Universidad de Oviedo

ESCUELA POLITÉCNICA DE INGENIERÍA DE GIJÓN.

MÁSTER UNIVERSITARIO EN INGENIERÍA DE TELECOMUNICACIÓN

ÁREA DE PROYECTOS DE INGENIERÍA

TRABAJO FIN DE MÁSTER Nº 201809

TECNOLOGÍAS DE REDES IOT APLICABLES A PROCESOS INDUSTRIALES

GONZÁLEZ DÍAZ, Carlos
TUTOR: RODRÍGUEZ MONTEQUÍN, Vicente
GARCÍA CORBATO, Andrés

FECHA: julio de 2018

Agradecimientos

Me gustaría agradecer a Andrés García Corbato y a Daniel González Castro por sus ideas, colaboración y búsqueda de alternativas durante el desarrollo de este proyecto. A Vicente Rodríguez Montequín por su completa disponibilidad, disposición y ayuda con el mismo.

También me gustaría agradecer a toda la sala B302 de *ArcelorMittal I+D*® por lograr que el entorno y ambiente de trabajo donde se llevó a cabo el proyecto fuese inmejorable.

Por último me gustaría dar las gracias a mi familia y amigos por estar siempre apoyándome y motivándome para conseguir mis metas.

Índice

1.- Introducción 1

1.1.- Resumen 1

1.2.- Objetivos y descripción 1

1.3.- Motivación 2

1.3.1.- Historia y concepto del Internet de las Cosas 3

1.3.2.- Principales áreas del IoT 5

1.3.2.1.- Automatización industrial 7

1.3.2.2.- Industria 4.0 8

1.3.3.- *Industrial Internet of Things* (IIoT) 11

1.3.4.- Aplicaciones de redes inalámbricas 12

1.3.4.1.- Topologías de red 12

1.3.4.2.- ZigBee (DigiMesh) 13

1.3.4.3.- Bluetooth Low Energy (BLE) 16

1.3.4.4.- 6lowPAN 16

1.3.4.5.- WiFi 17

1.3.4.6.- GSM 18

2.- Plataformas de hardware existentes 20

2.1.- SigFox 20

2.1.1.- Conectividad y velocidad 20

2.1.2.- Tecnología 21

2.2.- WirelessHART 22

2.2.1.- Tecnología 22

2.2.2.- Dispositivos y aplicaciones 23

2.3.- LORA 24

2.3.1.- Tecnología 24

2.3.2.- Dispositivos 25

2.4.- WIMON 100 SERIES 26

2.5.- WISE 28

2.6.- LIBELIUM 29
3.- Diseño y desarrollo de una red ADHOC 31
3.1.- Requisitos de la red 31
3.2.- Elección de componentes 32
3.2.1.- ConnectPort X4IA 32
3.2.2.- XBee SX868 34
3.2.3.- Arduino NANO 37
3.2.4.- Sensor RGB 39
3.2.4.1.- Funcionamiento TCS34725 39
3.2.5.- *KEPServerEX* 40
3.3.- Configuración ConnectPort X4IA 41
3.3.1.- *Home* 41
3.3.2.- *Configuration* 42
3.3.3.- *Applications* 43
3.4.- Configuración XBee SX868 44
3.4.1.- API frame 49
3.5.- Configuración Arduino NANO y sensor RGB 50
3.6.- Configuración *KEPServerEX* 52
3.6.1.- *Wireshark* 53
3.6.2.- *KEPServerEX* 54
4.- Diseño de un prototipo 61
4.1.- PCB: diseño y construcción 61
4.1.1.- PCB alimentada mediante toma de red 61
4.1.2.- PCB alimentada mediante batería 64
4.2.- Carcasa impresa en 3D: diseño y fabricación 67
5.- Pruebas Realizadas 70
5.1.- Comunicación modo API 70
5.2.- Comunicación modo AT 74
5.3.- Pruebas en un entorno industrial 76
6.- Balance de blancos 80
6.1.- Métodos de balance de blancos 80
6.1.1.- Auto White Balance (AWB) 81

6.1.2.- Gray Card (Tarjeta Gris) 82
6.1.3.- Kelvin 83
6.1.4.- ExpoDisc 84
6.1.5.- Hoja blanca de papel 84
6.2.- Exposímetro 85
6.2.1.- Exposímetro de luz reflejada 85
6.2.2.- Exposímetro de luz incidente 86
7.- Industrialización del prototipo 87
8.- Conclusiones y líneas de trabajo futuras 89
9.- Anexos 90
9.1.- Modbus 90
9.2.- Arduino Genuino: código 92
9.3.- ConnectPort X4IA: código 93
9.3.1.- Config_cpx4.py 93
9.3.2.- 868_TCP_send.py 94
10.- Planificación 101
10.1.- Tareas y Diagrama de Gantt 101
11.- Referencias 103

Índice de figuras

Figura 1.1.- Diagrama general de bloques del proyecto. 2
Figura 1.2.- Cita de Nikola Tesla sobre el IoT [5]. 3
Figura 1.3.- Descripción gráfica del Internet de las Cosas [2]. 4
Figura 1.4.- Principales compañías centradas en cada área IoT [4]. 6
Figura 1.5.- Conjunto de fases de un proyecto de automatización [8]. 8
Figura 1.6.- Revoluciones industriales a lo largo de la historia [7]. 9
Figura 1.7.- Topologías de red existentes [9]. 12
Figura 1.8.- Elementos de una red ZigBee dentro de diferentes topologías [10].. 14
Figura 1.9.- Protocolos situados en las distintas capas del modelo ISO/OSI [12].17
Figura 1.10.- Arquitectura GSM. 19
Figura 2.1.- Red SigFox [16]. 21
Figura 2.2.- Infraestructura de una red *WirelessHART* [26]. 23
Figura 2.3.- Ejemplo de arquitectura de red WIMON 100 [28]. 27
Figura 2.4.- Características de una red *WISE 4000* [23]. 29
Figura 2.5.- Esquema general de una red *LIBELIUM* [27]. 29
Figura 3.1.- ConnectPort X4IA. 34
Figura 3.2.- XBee SX868. 34
Figura 3.3.- Placa de desarrollo XBee SX868. 36
Figura 3.4.- Antena XBee SX868. 36
Figura 3.5.- Arduino NANO. 38
Figura 3.6.- Sensor RGB TCS34725. 39
Figura 3.7.-Esquemático sensor RGB TCS34725 [34]. 40
Figura 3.8.-Interfaz ConnectPort X4IA. 41
Figura 3.9.-"*Network*". 42
Figura 3.10.- "*XBee Network*". 43
Figura 3.11.- "*Python*". 44
Figura 3.12.- Interfaz gráfica software *XCTU*. 44
Figura 3.13.- Búsqueda automática de dispositivos en XCTU. (a) Selección del puerto, (b) características del dispositivo y (c) dispositivos descubiertos. 45

Figura 3.14.- Módulos que componen la red DigiMesh. 46

Figura 3.15.- Conexión entre los módulos (red DigiMesh). 47

Figura 3.16.- Configuración de un módulo XBee (XCTU)(I). 47

Figura 3.17.- Configuración de un módulo XBee (XCTU)(II). 48

Figura 3.18.- Trama modo API. 49

Figura 3.19.- Conexionado Arduino NANO-sensor RGB. (a) Conexionado desde el sensor, (b) conexionado desde el Arduino. 50

Figura 3.20.- Montaje en protoboard de Arduino NANO y sensor RGB. 51

Figura 3.21.- Interfaz gráfica *Arduino Genuino* 51

Figura 3.22.- Datos proporcionados por el sensor RGB TCS34725. 52

Figura 3.23.- Visualización de los datos del sensor RGB TCS34725. 52

Figura 3.24.- Interfaz principal Wireshark. 53

Figura 3.25.- Análisis de tramas en *Wireshark.* 54

Figura 3.26.- Creación de un canal (I). 55

Figura 3.27.- Creación de un canal (II). 55

Figura 3.28.- Creación de un canal (III). 56

Figura 3.29.- Creación de un dispositivo (I). 57

Figura 3.30.- Creación de un dispositivo (II). 57

Figura 3.31.- Creación de un dispositivo (III). 58

Figura 3.32.- Configuración de un dispositivo (I). 58

Figura 3.33.- Configuración de un dispositivo (II). 59

Figura 3.34.- Configuración de un dispositivo (III). 59

Figura 3.35.- Configuración de un dispositivo (IV). 60

Figura 4.1.- Esquemático PCB alimentada mediante toma de red. 62

Figura 4.2.- Diseño PCB alimentada mediante toma de red. 63

Figura 4.3.- PCB final. Capa "*TOP*" arriba y capa "*BOTTOM*" abajo. 64

Figura 4.4.- Esquemático PCB alimentada mediante batería. 65

Figura 4.5.- Diseño PCB alimentada mediante batería. 66

Figura 4.6.- Lumisfera 401-821 de *Sekonic*®. 67

Figura 4.7.- Tapa diseñada en 3D (renderizada). 68

Figura 4.8.- Carcasa diseñada en 3D (renderizada). 68

Figura 4.9.- Carcasa impresa en 3D. 69

Figura 5.1.- Configuración de mensaje enviado. 71

Figura 5.2.- Configuración PuTTy. 71

Figura 5.3.- Mensajes recibidos ConnectPortX4 (PuTTy). 72

Figura 5.4.- Configuración *KEPServerEX* para dos módulos XBee. 73

Figura 5.5.- Mensaje recibido desde XBeeB en *KEPServerEX*. 73

Figura 5.6.- Mensajes recibidos desde XBeeC en *KEPServerEX*. 74

Figura 5.7.- Mensajes recibidos ConnectPortX4 (PuTTy). 74

Figura 5.8.- Mensajes recibidos servidor *KEPServerEX*. 75

Figura 5.9.- Equipo de trabajo. 76

Figura 5.10.- Estudio sobre el nivel de señal recibida en módulos XBee en interiores [42]……..... 79

Figura 6.1.- Tarjeta gris [38]. 82

Figura 6.2.- Efecto de la variación de la temperatura de color [38]. 83

Figura 6.3.- Comparativa de los distintos métodos de balance de blancos [38]. .. 84

Figura 6.4.- Ejemplo de exposímetro [17]. 85

Figura 8.1.- Configuración manager en DigiEsp. 90

Figura 8.2.- Configuración Modbus en ConnectPort X4IA. 92

Figura 9.1.- Diagrama de Gantt del presente proyecto. 102

Índice de tablas

Tabla 1.1.- Ventajas y desventajas ZigBee. 15

Tabla 1.2.- Características WiFi [13]. 18

Tabla 2.1.- Características SigFox. 22

Tabla 2.2.- Características LoRaWAN. 25

Tabla 2.3.- Características *WIMON 100*. 27

Tabla 2.4.- Características WISE 4000. 28

Tabla 2.5.- Características sensores LIBELIUM. 30

Tabla 3.1.- Especificaciones técnicas ConnectPort X4IA [30]. 33

Tabla 3.2.- Especificaciones técnicas *XBee SX868* [31]. 35

Tabla 3.3.- Comparativa entre Raspberry Pi y Arduino NANO [32]. 37

Tabla 3.4.- Especificaciones técnicas *Arduino NANO* [33]. 38

Tabla 3.5.- Especificaciones técnicas *KEPServerEX* [35]. 40

Tabla 4.1.- PCB alimentada mediante toma de red: componentes. 61

Tabla 4.2.- PCB alimentada mediante batería: componentes. 65

Tabla 5.1.- Nivel de señal recibida por el ConnectPort X4IA en función de la potencia transmitida por el XBeeB. 77

Tabla 5.2.- Nivel de señal recibida por el XBeeB en función de la potencia transmitida por el XBeeA a una distancia entre ambos de 9m. 78

Tabla 5.3.- Nivel de señal recibida por el XBeeB en función de la potencia transmitida por el XBeeA a una distancia entre ambos de 12m. 78

Tabla 5.4.- Nivel de señal recibida por el XBeeB en función de la potencia transmitida por el XBeeA a una distancia entre ambos de 25m. 78

Tabla 6.1.- Temperaturas de color típicas. 81

1.- INTRODUCCIÓN

El presente proyecto se basa en el diseño, configuración y construcción de parte de una red tipo malla DigiMesh en 868MHz que se encargue de captar datos a través de un determinado tipo de sensor y los envíe hacia un servidor donde puedan ser visualizados. No se ha realizado el desarrollo completo de la red ya que la parte relacionada con actuadores o procesado de datos en la nube se salía del ámbito de este proyecto. Para lograr este fin se ha realizado un estudio sobre los dispositivos que mejor se adecuaban a las características buscadas, y sobre la configuración y programación óptima para posibilitar la comunicación entre módulos XBee.

Ha sido desarrollado en su totalidad en la empresa multinacional *ArcelorMittal*®. Concretamente en la parte dedicada a I+D en el departamento *Digital Factory* de Avilés, Asturias.

1.1.- Resumen

El proyecto ha consistido en el estudio del estado del arte de las redes inalámbricas existentes basadas en Internet de las Cosas, además se han elegido los componentes que componen la red a diseñar y se han realizado las configuraciones oportunas. Para posibilitar la comunicación entre elementos y con la finalidad de ver los datos deseados en un servidor se ha desarrollado código en el lenguaje *Python* (basado en *sockets*).

Se ha diseñado un prototipo que será uno de los elementos principales de la red ya que basará su funcionamiento en un sensor RGB, un Arduino y un módulo XBee. Este se ha realizado en una placa de circuito impreso a la que se le ha añadido una carcasa diseñada e impresa en 3D.

Para finalizar se han realizado una serie de pruebas para comprobar que la comunicación funcionaba de manera correcta.

1.2.- Objetivos y descripción

El sistema implementado constará de tres partes principales. La primera de ellas sería la correspondiente al nodo principal que será el encargado de captar los datos y de realizar el procesado de datos con la finalidad de enviarlos con la apariencia final buscada. La segunda parte es la que estaría compuesta del resto de módulos XBee (red tipo malla) y de una *Gateway* que sería la encargada de captar los datos enviados por dichos módulos y de enviarlos hacia el servidor. Por último, la tercera parte es aquella que está formada por un servidor y en este, será donde se podrán visualizar los datos que se han enviado desde el sensor y tomar a partir de los mismos las acciones que se estimen oportunas.

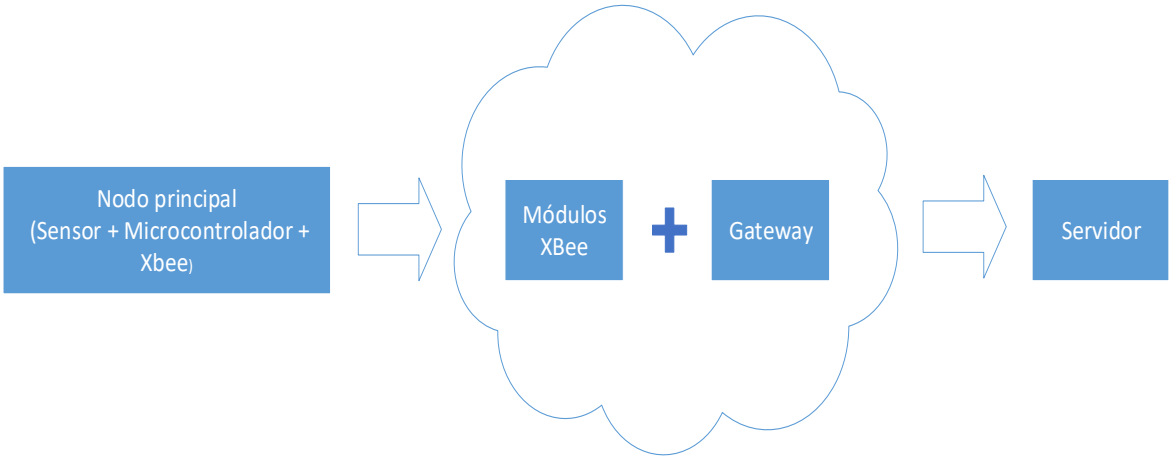


Figura 1.1.- Diagrama general de bloques del proyecto.

La idea principal es la de implementar un sistema de comunicaciones que permita el envío correcto de datos. Como se explicará y justificará en apartados posteriores de este documento se ha decidido realizar la comunicación con el protocolo DigiMesh (frente a otras alternativas como pudiera ser ZigBee), además, el punto crítico de la misma que sería la recepción de los datos provenientes de la malla y el posterior reenvío hacia el servidor se ha decidido realizar mediante *sockets* TCP/IP.

Una vez finalizada el diseño y la construcción de parte de la red se ha realizado el diseño y la construcción de un prototipo del nodo principal del sistema. Como culmen del proyecto se han realizado pruebas para comprobar el correcto funcionamiento de la misma.

1.3.- Motivación

El presente proyecto se ha desarrollado en el entorno industrial de la empresa *ArcelorMittal*®, debido al interés de la misma en fortalecer y aumentar sus posibilidades en el mundo de las comunicaciones inalámbricas (IoT). Una de las principales actividades es la de posibilitar el envío de datos (generados por sensores) hacia una sala completamente habilitada para poder monitorizarlos y tomar decisiones en función de los mismos.

Para realizar la comunicación entre el sensor y el servidor final, la empresa se encuentra en la actualidad estudiando diversas tecnologías y topologías de red que se puedan incorporar a un sector como es el del acero. Todo ello considerando y teniendo en cuenta que tipo de medidas se realizarían y en que localización (interiores o exteriores).

Hay otras áreas del Internet de las Cosas que están siendo trabajadas en la actualidad en *ArcelorMittal*®. Una de ellas es el estudio y procesamiento de datos (*Big Data*) en la nube y en servidores propios de la compañía que permiten obtener estimaciones estadísticas

para así prevenir posibles fallos o simplemente para mejorar el rendimiento en determinados procesos.

1.3.1.- Historia y concepto del Internet de las Cosas

El Internet de las Cosas [5] no es un concepto reciente y novedoso. El origen de los objetos conectados no es algo de hace pocas décadas, en realidad se remonta hasta los inicios tecnológicos del siglo XIX, en lo que se consideran los primeros experimentos de telemetría de la historia. El primero del que se tiene constancia fue el llevado a cabo en 1874 por científicos franceses. Estos instalaron dispositivos de información meteorológica y de profundidad de nieve en la cima del *Mont Blanc*. A través de un enlace de radio de onda corta, los datos eran transmitidos a París.

En 1926, Nikola Tesla en una entrevista a la revista "*Colliers"* anticipó lo que en la actualidad se entiende como conectividad a nivel global y la miniaturización tecnológica (ver Figura 1.2).

"Cuando lo inalámbrico esté perfectamente desarrollado, el planeta entero se convertirá en un gran cerebro, que de hecho ya lo es, con todas las cosas siendo partículas de un todo real y rítmico… y los instrumentos que usaremos para ellos serán increíblemente sencillos comparados con nuestros teléfonos actuales. Un hombre podrá llevar uno en su bolsillo"

Figura 1.2.- Cita de Nikola Tesla sobre el IoT [5].

La primera conexión entre dos dispositivos data de 1969 con la red ARPANET, creada por el departamento de Defensa de Estados Unidos. Por aquel entonces este tipo de comunicación necesitaba de personas que enviasen la información de un lado a otro y que en el lado de la recepción se procesasen los datos y se enviase una respuesta.

Posteriormente se implantó el protocolo TCP/IP y en 1990 se implementó la primera comunicación entre un cliente HTTP y un servidor en Internet. Un año después se creó la primera página web y seguidamente todo fue evolucionando hasta lo que conocemos hoy en día. También en 1990, Mark Weiser obtuvo el reconocimiento mundial gracias a su trabajo "*The Computer for the Twenty-First Century*" donde, por primera vez alguien hablaba sobre interconectar elementos que fuesen ordenadores.

En la actualidad el Internet de las Cosas es un concepto que está en constante crecimiento [2]. No obstante, hay que remontarse al año 1999 para encontrar el origen de este término, fue en el Instituto Tecnológico de Massachusetts (MIT) y su autor fue Kevin Ashton en el campo de la identificación por radiofrecuencia en red (RFID) y tecnologías de sensores.

Como concepto se refiere a la interconexión de digital de objetos cotidianos (aunque actualmente pueden ser de cualquier tipo) con Internet. El IoT debería de codificar de 50 a

100.000 millones de objetos y monitorizar el estado de los mismos y se estima que para el año 2020 ya haya unos 30.000 millones de dispositivos inalámbricos conectados a Internet.

Una de las principales razones por las que esto será posible será el aumento del denominado como Internet de Nueva Generación que permitirá entre otras cosas la implementación de una manera definitiva del protocolo IPv6 (siendo relegado IPv4). El principal problema a día de hoy es que el actual protocolo de direccionamiento es IPv4, lo cual limita mucho el número de asignaciones y que para solventar este problema de cantidad de direcciones se usan protocolos como puede ser NAT. Este permite que a una determinada red privada (compuesta de numerosos equipos que necesitan IP) se pueda direccionar hacia el exterior a partir de una sola IP pública.

El problema se solucionará cuando la principal forma de direccionamiento pase a ser IPv6 ya que se proporcionará una capacidad de 2^{128} direcciones de hosts diferentes lo cual hace muy complicado que se vuelva a tener el problema actual de IPv4.

Figura 1.3.- Descripción gráfica del Internet de las Cosas [2].

En resumen, el Internet de las Cosas [1] es una situación en la que cada objeto, animal y persona cuenta o contaría con un identificador único (UID) y puede transmitir datos en red sin necesidad de interacción alguna de humanos o máquinas. Según los expertos existirán al menos 5 niveles de inteligencia:

- Nivel 1: Identidad. El objeto será capaz de identificarse de manera única

- Nivel 2: Ubicación. Se podrá saber dónde está dicho objeto o dónde ha estado.

- Nivel 3: Estado. Será capaz de comunicar el estado en que se encuentra así como sus características.

- Nivel 4: Contexto. El objeto será capaz de percibir el entorno en que se encuentra.

- Nivel 5: Criterio. El objeto se comunica, se identifica, se ubica, analiza su entorno, decide y ejecuta en función de su criterio.

1.3.2.- Principales áreas del IoT

En la actualidad hay varias áreas en donde el IoT está siendo explotado y desarrollado. Estas áreas son: medios y publicidad, medio ambiente, industria e infraestructura, manufactura, energía, medicina y cuidado de la salud y logística y transporte [3].

Medios y publicidad

El IoT junto con el *Big Data* revolucionará la industria de los medios de comunicación ya que debido al gran procesamiento de información se pretende medir a los consumidores de manera que se pueda estimar y analizar la variedad creciente de estadísticas de comportamiento.

Medio ambiente

El uso de IoT en el mundo de la naturaleza permite la protección de la misma de distintas maneras. Entre ellas destaca la capacidad de medir la calidad del aire, la calidad del agua, las condiciones atmosféricas o del suelo, además de permitir optimizar recursos para generar prevención. También se puede usar para prevenir desastres naturales como tsunamis o terremotos.

Industria e infraestructura

Quizás la parte que más está en auge en la actualidad debido sobre todo a la inversión de las grandes empresas multinacionales en investigación y desarrollo. Se suele enfocar en su mayoría al campo de mejorar la eficiencia en los procesos, aumentar la calidad del servicio/producto y disminuir los costes de operación.

Un buen ejemplo de este campo (y del aumento del IoT) es la empresa *Michelin*®. Hasta hace 10 años, era una empresa dedicada a la venta de llantas. Sin embargo, aprovechó el auge de lo digital para realizar labores de desarrollo tecnológico e incluir sensores en las llantas de los camiones para que pudiesen enviar información vía SMS o un correo electrónico con el estado de los neumáticos, además de generar información que permitía ahorrar gasolina o agilizar rutas.

Manufactura.

El Internet de la Cosas permitirá establecer un control y gestión sobre la fabricación de equipos, manejo de los activos y el control de procesos de fabricación.

Energía

El IoT podrá usarse como complemento en los sistemas de detección y actuación conectados a Internet; además de optimizar y medir el consumo de energía en distintos dispositivos.

Medicina y cuidado de la salud

Dispositivos que se usarán para habilitar sistemas de notificación de emergencia y vigilancia remota. Sensores especializados que pueden ser equipados para supervisar la salud y el bienestar de las personas.

Logística y transporte

El IoT complementará las comunicaciones, control y tratamiento de la información a través de varios sistemas de transporte.

Se impulsa una interacción dinámica de control de tráfico inteligente, elección del estacionamiento, sistemas de cobro electrónico de peajes, gestión de logística y flota, control de vehículo, asistencia vial y seguridad.

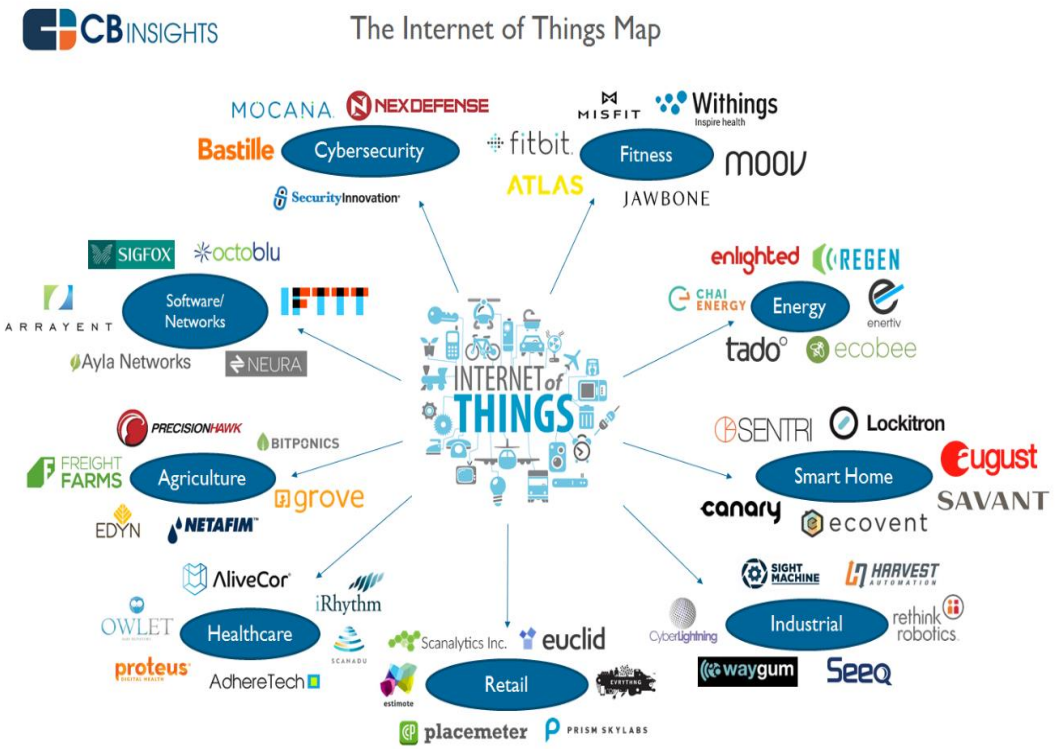


Figura 1.4.- Principales compañías centradas en cada área IoT [4].

1.3.2.1.- Automatización industrial

El proyecto realizado se centra dentro del área de "Industria e infraestructura" mencionada en párrafos anteriores. Por ello a modo introductorio se va a mencionar en que consiste la automatización industrial y las fases que se deben de seguir para que se pueda realizar de una manera correcta.

Se define la automática [8] como el conjunto de métodos y procedimientos que permiten la sustitución del operario en tareas físicas y mentales. De esta definición original se desprende la definición de la automatización como la aplicación de la automática al control de procesos industriales.

Un proceso [8] es aquella parte del sistema que a partir de la entrada del material, energía e información se genera una transformación sujeta a perturbaciones. En la industria, los procesos se dividen en procesos continuos (salida en forma continua), discretos (salida en forma de unidades) y *batch* (salida en forma de lotes).

En lo industrial [8], se enfatiza el control secuencial y la regulación continua. El control secuencial usa estados y transiciones en una secuencia ordenada, por otro lado, la regulación continua se basa en la acción de control proporcional, derivativo o integral respecto al error para conseguir así una regulación adecuada de la variable.

