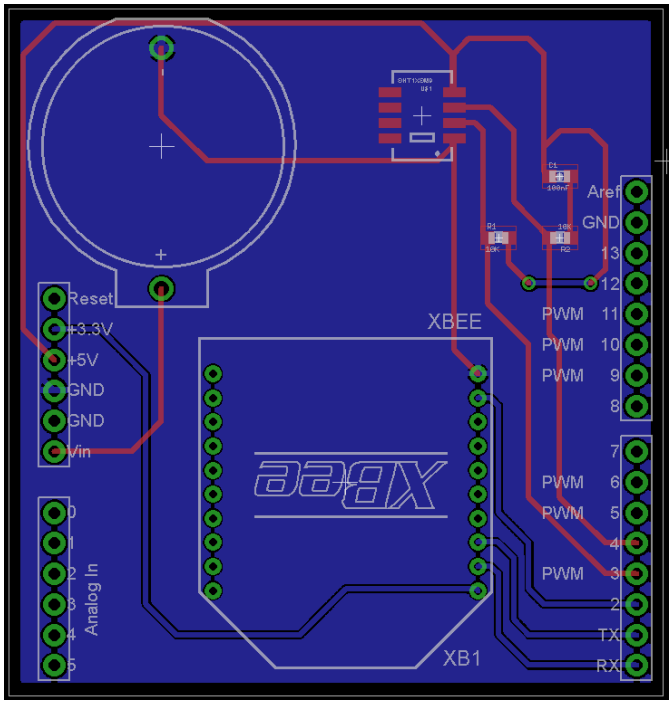


Figura 3.29. PCB enrutado con plano de masa.

Sección 4: Conclusiones.

Tras terminar el proyecto es hora de presentar conclusiones, analizar todos los pasos dados durante su realización y valorar el éxito del mismo.

Se recuerda que con el fin de completar los objetivos iniciales se ha dividido el proyecto en dos fases: la elección del hardware y software necesario y el desarrollo de las pruebas experimentales. Así, las conclusiones del proyecto se pueden dividir en dos partes diferentes.

En referencia a la elección de los materiales y el estudio teórico realizado al respecto:

- Zigbee ofrece ventajas importantes respecto a otras tecnologías de comunicación inalámbrica: bajo consumo de energía gracias a la posibilidad de poder tener los módulos Xbee en modo "sleep" y bajo coste, además de la posibilidad de interconexión de muchos nodos en una red.
- La elección de sensores es teniendo en cuenta aspectos como la comunicación del mismo, la precisión, el tamaño o el coste. En este sentido el SHT15 es la elección ideal.
- Con Arduino tenemos la posibilidad de interconectar todo de una manera sencilla y totalmente controlada. Se ha obtenido un grado de alta familiarización con Arduino.
- La batería es un punto crucial en el sistema y se ha realizado un estudio para la elección de la misma.

De las pruebas experimentales que se han hecho con los sensores se pueden sacar más conclusiones:

- Se han realizado pruebas con termistores para recordar el funcionamiento de los mismos visto durante la carrera y se ha comprobado la correcta operación.
- Se ha aprendido a emplear módulos Bluetooth JY-MCU y su programación y se ha empezado a conocer la tecnología.
- Comprobada la correcta operación de los módulos Xbee y su funcionamiento necesario en modo sleep, se pueden leer en la sección 3 las conclusiones obtenidas de su práctica en el interior de la cámara climática.

Finalmente, se puede decir que se han alcanzado los objetivos que planteaba el proyecto de forma satisfactoria.

De cara a futuros trabajos, se podrían añadir nuevos sensores inalámbricos, con el fin de desarrollar redes más complejas, y de esta forma tener una red con varias capas, poder hacer diferentes enrutamientos en función de los obstáculos, etc.

Sección 5: Presupuesto.