Fases de un proyecto de automatización industrial

Las fases de las que consta un proyecto son: automatización, supervisión, interacción, implementación y pruebas (ver Figura 1.5). Todas ellas deben de estar supervisadas por un operario.

Automatización: formado por los pasos de observación del proceso a controlar y generación de estados y transiciones, selección del automatismo, selección y cableado físico de sensores y actuadores y generación de un segundo gráfico de segundo nivel en su descripción tecnológica.

Supervisión: reúne información sobre los estados posibles que se han definido en la fase anterior, define los módulos a utilizar y además los representa mediante estados y transiciones. Para cada módulo se debe establecer la relación con el resto y estar atento a posibles fallos.

Interacción: se identifica cuando se realiza una acción concreta y sobre el panel de mando a que elementos y dispositivos se puede acceder. Los dispositivos son aquellos que se han definido en la fase de supervisión mediante módulos.

Implementación: elección del lenguaje de programación del automatismo y de la traducción de los gráficos realizados en dicho lenguaje.

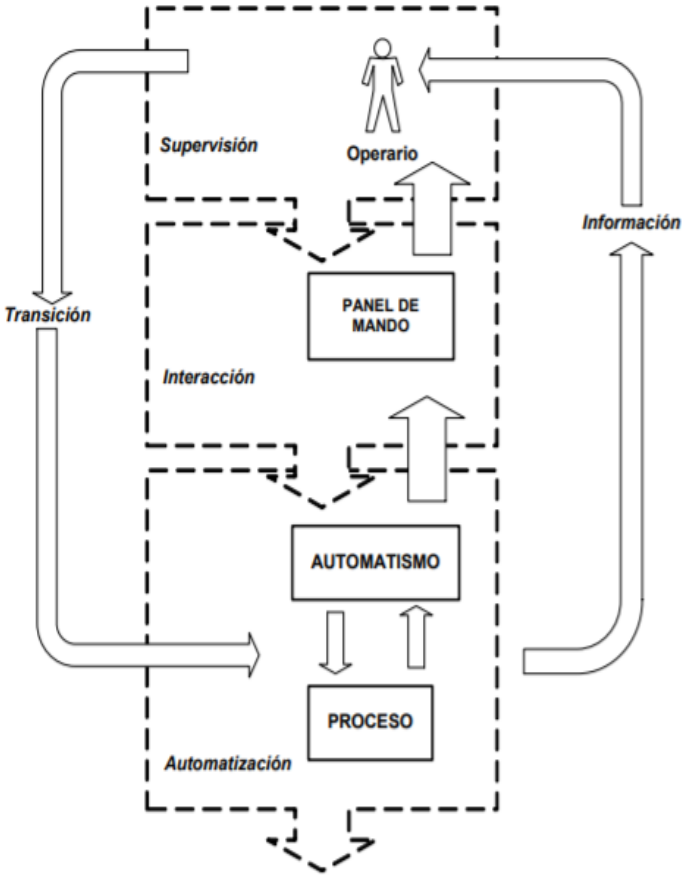


Figura 1.5.- Conjunto de fases de un proyecto de automatización [8].

1.3.2.2.- Industria 4.0

El término Industria 4.0 [6] se refiere a la cuarta revolución industrial, impulsada por la transformación digital, y significa un salto cualitativo en la organización y gestión de la cadena de valor del sector. La aparición de las tecnologías digitales permite la unión entre el mundo físico y el digital, es decir, se posibilita la vinculación del mundo físico al digital. Esta conexión habilita que dispositivos y sistemas colaboren entre ellos y con otros para crear una industria inteligente.

La evolución hacia la era digital proporciona beneficios a distintos niveles, entre los que destacan el nivel de proceso, negocio o producto.

Figura 1.6.- Revoluciones industriales a lo largo de la historia [7].

El proceso se basa en siete aspectos clave: estándares clave, mejora en las decisiones, empoderamiento de las personas, ciberseguridad, nueva generación de sensores, máquinas valoradas como cibersistemas físicos y *Cloud Computing* y *Big Data*.

Estándares clave

Un aspecto principal es el de desarrollar estándares que permitan desarrollar la interoperabilidad para interconectar diferentes tipos de dispositivos. Dicha capacidad debe de ser sencilla y práctica.

El MTConnect [6] es un estándar que posee un grupo de expresiones predefinidas, con las cuales las máquinas-herramienta pueden comunicarse en un lenguaje común interpretable por determinadas aplicaciones. Está construido en la tecnología estándar de Internet (HTTP, TCP/IP, XML y Ethernet) y permite que las características principales (arquitectura abierta, protocolo abierto, datos abiertos…) que son críticas a los sistemas sean compatibles con el IoT Industrial.

OPC-UA [6] es otro estándar que se basa en la "vinculación e inclusión de objetos para control de proceso – arquitectura unida". Protocolo que permite que los sistemas de automatización se entiendan entre sí.

Mejora en las decisiones

La gran cantidad de datos que permite obtener el IoT facilita conseguir datos estadísticos a partir de los cuales tomar decisiones lo más precisas y oportunas. De esta manera se pueden alcanzar los objetivos estratégicos de una manera más sencilla.

Empoderamiento de las personas

A pesar de que se pudiese pensar que el IoT podría suponer que las personas tuviesen un papel menos importante en la industria esto no es así, sino todo lo contrario, ya que las implicaciones, deberes y responsabilidades aumentarán en proporción al aumento del Internet de las Cosas.

Ciberseguridad

Conectar máquinas-herramienta a una aplicación basada en la red o en la nube crea un gran número de vulnerabilidades.

Nueva generación de sensores

Los sensores básicos detectan y miden las características físicas o las condiciones de un dispositivo, mientras que los sensores inteligentes desarrollan otras funciones como la de convertir lecturas analógicas a un formato digital o la de transmitir estos datos hacia una aplicación software entre otras.

Los sensores proveen información sobre cambios significativos. Una gran prioridad es poner los sensores donde puedan detectar y reportar automáticamente los cambios que afectan la viabilidad del componente. Esta información proveerá alertas de fallos inminentes y hace que el tiempo de inactividad no esperada ocurra excepcionalmente. Los sensores son claves para la detección, intervención y prevención.

Máquinas valoradas como cibersistemas físicos

Un cibersistema es un sistema en el cual ordenadores embebidos monitorean y controlan procesos físicos a través de un lazo de realimentación en un ambiente en red.

El aspecto más útil del cibersistema es el lazo de control, ya que crear varios en distintos niveles permite aumentar el IoT Industrial ya que varios procesos podrían estar controlados por un único operario.

Cloud Computing* y *Big Data

La computación en la nube (*Cloud Computing*) significa que una aplicación está siendo ejecutada en un ordenador remoto en lugar de uno que se encuentre en el mismo lugar físico. Los usuarios interactúan con la aplicación en la nube a través de una red, generalmente Internet.

El almacenamiento remoto de datos también puede estar basado en la nube. Generalmente, la capacidad de la nube para almacenar y procesar datos es casi ilimitada. Además el hecho de realizar estas acciones de manera remota es más económico, flexible y seguro que otras alternativas.

Big Data es la capacidad de analizar grandes cantidades de datos con la finalidad de obtener patrones estadísticos realizados para ser capaces de visualizar tendencias significativas o emergentes. La capacidad de recolectar y acceder a enormes masas de datos generados en un entorno industrial es lo que aporta el *Big Data* a la empresa actual.

1.3.3.- *Industrial Internet of Things* (IIoT)

El IIoT [43] [44] se suele presentar como una revolución que modifica la manera de visualizar y entender la industria. No obstante no se trata de una revolución sino de una evolución constante que tiene sus orígenes en las tecnologías y funcionalidades desarrolladas por las empresas hace 15 años. Como se trata de un proceso que está todavía en constante progreso, se estima que el IIoT alcance su punto máximo para el año 2030.

El Internet Industrial de las Cosas no se basa en desmantelar los sistemas de automatización actuales y sustituirlos por otros nuevos. Su potencial radica en la capacidad de vincular los sistemas de automatización, la planificación empresarial, la programación y los sistemas de vida del producto.

IIoT permite el paso en el mundo de la industria hacia modelos más flexibles accediendo así a sistemas móviles basados en estándares de Internet. Además de la generación y recopilación de datos se debe de prestar atención a otro tipo de detalles como son la creación de un sistema avanzado de medición (donde el dispositivo que recoge los datos hace un procesado sobre los mismos antes de enviarlos), una *Gateway* (con la capacidad de agregar datos, mostrar información, configurar dispositivos de la red, enviar datos hacia el servidor correcto en el momento adecuado, etc…), uso de aplicaciones o servicios (que permitan analizar en detalle los datos recibidos y tomar decisiones en función de los mismos) y un entorno de desarrollo abierto (que permita desarrollar una plataforma común).

Se deberán de mejorar varias áreas antes de que el IIoT pueda incorporarse en su totalidad en una industria y no solamente en determinadas funciones dentro de la misma. Algunas de estas son: la consolidación de más estándares en el entorno IIoT, la protección de los datos (ciberseguridad), la adaptación de los técnicos al nuevo conjunto de habilidades, etc…

Algunas de las principales características del IIoT son:

Resistencia

Los dispositivos se encuentran situados normalmente en ámbitos donde soportan condiciones extremas tales como temperatura, fuertes niveles de presión, humedad… Por tanto deben de poseer la capacidad de ser inmunes a los factores externos que pueden provocar el fallo en la red.

Escalabilidad

Los sensores usados en las redes industriales generan una gran cantidad de datos dentro de los cuales hay mucha variedad de variables, para evitar la pérdida de los mismos y posibilitar la adición de otros módulos se necesita de una red que permita el aumento de dispositivos sin la necesidad de realizar una reconfiguración completa de la misma.

Accesibilidad y mantenimiento

En ocasiones, los dispositivos encargados de captar los datos del IIoT tienden a estar situados en lugares donde la accesibilidad o el mantenimiento de los mismos implican una gran dificultad y por tanto un gran desembolso económico para la empresa. Por ello es necesario dotar a aquellos elementos que lo requieran de una gran autonomía (hay otros que no requerirán de esta característica ya que por ejemplo pueden estar alimentados desde una toma de red) y de un sistema de control y vigilancia que detecte errores en la red.

Seguridad y fiabilidad

Los dispositivos del Internet Industrial de las Cosas necesitan de un mayor nivel de seguridad que aquellos usados para ámbitos en los que un fallo no suponga ningún tipo peligro. En el ámbito industrial es distinto ya que un error podría provocar la fatalidad en una planta o instalación de la empresa o incluso en áreas cercanas que podrían estar habitadas.

1.3.4.- Aplicaciones de redes inalámbricas

En este apartado se mencionaran los distintos tipos de redes y los protocolos y topologías de red más habituales a día de hoy en el mundo del Internet de las Cosas. Se comentarán entre otras cosas la utilidad de los mismos, sus características, ventajas y desventajas, etc…

1.3.4.1.- Topologías de red

A modo introductorio se realiza un breve estudio sobre las distintas topologías que puede tener hoy en día las redes y los protocolos de los que se hablará más adelante. Las topologías son las siguientes [9]:

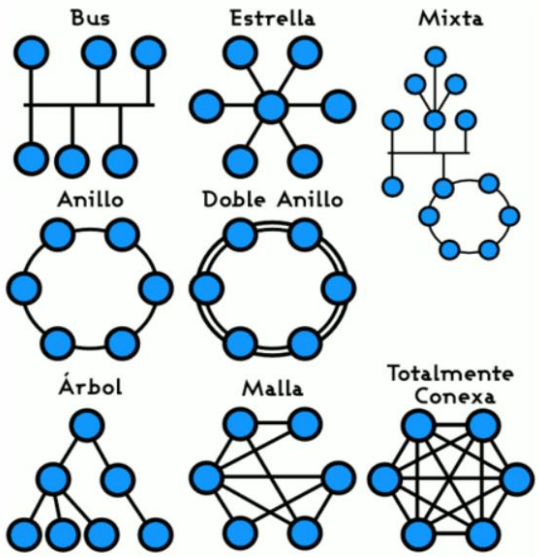


Figura 1.7.- Topologías de red existentes [9].

Punto a punto

Las redes punto a punto son aquellas en las que el canal está compuesto únicamente por dos tipos de nodos. En ella ambos poseen los mismos roles, es decir, se pueden comportar como emisor y como receptor.

Estrella

Una red en estrella es aquella que está formada por un gran número de dispositivos y todos ellos se conectan a un punto central que es el encargado de posibilitar la comunicación. Este nodo puede ser denominado conmutador, repetidor o concentrador. Los elementos de la red no están conectados entre sí de manera directa y el nodo principal suele estar provisto de los medios necesarios para evitar problemas relacionados con el eco.

Su uso más típico es el de las redes WAN (*Wide Area Network*).

Malla

Una red en malla es una topología de red en la que cada nodo está conectado a todos los nodos. De esta manera es posible llevar los mensajes de un nodo a otro por distintos caminos. Si la red de malla está perfectamente conectada, no puede haber interrupción en las comunicaciones.

Híbrido

En la topología híbrida o topología mixta las redes pueden utilizar diversas topologías para conectarse. La topología mixta es una de las más frecuentes y se deriva de la unión de varios tipos de topologías de red. Por ejemplo: topología en árbol, estrella-estrella, bus-estrella, etc…

Su implementación se debe a la complejidad de la solución de red, o bien al aumento en el número de dispositivos, lo que hace necesario establecer una topología de este tipo. Tienen un coste muy elevado debido principalmente a su administración y mantenimiento.

1.3.4.2.- ZigBee (DigiMesh)

ZigBee [25] es un estándar de comunicaciones inalámbricas diseñado por *ZigBee Alliance*®. Es un conjunto estandarizado de soluciones que pueden ser implementadas por cualquier fabricante. Está basado en el estándar IEEE 802.15.4 de redes inalámbricas de área personal (*WPAN*) y tiene como objetivo las aplicaciones que requieren comunicaciones seguras con baja tasa de envío de datos y maximización de la vida útil de sus baterías.

Permite que dispositivos electrónicos de bajo consumo puedan realizar sus comunicaciones inalámbricas. Es especialmente útil para redes de sensores en entornos industriales, médicos y, sobre todo, domóticos.

Normalmente opera en la banda libre de 2.4 GHz, sin embargo también puede funcionar en las bandas (también libres) de 784 MHz en China, de 868 MHz en Europa y 915 MHz en Estados Unidos y Australia. Dependiendo de la frecuencia de trabajo la tasa de transmisión puede ir desde los 250 kbps en 2.4 GHz hasta los 20 kbps en 868 MHz.

Una red ZigBee en teoría puede estar formada por hasta 65.535 nodos. La realidad es que se pueden manejar grandes cantidades de dispositivos, del orden de miles, pero sin llegar a tal cantidad.

La topología más usada en ZigBee es la de malla ya que permite que si un camino de deja de funcionar la comunicación no deja de funcionar sino que elige un camino alternativo.

Como modulaciones más típicas se usan BPSK (símbolos binarios) y O-QPSK (símbolos hexadecimales).

Tipos de dispositivos

Una red ZigBee consta principalmente de 3 tipos distintos de dispositivos [10]:

- Coordinador ZigBee: es el dispositivo indispensable en la red, y por tanto debe de existir uno en cada una. Sus funciones son las de controlar la red y los caminos que deben seguir los dispositivos para conectarse entre ellos.
- Router ZigBee: interconecta dispositivos separados en la topología de la red, además de ofrecer a nivel de aplicación para la ejecución de código de usuario.
- Dispositivo final: posee la funcionalidad para comunicarse con el coordinador o el router ZigBee. Este tipo de nodo puede estar dormido la mayor parte del tiempo aumentando así la vida media de sus baterías. Además al tener requerimientos muy pequeños de memoria es más baratos.

Los coordinadores o routers también son llamados dispositivo de funcionalidad completa (FFD) o nodo activo y los dispositivos finales como los sensores o actuadores son llamados dispositivo de funcionalidad reducida (RFD) o nodo pasivo.

Los dispositivos FFD son capaces de comunicarse entre sí y con los RFD, mientras que los dispositivos RFD solo se pueden comunicar con los FFD pero no entre sí

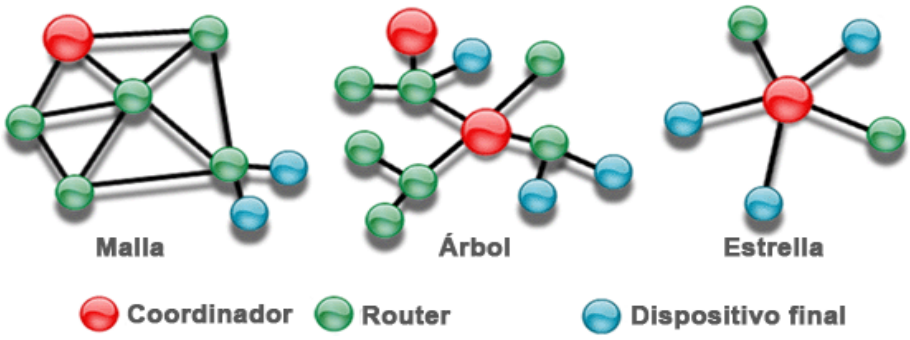


Figura 1.8.- Elementos de una red ZigBee dentro de diferentes topologías [10].

Ventajas y desventajas.

En la Tabla 1.1 se ven las principales ventajas y desventajas de las redes ZigBee.

Ventajas	Desventajas
Conexiones punto a punto y multipunto	Baja tasa de transferencia
Direccionamiento de información y refrescamiento de la red	Solo manipula texto
Banda libre ISM	Menor cobertura ya que WPAN
Óptimo para redes de baja tasa de frecuencia de datos	Búsqueda de compatibilidad con Bluetooth para mejorar prestaciones.
Alojamiento de 16 a 64 bits	-
Reducción de tiempos de espera	-
65000 nodos por red	-
Bajo consumo de energía y precio	-
Bajo nivel de radiación	-
AES cifrado, conexión segura	-

Tabla 1.1.- Ventajas y desventajas ZigBee.

DigiMesh

DigiMesh [29] es muy semejante a ZigBee, pero se diferencia de este en algunos aspectos. DigiMesh tiene un solo tipo de nodo (al tratarse de una red homogénea) y por tanto no existen relaciones entre nodos primarios y secundarios. Todos los nodos se pueden configurar como dispositivos alimentados por baterías o de baja potencia.

Como ventajas se permite una configuración más simple de la red, mayor flexibilidad y confiabilidad. DigiMesh a diferencia de ZigBee donde solo los dispositivos finales se podían suspender, permite que además los enrutadores o coordinadores lo hagan. Se elimina el punto único de fallo debido a una gran sincronización temporal, permite un mayor control sobre el espacio del código, un alcance mucho mayor que ZigBee y una configuración más sencilla.

Debido por tanto a su similitud con ZigBee y a que además incluye otras posibilidades adicionales (como permitir mayor control sobre los dispositivos) se ha decidido usar este protocolo en el presente proyecto.

El único inconveniente que se podría encontrar al usar DigiMesh es que al ser un protocolo propietario de la empresa *Digi*®, no se garantiza que todos los dispositivos XBee se puedan actualizar mediante *firmware* hacia el protocolo ZigBee o viceversa.

1.3.4.3.- Bluetooth Low Energy (BLE)

Bluetooth (IEEE 802.15.1) [11] es un estándar que permite desplegar redes inalámbricas de área personal (PAN); es decir, vincular sin cables varios dispositivos que están dentro de un radio cercano. Este estándar, nació en *Ericsson*® a mediados de los años 90, y a día de hoy es una de las conexiones habituales dentro de dispositivos móviles y PCs, abriendo la puerta a la conexión de todo tipo de periféricos y al intercambio de datos desde que fuese ratificado como estándar dentro del IEEE 802.15 en el año 2002.

A pesar que las distintas revisiones del estándar Bluetooth habían mejorado la tasa de transmisión de datos (de 1 Mbps a 24 Mbps), este tipo de conexiones seguían penalizando mucho la autonomía de los terminales y, prácticamente, podían agotar la batería de un Smartphone. BLE se basa en la reducción del consumo, minimizar la potencia de transmisión y el radio de cobertura.

Las principales características son: velocidad de transmisión de 1Mbps, conexión cifrada usando AES (de 128 bits) y códigos de redundancia para minimizar las transmisiones erróneas. Mientras con el Bluetooth convencional la batería de un dispositivo podría durar en torno a horas o días, la implementación del BLE hace que un dispositivo pueda estar encendido durante varios meses. BLE está pensado para dispositivos basados en domótica, salud y deporte.

1.3.4.4.- 6lowPAN

6lowPAN [12] es un protocolo que se encuentra entre las capas de enlace y red del modelo ISO/OSI y su nombre proviene del acrónimo del inglés "*IPv6 Over Low Power Wireless Personal Area Networks*". Se usa en dispositivos de bajas especificaciones como sensores o dispositivos dentro del mundo IoT.

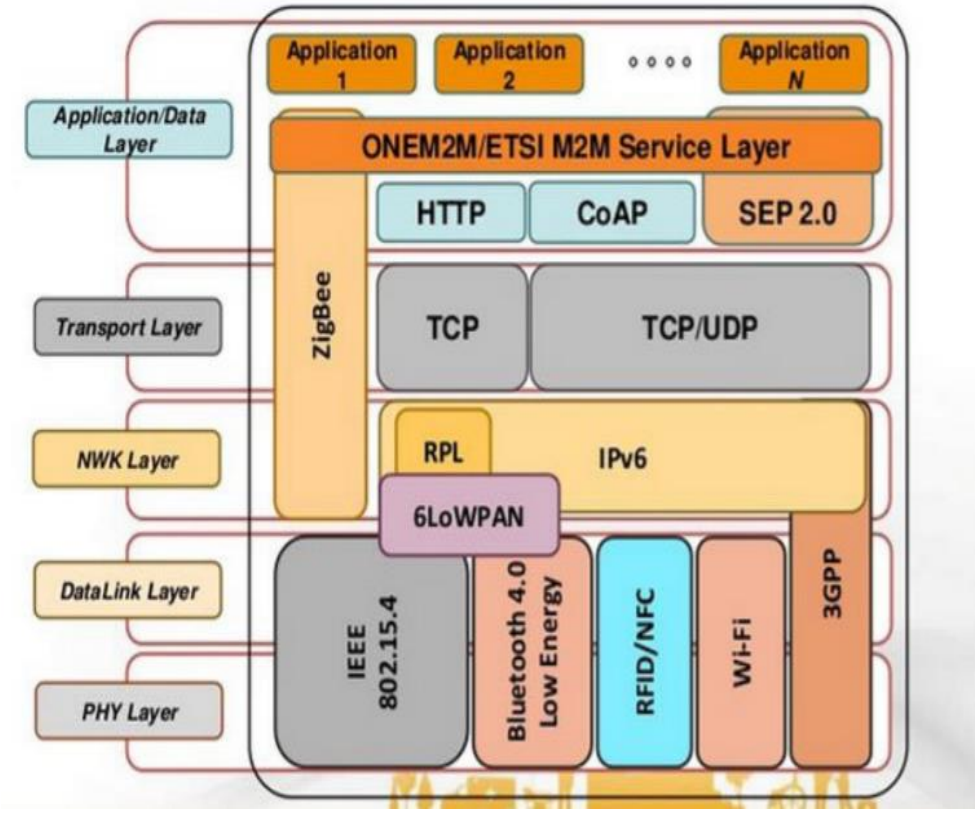


Figura 1.9.- Protocolos situados en las distintas capas del modelo ISO/OSI [12].

Se basa en el IEEE 802.15.4 ya que se suele usar para redes PAN de baja velocidad y basado en redes que necesitan de baja transmisión de datos y larga duración de la alimentación. En dicho estándar se definen varios parámetros, desde la banda de frecuencia hasta la modulación soportada.

1.3.4.5.- WiFi

Protocolo de comunicación definido en el estándar IEEE 802.11. Este tipo de redes cumplen dos tipos de requisitos, los correspondientes a los estándares Ethernet y los correspondientes a aquellos definidos en los recursos radioeléctricos. También se debe de tener en cuenta la manera o el orden en que cada uno de los dispositivos de red envía la información a otros.

Principales características WiFi	
Limitaciones	Cobertura, alcance y velocidades
Alcance	100 metros (dependiendo versión)
Velocidad	1 a 11 Mbps
Calidad de servicio	No considerada en versiones *b* y *gr*
Seguridad	Básica. Salvo en versión 802.11i
Diferentes versiones	802.11b (2.4GHz, velocidad de 11Mbps), 802.11a (5GHz, velocidad de 54Mbps), 802.11g (2.4GHz, velocidad de 54Mbps), 802.11n (empleo de MIMO), 802.11e (priorización de tráfico), 802.11i (seguridad a nivel del cableado).

Tabla 1.2.- Características WiFi [13].

1.3.4.6.- GSM

La red GSM (Sistema Global de Comunicaciones Móviles) [14] o estándar de telefonía móvil de segunda generación 2G fue a comienzos del siglo XXI el método de comunicación más usado alrededor del mundo.

En la actualidad debido a su bajo tasa de transmisión de datos y velocidad en comparación con los estándares más modernos su uso está restringido a pocas áreas. Una de estas es la del Internet de las Cosas, debido a que como norma general los paquetes transmitidos son de tipo texto y por tanto el tamaño será mucho menor que si se transmitiese audio, video o ambos.

La frecuencia a la que opera es variada, en Europa usa las bandas de frecuencia de 900 MHz y 1.800 MHz. La tasa de transferencia es de unos 9.6 kbps.

Arquitectura

En una red GSM, el usuario está identificado mediante la tarjeta SIM de su terminal móvil. Como toda red de telefonía móvil está compuesta de estaciones base, y todas ellas están conectadas a un Controlador de Estaciones Base (BSC) que regula los recursos. Además, todo el conjunto mencionado anteriormente (controlador y estaciones) se denota en su totalidad como Subsistema de Estaciones Base (BSS).

Los controladores de estaciones base están físicamente conectados al Centro de Conmutación Móvil (MSC).

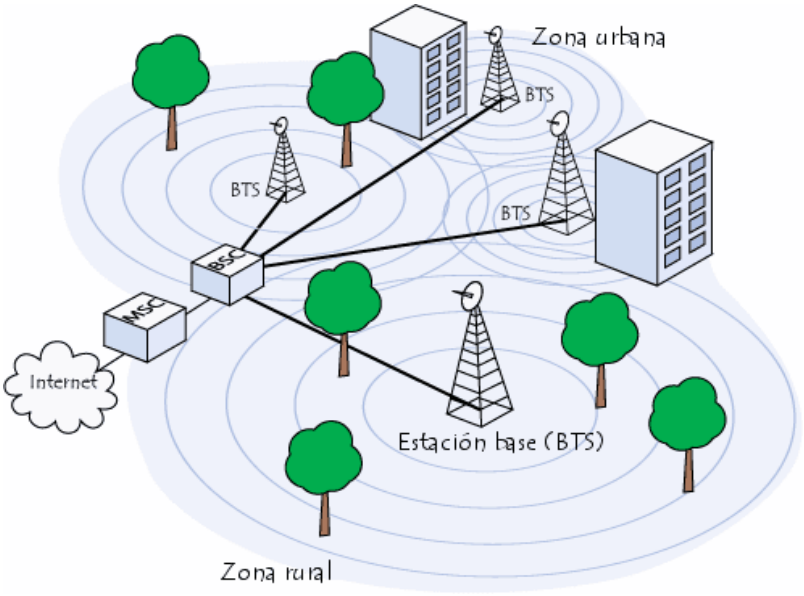


Figura 1.10.- Arquitectura GSM.

2.- PLATAFORMAS DE HARDWARE EXISTENTES

En este capítulo se mostrarán y describirán las funciones, características y elementos que forman parte de las plataformas de redes que ya están totalmente desarrolladas en la actualidad y están disponibles para ser implementadas de manera directa en distintos entornos, entre ellos el industrial.

2.1.- SigFox

Red [15] [16] que proporciona servicio de cobertura de bajo consumo, es inalámbrica y fue creada para que funcione e interactúe con dispositivos de bajo consumo energético (como pudiesen ser sensores) con tasas de transferencia de datos de hasta 12 bytes.

SigFox usa UNB (*Ultra Narrow Band*), que se basa en el uso de canales de espectro muy estrecho con la finalidad de aumentar el alcance de la transmisión con una necesidad de energía lo más baja posible.

Su funcionamiento es muy similar al de las redes de telefonía móvil ya que este se basa en el uso de estaciones móvil. A diferencia de las usadas en telefonía, las usadas en SigFox tienen la particularidad de que los dispositivos y sensores no están sujetos a una sola estación base específica ya que cualquiera de ellas puede enviar la información hacia la nube.

2.1.1.- Conectividad y velocidad

La cobertura de red satisface prácticamente las necesidades de casi todos los países europeos como Francia (país originario), Holanda, Portugal, Reino Unido, España… Pretende estar presente en unos 60 países a nivel global en un plazo de cinco años.

Los dispositivos que maneja esta red están la mayor parte del tiempo en reposo y transmitiendo mensajes de pequeño tamaño. La velocidad que proporciona la red hoy en día varía entre los 10kbps y 1kbps.

La banda de frecuencias usada por SigFox es de 868 MHz en Europa y de 915 MHz en Estados Unidos.

Cada estación base es capaz de dar servicio a un millón de objetos conectados. La red es escalable y permite por tanto aumentar o disminuir la capacidad de cada una de ellas mediante el aumento o la disminución de la densidad de la misma. Esta densidad está pensada para zonas de rurales en un rango de 30-50 km y de 3-10 km para zonas urbanas.

2.1.2.- Tecnología

El protocolo que usa la red SigFox es propietario (es decir, creado por la propia empresa). Es compatible con la mayoría de dispositivos que existen en la actualidad y el número de mensajes que permite enviar al día es de 140 mensajes (basado en suscripciones).

Con respecto a la seguridad, se aplica protección anti-respuesta, mensajes de aleatorización o secuenciación. Pero como los mensajes viajan de manera inalámbrica se incrementan los mensajes de sufrir ataques *man in the middle*.

La potencia transmitida varía entre los -20 y los 20 dBm y siempre teniendo en cuenta que no se pueden superar los 25 mW que se especifican en la banda ISM. El consumo varía entre los 45 y los 55 mA en modo activo, este consumo desciende a µA cuando el dispositivo está en modo reposo. Debido a estas características de consumo, la vida útil puede llegar a ser de varios años.

En cuanto a la recepción y procesado de datos, la red se basa en los dispositivos (emisores de mensajes) y el punto final de la red o *backend* que recibe esos mensajes y los procesa para generar un resultado.

SigFox ofrece el servicio llamado *SigFox Cloud* junto con una aplicación conocida como *SigFox Backend* desde la cual se pueden gestionar los dispositivos y visualizar los mensajes transmitidos y recibidos así como modificar la configuración de la red.

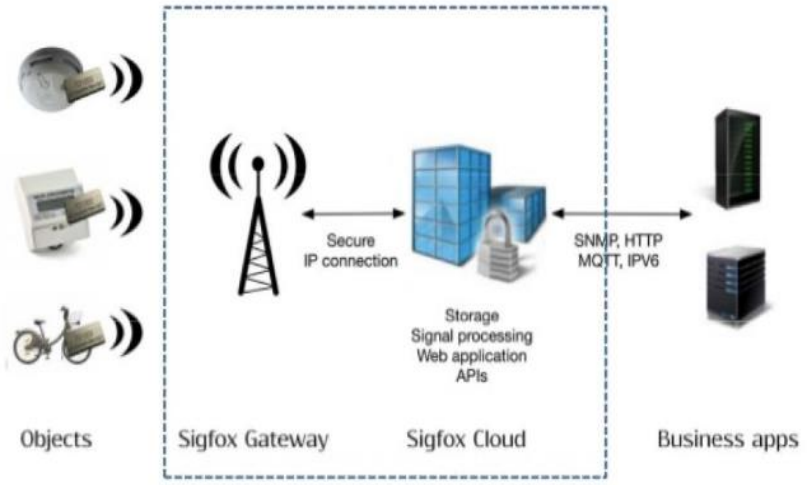


Figura 2.1.- Red SigFox [16].

Los dispositivos disponibles y compatibles para esta red se encuentran a la venta en los principales proveedores de componentes del mercado (*Texas Instrument®, Intel®, Silicon Labs®*...). Además siempre se busca la posibilidad de ofrecer soporte y disponibilidad para permitir así la interoperabilidad entre equipos de distintos fabricantes.

La compañía también ofrece la posibilidad de certificar los dispositivos la marca *SigFox Ready*, este proceso pretende clasificar los dispositivos en función de la cobertura y alcance radio. Dichas categorías van de 0 a 3, donde 0 es la de menor alcance y cobertura y 3 la que más.

En la Tabla 2.1 se resumen las características más notorias de esta red.

Principales características *SigFox*	
Facilidad de uso	Red ya desplegada. Solo integrar dispositivos
Operatividad	Fácil monitorización y gestión de dispositivos
Alcance	Asegura conectividad de objetos a larga distancia sin necesidad de implementar infraestructuras locales
Independencia de la frecuencia	Cobertura a nivel mundial y adaptable a cualquier entorno
Bajo consumo	Larga vida útil baterías
Bajo coste	Permite conectar a la red cualquier tipo de red en grandes cantidades

Tabla 2.1.- Características SigFox.

2.2.- WirelessHART

WirelessHART [18] [19] es una red inalámbrica basada en el estándar HART de amplia difusión, tal como lo evidencian los 20 millones de dispositivos de campo compatibles con HART. Fue concebida originalmente como una ampliación del bucle de corriente de 4 a 20 mA con el fin de proporcionar dispositivos de campo con mayor funcionalidad.

Dicho estándar fue publicado en septiembre de 2007 y basa su uso en la banda de 2.4 GHz. Como ya hay varias tecnologías trabajando en esa frecuencia, con el fin de evitar interferencias lleva a cabo una búsqueda especial de canales no utilizados.

2.2.1.- Tecnología

Utiliza una red de malla donde todas las estaciones de radio componen una red y en la que cada componente actúa simultáneamente como fuente de señal y como repetidor. El transmisor originalmente envía un mensaje hacia su vecino más próximo y este lo reenvía por la ruta más eficiente. Si una ruta se cancela debido por ejemplo a un obstáculo o un

receptor caído, entonces se obtiene de manera automática una ruta alternativa, de esta manera se aumenta la fiabilidad de la transmisión.

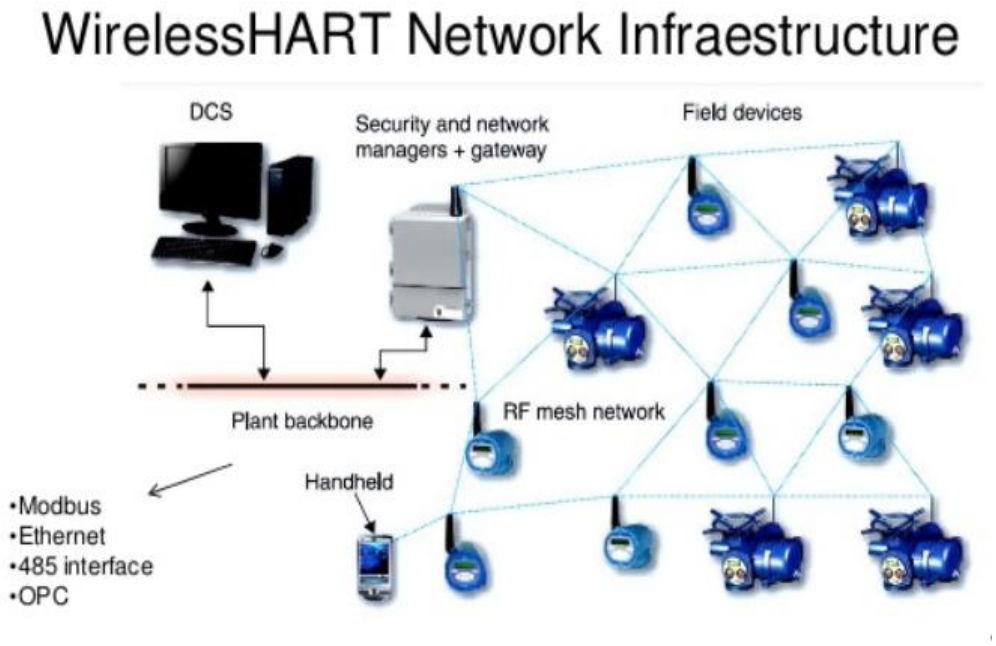


Figura 2.2.- Infraestructura de una red *WirelessHART* [26].

La red ofrece una mayor seguridad de transmisión gracias a rutas alternativas y redundantes. Los usos de este tipo de red son diversos y van desde la parametrización de dispositivos, supervisión de valores instrumentales y medioambientales, supervisión del rendimiento hasta la gestión de energía.

La comunicación en la red inalámbrica se coordina con TDMA ("*Time Division Multiple Access*"), que sincroniza los dispositivos integrantes de la red en etapas en 10 ms. Esto permite una red de alta fiabilidad y reduce los tiempos de adelanto y retardo de cada estación.

Como medida contra los colapsos, se usa la técnica del espectro ensanchado por salto de frecuencia. Los 15 canales definidos en el estándar IEEE 802.15.4 se utilizan en paralelo. La combinación de los 15 canales con los 10 ms de sincronización permite obtener 1500 conexiones cada segundo.

2.2.2.- Dispositivos y aplicaciones

Una red WirelessHART consta de *Gateways* y de adaptadores.

Una ***Gateway*** gestiona la red inalámbrica y la conecta a un sistema de control SCADA mediante un bus de campo. Las señales de los dispositivos se reciben y envían mediante el protocolo de bus adecuado.

Un **adaptador** constituye el otro extremo de la ruta de comunicaciones. Los dispositivos de campo existentes se pueden complementar con un adaptador. En la actualidad hay 3 tipos.

-Adaptadores alimentados por bucle: se incluyen en un bucle existente de donde extraen la energía para su funcionamiento.

-Adaptadores con batería: suministran la alimentación al dispositivo mediante una batería. Se puede configurar el dispositivo de campo de manera que solo se active para enviar los datos cada un determinado tiempo.

-Alimentación externa: obtiene la alimentación para el dispositivo de campo y el adaptador.

Como aplicaciones destacan la extracción de aceite en máquinas, la producción de catalizadores, monitorización del nivel de agua, incineraciones de residuos tóxicos, actualización de inventario… Se suele usar en general para monitorizar y optimizar.

2.3.- LORA

LoRaWAN [20] es una especificación para redes de baja potencia y área amplia. Está diseñada específicamente para dispositivos de bajo consumo de alimentación, que operan en redes de alcance local, regional, nacional o global.

El estándar de red se ajusta a los requerimientos necesarios del Internet de las Cosas, tales como conexiones bidireccionales seguras, bajo consumo de energía, largo alcance de comunicación…

Permite la interconexión entre dispositivos inteligentes, otorgando libertad de uso en distintos ámbitos.

2.3.1.- Tecnología

En la Tabla 2.2 se puede observar un resumen de las principales características de las redes LoRaWAN.

Una red LoRa consta de tres elementos, dispositivos finales (permiten la conexión de los objetos a la propia red), pasarelas (reciben las transmisiones y las reenvían hacia un servidor determinado) y servidores (recepción de datos y procesamiento de los mismos).

Principales características LoRaWAN	
Facilidad de uso	Red ya desplegada. Solo integrar dispositivos
Banda	ISM (868 y 433 MHz)
Modulación	DSSS
Bajo Consumo	25 mA en transmisión y 10 mA en recepción
Robustez	Frente a interferencias y efecto *Doppler*
Ajuste dinámico de potencia	ADR (*"Adaptative Data Rate")*
Velocidad	0.3 kbps a 5486 kbps
Sensibilidad	-137 dBm a -123 dBm
Alcance	2 km a 14 km

Tabla 2.2.- Características LoRaWAN.

2.3.2.- Dispositivos

En una red LoRaWAN, los dispositivos se dividen en clases, según las funcionalidades que soporten. Existen en la actualidad 3 tipos de clases, siendo la clase A la más restrictiva y la clase C la que más características posee.

- **Clase A**

  Clase más eficiente desde un punto de vista energético y adecuada para aplicaciones que prácticamente solo requieran de envío de datos.

- **Clase B**

  Permite la creación de ventanas de recepción sin la necesidad de que se produzca una transmisión previa.

- **Clase C**

  En esta clase los dispositivos se encuentran en modo de transmisión permanente que sólo se interrumpe cuando se produce una transmisión. Presenta mejor latencia de conexión. Representan un balance entre el consumo de energía y la capacidad de recepción.

El número de dispositivos limita la capacidad de la red, siendo a mayor número de dispositivos menor rendimiento. El valor máximo de transferencia en bytes por hora y

dispositivo para una carga de 10 bytes en una red de 250 dispositivos finales es de 3670 bytes mientras que si la red es de 5000 dispositivos esta cifra se reduce a 180 bytes.

Además, con la finalidad de no disminuir el rendimiento, se recomienda que los dispositivos finales se encuentren lo más cerca posible de la pasarela.

En cuanto a los dispositivos destacan los siguientes:

Transceptor LoRa

Dispositivo formado por un solo circuito integrado que contiene el modulador LoRa y un amplificador. No incluye la antena ni el cristal. La pila del protocolo LoRaWAN se debe implementar en un microcontrolador externo (MCU).

Front-end LoRa

Son varios los dispositivos de este tipo sobre los que destacan los transceptores de la familia SX12XX de *Semtech*®, empresa propietaria de la tecnología LoRa.

Modem LoRa

Dispositivo que junto al transceptor incluye circuitería que implementa la pila del protocolo LoRaWAN. Este tipo de dispositivos se controlan mediante un MCU externo a través de una interfaz de órdenes ASCII sobre UART.

System-on Chip LoRa

Dispositivo que integraría el transceptor, la antena, la pila del protocolo LoRaWAN y el MCU en un solo circuito integrado.

2.4.- WIMON 100 SERIES

WIMON 100 [21] es un sistema propietario de la empresa *ABB*®. Se caracteriza por activación automática, monitorización inalámbrica de motores eléctricos y equipo rotativo. La información que produce la red es accesible de manera remota por los usuarios que así lo deseen.

Es un sistema fácilmente instalable y los costes derivados de la ingeniería, planificación, materiales, costes, etc… suelen ser menores que los de otro tipo de sistemas tradicionales (sistemas cableados).

Los elementos que componen la red son:

- Sensor WIMON 100 AB.
- Gateway.
- WIMON DataManager.
- Servidor OPC.

Este tipo de red es muy semejante a la red WirelessHART explicada en el sub-apartado 2.2.

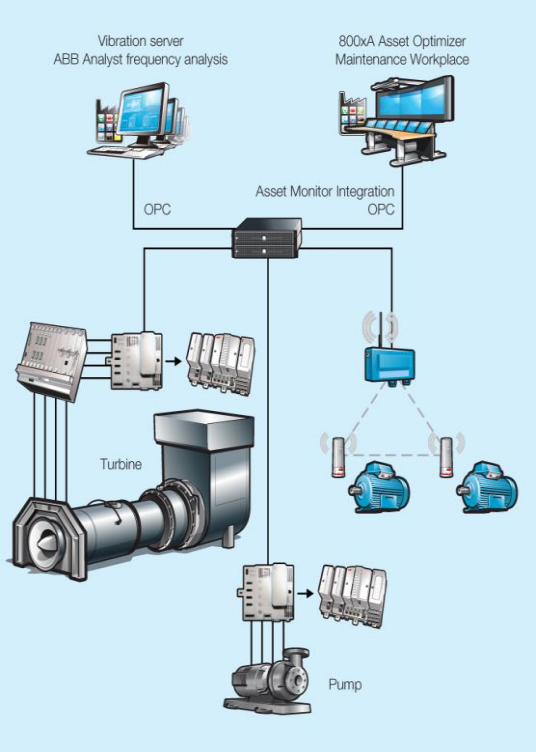


Figura 2.3.- Ejemplo de arquitectura de red WIMON 100 [28].

Como beneficios de este tipo de red se destaca que detecta de manera temprana posibles fallos, reduce costes, se puede activar y desactivar de manera remota, eliminación de espureos en frecuencia…

En la Tabla 2.3 se resumen las características de la misma:

Principales características *WIMON 100*
Recolección de datos de manera automática
Larga vida de la batería (5 años)
Fácil integración
Certificado ATEX Zone 0
Temperatura soportada: -40 a 80ºC
Sin límites de equipos para monitorizar
Software compatible con el sistema de ciberseguridad estándar *ABB*

Tabla 2.3.- Características *WIMON 100*.

2.5.- WISE

Plataforma hardware que permite dar soporte a distintos tipos de necesidades industriales. En concreto WISE 4000 [22] [23] se basa en comunicación inalámbrica (dentro del mundo del Internet de las Cosas). Incluye adquisición de datos, procesado y funciones de publicación. Además de esto, provee al usuario de pre-escalado de datos, lógica de datos y funciones de *datalogger*. Dichos datos pueden ser consultados a través de un dispositivo móvil.

En la Tabla 2.4 se pueden ver explicadas las principales características de este sistema.

Principales características *WISE 4000*	
Servicio web RESTful	Estilo de arquitectura de software usado para la creación de servicios web escalables. Se basa en transferencia de hipertexto (HTTP) y en el uso de peticiones GET, POST, PUT, DELETE… Los datos pueden ser recuperados vía HTML, JSON, XML… Soporta HTTPS y cifrado TLS.
Almacenamiento de datos	Almacenamiento de hasta 1000 muestras de datos en un determinado tiempo. Los datos se guardan periódicamente y si se llegase a llenar la memoria se descartarían los más antiguos.
IoT Cloud	Los datos almacenados se pueden consultar en la nube y ser descargados de esta en formato *.csv
Protocolo MQTT	Modelo de mensajes basado en publicador/suscriptor para dispositivos de pequeño ancho de banda, alta latencia y redes no seguras.
Comunicación inalámbrica	Puede ser vía WiFi, 3G, LPWAN… En función de las necesidades de la comunicación inalámbrica.

Tabla 2.4.- Características WISE 4000.

Figura 2.4.- Características de una red *WISE 4000* [23].

2.6.- LIBELIUM

Plataforma hardware [24] pensada para el uso en ciudades inteligentes. Permite integrar el sistema de monitorizar el ruido, la polución, la salud de los edificios y el malgasto de energía.

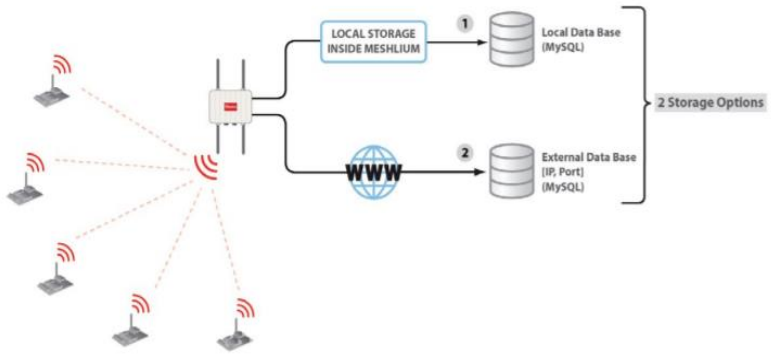


Figura 2.5.- Esquema general de una red *LIBELIUM* [27].

La capacidad de medir los parámetros mencionados, combinados con una buena infraestructura permite ofrecer diversos servicios simultáneamente. Entre ellos destacan la monitorización de niveles de gas, la detección de radiación y de plazas libres en un aparcamiento.

En la Tabla 2.5 se pueden ver resumidas las características principales de los sensores usados.

Principales características sensores LIBELIUM	
Sensores de gas calibrados	Mide niveles de CO, NO, NO_2 y SO_2.
Sensor de ruido/sonido	Cumple especificaciones IEC 61672 para niveles de sonido. Niveles de precisión de ±0.5dBA (1KHz).
Sensor de partículas de polvo	Mide concentraciones de los niveles PM1, PM2.4 y PM10. Precisión de 0.1 ppm.
Sensor grietas	Mide grietas de hasta 10 μm en infraestructuras.
Sensor de ultrasonidos	Capacidad para medir los niveles de basura en contenedores.

Tabla 2.5.- Características sensores LIBELIUM.

Las placas PCB usadas en las *Smart Cities* requieren de una tensión de alimentación que debe de estar comprendida entre 3.3V y 5V. Los sensores son alimentados en grupos. En cuanto al consumo, varía en función del número de sensores que se encuentren en funcionamiento. El consumo mínimo constante es de 400 μA y el caso de mayor consumo es de 37.5 mA que es el que se corresponde cuando está activo el sensor de partículas de polvo. El consumo promedio suele ser de 15 mA.

Muy semejante a las redes SigFox.

3.- DISEÑO Y DESARROLLO DE UNA RED ADHOC

En este capítulo se mostrarán los dispositivos elegidos a la hora de desarrollar la red ADHOC (red normalmente de tipo malla que basa su funcionamiento en el constante reenvío de información entre nodos de manera inalámbrica) en la que se basa el presente proyecto. Se explicarán sus principales características, y también se ilustrará el proceso de configuración de cada uno de ellos hasta llegar al correcto funcionamiento de la red global.

3.1.- Requisitos de la red

Una vez realizado un estudio sobre las principales redes existentes en la actualidad, se establecen los requisitos que debe de cumplir la red a diseñar.

La red deberá de contar con tres tipos de elementos. El primero de ellos debería de ser un elemento capaz de recoger los datos proporcionados por un determinado tipo de sensor, tener la capacidad de procesarlos y posteriormente enviarlos a otro nodo de la red. Este otro tipo de nodo, sería aquel que debe de ser capaz de reenviar los datos que recibe y enviarlos a una puerta de enlace. El último elemento sería un *Gateway* y tendría la capacidad de reenviar los datos hacia un servidor o de interactuar con un actuador.

La banda de trabajo elegida es la Industrial-Científico-Médica (ISM) ya que se trata de una frecuencia (868 MHz) que permite mayor alcance que otras (como puede ser 2.4 GHz). Además de ser de ámbito libre, posee características como tolerancia frente a errores y mecanismos de protección frente a interferencias (espectro ensanchado).

La potencia de transmisión de cada elemento deberá ser menor a 100 mW, ya que se busca que se trate de una red de baja potencia.

Se deberá de usar un protocolo que permita usar una estructura de red tipo malla (DigiMesh), buscando siempre aumentar la robustez de la misma.

Para el caso analizado se pretende dar cobertura a un pequeño entorno industrial donde habrá numerosas dispositivos que podrán atenuar el nivel de señal o interferir en la misma, además de otros elementos como pudiesen ser muros. Por ello se pretende garantizar una correcta recepción de datos para una distancia de 25 m (lugar donde se han realizado las pruebas).

Además siempre se garantizará que todos los nodos de la red tengan conexión con el resto con mayor o menor nivel de señal para así poder garantizar en la medida de lo posible la estructura tipo malla.

3.2.- Elección de componentes

Como se ha comentado al inicio del presente documento y del actual capítulo el fin de este proyecto es el de desarrollar una red inalámbrica basada en el protocolo DigiMesh y mediante el uso de dispositivos XBee, todo ello en la frecuencia de 868 MHz. Todos los dispositivos que se mencionan a continuación corresponden a la empresa *Digi*®.

3.2.1.- ConnectPort X4IA

En la Figura 3.1 se puede observar la apariencia del dispositivo ConnectPort X4IA. En ella se puede observar como tiene tres conectores para antenas, dos de ellas son para bandas de telefonía y una de ellas de ZigBee (DigiMesh) en 868 MHz, siendo esta última la usada en el presente proyecto.

La funcionalidad de este componente de la red es la de usarse como *Gateway*, es decir, será el encargado de captar todos los datos que les llegan a través de los nodos conocidos como XBee. Una vez recibidos los datos de los dispositivos (que estén dentro de una lista de posibles emisores) se reenviarían hacia un servidor que se encontraría corriendo en un ordenador.

Una de las principales razones para la elección de este dispositivo ha sido la gran capacidad de memoria y procesamiento que posee ya que otros módulos como pudiese ser un XBee en el rol de coordinador necesitaría de un PIC externo que aumentase sus prestaciones y que le permitiese realizar funciones de nivel superior. De esta manera se evitan problemas de latencia, aumenta la velocidad y la disponibilidad, y se intenta que todo esté más cerca de la propia empresa ("*Edge Computing"*) evitando así el uso obligatorio de almacenamiento en la nube (ya que se podría almacenar en servidores propios). Además permite la adaptabilidad de diferentes protocolos al protocolo IP, la conectividad hacia un servidor centralizado, notificación de alarma mediante correo electrónico y un sistema operativo a través del cual se puede conectar y controlar dispositivos XBee. En caso de ser necesaria la transmisión de datos vía 2G o 3G, este dispositivo incluye antenas para posibilitar dicha comunicación.

Se destaca que se permiten subir archivos Python y ser ejecutados dentro del mismo.

En la Tabla 3.1 se muestran las principales características.

Especificaciones técnicas *ConnectPort X4IA*	
WWAN	
LTE CAT 1	N/A
HSPA+ (U9)	850/900/1700(AWS)/1900/2100MHz; Transfer rate (max): 5.76 Mbps up, 21 Mbps down
CDMA 1XRTT (BX)	800/1900 MHz; Transfer rate (max): 153 kbps up/down
Conector	1 o 2 SMA
Ranuras SIM	2
RF	
Estándar	ZigBee (2.4GHz), DigiMesh (2.4 GHz o 900 MHz), Digi XBee-PRO 900HP, 802.15.4 o 868
Conector	1 x RP-SMA
Interfaz cableado	
Serial	RS-232/422/485 (elección por software)
Ethernet	RJ-45
Alimentación	
Entrada	6-30 VDC
Consumo máximo	10.4 W
Dimensiones	
Dimensiones	13.33 cm x 8.50 cm x 2.54 cm
Peso	0.20 kg

Tabla 3.1.- Especificaciones técnicas ConnectPort X4IA [30].

Figura 3.1.- ConnectPort X4IA.

3.2.2.- XBee SX868

El dispositivo elegido para transmitir los datos del sensor elegido y de reenviarlos a través de la red ha sido el módulo XBee SX 868. Principalmente se ha elegido este modelo ya que pertenece a la familia RF que opera entre las frecuencias 863 y 870 MHz en Europa.

Figura 3.2.- XBee SX868.

En la Tabla 3.2 se pueden ver las principales especificaciones técnicas.

Especificaciones técnicas *XBee SX868*	
Hardware	
Banda de frecuencia	863 a 870 MHz
Opciones de antena	U.FL o pad RF
Parámetros de comunicación	
Velocidad de datos RF	10 kbps o 80 kbps.
Velocidad de datos UART	Hasta 921 kbps
Velocidad de datos SPI	Hasta 6 Mbps
Potencia transmitida	Hasta 13 dBm
Sensibilidad	-106 dBm @ 80 Kbps, -113 dBm @ 10 Kbps
Características	
I/O	13 Digital I/O
Entradas analógica	4 canales de 10 bits
Temperatura	-40ºc a +85ºC
Topologías de red	DigiMesh, Repetidor, Punto a Punto, Punto a Multipunto y Peer to Peer
Alimentación	
Entrada	2.4 a 3.6 VDC
Corriente en la transmisión	55 mA
Corriente en la recepción	40 mA
Corriente en modo dormido	1.8 µA

Tabla 3.2.- Especificaciones técnicas *XBee SX868* [31].

Además se ha adquirido una placa de desarrollo denominada *XBee SX868 development board*, la finalidad de la misma es la de permitir configurar el dispositivo XBee con mayor facilidad a través del software *Digi XCTU*. La placa puede visualizarse en la Figura 3.3.