Gastos materiales

Descripción	Coste Unitario	Cantidad	Importe
Placa Arduino Uno	20,00 €	2	40,00 €
Módulo Bluetooth JY-MCU	5,40 €	1	5,40 €
Módulo ZigBee XB24 XBEE ZNET	27,77 €	3	83,31 €
Breakout Board para el módulo Xbee	2,95 €	2	5,90 €
Xbee Explorer USB	24,95 €	1	24,95 €
Sensor de temperatura y humedad SHT15 + breakout	30,75 €	1	30,75 €
Sensor de temperatura y humedad DHT11 + adaptador 3 pines	8,06 €	1	8,06 €
Pila alcalina Energizer 9V	13,54 €	2	27,08 €
Conector para pila de 9V	1,89 €	2	3,78 €
NTC NJ28RA0104F	1,31 €	1	1,31 €
Pack 10 Resistencias	1,00 €	1	1,00 €
Cable USB 2.0 a Mini USB 1.8m	1,45 €	1	1,45 €
Cable USB 2.0 tipo A-macho a tipo B-macho (trenzado, 1,8m)	5,12 €	1	5,12 €
Importe total			**238,11 €**

Gastos mano de obra

Puesto	Descripción del trabajo	Trabajadores	Horas	Salario	Importe
Ingeniero	Realización de la totalidad del proyecto	1	450	15,00 €	6.750,00 €
Importe total					**6.750,00 €**

Importe total gastos directos	6.988,11 €
Importe total gastos indirectos (10% gastos directos)	698,81 €
Importe total sin IVA	7.686,92 €
Importe total del proyecto (IVA 21%)	**9.301,17 €**

Tras la elaboración de un presupuesto para el desarrollo completo de un producto siempre se plantea una pregunta: ¿puede llegar a ser rentable la venta del producto en cuestión o no interesa llevarlo al mercado porque simplemente su precio no rentabiliza su utilidad?

De un simple análisis del presupuesto se desprende que la creación, fabricación y posible venta de módulos inalámbricos sensores de temperatura y humedad como el que nos ocupa carece absolutamente de sentido si sólo se van a fabricar dos, ya que cada uno se debería vender por separado a un precio de 4650,85€ para tan solo recuperar la inversión. Luego este proyecto tal y cómo está aquí explicado no es viable de ser llevado al mercado.

A partir de aquí es importante notar que la creación de nuevos módulos en cadena podría cambiar este análisis completamente ya que gran parte de las horas de trabajo invertidas en el desarrollo e investigación del módulo no serían tenidas en cuenta para posteriores productos. Además parte de los materiales expuestos en la tabla no serían empleados, aunque siempre se emplearían sensores SHT15, más caros que los DHT11.

Así, por ejemplo, supongamos que de las 450 horas invertidas para el desarrollo de dos módulos, hiciesen falta 10 horas para la creación de nuevos módulos idénticos y supongamos también que se creasen 500 módulos en lugar de 2:

- El coste material de un módulo se reduciría a 110,79€.
- El coste de mano de obra por cada módulo bajaría hasta los 150€.
- Los módulos deberían ser vendidos a 280€ para recuperar la inversión inicial.

Como se puede apreciar los gastos se reducen enormemente. Teniendo en cuenta que aunque la fabricación fuese masiva un módulo nunca podría bajar de los 260,79€, se deberían buscar nuevos métodos de reducción de gastos, como la compra de materiales más baratos o la disminución del sueldo de los trabajadores para poder ofrecer un producto con una mejor relación calidad precio que fuese aceptado por los compradores y permitiese abrirse un hueco en el mercado.

Sección 6: Bibliografía.

Libros consultados:

[L1] Fernández Martínez, R. (2009). Redes Inalámbricas de sensores: teoría y práctica. Universidad de la Rioja: Servicio de publicaciones.

[L2] Guía usuario Xbee.

Páginas web visitadas:

[W1] http://www.efficity.es/wireless-sensor-network_post_wireless-sensor-network-24291757.htm

[W2]http://microcontroladores2utec.files.wordpress.com/2009/11/180909_articulo_colaboracion_boletin_fica_omar_otoniel_flores.pdf

[W3] http://arduino.cc/

[W4] http://es.wikipedia.org/wiki/Bluetooth

[W5] http://www.sensorsmag.com/networking-communications/batteries/a-practical-guide-battery-technologies-wireless-sensor-netwo-1499

[W6] http://www.mfbarcell.es/conferencias/wsn.pdf

[W7]http://research.microsoft.com/pubs/192688/IWS%202013%20wireless%20power%20consumption.pdf

[W8] http://deeea.urv.cat/public/PROPOSTES/pub/pdf/2114pub.pdf

Referencias de imágenes:

[I1] http://en.wikipedia.org/wiki/Wireless_sensor_network

[I2] http://silicio.mx/nodo-de-sensor-inalambrico-kit-solar

[I3] http://www.micro4you.com/files/sensor/DHT11.pdf

[I4]http://www.sensirion.com/fileadmin/user_upload/customers/sensirion/Dokumente/Humidity/Sensirion_Humidity_SHT1x_Datasheet_V5.pdf

[I5]http://www.sensirion.com/fileadmin/user_upload/customers/sensirion/Dokumente/Humidity/Sensirion_Humidity_SHT25_Datasheet_V2.pdf

[I6] http://www.robotshop.com/ca/en/sfe-sht15-humidity-temperature-sensor.html

[I7]http://www.wingstech.com.ph/RequestQuote/ItemDetail?recoId=1468&process=ViewDetail&quantity=0

[I8] http://www.futurlec.com/Atmel/ATMEGA8.shtml

[I9] http://www.cortoc.com/p/arduino.html

[I10] http://es.wikipedia.org/wiki/Bluetooth

[I11] http://plataformaszigbee.blogspot.com.es/2012/05/practica-1-configuracion-y-conceptos.html

[I12] http://www.engineersgarage.com/articles/what-is-zigbee-technology?page=3

[I13] http://www.zigbee.org/

[I14] http://www.giant-bicycles.com/en-us/gear/product/neos.pro.ant.wireless.computer.hrm/511/47481/

[I15] http://uk.farnell.com/digi-international/xb24-bwit-004/zigbee-module-xbee-znet-wire-ant/dp/1546390

[I16] http://es.farnell.com/rf-solutions/easybee-so/zigbee-module-rf-txrx-802-15-4/dp/1330069

[I17] http://pdf1.alldatasheet.es/datasheet-pdf/view/248146/ETC2/ZB21.html

[I18] http://www.datasheetarchive.com/dlmain/SFDatasheet-3/sf-00061089.pdf

[I19] https://www.sparkfun.com/products

[I20] http://www.sensorsmag.com/networking-communications/batteries/a-practical-guide-battery-technologies-wireless-sensor-netwo-1499

[I21] http://www.acsenvironmentaltestchambers.com/Customer/Winkratos

[I22] http://dangerousprototypes.com/2011/10/26/a-guide-to-seeedstudio%E2%80%99s-fusion-pcb-service/

Sección 7: Anexos.

7.1. Anexo I. Códigos de programas.

7.1.1. Códigos Arduino.

7.1.1.1. Configuración de los módulos Bluetooth mediante comandos AT.

```
////////////////////////////////////////////////////////////////
//        Cambio de la configuración del módulo bluetooth      //
//                      mediante comandos AT.                  //
////////////////////////////////////////////////////////////////
// Variable para control de configuración
boolean conf = false;
int espera = 1000;

void setup(){
   pinMode(13,OUTPUT);     // Pin13 para control del proceso
   Serial.begin(9600);    // Velocidad del módulo bluetooth
   digitalWrite(13, LOW);  // Apagamos el led 13
}
void loop(){
  if (conf){
    // Parpadeo para verificar su funcionamiento
    digitalWrite(13, HIGH);   // Enciende el led
    delay(espera);            // Espera
    digitalWrite(13, LOW);    // Apaga el led
    delay(espera);            // Espera
  }
  else{
    digitalWrite(13, HIGH); // Empieza el tiempo de espera para reconectar
    Serial.println("Configuracion del modulo bluetooth");
    Serial.println("Conecte el TX del modulo BT al RX del Arduino y el RX del
modulo BT al TX del Arduino");
    delay(5000);           // Espera de 15 segundos (se puede cambiar)
    digitalWrite(13, LOW);  // Tiempo de espera agotado para reconectar
```

```
    Serial.print("AT");     // Empieza la comunicación por el módulo BT
    delay(espera);          // Espera para el envío de comandos
    Serial.print("AT+NAMEGoicoARD"); // Nombre del dispositivo
    delay(espera);                   // Espera para el envío de comandos
    Serial.print("AT+BAUD7");   // Establecemos la velocidad en 9600
    delay(espera);              // Espera para el envío de comandos
    Serial.print("AT+PIN9876");  // Establecemos el pin de vinculación
    delay(espera);               // Espera para el envío de comandos
    //En este punto debe estar configurado
    digitalWrite(13, HIGH); // Todo ha funcionado según lo esperado
    conf = true;     // Para no volver a configurarlo, salvo que se resetee
  }
}
```