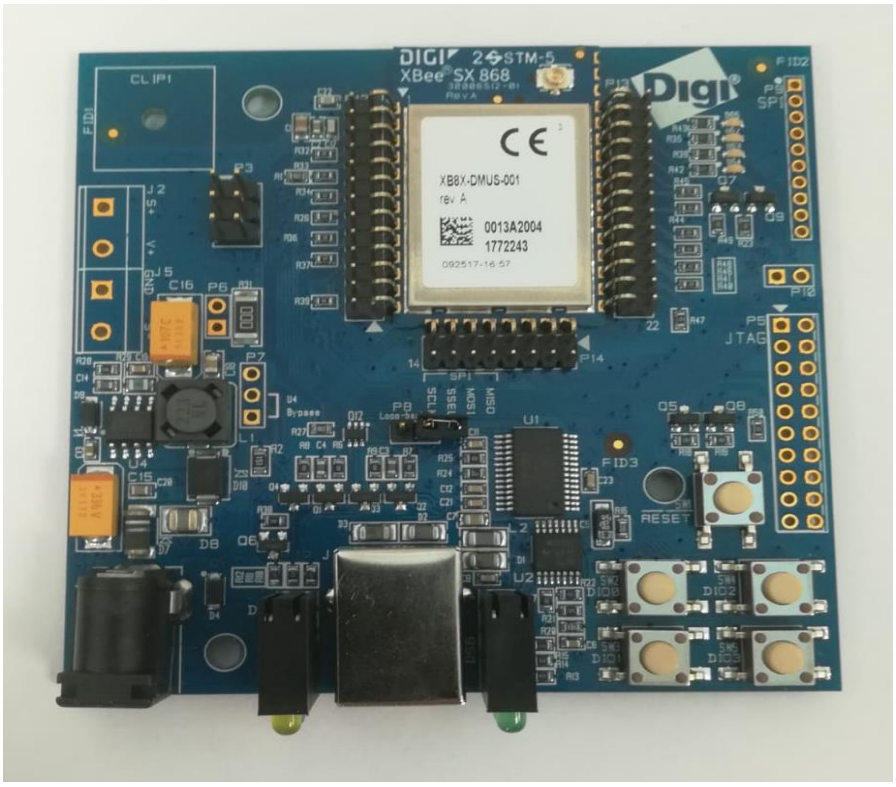


Figura 3.3.- Placa de desarrollo XBee SX868.

En el mencionado kit de desarrollo además de la placa que permite configurar el XBee vienen incluidos una antena para el propio dispositivo (ver Figura 3.4) y una fuente de alimentación de 12 VDC. Se puede destacar que esta PCB ajusta de manera automática los niveles de tensión a los 3.3 V que necesita el XBee para funcionar de una manera óptima.

Figura 3.4.- Antena XBee SX868.

La antena es un dipolo de media onda y además de trabajar en la banda de 868 MHz, posee una ganancia de 2.1 dBi.

3.2.3.- Arduino NANO

En el nodo principal de la red (aquel que consta de un sensor encargada de tomar las medidas necesarias) se necesita de un elemento que sea capaz de procesar los datos que envía el sensor y de enviarlos hacia el XBee de manera que este último los vaya reenviado de nodo en nodo hasta llegar al a *Gateway*.

Por tanto se han barajado dos alternativas a la hora de elegir un elemento que se encargue de tal fin. Dichos elementos han sido Arduino (entre todos sus modelos) y Raspberry Pi 3 (última versión).

En la Tabla 3.3 se puede ver una comparativa a nivel general entre ambos.

	Arduino vs Raspberry Pi	
	Raspberry Pi	Arduino
CPU	Mucha más potencia de cálculo	Parte principal, encargado de dirigir operaciones
Memorias	Dispositivos externos	Incluidos en el propio dispositivo
Velocidad de operación	Rápida	Lenta (en comparación Raspberry Pi)
Tamaño	Mayor tamaño	Menor tamaño
Coste	Más costoso	Menos costoso
Interferencias	Más susceptibles debido a su tamaño y cableado externo	Bajo, debido al gran nivel de integración

Tabla 3.3.- Comparativa entre Raspberry Pi y Arduino NANO [32].

A partir de la tabla anterior, se puede deducir que las Raspberry Pi son una buena opción para aquellos sistemas en los que los requerimientos de procesamiento son elevados y no importa el tamaño ocupado. No obstante, en el caso actual se busca todo lo contrario ya que el dispositivo en sí solamente debe de ser capaz de procesar los datos enviados por un sensor cada un determinado tiempo y esto no requiere de un procesamiento elevado, además la elección del Arduino, en concreto el modelo NANO se debe al pequeño tamaño del mismo y a que se adapta de una manera correcta tanto al XBee como al sensor elegido.

En la Tabla 3.4 se pueden ver las principales especificaciones del Arduino NANO.

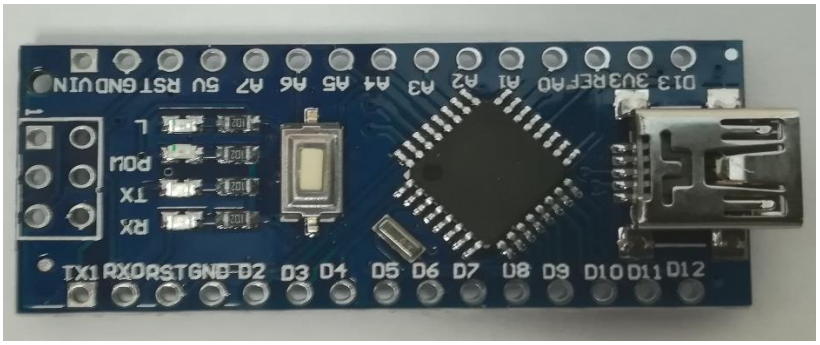


Figura 3.5.- Arduino NANO.

Especificaciones técnicas *Arduino NANO*	
Microcontrolador	ATmega328
Arquitectura	AVR
Memoria Flash	32 KB
SRAM	2KB
Velocidad de reloj	16 MHz
Pines analógicos	8
EEPROM	1KB
Corriente en los pines I/O	40 mA
Tensión de alimentación	7-12 V
Pines digitales	22
Salidas PWM	6
Consumo de potencia	19 mA
Tamaño	18x45mm
Peso	7g

Tabla 3.4.- Especificaciones técnicas *Arduino NANO* [33].

3.2.4.- Sensor RGB

Sensor encargado de tomar medidas tales como: temperatura de color, cantidad de rojo, cantidad de verde, cantidad de azul y luminancia con la finalidad de realizar un balance de blancos. Este proceso será explicado más en detalle en el Capítulo 6 del presente documento.

Se ha elegido este tipo de sensor (Figura 3.6) debido a que la empresa *ArcelorMittal*® necesita realizar la corrección de fotografías tomadas en el amanecer y en el atardecer, situaciones donde el nivel de luz es crítico a la hora de tomar imágenes que se correspondan con lo que se ve a través de los ojos.

Figura 3.6.- Sensor RGB TCS34725.

Además este sensor de color digital [34] es compatible con el Arduino NANO elegido en el apartado anterior. El TCS34725 es un circuito integrado completo que realiza un tratamiento digital de la medición de color. La comunicación entre el sensor y el Arduino se realiza por I2C.

3.2.4.1.- Funcionamiento TCS34725

El TCS34725 basa su funcionamiento en una matriz de 3x4 fotodiodos, junto con 4 conversores A/D de 16 bits que realizan la medición de los fotodiodos.

La matriz de 3x4 está formada por fotodiodos filtrados para rojo, verde, azul y sin filtro (*clear*). Todos los sensores están filtrados para infrarrojos.

Los ADC integran la medición de los fotodiodos, que es transferida a los registros internos del TCS34725, el cual incorpora un doble buffer para asegurar la integridad de los datos.

El estado del sensor y el estado de energía del sensor está controlado por una máquina de estados interna, que controla todas las funciones del TCS34725.

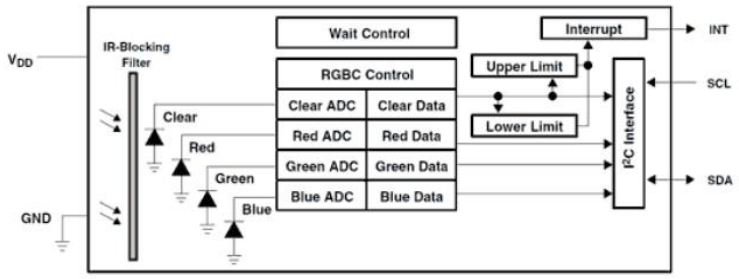


Figura 3.7.-Esquemático sensor RGB TCS34725 [34].

3.2.5.- *KEPServerEX*

Como servidor para visualizar los datos recibidos a través del ConnectPort X4IA se ha usado *KEPServerEX* (de la empresa *Kepware*®) por su amplia capacidad para soportar diferentes configuraciones. Usado principalmente como servidor para automatización de datos en entornos industriales, permite conectar, gobernar, monitorizar y controlar diferentes dispositivos y aplicaciones software a través de una simple interfaz gráfica.

KEPServerEX [35] aprovecha OPC (el estándar de interoperabilidad en la industria de la automatización) y el estar centrado en los protocolos de las Tecnologías de la Información (TI) tales como SNMP, ODBC y servicios WEB.

En la Tabla 3.5 se pueden observar las principales características de este tipo de servidor.

Características *KEPServerEX*	
Campos de trabajo	IoT, IT, Inteligencia de Negocios y Excelencia Operacional en la Empresa
Compatibilidad	VMWare y Hiper-V
Seguridad	SSL y TLS
Accesibilidad	OPC UA, OPC DA, OPC AE, OPC HDA, interfaces automatizadas, interfaces IT, interfaz en la nube.
Optimización	Redundancia, balanceado de carga, comunicaciones, enlace maquina a máquina.
Agregación	Plataforma centralizada, configuración unificada, almacenamiento de datos
Conectividad	Drivers, Ambientes de telemetría, rápido despliegue y simulación

Tabla 3.5.- Especificaciones técnicas *KEPServerEX* [35].

Una de las características más destacadas es que posee la opción de establecer un cliente configurable (U-CON), el cual permite crear a los usuarios de una manera rápida y sencilla una especie de driver para aquellos dispositivos que se conectan por serial o Ethernet (y que no poseen de uno específico). Esta característica ha sido usada en la configuración del servidor tal y como puede comprobarse en el sub-apartado 3.5.

3.3.- Configuración ConnectPort X4IA

En el presente apartado se mostrará la interfaz gráfica del ConnectPort X4IA, dentro de la misma se procederá a explicar los cambios realizados y el porqué de los mismos.

El código subido a este dispositivo ha sido programado en el lenguaje Python y está basado en *sockets* TCP/IP. Un socket es una abstracción software y que representa el extremo de una conexión (está identificado por IP y puerto), permite ignorar los detalles de la capa de transporte y de red. Es decir, un *socket* es un hilo existente que permite enviar y recibir información desde una máquina remota.

Tanto el código al que se hace referencia en el párrafo anterior como otra configuración barajada (y basada en el protocolo Modbus) pueden verse explicadas en los Anexos 8.3 y 8.1 respectivamente.

3.3.1.- *Home*

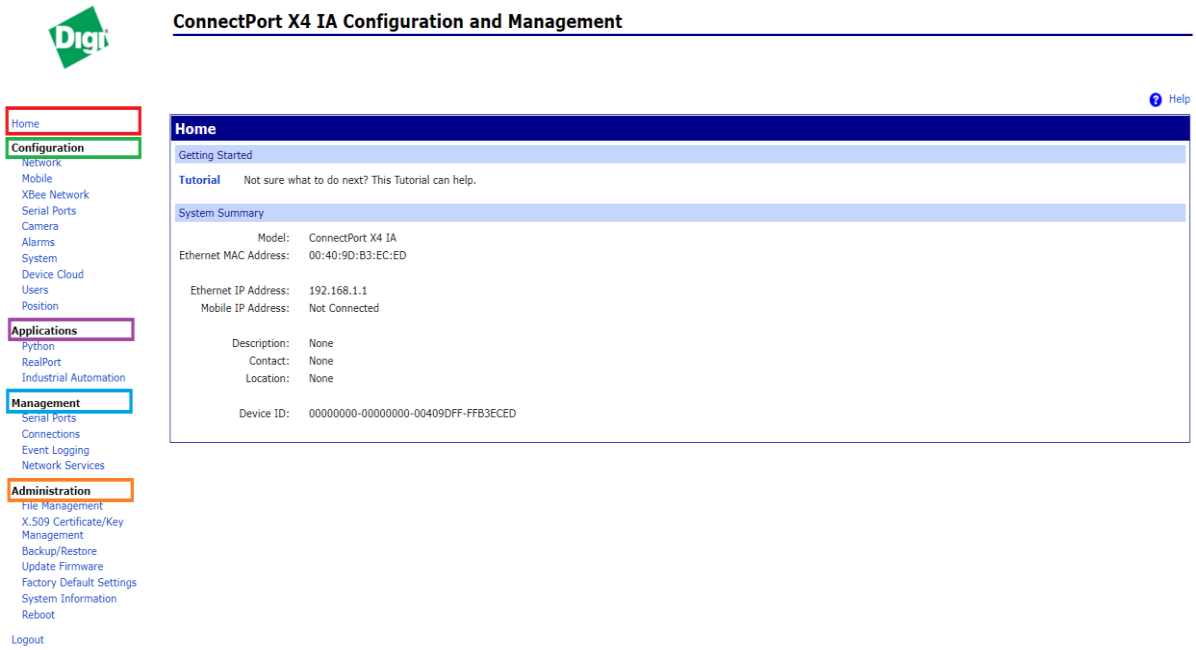


Figura 3.8.-Interfaz ConnectPort X4IA.

En la Figura 3.8 se puede ver la ventana principal de la interfaz gráfica del ConnectPort X4IA. Para acceder a la misma es necesario conectar el dispositivo con un ordenador mediante un cable Ethernet y posteriormente escribir en la barra de búsqueda de un navegador la dirección IP del *Gateway*. Dicha IP se puede obtener de manera sencilla mediante la línea de comandos que está disponible para todos los ordenadores.

En la imagen anterior se pueden ver recuadradas los distintos apartados que componen la interfaz, estos son: "*Home*", "*Configuration*", "*Applications*", "*Management*" y "*Administration*".

"*Home*" (recuadrada en color rojo en la Figura 3.8 y visible en la misma imagen) permite observar el modelo de dispositivo, la dirección MAC e IP del mismo y su identificador. Se destaca que no se puede realizar ninguna configuración en la misma.

3.3.2.- *Configuration*

"*Configuration*" permite configurar la red XBee, las características principales del dispositivo, posibilidad de conectar a la nube, puertos serie, etc… Dentro de las mismas se han modificado las opciones de "*Network*" y "*XBee Network*" y el resto se han dejado por defecto.

Dentro de la pestaña "*Network* se observan que hay diferentes opciones de configuración pero solamente se ha modificado la correspondiente a "*Ethernet IP Settings*" (Figura 3.9) donde se ha usado una IP estática en lugar de una dinámica asignada de manera automática por el protocolo DHCP. Se puede comprobar que la dirección es la IP 192.168.1.1 con su correspondiente máscara de subred. Dicha dirección será de utilidad a la hora de realizar cambios en la configuración del mismo y la hora de indicar al servidor de donde debe de esperar los datos que reciba.

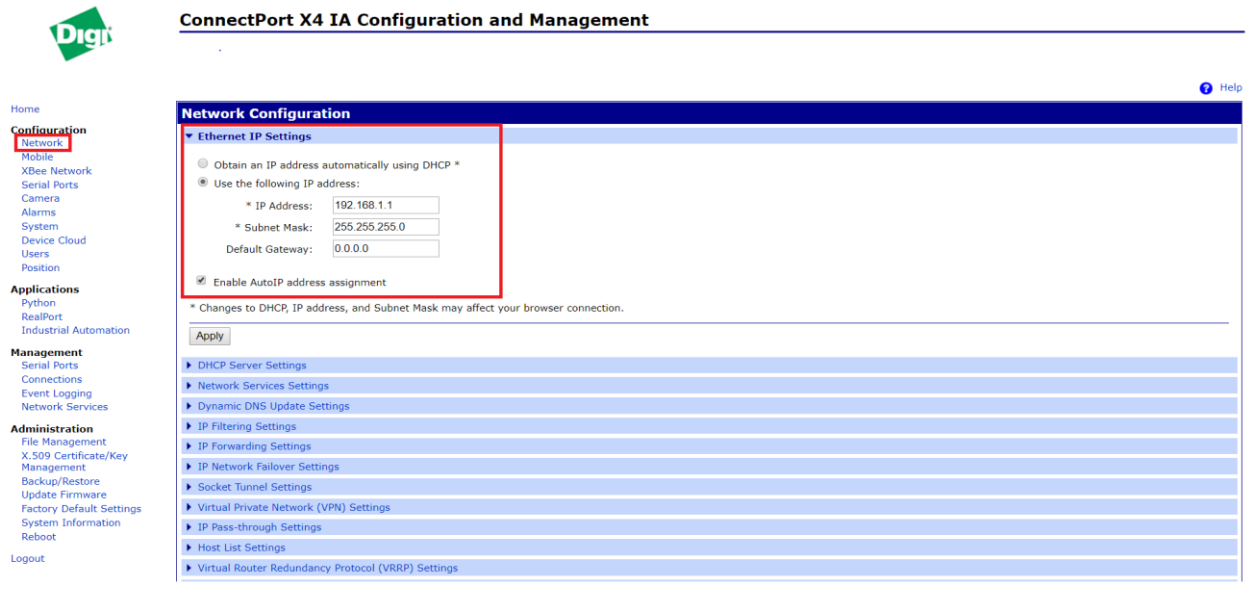


Figura 3.9.-"*Network*".

En la pestaña "*XBee Network*" se pueden observar los nodos que componen la red XBee. En este caso se puede ver que la red está compuesta por el propio ConnectPort X4IA (que tiene el rol asignado de coordinador) y por dos dispositivos XBee: "XBEE_A" y "XBEE_B" que tienen la función de *routers*, en todos ellos se puede ver el nombre que poseen, su dirección extendida, el tipo de rol que ocupa dentro de la red y el tipo de dispositivo que se trata.

Para comprobar que la red está compuesta por los dispositivos esperados se debe de hacer clic en el botón "*Discover XBee Devices*".

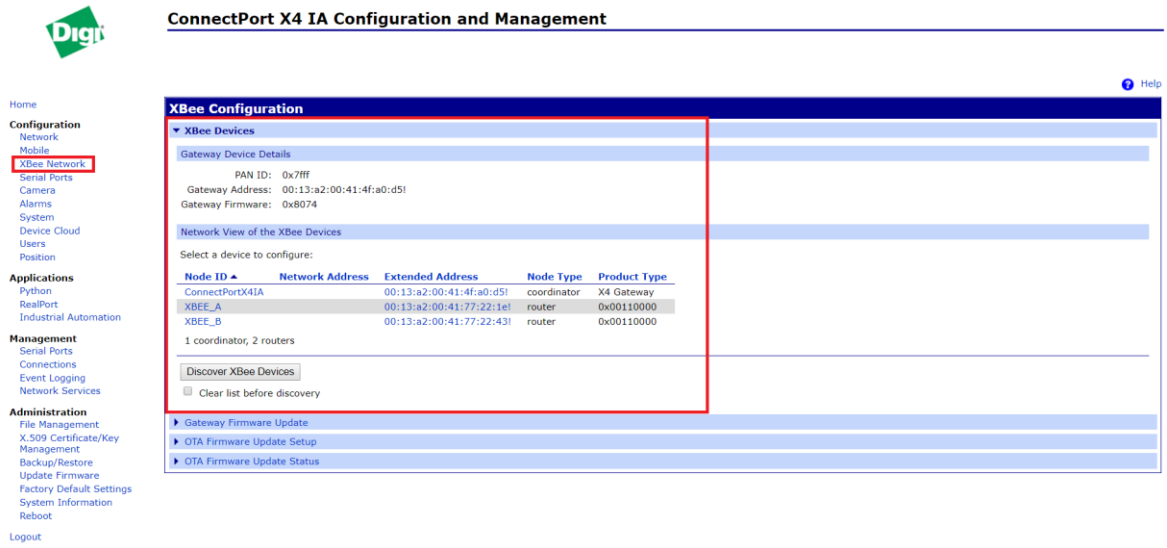


Figura 3.10.- "*XBee Network*".

3.3.3.- *Applications*

"*Applications*" es aquella opción que permite configurar o subir los archivos que indican al ConnectPort X4IA que protocolos o métodos debe de usar para obtener y redireccionar los datos. Dentro de este se pueden comprobar que hay tres opciones y dentro de las mismas solo se ha usado aquella denotada como "*Python*".

"*Python*" permite subir aquellos archivos que se iniciarán en el arranque del dispositivo. En la Figura 3.11 se pueden ver recuadrados los archivos que componen el ConnectPort X4IA, dentro de todos ellos se han usado para posibilitar la configuración los denotados como "868_TCP_send.py" y el "config_cpx4.py" (ver Anexo 8.3).

El resto de archivos como "zigbee.py" o "python.zip" son los necesarios para que el dispositivo entienda el lenguaje Python y contenga la librería ZigBee de manera que se puedan compilar y ejecutar los archivos mencionados en el párrafo anterior.

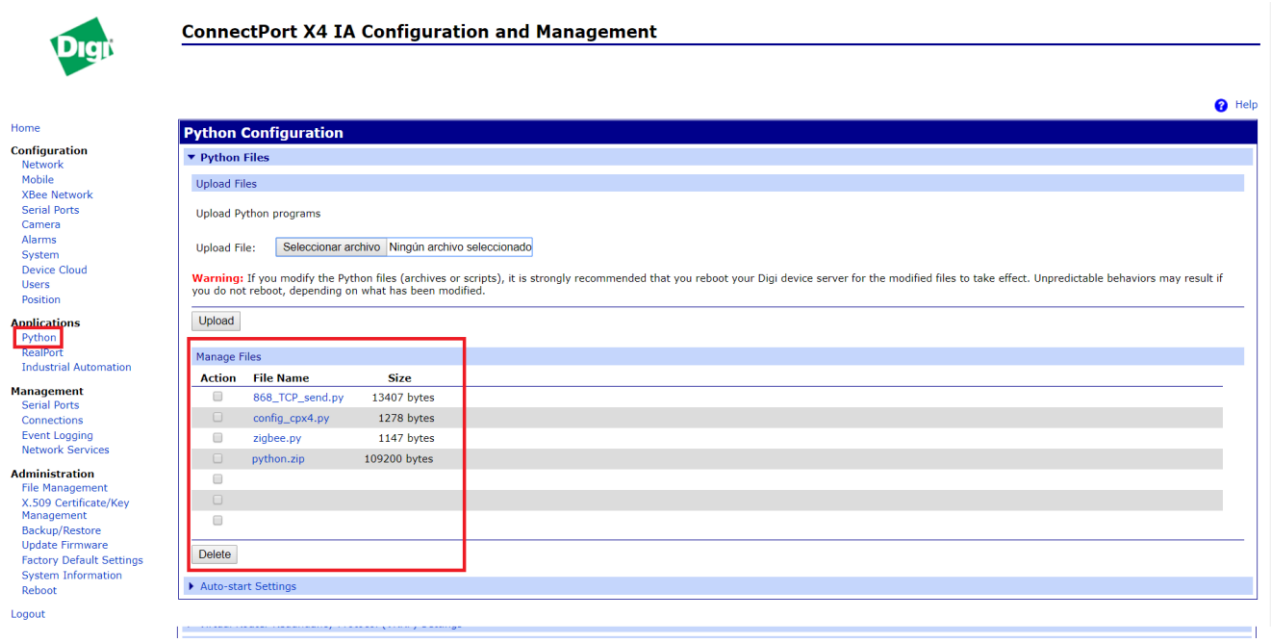


Figura 3.11.- "*Python*".

3.4.- Configuración XBee SX868

El software usado para la configuración de los dispositivos XBee SX868 y permitir así la comunicación entre los mismos y con el Gateway (ConnectPort X4IA) se realiza mediante el *Digi XCTU* de la empresa *Digi*®. En este sub-apartado, además de explicar la configuración de los dispositivos se mostrarán las principales características del programa usado para tal fin.

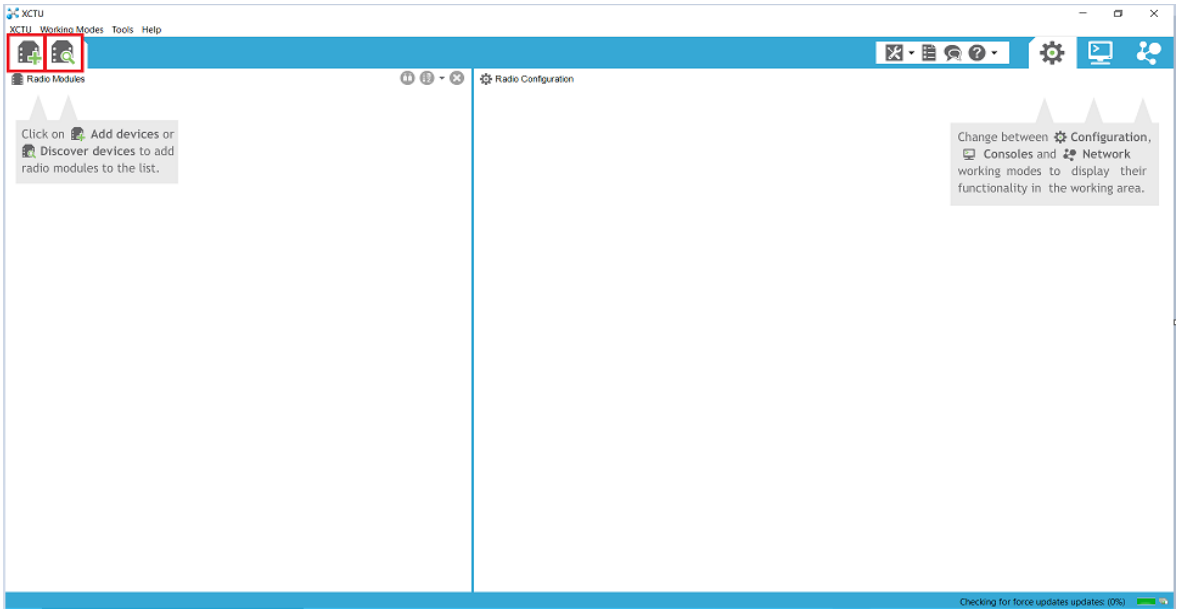


Figura 3.12.- Interfaz gráfica software *XCTU*.

En la Figura 3.12 se puede observar la interfaz principal del programa. Como se destaca en la misma (remarcado en color rojo), se pueden ver las opciones de las que dispone el programa para detectar dispositivos que funcionen bajo el protocolo ZigBee o uno semejante (por ejemplo DigiMesh). La opción de la izquierda se usa para añadir a la red dispositivos de forma manual, para ello se debe de conocer de antemano el nombre del dispositivo y el puerto al que este está conectado, así mismo se deben de configurar las características del puerto serie; si todo se ha realizado de manera correcta el dispositivo debería de aparecer como se puede ver recuadrado en azul en la Figura 3.12. La opción de la derecha (ver Figura 3.12), por el contrario permite buscar dispositivos de manera automática que estén conectados al ordenador, esta característica es útil si se desconoce tanto el nombre como la configuración de este.

Como se puede ver en la Figura 3.13 el método utilizado para localizar los dispositivos XBee de los que consta la red (dos en este caso) ha sido el de la búsqueda automática y en ella se pueden visualizar los pasos realizados. El primero de ellos se basa en seleccionar los puertos USB donde se quiere realizar el análisis; el segundo de ellos se basa en seleccionar aquellas características que tienen los módulos a buscar (tasa de baudios, si hay bit de paridad, bit de stop, etc…), en el caso actual la tasa de baudios es de 9600, los bits de datos son ocho, no hay bit de paridad ni control de flujo y sí hay un bit de stop; el tercer paso es el de añadir a la red aquellos elementos que se correspondan con la búsqueda realizada.

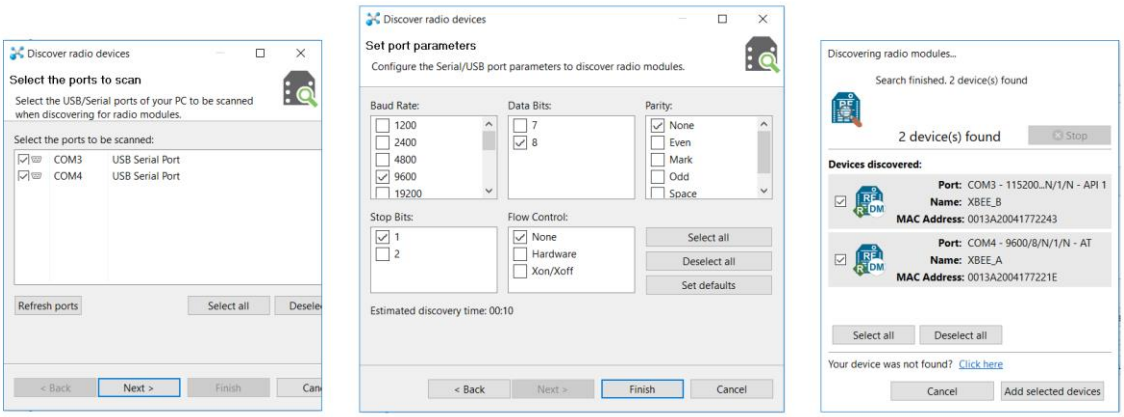


Figura 3.13.- Búsqueda automática de dispositivos en XCTU. (a) Selección del puerto, (b) características del dispositivo y (c) dispositivos descubiertos.

En la Figura 3.14 se pueden ver los módulos encontrados y que conforman la red DigiMesh. En color rojo se recuadra la opción que permite buscar dispositivos de manera remota que forman también parte de la red, pero que no se configuran en principio desde el XCTU. Un claro ejemplo de esto último es el ConnecPort X4IA el cual puede verse recuadrado en color verde en la imagen (se comprueba cómo los dos XBee lo encuentran sin problema.

En color violeta se recuadra el rol que tienen los dispositivos y el protocolo de comunicaciones que usan. Para el caso de los XBee el rol viene determinado por la letra "R" que significa "*Router*" y para el caso del ConnectPort por la letra "C" de "*Coordinador*", por otra parte las iniciales "DM" indican que el protocolo usado en la red y el que viene configurado en los dispositivos es DigiMesh.

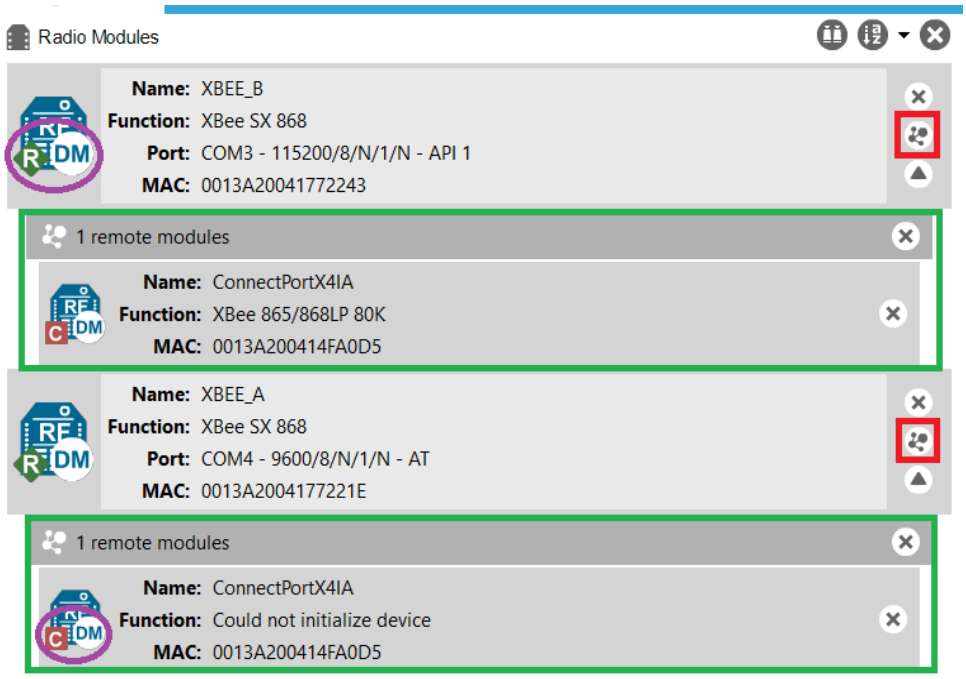


Figura 3.14.- Módulos que componen la red DigiMesh.

A la hora de configurar dos módulos XBee hay dos modos principales de configuración: el modo AT y el modo API. Cuando el dispositivo se encuentra trabajando en modo AT, todos los datos recibidos vía serial se van almacenando en una cola para realizar una transmisión RF; si se reciben los datos vía RF se realiza el proceso inverso. En el modo API el modo de operación es una alternativa al modo AT donde el funcionamiento está basado en el uso de enviar las tramas API (en el modo AT no existen tramas).

Una de las principales ventajas del modo API, es que posee la opción de ver como los módulos efectivamente se encuentran están todos interconectados entre sí (red DigiMesh). Para ello sirve con hacer clic en el botón recuadrado en rojo en la Figura 3.15, en ella se comprueba cómo entre cada dispositivo hay una línea, esta línea indica que entre dos elementos de la red hay conectividad, además el color de la línea indica la calidad de la comunicación (verde sería la mejor conexión y rojo la peor). Encima de cada línea se puede ver la potencia expresada en dBm.

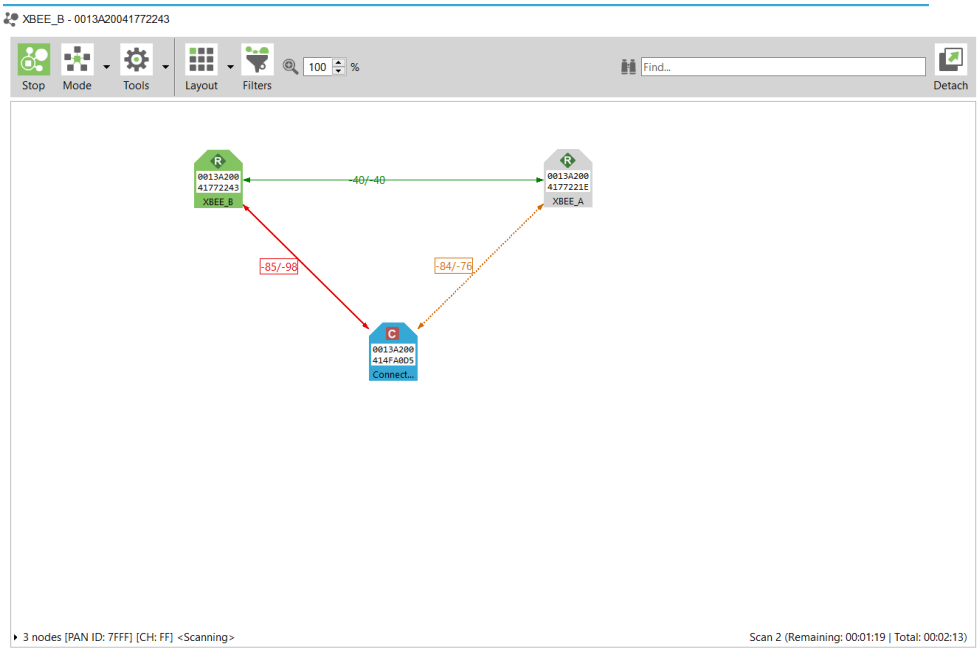


Figura 3.15.- Conexión entre los módulos (red DigiMesh).

La configuración principal de los dispositivos es independiente del modo de trabajo en el que se encuentre, ya que esto último solamente modificará la estructura del envío de datos, mientras que el resto del proceso se realiza de manera independiente.

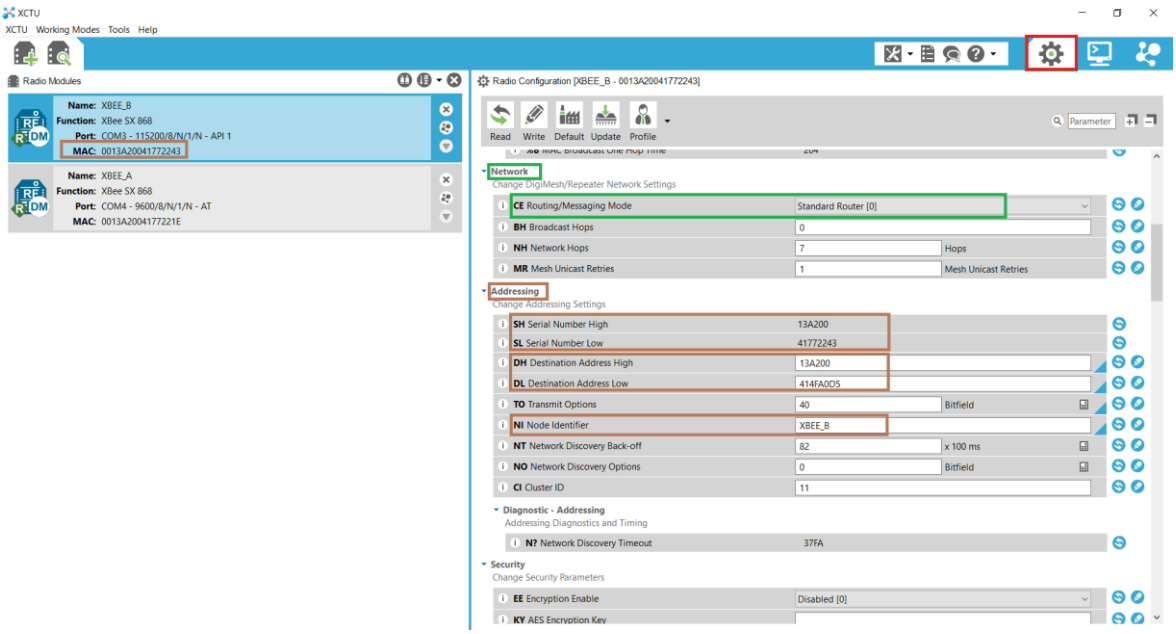


Figura 3.16.- Configuración de un módulo XBee (XCTU)(I).

En la Figura 3.16 se puede observar parte de la configuración necesaria realizada en un XBee. En primer lugar se debe de cliquear en el botón rojo para iniciar el proceso,

seguidamente en la opción recuadrada en verde "*Network"* se puede ver que hay cuatro opciones de configuración, en la primera de ellas *"Routing/Message Mode"* se elige el rol que tiene el dispositivo configurado dentro de la red, en este caso *"Standard Router"* (en el caso del ConnectPort X4IA sería *"Coordinator"*).

Recuadrado en color marrón y en el apartado *"Addressing"* se puede ver la dirección de identificación del dispositivo *"Serial Number"* que, como es lógico, coincide con la dirección MAC del XBee. Además, en la opción *"Destination Address"* se puede ver la dirección del ConnectPort X4IA (ver Figura 3.14 dirección MAC) ya que esta será el dispositivo final al que se debe de hacer llegar los datos, bien de manera directa desde el dispositivo XBee o bien a través de otro elemento de la red. Para finalizar, en *"Node Identifier"* se puede poner un nombre al dispositivo de cara a diferenciarlos del resto.

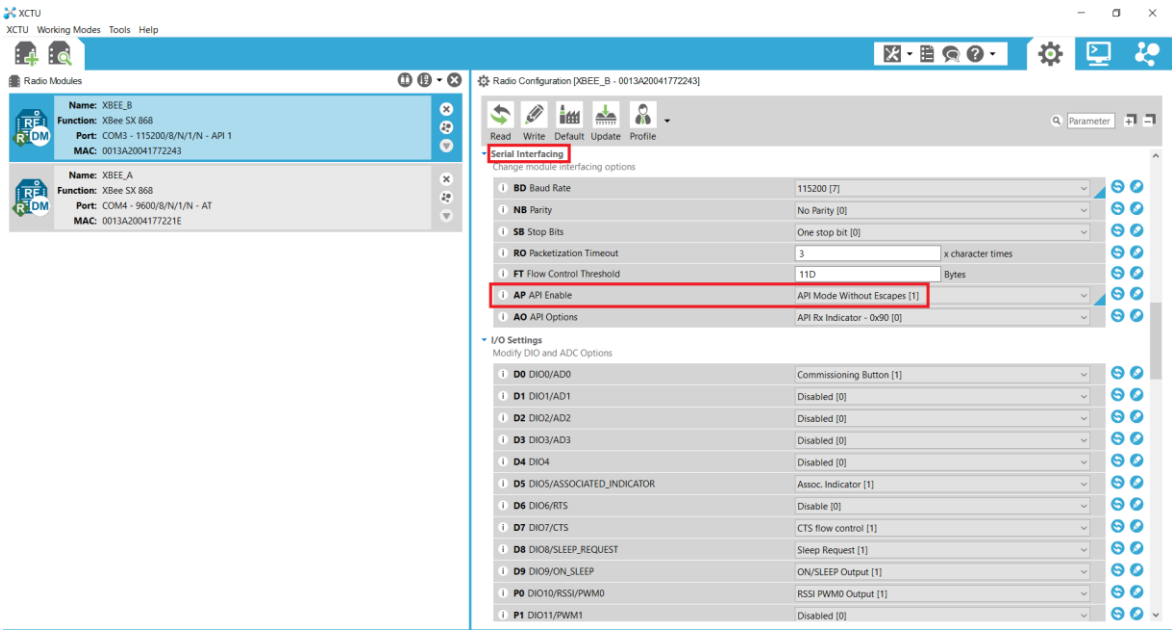


Figura 3.17.- Configuración de un módulo XBee (XCTU)(II).

En la Figura 3.17, en la opción "*Serial Interfacing*", se puede elegir el modo de trabajo del dispositivo que puede ser API o AT. La principal ventaja del modo AT es que permite realizar conexiones punto a punto de manera inmediata y no necesita de una estructura de paquetes como el modo API donde los datos deben de incorporarse en tramas predefinidas para ser transmitidas.

3.4.1.- API frame

A la hora de crear una trama desde el modo API, se deben de realizar los siguientes pasos.

Una vez se tenga el dispositivo con el que se quiera enviar la trama identificado, seleccionado y configurado en modo API, se debe de hacer clic en el botón recuadrado en rojo en la Figura 3.18.

Seguidamente se debe de seleccionar la opción de añadir nueva trama (botón violeta en Figura 3.18). Una vez realizado este paso aparecerá una ventana que permite seleccionar el tipo de trama; en el caso de la figura se ha elegido el tipo "*0x10 Transmit Request*" con la finalidad de crear un mensaje *broadcast* que permita comprobar que la red funciona de manera correcta.

Dentro de la trama creada se debe de especificar que se trata de un mensaje *broadcast*, para ello se deben de rellenar los parámetros recuadrados en color marrón en la Figura 3.18 con la información visible en la misma.

Una vez creada la trama, se puede editar la misma sin más que hacer doble clic sobre la misma dentro del campo "*Send Frames*". Además, se puede seleccionar el periodo con el que se quiere enviar la trama, para ello hay que ir al campo "*Send sequence*" y seleccionar el tiempo deseado; se destaca que también se puede enviar un número determinado de tramas o enviar de manera continua mientras el dispositivo se encuentre alimentado.

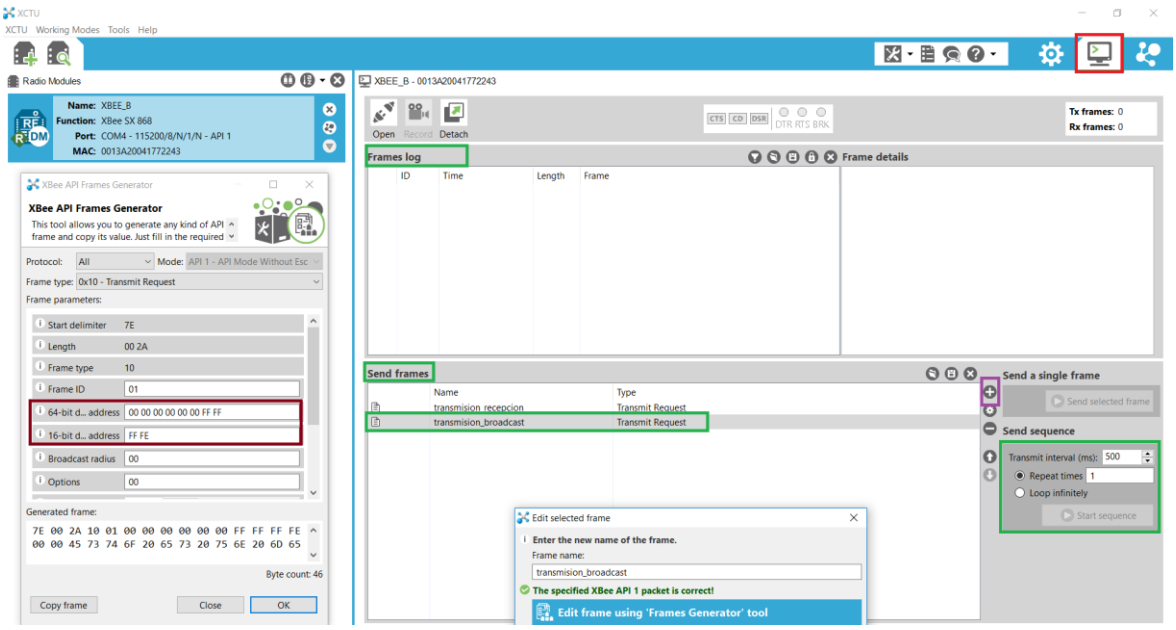


Figura 3.18.- Trama modo API.

3.5.- Configuración Arduino NANO y sensor RGB

En este apartado se mostrarán tanto el conexionado físico entre el Arduino NANO y el sensor RGB. El código de Arduino usado se encuentra visible en el Anexo 8.2.

En la Figura 3.19 se puede observar cómo sería el conexionado físico entre el sensor RGB y el microcontrolador. En ella se puede observar cómo no es necesario ningún elemento extra como pudiesen ser resistencias o condensadores, ya que ambos elementos ya vienen adaptados para tal fin.

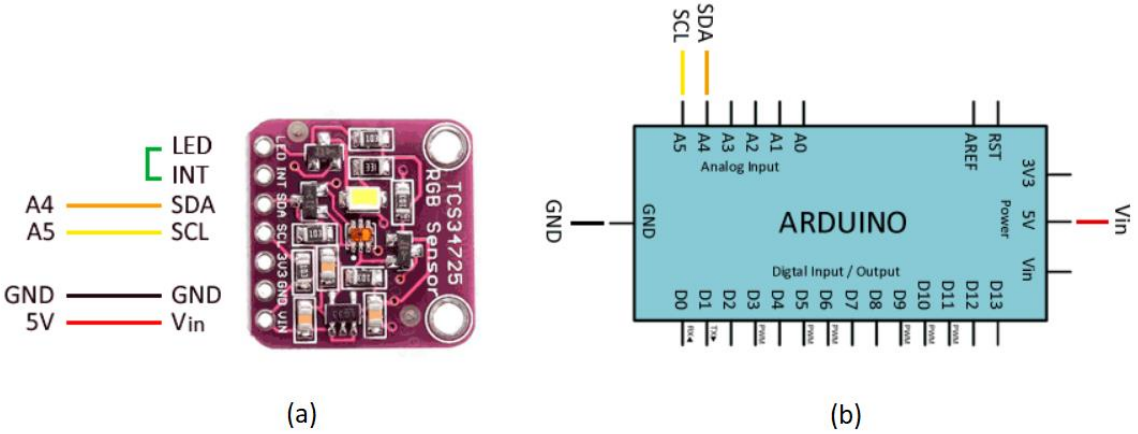


Figura 3.19.- Conexionado Arduino NANO-sensor RGB. (a) Conexionado desde el sensor, (b) conexionado desde el Arduino.

Como se puede comprobar a partir de la Figura 3.19, el protocolo de comunicación entre los dos dispositivos se basa en el uso de I2C ya que hace uso de los pines típicos del mismo que son SDA (datos) y SCL (reloj). El conexionado es bastante sencillo y es suficiente con unir mediante cables/pistas de cobre los pines que se ven en la Figura anterior.

En la Figura 3.20 se puede ver el montaje real, realizado sobre una protoboard.

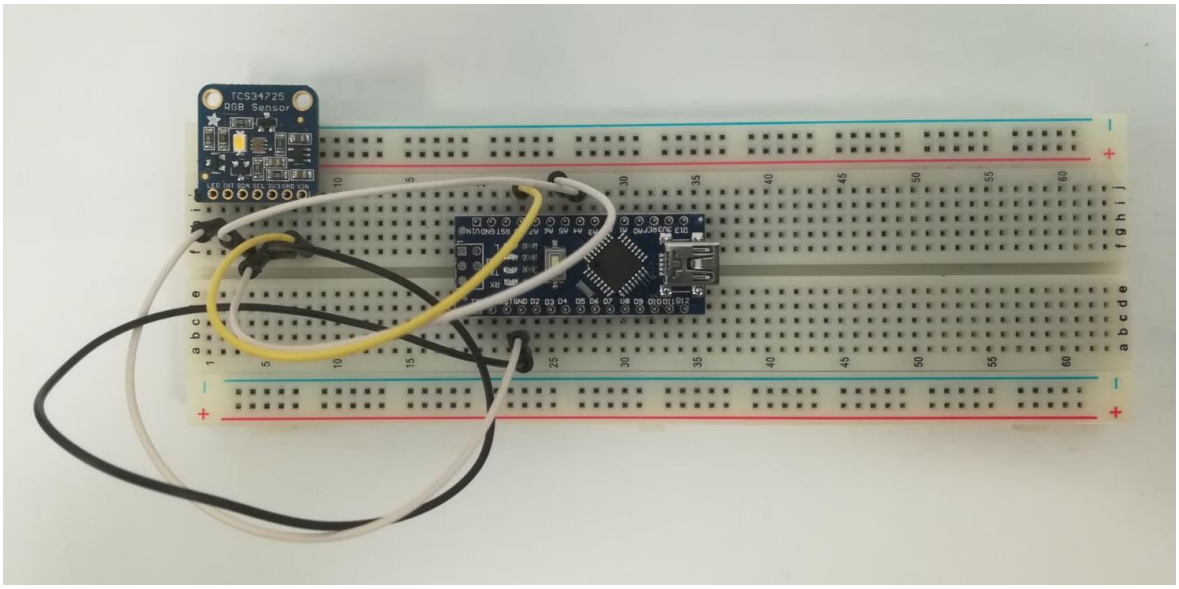


Figura 3.20.- Montaje en protoboard de Arduino NANO y sensor RGB.

El software usado para posibilitar la comunicación entre los elementos mencionados es el *Arduino Genuino*, cuya interfaz gráfica puede verse en la Figura 3.21. En ella (recuadrado en color rojo) se puede ver como se debe de elegir el tipo de modelo de Arduino con el que se está trabajando, el procesador del mismo y el puerto COM al que está conectado.

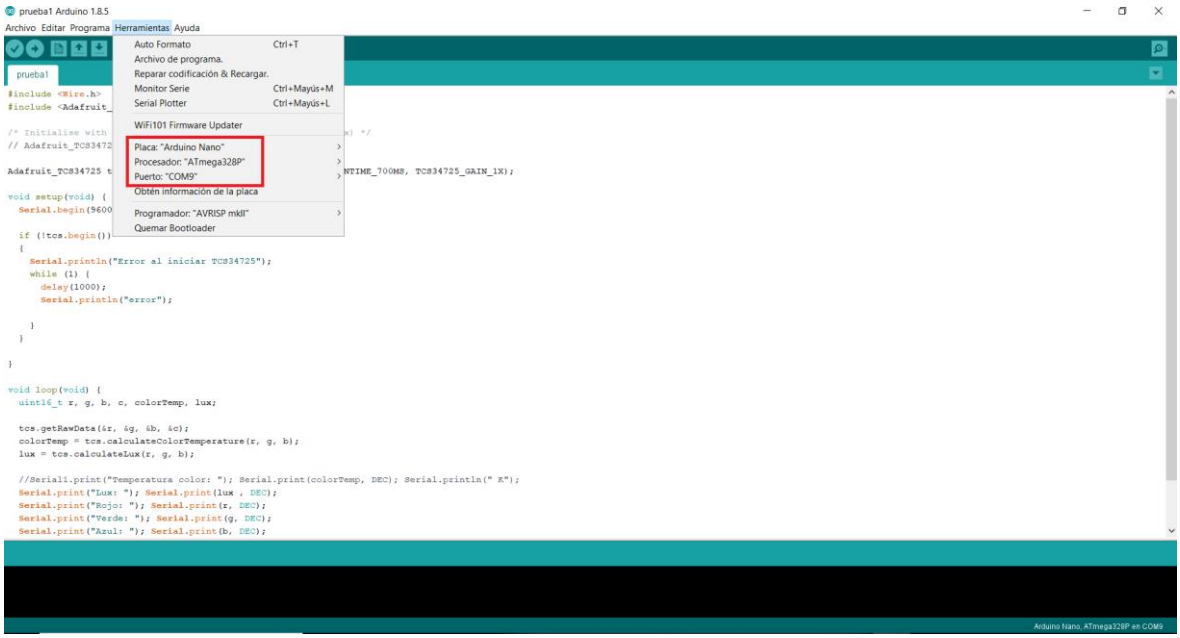


Figura 3.21.- Interfaz gráfica *Arduino Genuino*.

El código usado se encuentra en el Anexo 8.2 y basa su funcionamiento principalmente en el uso de la librería "Adafruit_TCS34725" desarrollada por *Adafruit*® para el sensor RGB. A partir de la misma se pueden obtener de una manera sencilla aquellos valores que proporciona el sensor como son la temperatura de color, la cantidad de color rojo, verde, la luminancia, etc…

```
Serial.print("Temperatura color: "); Serial.print(colorTemp, DEC); Serial.println(" K");
Serial.print("Lux: "); Serial.print(lux , DEC);
Serial.print("Rojo: "); Serial.print(r, DEC);
Serial.print("Verde: "); Serial.print(g, DEC);
Serial.print("Azul: "); Serial.print(b, DEC);
Serial.print("Clear: "); Serial.print(c, DEC);
```

Figura 3.22.- Datos proporcionados por el sensor RGB TCS34725.

En la Figura 3.23 puede observar como el sensor y el Arduino funciona de manera correcta y donde los porcentajes de cada color se van modificando en cada medida ya que las medidas se han realizado en distintas condiciones de luz.

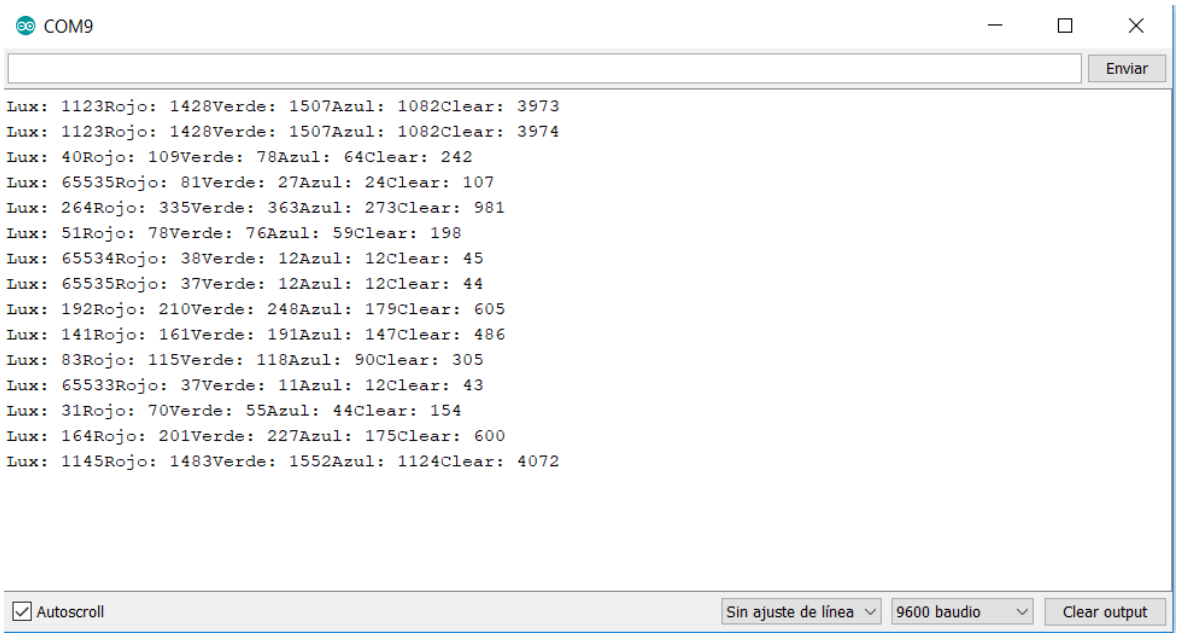


Figura 3.23.- Visualización de los datos del sensor RGB TCS34725.