7.1.1.2. NTC + Puerto serie + Módulo inalámbrico.

```
//////////////////////////////////////////////////////////
//        Programa que recoge datos del puerto A0       //
//       de Arduino y se los pasa al serial. De ahí     //
//    posteriormente se recogen y se grafican (PDX DAQ) //
//              Válido para NTC+BT y NTC+ZB             //
//////////////////////////////////////////////////////////
// Variable que muestra el índice de la medida
int i = 0;
void setup() {
  // Se inicia el puerto serie a 9600 baudios
  Serial.begin(9600);
  // Se borra todo lo que había previamente
  Serial.println("CLEARDATA");
  // Se imprimen los nombres de cada columna
  Serial.println("LABEL,Time,i,Temperatura");
}
void loop() {
  // Se imprime en el puerto serial cada fila con: Tiempo, iteración, lectura
  Serial.print("DATA,TIME,");
  Serial.print(i);
```

```
  Serial.print(",");
  Serial.println(analogRead(0));
  // Se aumenta en uno el índice de filas
  i++;
  delay(500);
}
```

7.1.1.3. Sensor SHT15.

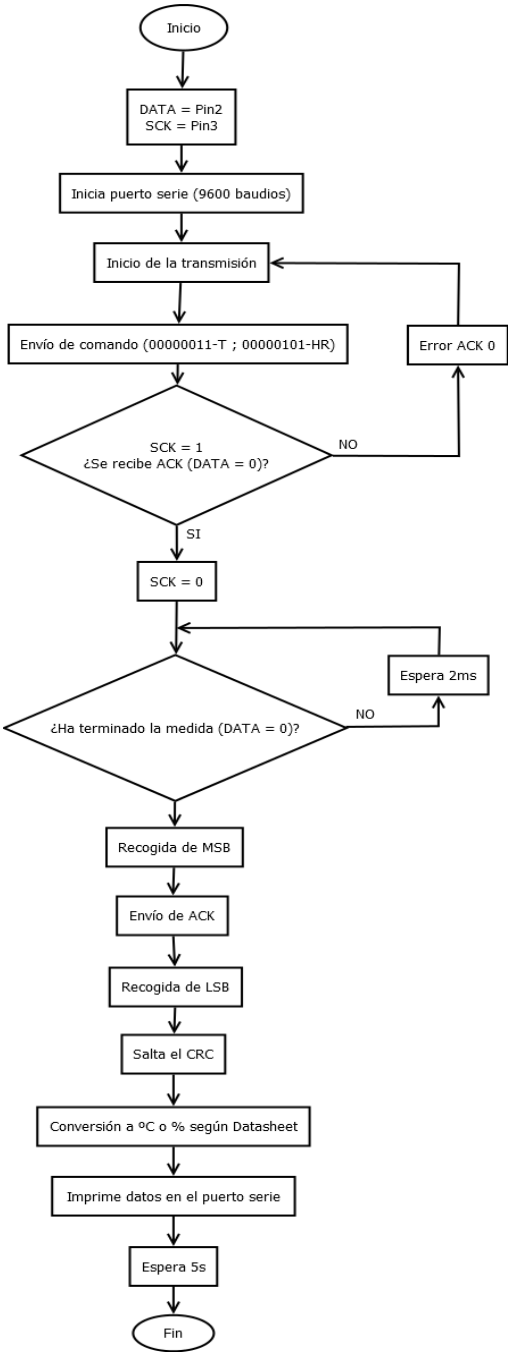
```
//////////////////////////////////////////////////////////////////////
//     Programa encargado de realizar toda la comunicación con el      //
//    sensor SHT15 mediante las directrices aportadas en la Sección    //
//     3 del proyecto, con el fin de obtener temperatura (en ºC) y     //
//    humedad relativa (en %) e imprimirlas en el puerto serie donde   //
//  serán recogidas por el módulo inalámbrico para su posterior envío. //
//          Incluye el modo de bajo consumo (pin hibernation)          //
//                        para los módulos Xbee.                       //
//////////////////////////////////////////////////////////////////////
// Se definen los pines de Arduino conectados
// al reloj y a la línea de datos del sensor
int SHT_clockPin = 3;  // pin used for clock
int SHT_dataPin  = 2;  // pin used for data
// Variable long que contiene los milisegundos pasados desde el reset del
programa
unsigned long time;
void setup(){
  // Se abre el puerto a 9600 baudios
  Serial.begin(9600);
}
void loop(){
  // Variables que contienen la temperatura en ºC y la HR en %
  float temperature = getTemperature();
  float humidity = getHumidity();
  // Pin 6 configurado como pin de hibernación en modo salida
  // (prescindir del pin 6 si no se desea modo de bajo consumo)
  pinMode(6, OUTPUT);
```

```
  // Se desactiva el modo de bajo consumo para enviar
  digitalWrite(6, LOW);
  delay(10); // Tiempo de guarda
  // Envío de tiempo, temperatura y HR por parte del sensor
  Serial.print("DATA1"); //SHT15
  Serial.print(";");
  time = millis(); // Devuelve los ms desde que se inicio el programa
  Serial.print(time);
  Serial.print(";");
  Serial.print(temperature);
  Serial.print(";");
  Serial.print(humidity);
  Serial.print(";");
  Serial.print("\n");
  delay(10); // Tiempo de guarda