3.6.- Configuración *KEPServerEX*

El servidor usado para visualizar los datos recogidos por la *Gateway* ha sido *KEPServerEX*. Para realizar la configuración, además del propio programa se ha usado el software *Wireshark* para así poder conseguir información que será de utilidad a la hora de realizar una correcta configuración.

3.6.1.- *Wireshark*

En la Figura 3.24 se comprueba como el programa *Wireshark* permite captar los datos que provienen de cualquier puerta de entrada o salida del ordenador en el que está instalado. En este caso se puede observar como hay cinco opciones: "*VMware Network Adapter VMnet1*", "*VMware Network Adapter VMnet8*", "*Ethernet2*", "*Wi-Fi*", "*Conexión de área local* 13*" y "*USBCap1*".

En el caso actual el tráfico que resulta interesante de analizar se encuentra en la opción "*Ethernet2*", por ser el medio que comunica el ConnectPort X4IA con el ordenador donde se encuentra corriendo el servidor.

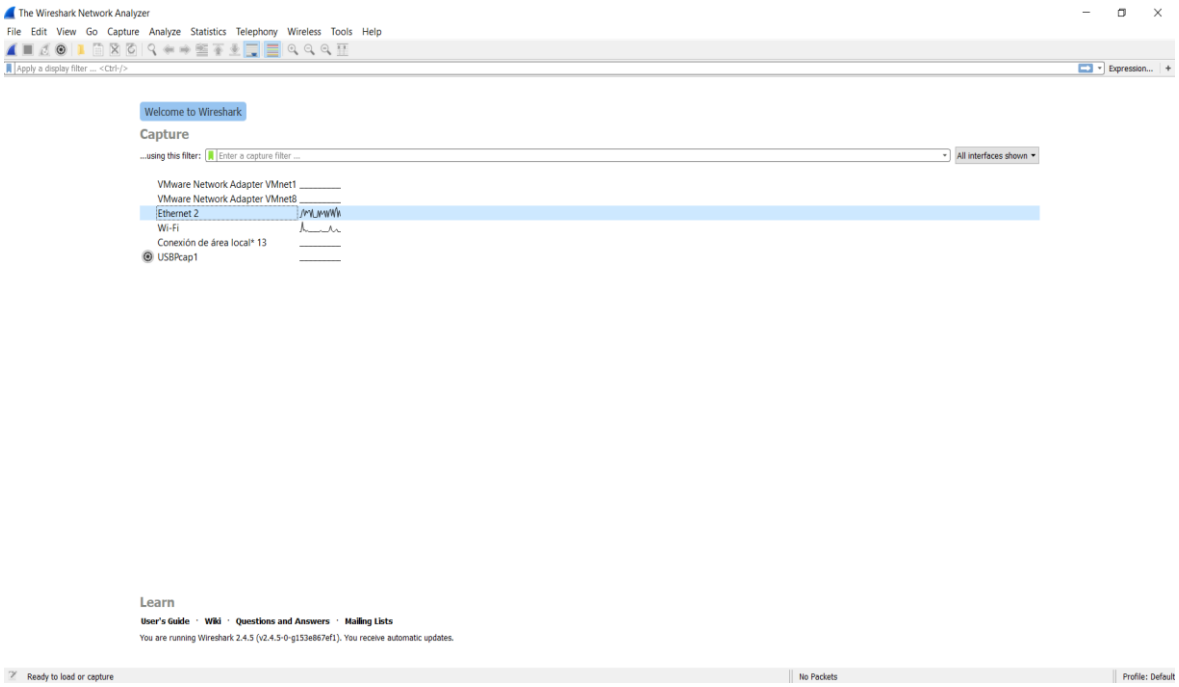


Figura 3.24.- Interfaz principal Wireshark.

Sabiendo que la dirección del XBee usado en el caso analizado es la "[00:13:a2:00:41:77:22:1e]!" y que esta, como se puede comprobar en el código "config_cpx4.py" (Anexo 8.3.2) está asociada con el puerto 50012. Por ello, se ha realizado un filtro en el *Wireshark* para determinar los datos que se envían hacia el *KEPServerEX* a través del XBee.

El filtro realizado se puede ver resaltado en color verde en la Figura 3.25 y está definido de la siguiente manera: "udp.port==50012". Una vez activado el mismo, se puede comprobar como la dirección fuente se corresponde con la dirección IP 192.168.1.1 y la dirección destino con la IP 192.168.1.100. Estas direcciones se corresponden respectivamente con la dirección de la *Gateway* y del *KEPServerEX*.

Además, en la Figura 3.25, se puede observar recuadrado en color rojo la longitud de los datos que viajan a través del puerto mencionado. Dicha longitud es de 54 bytes y como se verá más adelante será un dato necesario para realizar la configuración de tal manera que la visualización en el servidor se haga de manera correcta.

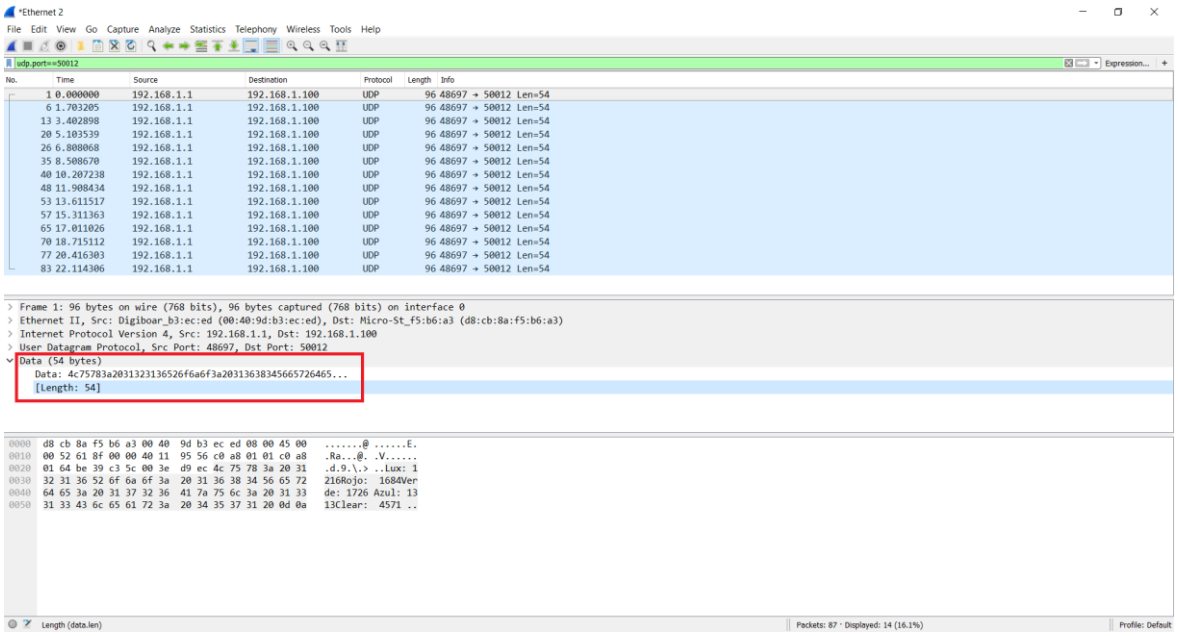


Figura 3.25.- Análisis de tramas en *Wireshark*.

3.6.2.- *KEPServerEX*

Una vez recopilados datos como la longitud de los datos recibidos o la dirección de destino se procede a la configuración del *KEPServerEX*. Para ello se deben de realizar los pasos que se enumeran a continuación.

- **Adición de un canal**

En la Figura 3.26 se puede comprobar como en primer lugar hay que añadir un canal. A pesar de que hay diferentes opciones de configuración, la que mejor se ajusta a lo que se quiere realizar (comunicación mediante sockets) es la opción de "*USER-CONFIGURABLE*" (U-CON) ya que es la que está más relacionada con los protocolos de comunicación TCP y UDP.

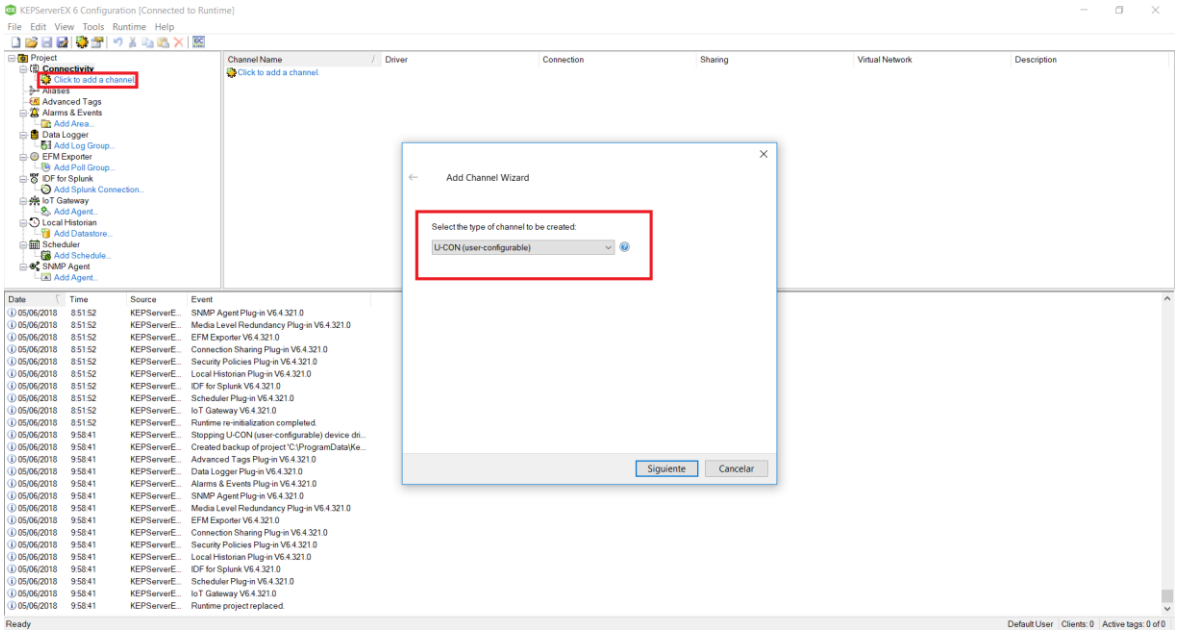


Figura 3.26.- Creación de un canal (I).

Seguidamente y como se comprueba en la Figura 3.27 se deben de elegir el medio físico (Ethernet), el adaptador de la red (identificado por la IP del servidor 192.168.1.100), el estilo del driver de la comunicación (no solicitado), el puerto (50012) y el protocolo de comunicación (UDP).

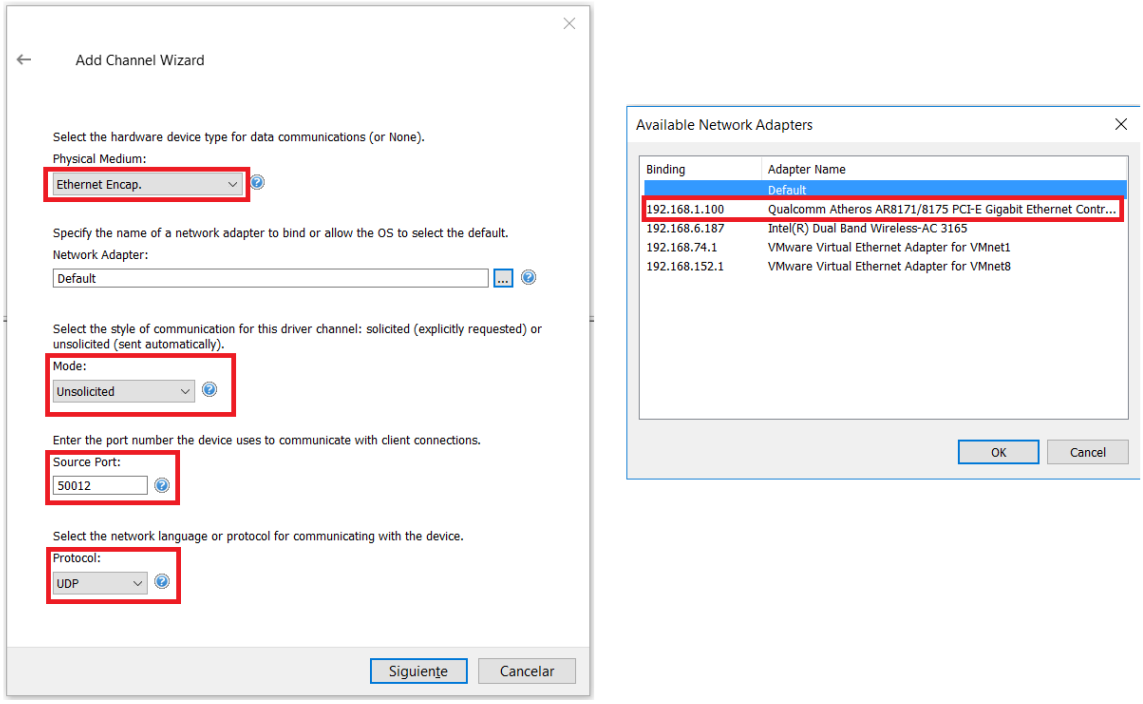
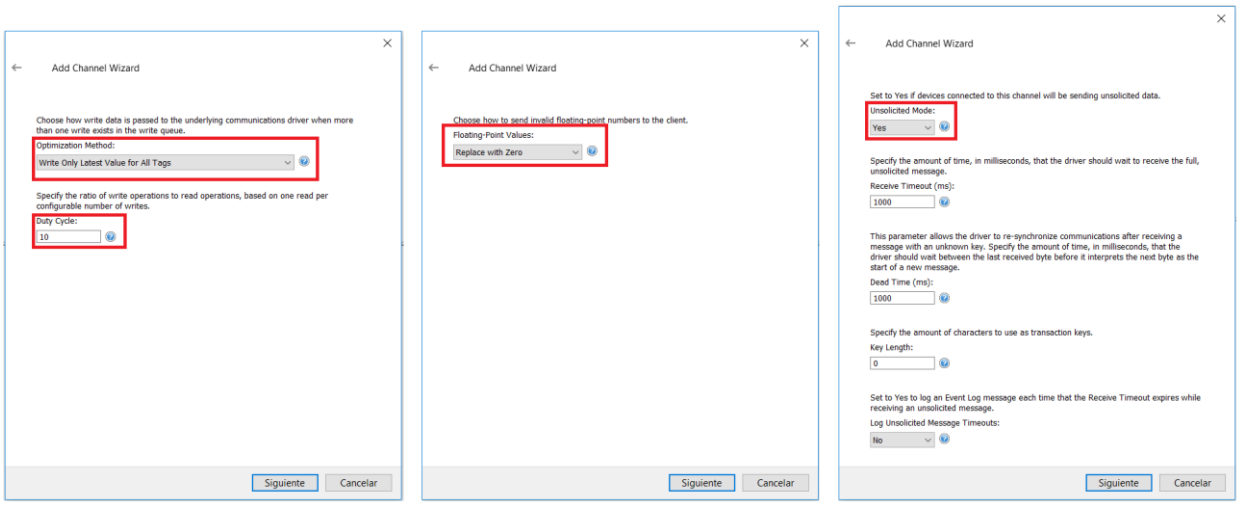


Figura 3.27.- Creación de un canal (II).

En la Figura 3.28 (a) se elige el método de optimización "*Write Only Latest Value for All Tags*" ya que es la opción más útil cuando se quiere visualizar siempre el último valor enviado por el dispositivo, además el ciclo de trabajo se fija a 10 por ser el valor recomendado para este método.

La opción de "*Replace with Zero*", en la sección de los valores de los puntos flotantes (Figura 3.28 (b)), se debe a que los valores flotantes no normalizados en ocasiones se interpretan como valores infinitos, por tanto para evitar esto se sustituyen por ceros antes de enviarlo a los clientes. Por último en la Figura 3.28 (c), se activa el modo no solicitado ya que el servidor no tiene que realizar peticiones para recibir datos de los dispositivos de la red DigiMesh.

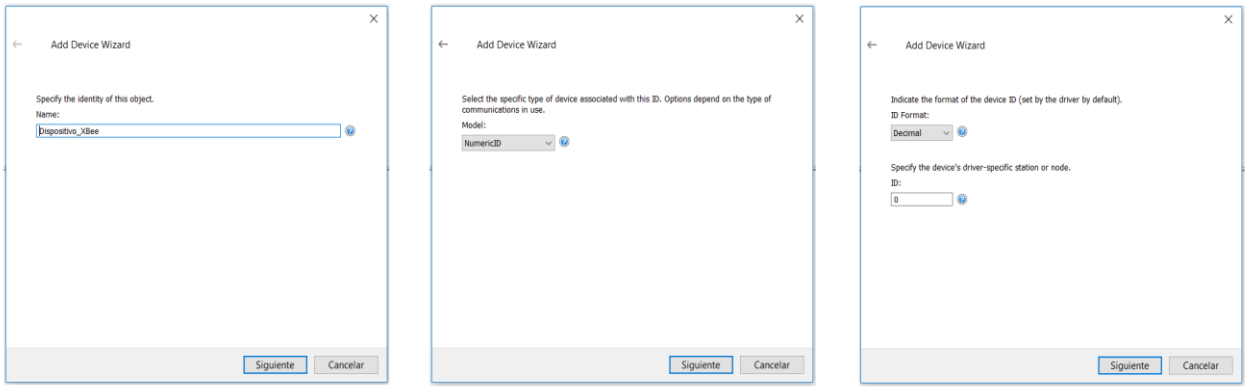


(a) (b) (c)

Figura 3.28.- Creación de un canal (III).

- **Adición de un dispositivo**

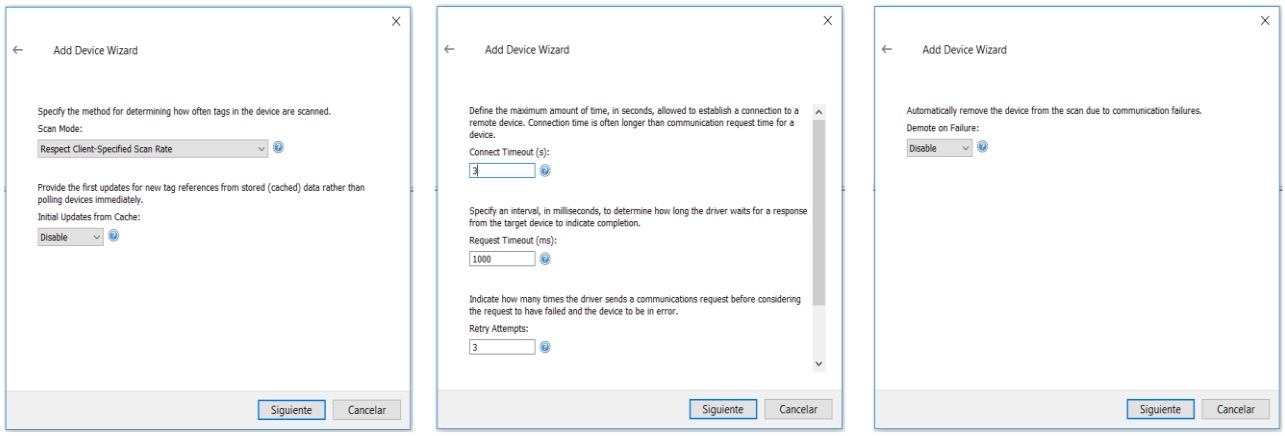
En la Figura 3.29 se muestran los primeros pasos que se deben de realizar para la configuración de un dispositivo. En ella se comprueba (de izquierda a derecha) como se le debe de dar un nombre al dispositivo, se debe de indicar si su ID es numérica o texto y en caso de ser numérica (como es el caso analizado) el formato del mismo y el número que se le asigna al dispositivo.



(a) (b) (c)

Figura 3.29.- Creación de un dispositivo (I).

En la Figura 3.30 se observa cómo se debe de elegir el modo de escaneo, en este caso se respeta la tasa de envío determinada por el cliente y se desactivan las actualizaciones desde caché. Se especifica el tiempo de máximo para establecer una comunicación con un dispositivo remoto (en este caso tres segundos), el intervalo de espera por una respuesta (un segundo) y el número de intentos para comprobar que una conexión ha fallado (tres intentos). Además una vez la conexión está caída, esta se elimina de manera automática.



(a) (b) (c)

Figura 3.30.- Creación de un dispositivo (II).

En la Figura 3.31 se indica como en caso de que haya una etiqueta duplicada, la opción a realizar es la de eliminarla, además se activa la opción de crear sub-grupos para la generación de etiquetas nuevas. Para finalizar se especifica el tiempo máximo que se debe de esperar por una petición no solicitada (en este caso un segundo).

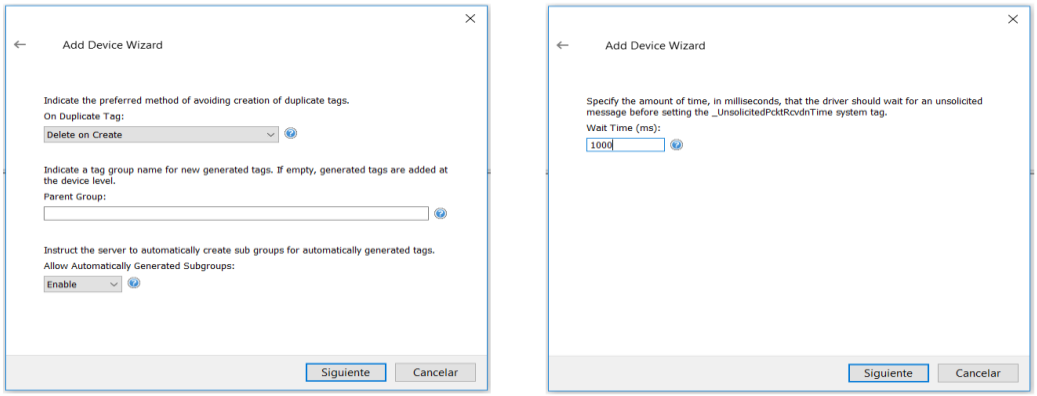


Figura 3.31.- Creación de un dispositivo (III).

- **Configuración de un dispositivo**

Una vez creado el dispositivo se procede a la configuración de este para leer de manera correcta la información enviada por el sensor. Para ello y como se puede comprobar en la Figura 3.32 se procede a crear un nuevo bloque de *tags* (denotado como "*Unsolicited*" en Figura 3.32 (b)) y dentro de este se realiza la creación de una orden de lectura ("*Read Response*").

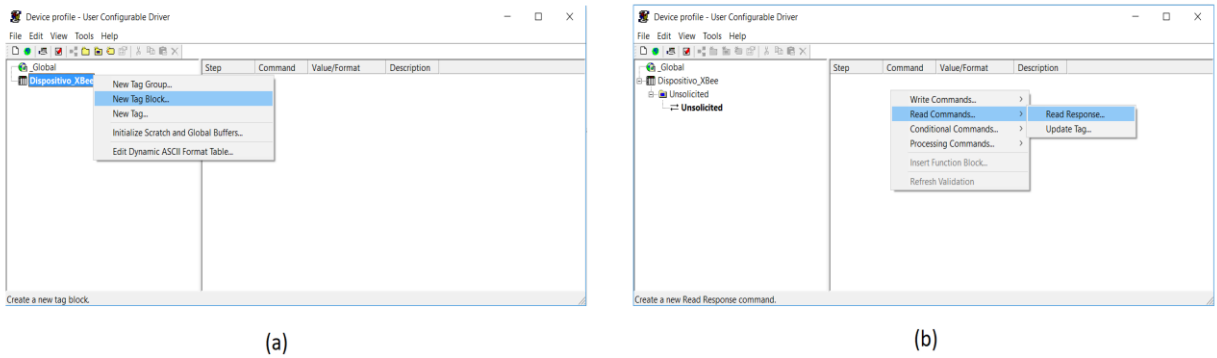


Figura 3.32.- Configuración de un dispositivo (I).

Seguidamente se procede a la creación de dicho comando. Para ello y como se puede visualizar en la Figura 3.33 (recuadrado en color rojo) se elige el tipo de trama *"Frame has known length"* ya que gracias al uso del programa Wireshark se sabe que la longitud de los datos es de 54 bytes. Este dato se introduce en el campo *"Number of bytes"* y además se dejan activadas por defecto las opciones de *"Clear RX buffer before read"* y *"Log timeout errors"*.

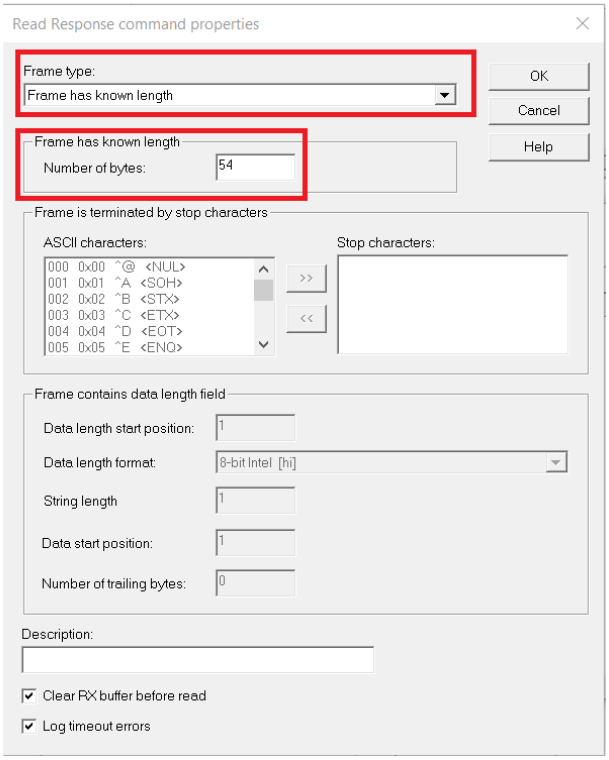


Figura 3.33.- Configuración de un dispositivo (II).

Una vez creada y configurada la orden de lectura se procede a la adición de un *tag* al bloque previamente creado. Para ello y como se puede ver en la Figura 3.34 (a) se nombra al *tag*, se selecciona el tipo (en el caso actual de tipo "*string*" ya que es lo que envía el XBee) y el formato del mismo (*"ASCII String"*). Se selecciona además la extensión del *string* (Figura 3.34 (b)).