  // Se activa el modo de bajo consumo
  // durante los 5 segundos que no se envía nada
  digitalWrite(6, HIGH);
  delay(5000);
}
float getTemperature(){
  // Devuelve la temperatura en ºC
  SHT_sendCommand(B00000011, SHT_dataPin, SHT_clockPin); // Envía comando de
temperatura
  SHT_waitForResult(SHT_dataPin); // Espera la confirmación de solicitud
  int val = SHT_getData(SHT_dataPin, SHT_clockPin); // Recoge el valor de T
del sensor
  SHT_skipCrc(SHT_dataPin, SHT_clockPin); // Se salta el CRC
  return (float)val * 0.01 - 40; // Se convierte a ºC según datasheet
}
float getHumidity(){
  // Devuelve la humedad relativa en %
  SHT_sendCommand(B00000101, SHT_dataPin, SHT_clockPin); // Envía comando de
HR
  SHT_waitForResult(SHT_dataPin); // Espera la confirmación de solicitud
  int val = SHT_getData(SHT_dataPin, SHT_clockPin); // Recoge el valor de HR
del sensor
```

```
  SHT_skipCrc(SHT_dataPin, SHT_clockPin); // Se salta el CRC
  return -4.0 + 0.0405 * val + -0.0000028 * val * val; // Se convierte a %
según datasheet
}
void SHT_sendCommand(int command, int dataPin, int clockPin){
  // Envío de comando de inicio de la transmisión
  pinMode(dataPin, OUTPUT);
  pinMode(clockPin, OUTPUT);
  digitalWrite(dataPin, HIGH);
  digitalWrite(clockPin, HIGH);
  digitalWrite(dataPin, LOW);
  digitalWrite(clockPin, LOW);
  digitalWrite(clockPin, HIGH);
  digitalWrite(dataPin, HIGH);
  digitalWrite(clockPin, LOW);
  // Introducción del comando de T o HR mientras se realiza el Toggle del
reloj
  // (los 3 MSB son de dirección y son 000, los últimos 5 son el comando)
  shiftOut(dataPin, clockPin, MSBFIRST, command);
  // Comprueba que se recibe el ACK
  digitalWrite(clockPin, HIGH);
  pinMode(dataPin, INPUT);
  if (digitalRead(dataPin)) Serial.println("ACK error 0");
  digitalWrite(clockPin, LOW);
}
void SHT_waitForResult(int dataPin){
  // Espera la respuesta del SHT15
  pinMode(dataPin, INPUT);
  int ack;
  //Se esperan dos segundos hasta leer el valor
  for (int i = 0; i < 1000; ++i){
    delay(2);
    ack = digitalRead(dataPin);
    if (ack == LOW) break;
  }
}
```

```
int SHT_getData(int dataPin, int clockPin){
  // Se recogen los datos del sensor
  // Se toman los MSB
  pinMode(dataPin, INPUT);
  pinMode(clockPin, OUTPUT);
  byte MSB = shiftIn(dataPin, clockPin, MSBFIRST);
  // Se envía el ACK necesario
  pinMode(dataPin, OUTPUT);
  digitalWrite(dataPin, HIGH);
  digitalWrite(dataPin, LOW);
  digitalWrite(clockPin, HIGH);
  digitalWrite(clockPin, LOW);
  // Se toman los LSB)
  pinMode(dataPin, INPUT);
  byte LSB = shiftIn(dataPin, clockPin, MSBFIRST);
  // Se devuelve la combinación de MSB y LSB
  return ((MSB << 8) | LSB); //combine bits
}
void SHT_skipCrc(int dataPin, int clockPin){
  // Se salta el CRC del sensor
  pinMode(dataPin, OUTPUT);
  pinMode(clockPin, OUTPUT);
  digitalWrite(dataPin, HIGH);
  digitalWrite(clockPin, HIGH);
  digitalWrite(clockPin, LOW);
}
```