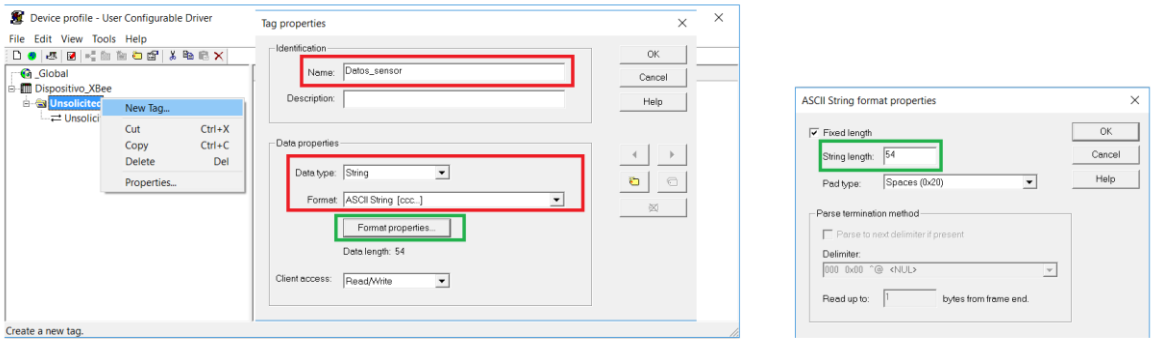


Figura 3.34.- Configuración de un dispositivo (III).

Para finalizar la configuración del dispositivo se debe de indicar como se debe de actualizar el *tag* creado. Para ello, se debe de seleccionar la opción "*Update Tag*" (Figura

3.35 (a)) y seguidamente elegir la etiqueta a configurar, siendo en este caso "Datos_sensor". Posteriormente se elige desde donde se deben de actualizar los datos, en este caso desde el *buffer* de lectura y desde que byte recibido se debe de actualizar la información (desde el primero de ellos ya que es interesante visualizar todo lo recibido).

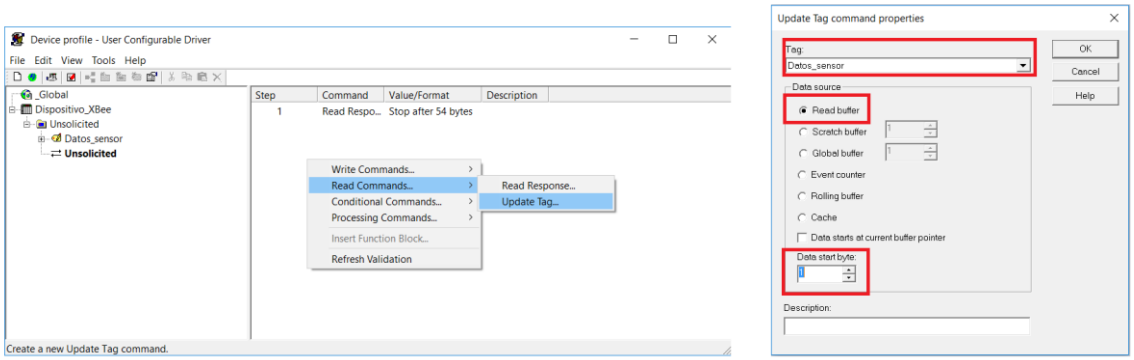


Figura 3.35.- Configuración de un dispositivo (IV).

Se destaca que para posibilitar la comunicación entre la *Gateway* y el servidor se deben de abrir aquellos puertos por los que fluya la misma, ya que en caso contrario los datos serían rechazados y no se podrían visualizar.

4.- DISEÑO DE UN PROTOTIPO

En este capítulo se explicará el diseño del subsistema compuesto por el Sensor RGB, el Arduino NANO y el XBee. Inicialmente se planteó una primera versión del prototipo basado en placa protoboard (ver punto 3.4.-Configuración Arduino NANO y sensor RGB) y al comprobar que esta funcionaba de manera correcta se procedió al diseño y construcción de una placa basada en circuito impreso PCB ("*Printed Circuit Board*").

4.1.- PCB: diseño y construcción

Para realizar los dos prototipos a los que se hace mención en este sub-apartado se ha usado el programa de diseño *Autodesk Eagle*. Este programa, al igual que cualquier otro del mismo fin basa su funcionamiento en el uso de librerías de componentes de distintos fabricantes y características.

Aquellos componentes usados que no se encontraban implementados en las librerías han sido realizados mediante la herramienta que dispone el software para crear tanto la huella del dispositivo como su símbolo en el esquemático.

4.1.1.- PCB alimentada mediante toma de red

En la Figura 4.1 se puede ver el esquemático realizado y que servirá como base para posteriormente realizar la mencionada PCB. Los principales elementos que componen el prototipo pueden verse en la Tabla 4.1.

PCB alimentada mediante toma de red
Conector JACK hembra
Condensador desacoplo
Arduino Nano
Sensor RGB TCS34725
Tira de pines 2 mm
Tira de pines de 2.5 mm

Tabla 4.1.- PCB alimentada mediante toma de red: componentes.

Se puede comprobar como todos los componentes incluidos en la Tabla 4.1 se ven reflejados en la Figura 4.1. Con respecto al esquemático, se puede ver cómo están relacionados entre sí los componentes sin más que fijarse en el nombre de las pistas que salen de cada uno de ellos (aquellas pistas que tienen el mismo nombre están unidas).

La alimentación principal de la placa desarrollada viene dada por un conector JACK hembra a través del cual se proporciona una tensión de entrada continua de 12 V, esta tensión es la encargada de alimentar el Arduino Nano (ya que como se puede observar en las especificaciones técnicas del mismo esta tensión está dentro de los rangos permitidos). Una vez alimentado el Arduino, este será el encargado de alimentar el sensor TCS34725 mediante su salida de 5V.

La comunicación entre el sensor y el Arduino se realiza mediante el protocolo I2C (ver líneas SDA y SCL) y entre el Arduino y el sensor XBee se realiza mediante UART, es decir usa simplemente las líneas de TX y RX de ambos dispositivos para posibilitar la comunicación (TX de XBee con RX de Arduino y viceversa). La velocidad con la que se transmiten los datos, viene dada por el software usado en el microcontrolador.

En esta primera versión el módulo XBee se alimenta de manera independiente al Arduino y al sensor. Esto se debe principalmente a que la placa de desarrollo proporciona unos elementos extra de seguridad que garantizan la protección del mismo ante por ejemplo sobretensiones o sobrecorrientes.

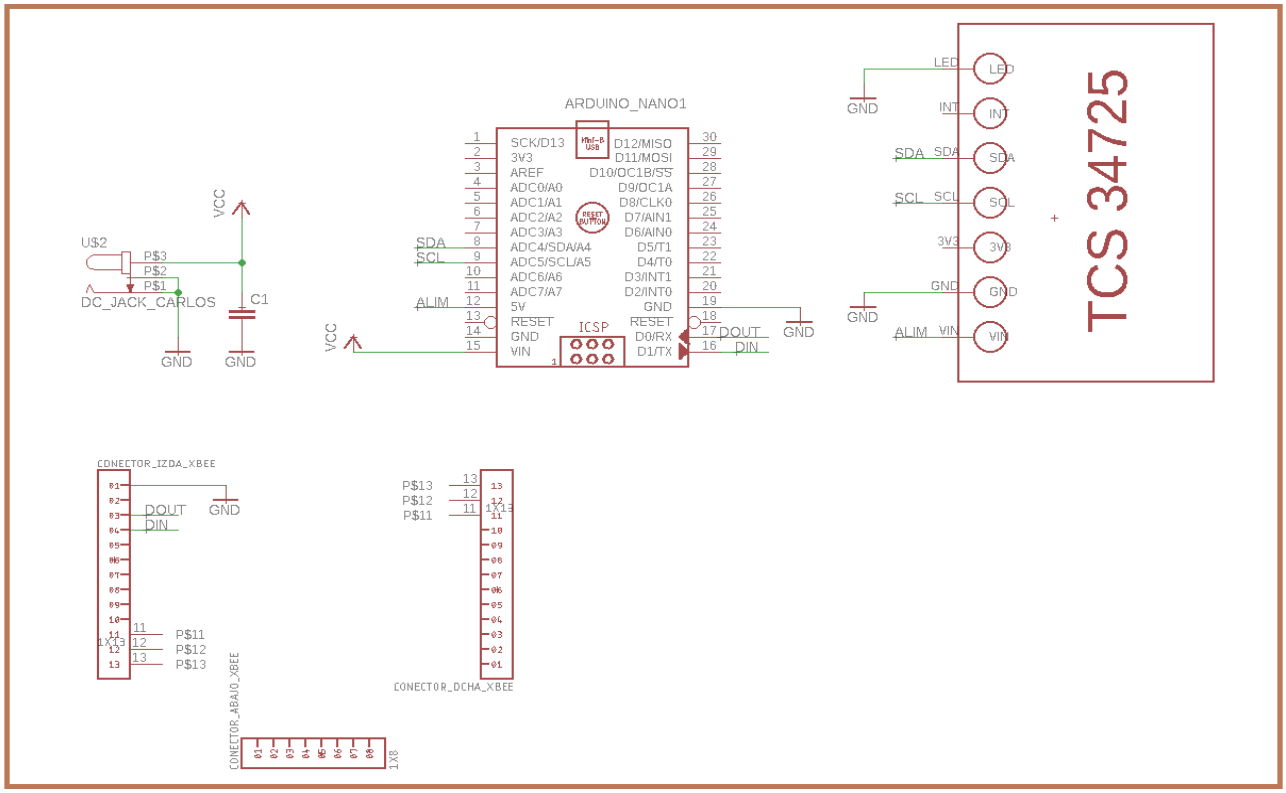


Figura 4.1.- Esquemático PCB alimentada mediante toma de red.

En la Figura 4.2 se puede visualizar el diseño de la PCB y la disposición de elementos de la misma. El trazado de pistas de alimentación "*VCC*" y de masa "*GND*" se ha realizado con un grosor de 50 mils (por ser por regla general aquellas que más problemas pueden generar), el resto de pistas se ha trazado con uno de 25 mils.

Se puede comprobar como todos los elementos se han situado en la capa "*TOP*" mientras que las pistas se han realizado por ambas (tanto capa "*TOP*" (color rojo) como por la "*BOTTOM*" (color azul)).

La distancia entre los componentes y el tamaño de la PCB ha sido el menor posible siempre garantizando la compatibilidad con la placa de desarrollo de los XBee. Además se ha intentado minimizar el área con la finalidad de evitar posibles bucles de corriente.

No se ha realizado el diseño de un plano de masa como tal, sino que se ha aprovechado el que ya incorporan las placas de desarrollo XBee y al que se une mediante la tira de pines de 2mm.

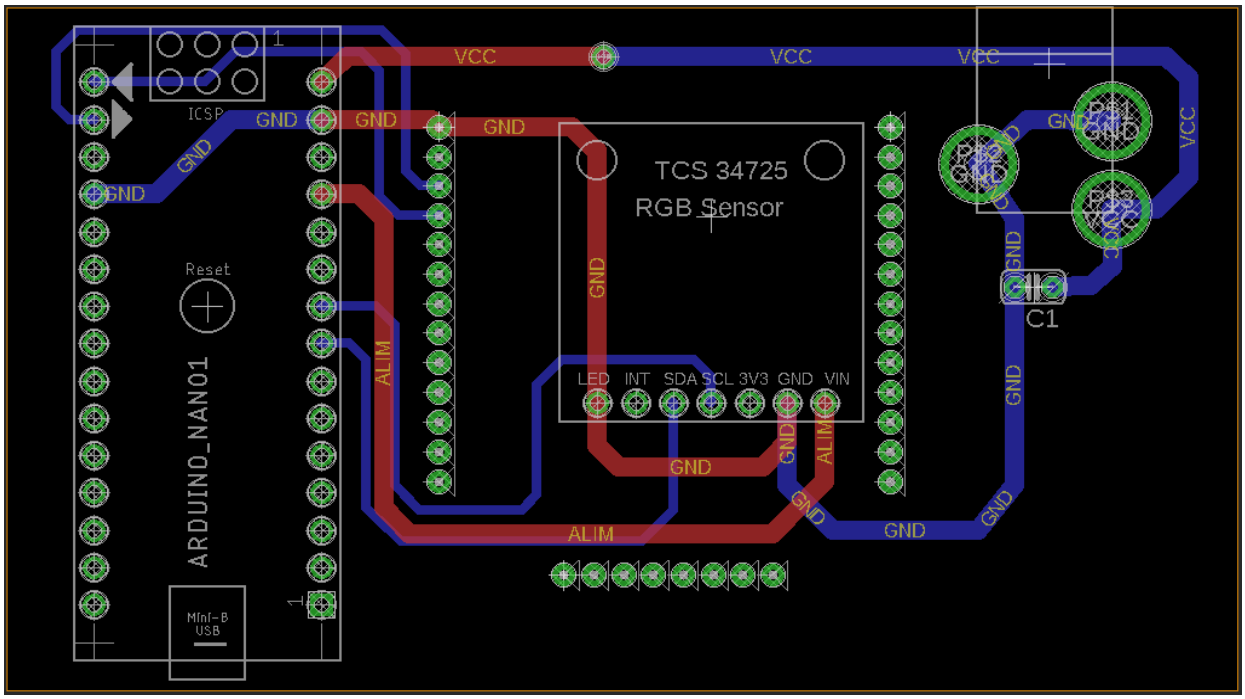


Figura 4.2.- Diseño PCB alimentada mediante toma de red.

En la Figura 4.3 se puede observar el resultado de la PCB final

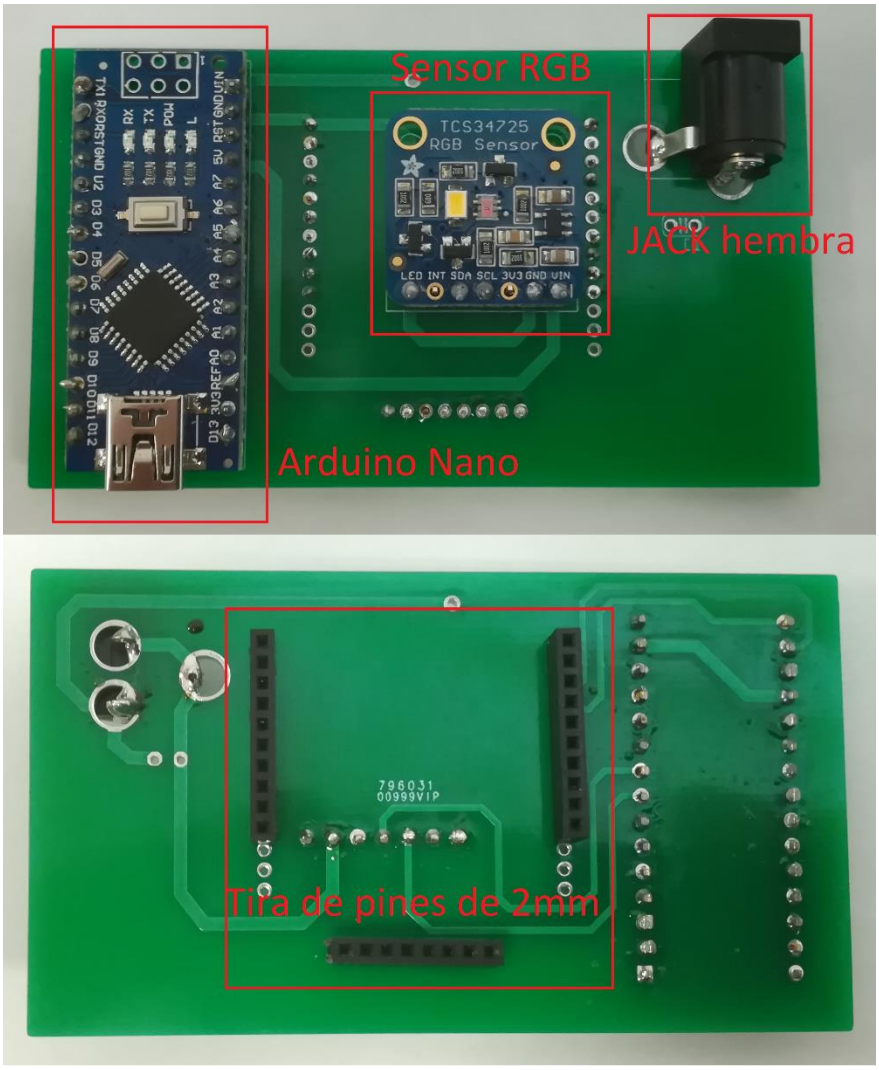


Figura 4.3.- PCB final. Capa "*TOP*" arriba y capa "*BOTTOM*" abajo.

4.1.2.- PCB alimentada mediante batería

En este sub-apartado se ha realizado el esquemático y el diseño de una PCB portátil. Dicha PCB está alimentada solamente por una batería recargable que proporcionaría una autonomía mínima de varias horas.

En la Figura 4.4 se puede visualizar el esquemático realizado. Se comprueba como a diferencia del esquemático de la Figura 4.1, este ya no consta de alimentación mediante conector JACK hembra sino que está alimentado mediante dos baterías 18650 de 3.7 V conectadas en serie, desaparece también la tira de pines de 2mm (ya que en este caso no se usará la placa de desarrollo XBee) y aparece el módulo XBee y un convertidor reductor de voltaje que proporcionaría a su salida la tensión necesaria para alimentar el módulo inalámbrico.

La principal novedad es la inclusión del convertidor reductor y para un rango de tensiones de entrada comprendido entre los 4 y los 36 V de voltaje continuo [36] provee siempre a su salida una tensión constante de 3.3V que permite la alimentación del XBee sin necesidad del uso de la placa de desarrollo. Se ha elegido un convertidor en lugar de un regulador de tensión o un divisor resistivo por ser este mucho más eficiente desde el punto de vista del rendimiento que los otros dos.

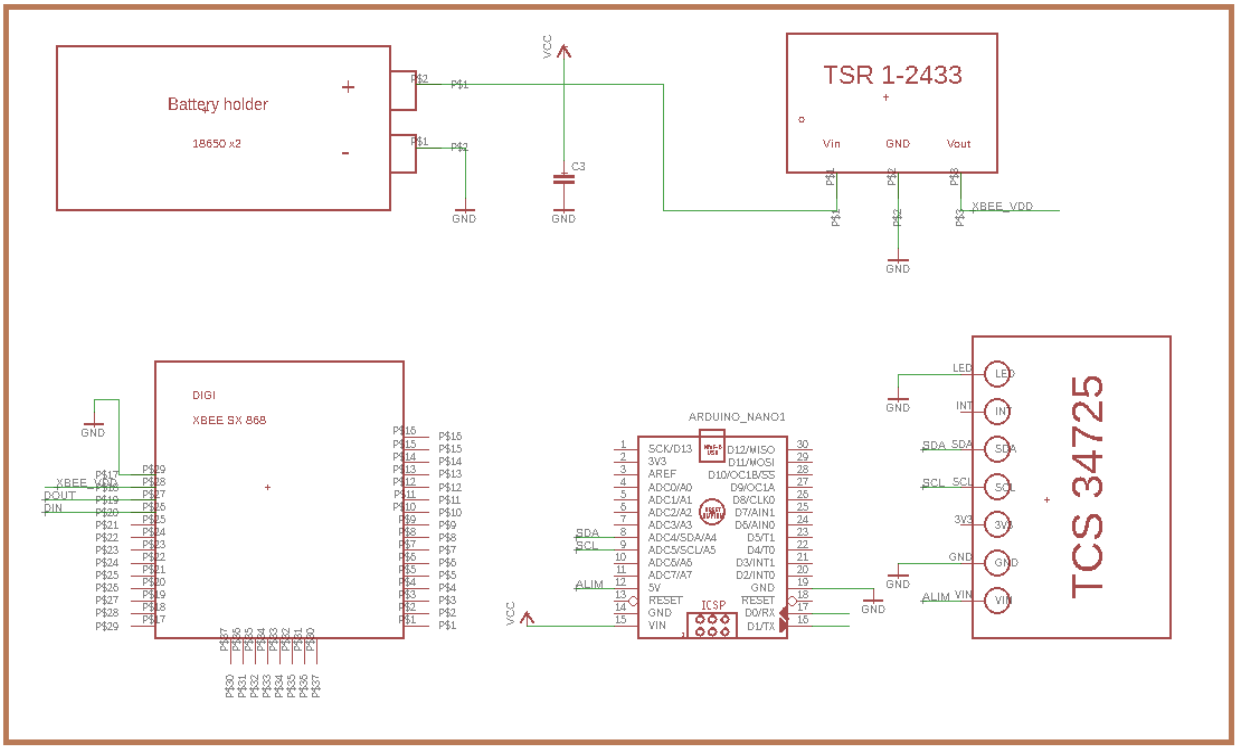


Figura 4.4.- Esquemático PCB alimentada mediante batería.

En la Tabla 4.2 se pueden ver los componentes usados para el diseño de la PCB.

PCB alimentada mediante batería
Batería recargable 18650 de 3.7V (x2)
Módulo XBee 868
Arduino Nano
Sensor RGB TCS34725
Convertidor reductor TSR 1-2433

Tabla 4.2.- PCB alimentada mediante batería: componentes.

En la Figura 4.5 se puede ver la disposición de la huella de los elementos en la placa diseñada. El tamaño de esta es mayor que el de la diseñada en el apartado anterior, principalmente por la aparición del módulo XBee y de las dos baterías de 3.7V ya que estos ocupan más espacios que el conector JACK hembra y las tiras de pines de 2mm. No obstante el tamaño de la misma no supera los 15 cm por lado.

Se destaca también que todos los elementos se sitúan en la capa "*TOP*", incluido el dispositivo XBee que es de tipo *SMD*. Se ha incluido también un plano de masa (polígono azul en la Figura 4.5). Al igual que en el primer diseño se han intentado evitar los ángulos de 90º en el trazado de todas las pistas de la placa.

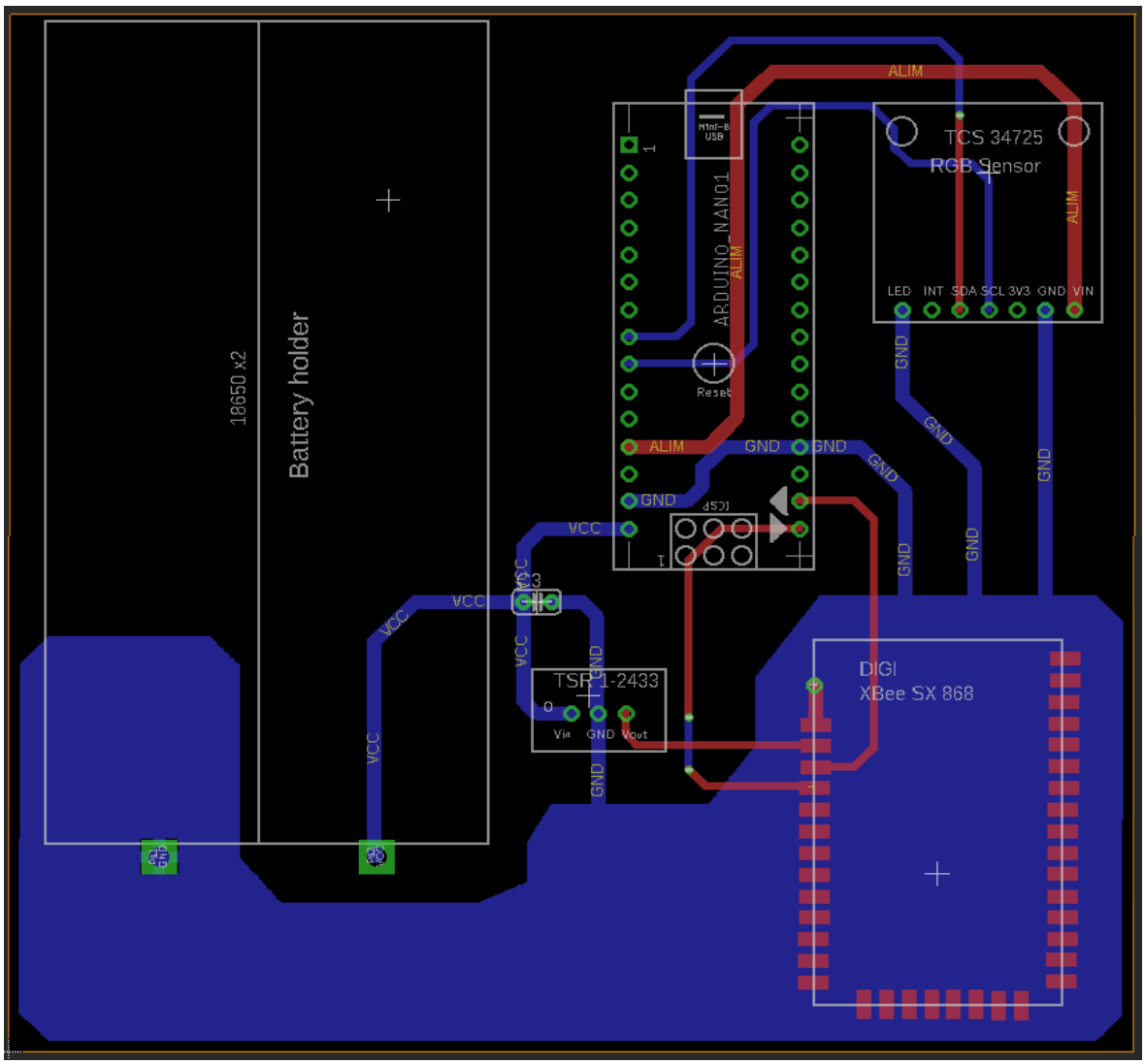


Figura 4.5.- Diseño PCB alimentada mediante batería.

La duración mínima de este prototipo portátil (sin necesidad de recargar las baterías) debe de ser de varias semanas. Como los dispositivos que componen la PCB no consumen demasiado (orden de decenas de mA) y no emiten de manera continuada no parece

descabellado pensar en que se pueda lograr tal objetivo sin más que elegir unas baterías del modelo 18650 de una capacidad adecuada.

No obstante en caso de querer aumentar aún más la duración de la autonomía del prototipo se podría buscar la sustitución del módulo Arduino por otro microcontrolador de propósito más específico.

4.2.- Carcasa impresa en 3D: diseño y fabricación

En este apartado se ha realizado el diseño de una carcasa impresa en 3D para el prototipo construido que se alimenta mediante toma de red. El desarrollo se ha realizado mediante el software *SolidWorks* que está especializado en el modelado mecánico en 2D y 3D.

Se destaca que para el diseño de la carcasa se ha decido añadir un elemento extra conocido como lumiesfera. La principal función de este elemento es la de concentrar la luz de tal manera que las medidas del sensor RGB sean más precisas, es decir mediría en mayor medida la luz incidente y menos la luz reflejada. En la Figura 4.6 se puede ver la lumiesfera elegida.

Figura 4.6.- Lumisfera 401-821 de *Sekonic*®.

La carcasa consta de dos partes, la primera de ellas es la parte principal y es donde va situado el prototipo, la segunda de ellas es la tapa de la misma que sería la encargada de proteger el dispositivo y es el lugar donde se situará la lumiesfera a la que se hace mención en el párrafo anterior.

Los elementos diseñados se pueden ver en la Figura 4.7 y 4.8. Se destaca que para que tuviesen un aspecto más realista se han visto sometidos a un proceso de renderización.

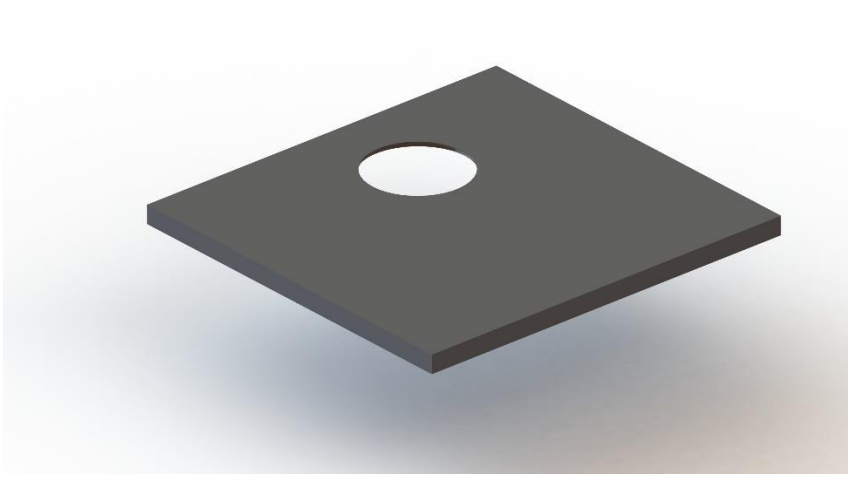


Figura 4.7.- Tapa diseñada en 3D (renderizada).