El siguiente diagrama de flujo describe el protocolo de comunicación con el sensor en una iteración sin el modo de bajo consumo.



7.1.1.4. Sensor DHT11.

```
//////////////////////////////////////////////////////////////////////
//       Programa encargado de recoger los valores de temperatura     //
//       y humedad del sensor DHT11 y enviarlos al puerto serial      //
//      donde serán recogidos y enviados por el módulo inalámbrico.   //
//     Se activa el modo bajo consumo. La lectura de valores del      //
//      sensor se realiza mediante su librería de acceso público.     //
//////////////////////////////////////////////////////////////////////
```

```
// Se añade la libreria DHT del sensor
#include "DHT.h"
// Se define el pin donde se conecta el terminal de datos del sensor
#define DHTPIN 2
// S define el tipo de sensor de la librería, en este caso, el DHT11
#define DHTTYPE DHT11   // DHT 11
DHT dht(DHTPIN, DHTTYPE);
// Variable que lleva el tiempo en milisegundos desde que se inició el programa
unsigned long time;
void setup(){
  // Se inicializa el puerto serie a 9600 baudios
  Serial.begin(9600);
  // Se inicia el sensor
  dht.begin();
}
void loop(){
  // Almacena en unas variables los valores de temperatura y humedad relativa
  int h = dht.readHumidity();
  int t = dht.readTemperature();
  // Comprobamos si lo que devuelve el sensor es valido, si no son numeros algo
esta fallando
  if (isnan(t) || isnan(h)){
    // Comprueba que se estén leyendo números
    Serial.println("Fallo al leer del sensor DHT");
  }else{
    // Se imprimen en el puerto serie los valores leídos junto al tiempo
    // Se activa el modo de hibernación de pin para bajo consumo tras cada
envío
    pinMode(6,OUTPUT);
    digitalWrite(6,LOW);
    delay(10);
    Serial.print("DATA2");
    Serial.print(";");
    time = millis();
    Serial.print(time);
    Serial.print(";");
```

```
    Serial.print(t);
    Serial.print(";");
    Serial.print(h);
    Serial.print(";");
    Serial.print("\n");
    delay(10);
    digitalWrite(6,HIGH);
  }
  delay(5000);
}
```

7.1.2. Códigos Processing.

7.1.2.1. Lectura de puerto serie y almacenamiento.

```
////////////////////////////////////////////////////////////////
//     Programa encargado de leer y escribir en un fichero     //
//      los datos provenientes de los dos nodos sensores       //
//      a través del puerto serial en el nodo coordinador      //
////////////////////////////////////////////////////////////////
// Se importa la librería Serial de processing
import processing.serial.*;
Serial myPort;
// Carácter ASCII fin de línea 10 = salto de carro
int lf = 10;
// String de buffer para leer línea serial
String myString = null;
// Vector de strings donde se almacenarán los datos para guardar en un
archivo *.txt
String[] Archivo = new String[6000];
// Indice de adquisición
int vuelta = 0;
void setup() {
  // Se imprime la lista de puertos disponibles y se inicia uno a 9600
baudios, en este caso el COM0
  println(Serial.list());
```

```
  myPort = new Serial(this, Serial.list()[0], 9600);

  // Se lee la primera línea que llega al puerto serial hasta que haya un
salto de carro

  myString = myPort.readStringUntil(lf);

  myString = null;

}

void draw() {

    while (myPort.available() > 0) {

      // Se leen una a una las líneas que llega al puerto serial

      myString = myPort.readStringUntil(lf);

      if (myString != null) {

        println(vuelta);

        println("********************");

        // Convierte los datos del Serial en enteros

        int[] data=int(split(myString, ";"));

        // Almacena los datos en un vector de strings

        Archivo [vuelta] = myString;

        // Se guarda en el archivo de texto la línea leída

        saveStrings("DataJulen.txt", Archivo);

        // Se aumenta el índice de adquisición

        vuelta++;

      } //end else del proceso.

    }

}
```

7.1.3. Códigos Matlab.

7.1.3.1. Lectura de archivo y mostrado en pantalla.

```
function datosSerial()
    close all;
    clc;
    %% Abre un explorador para seleccionar el archivo fuente
    [NombreArchivo,DirArchivo] = uigetfile('*.txt','Seleccione un
archivo de texto');
    if(isequal(NombreArchivo,0))
        return;
    else
        Archivo = fullfile(DirArchivo,NombreArchivo);
```

```
    end
    %% Abre el archivo
    Dat = fopen(Archivo,'r');
    %% Creación de las dos figuras para humedades y temperaturas
    figure('Name','Sensor temperatura y humedad relativa');
    subplot(2,1,1)
    axis([0 450 -0.06 0.02])
    xlabel('Time(minutes)');ylabel('Temperature (°C)')
    title('Temperatura');
    subplot(2,1,2)
    title('Humedad Relativa');
    xlabel('Tiem (minutes)');
    ylabel('HR(%)');
    grid on;
    hold on;
    %% Variable que lleva el índice de muestras
    i=1;
    %% Lee el archivo mientras haya datos
    while ~feof(Dat)
        leer_linea = fgetl(Dat);
        %% Separa las variables de tiempo, T y HR
            lect_v= strsplit(leer_linea, ';');
            %% Distingue si pertenece a SHT15
            if strcmp('DATA1',lect_v(1))
                %% Coge las 3 variables y las almacena en un vector
que se dibuja
                time1(i)=str2double(lect_v(2))/1000;
                T1(i)=str2double(lect_v(3));
                HR1(i)=str2double(lect_v(4));
                subplot(2,1,1)
                plot (time1, T1, '.b')
                subplot(2,1,2)
                plot (time1, HR1, '.b')
            %% Distingue si pertenece a DHT11
            elseif strcmp('DATA2',lect_v(1))
                 %% Coge las 3 variables y las almacena en un vector
que se dibuja
                time2(i)=str2double(lect_v(2))/1000;
                T2(i)=str2double(lect_v(3));
                HR2(i)=str2double(lect_v(4));
                subplot(2,1,1)
                plot (time2, T2, '.k')
                subplot(2,1,2)
                plot (time2, HR2, '.k')
            end
            %% Aumenta el índice de muestra
            i=i+1;
    end
    %% Cierra el fichero fuente
    fclose(Dat);
    clear all;
end
```

7.2. Anexo II. Hojas de características de fabricantes.

DHT11: http://www.micro4you.com/files/sensor/DHT11.pdf

SHT25:http://www.sensirion.com/fileadmin/user_upload/customers/sensirion/Dokumente/Humidity/Sensirion_Humidity_SHT25_Datasheet_V2.pdf

SHT1X:http://www.sensirion.com/fileadmin/user_upload/customers/sensirion/Dokumente/Humidity/Sensirion_Humidity_SHT1x_Datasheet_V5.pdf

AVR: http://www.atmel.com/Images/doc8161.pdf

ZMN2400: http://www.datasheetarchive.com/dlmain/SFDatasheet-3/sf-00061089.pdf

Energizer 522: http://data.energizer.com/PDFs/522.pdf

AMS117 Regulator: http://www.advanced-monolithic.com/pdf/ds1117.pdf

NTC NJ28RA0104F: http://html.alldatasheet.es/html-pdf/515754/AVX/NJ28RA0104F/122/2/NJ28RA0104F.html