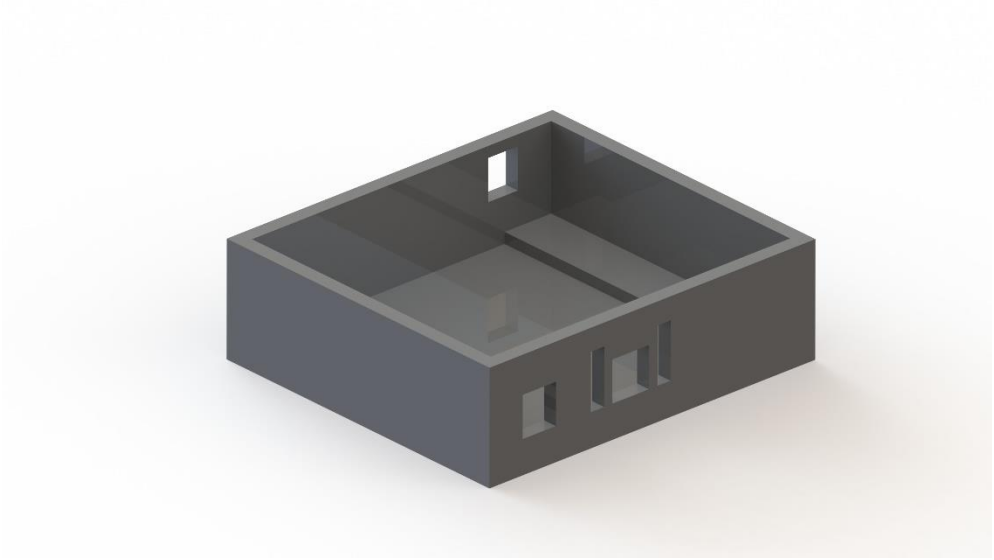


Figura 4.8.- Carcasa diseñada en 3D (renderizada).

En la Figura 4.9 se puede ver una imagen de la carcasa ya impresa y ensamblada. Además las medidas correspondientes a las Figuras 4.7 y 4.8 se pueden ver en los planos 201809-8 y 201809-7 respectivamente.

El material con el que se ha fabricado ha sido PLA [39] (ácido poliláctico) que es un polímero biodegradable derivado del ácido láctico de origen vegetal y renovable al 100% ya que proviene de la extracción de distintos tipos de azúcares. Destaca por su resistencia a la tracción, su módulo de elasticidad, su baja densidad, su estabilidad a la luz ultravioleta y por su capacidad para ser formulado de manera rígida o flexible.

La técnica de impresión usada ha sido FDM [40] o modelado por deposición fundida. Dicha técnica se basa en el uso de tres elementos principales: una placa en la que se imprime la pieza, una bobina de filamento que sirve como material de impresión y una cabeza de

extrusión. Basa su funcionamiento en la succión y fundido del filamento por el extrusor de la impresora 3D, que deposita el material de forma precisa capa por capa sobre la placa de impresión. La boquilla mediante la que se imprime se mueve en 3 ejes: X, Y y Z.

Figura 4.9.- Carcasa impresa en 3D.

5.- PRUEBAS REALIZADAS

En el presente capítulo se expondrá el correcto funcionamiento de los dispositivos. En el apartado 5.1 se usarán dos módulos XBee configurados en modo API, se mostrará la creación de una trama en dicho modo (se envirará una trama por dispositivo) y como esta llega de manera correcta al ConnectPort X4IA, seguidamente se podrá comprobar como los datos enviados se reciben de manera correcta en el servidor.

En el apartado 5.2 se mostrará el funcionamiento del prototipo diseñado. Para ello, y a diferencia del apartado 5.1, el módulo está configurado en modo AT. El proceso visualizado es el mismo que en el modo API, la principal diferencia es que los datos que se envían son los proporcionados por el sensor RGB.

5.1.- Comunicación modo API

En este sub-apartado se han usado dos módulos XBee en modo API, para comprobar que ambos dispositivos funcionan de manera correcta y que los mensajes enviados por los mismos se reciben a la perfección en primer lugar en el ConnectPort X4IA y en segundo lugar en el *KEPServerEX*.

Para ello y en primer lugar se configuran los dispositivos en modo API con una tasa de baudios de 115200 de la misma manera que se explicaba en la Figura 3.17 del sub-apartado 3.3.

Seguidamente se procede a configurar las tramas a enviar de acuerdo a la Figura 3.18 y se enviarán dos tramas de datos (como la visible en la Figura 5.1). Recuadrado en color rojo se puede ver el campo de la dirección a la que serán enviados los mensajes, dicha dirección es por supuesto la del ConnectPort X4IA. En el campo "*RF data*" se ha introducido el mensaje en ASCII "SOYXBEEC" para uno de los módulos XBee y para el otro "SOYXBEEB". De esta manera, se comprobará la capacidad de la *Gateway* de recibir información de más de un dispositivo.

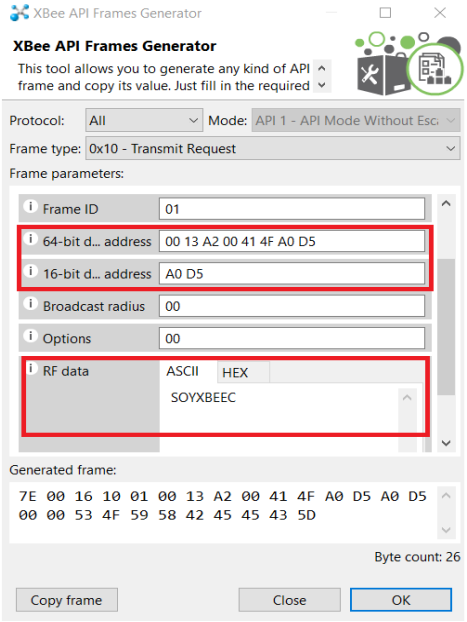


Figura 5.1.- Configuración de mensaje enviado.

Posteriormente se realiza una conexión al ConnecPort X4IA mediante el programa *PuTTY*, para acceder al dispositivo se realiza una configuración como la que se puede ver en la Figura 5.2. En ella se puede ver como se indica: el protocolo de conexión usado que es Telnet, el puerto usado (el más habitual del protocolo) que es el 23 y la dirección IP del elemento que es la 192.168.1.1.

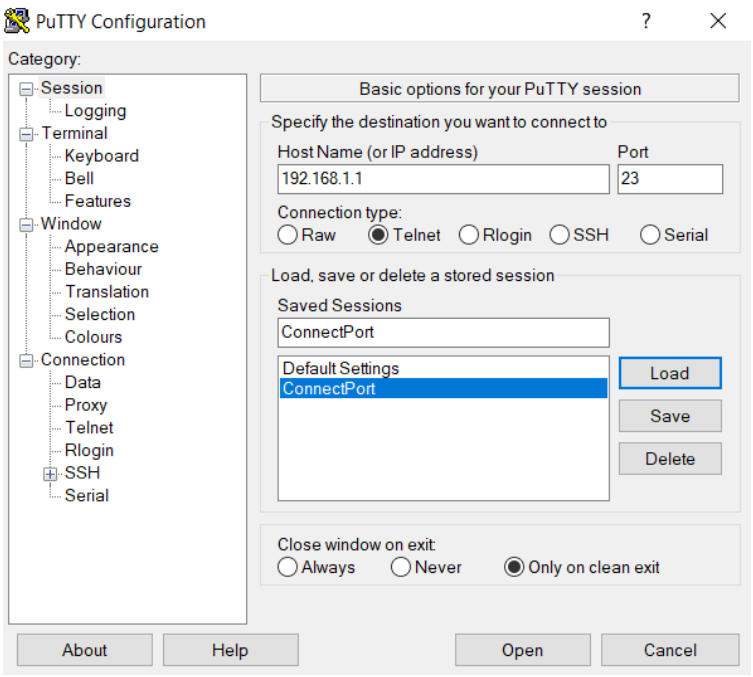


Figura 5.2.- Configuración PuTTy.

En la Figura 5.3 se puede ver como los mensajes enviados por los módulos XBee llegan de manera correcta a la *Gateway*. Se recuadra en rojo en la Figura la dirección de origen de los mensajes, se puede comprobar como hay dos distintas una por elemento emisor.

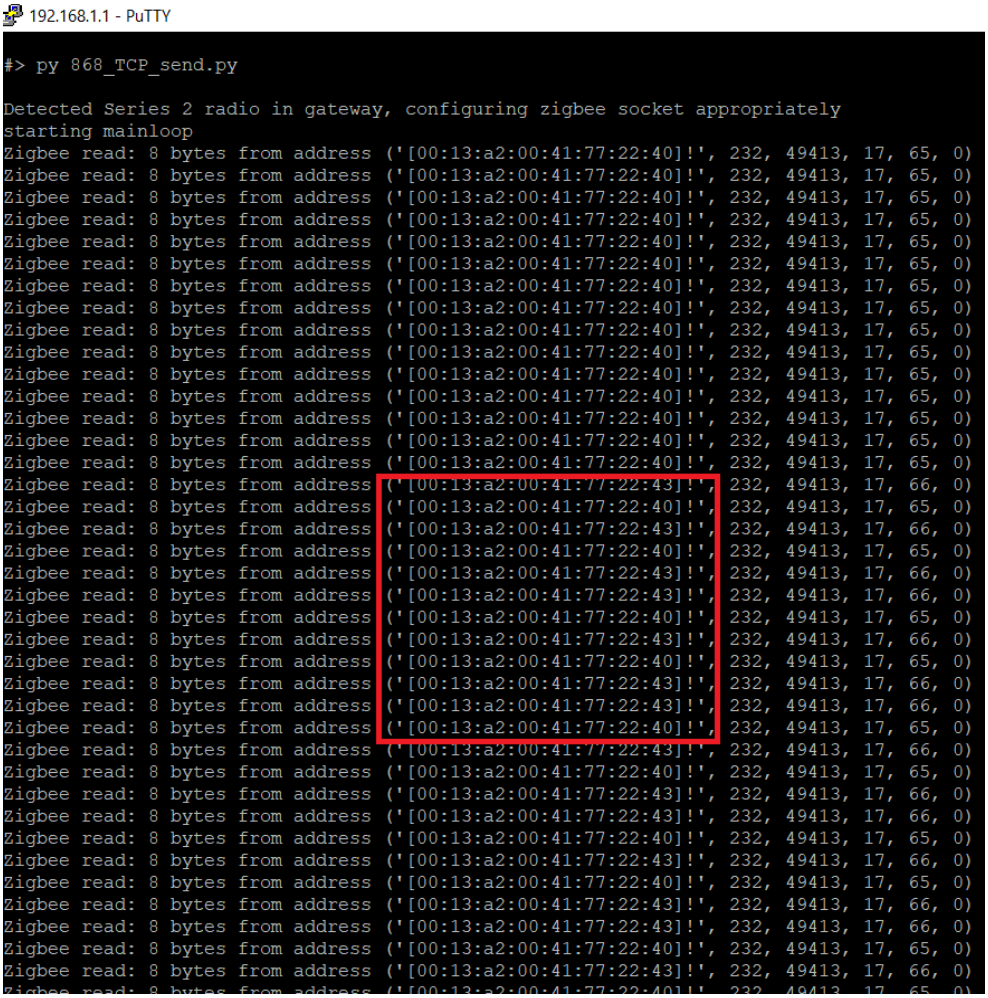


Figura 5.3.- Mensajes recibidos ConnectPortX4 (PuTTy).

Seguidamente y como se explicó en la configuración del *KEPServerEX* se debe de crear un canal por cada dispositivo XBee del cual se espere la llegada de datos, en este caso al haber dos dispositivos habrá dos canales.

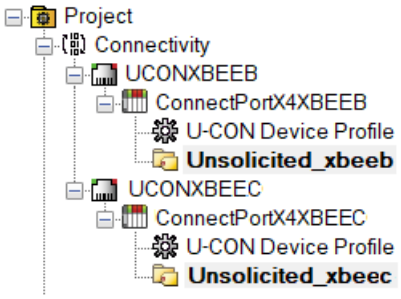


Figura 5.4.- Configuración *KEPServerEX* para dos módulos XBee.

En las Figuras 5.5 y 5.6 se puede comprobar como los mensajes enviados desde los dispositivos XBee se pueden visualizar en el servidor. Se destaca que para ello ha de inicializarse un cliente "OPC Client".

En la Figura 5.5 al enviarse siempre el mismo mensaje no habrá indicios de actualización (ver campo "*Update count*" siempre a 1), ya que esto solo ocurre cuando los datos se modifican de un mensaje a otro.

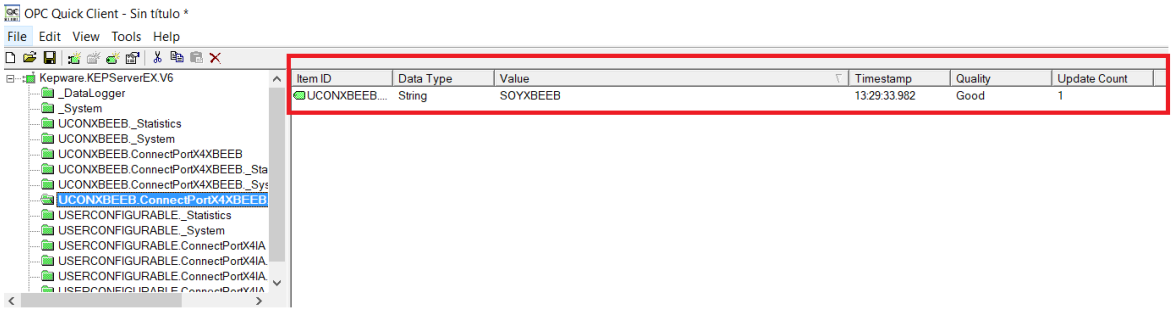


Figura 5.5.- Mensaje recibido desde XBeeB en *KEPServerEX*.

En la Figura 5.6 se ve como el campo "*Update Count*" se tiene un valor de 1211 y de 1232, debido a que uno de los módulos XBee está enviando dos mensajes distintos de manera continua.

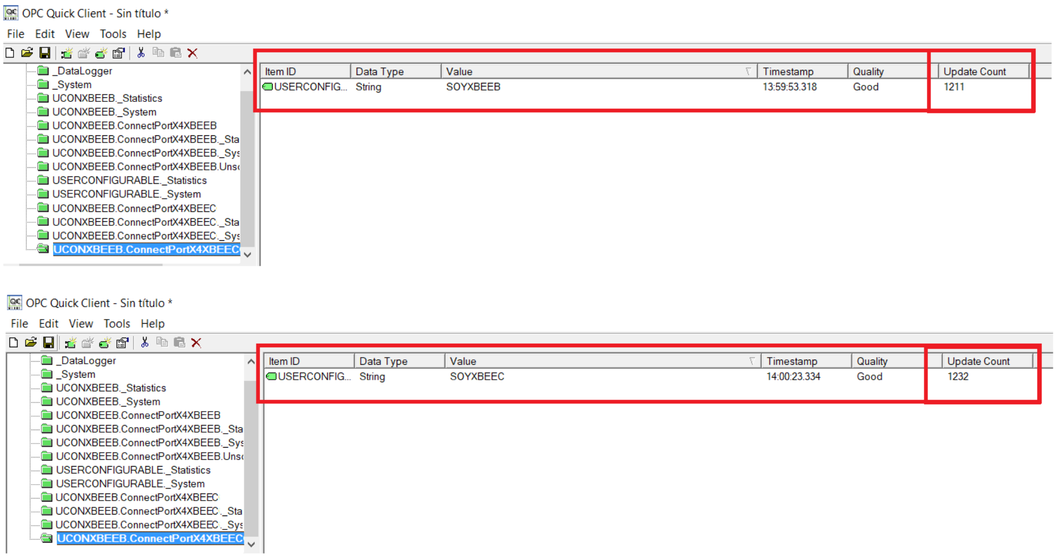


Figura 5.6.- Mensajes recibidos desde XBeeC en *KEPServerEX*.

5.2.- Comunicación modo AT

En este sub-apartado se mostrará el correcto funcionamiento del prototipo diseñado para el presente proyecto. En este caso, y a diferencia del apartado anterior, se usa el modo de comunicación AT, para ello no es necesario el uso del software *XCTU* ya que este modo no usas tramas propiamente dichas, sino que envía los datos según le llega vía UART (pines TX, RX y GND).

En la Figura 5.7 se puede comprobar como los datos enviados por el prototipo llegan de manera satisfactoria al ConnectPort X4IA.

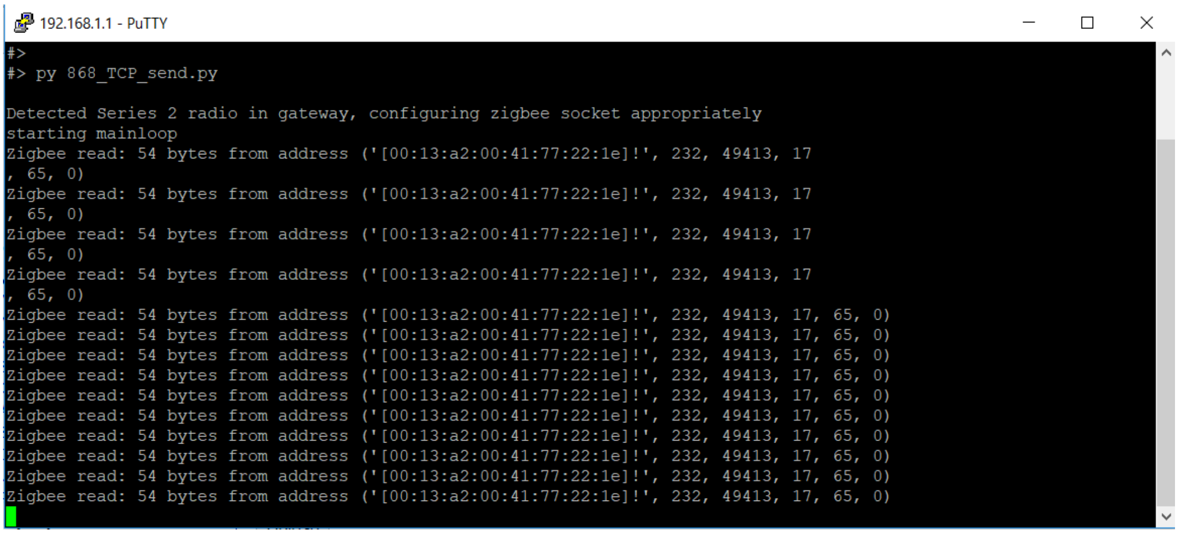


Figura 5.7.- Mensajes recibidos ConnectPortX4 (PuTTy).

En este caso se reciben como se puede comprobar 54 bytes puesto que la información que se envía desde el sensor RGB es mayor que la creada en el caso anterior desde los propios módulos XBee.

En la Figura 5.8 se puede visualizar la llegada de los datos hasta el servidor y por tanto comprobar que el sistema de comunicación diseñado e implementado funciona correctamente. Como los datos medidos por el sensor varían continuamente el campo "*Update Count*" aumenta de manera proporcional a los mensajes recibidos.

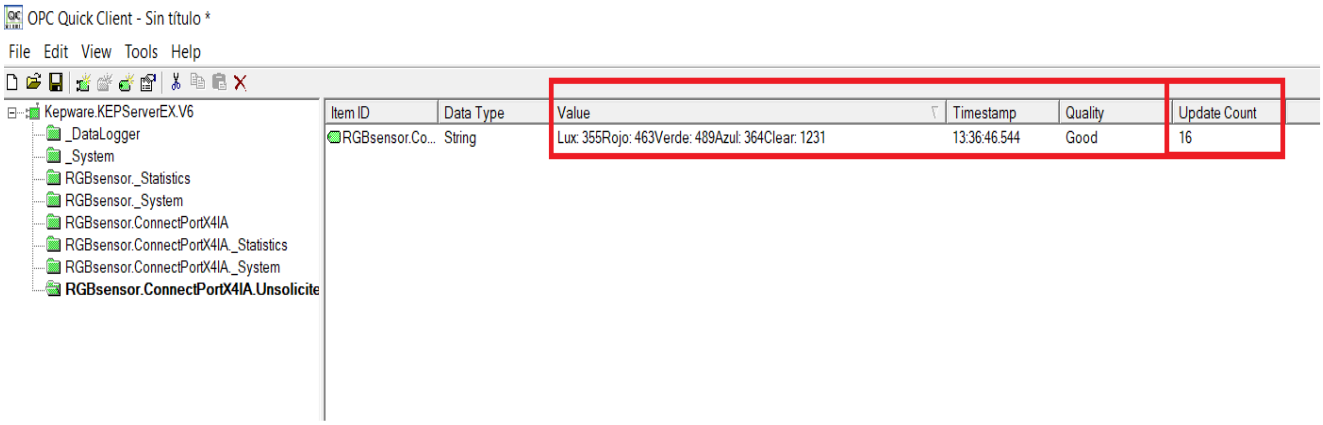


Figura 5.8.- Mensajes recibidos servidor *KEPServerEX*.

En la Figura 5.9 se muestra una imagen del equipo de trabajo completo (formado por el prototipo, el ConnectPort X4IA y un ordenador donde corre el servidor *KEPServerEX*). Se destaca que el número de prototipos desarrollados podría ser bastante elevado y para recibir los datos de manera correcta de cada uno de ellos simplemente habría que añadir un dispositivo nuevo en el servidor. En cuanto al proceso de comunicación no debería de haber interferencias puesto que se identifica cada dispositivo por su MAC, además la llegada de datos no debería de fallar por ser una estructura de red tipo malla.

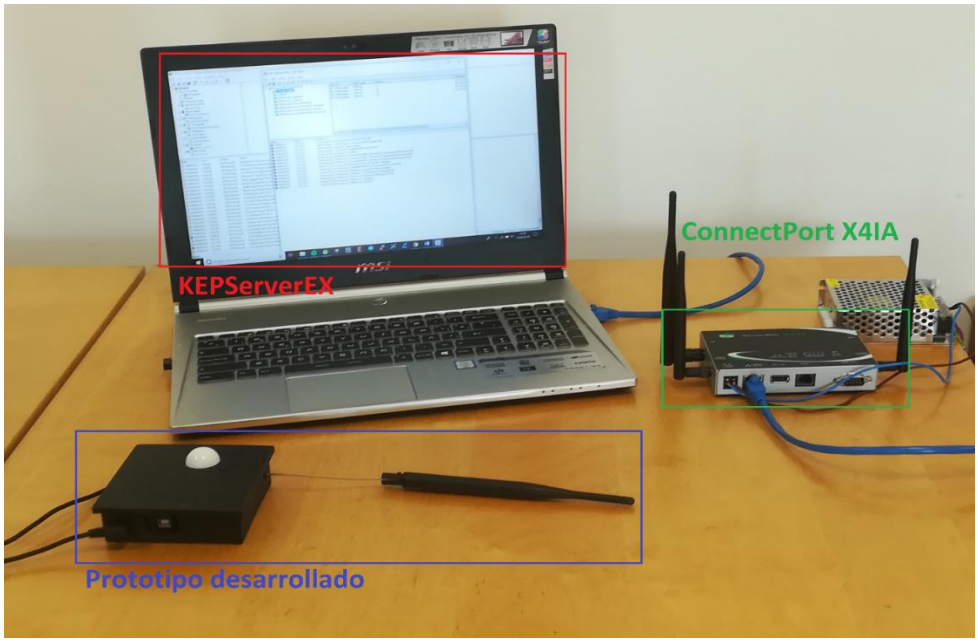


Figura 5.9.- Equipo de trabajo.

5.3.- Pruebas en un entorno industrial

Las pruebas mostradas en este sub-apartado se han realizado en un pequeño hangar de una distancia de largo de unos 25 m de longitud de un extremo a otro.

El entorno de trabajo consta de dos módulos XBee y el ConnectPort X4IA. La *Gateway* se sitúa en una sala en una posición fija separada por varios muros de lo que son los dispositivos XBee. Uno de los XBee se sitúa cerca de la entrada del hangar, mientras que el otro se va separando del mismo a distintas distancias. También se modifica la potencia transmitida de cada uno de los módulos inalámbricos (que ya incluye la ganancia de la antena) para ver como de crucial es la modificación del parámetro.

Se sabe de antemano que la sensibilidad de los dispositivos de la red es de unos -115 dBm. En las siguientes tablas se denota como XBeeB al módulo fijo y más cercano a la entrada del lugar donde se han realizado las pruebas y como XBeeA al elemento que se irá alejando de manera progresiva. Además cuando se hace referencia a CPX4 se refiere al ConnectPort X4IA.

La distancia fija y aproximada entre el ConnectPort X4IA y el XBeeB será de aproximadamente unos 20 m.

	Potencia transmitida XBeeB			
	9 mW	*15 mW*	*25 mW*	*32 mW*
Señal recibida CPX4 (dBm)	-86	-86	-85	-85

Tabla 5.1.- Nivel de señal recibida por el ConnectPort X4IA en función de la potencia transmitida por el XBeeB.

A partir de la Tabla 5.1 se puede concluir que la potencia recibida por el ConnectPort X4IA es muy semejante independientemente de la potencia con la que se transmita por el XBeeB. Esto se debe principalmente a que la Gateway y el módulo inalámbrico no se encuentran en la misma sala, sino en salas diferentes y separadas entre sí por varios muros fabricados en distintos materiales, si además de esto se añade el hecho de que se está en un entorno industrial y que hay máquinas u otros elementos que añaden una atenuación extra a la señal transmitida se puede justificar el nivel de señal recibido.

Además también se puede llegar a justificar el hecho de que el nivel de señal recibida sea muy semejante independientemente de la potencia con la que se transmite debido principalmente a que se trata de potencias de muy bajo nivel y al aumentar la distancia el nivel de potencia recibida disminuye. Esto se puede razonar mediante la siguiente fórmula:

$$P_{RX} = \frac{P_{TX}}{4\pi \cdot R^2} = \frac{10\ mW}{4\pi \cdot 20^2} = 1.98\ \mu W/m^2 \qquad (5.1)$$

Dónde P_{RX} es la potencia recibida por metro cuadrado, P_{TX} la potencia transmitida y R la distancia entre elemento transmisor y receptor. Se puede comprobar sin más que pasar a unidades logarítmicas el resultado obtenido que en el mejor de los casos y sin tener en cuenta elementos que puedan atenuar la señal el nivel de potencia recibido aproximado de -57 dBm (que dependerá también de factores como el tipo de tecnología usada, protocolo, etc…).

En las Tablas 5.2, 5.3 y 5.4 se puede ver la relación de potencia entre los módulos XBeeB y XBeeA para diferentes distancias.

	Potencia transmitida XBeeA a una distancia de 9 m			
	9 mW	*15 mW*	*25 mW*	*32 mW*
Señal recibida XBeeB (dBm)	-49	-45	-43	-41

Tabla 5.2.- Nivel de señal recibida por el XBeeB en función de la potencia transmitida por el XBeeA a una distancia entre ambos de 9m.

	Potencia transmitida XBeeA a una distancia de 12 m			
	9 mW	*15 mW*	*25 mW*	*32 mW*
Señal recibida XBeeB (dBm)	-47	-46	-44	-43

Tabla 5.3.- Nivel de señal recibida por el XBeeB en función de la potencia transmitida por el XBeeA a una distancia entre ambos de 12m.

	Potencia transmitida XBeeA a una distancia de 25 m			
	9 mW	*15 mW*	*25 mW*	*32 mW*
Señal recibida XBeeB (dBm)	-98	-87	-83	-83

Tabla 5.4.- Nivel de señal recibida por el XBeeB en función de la potencia transmitida por el XBeeA a una distancia entre ambos de 25m.

La diferencia de distancia entre las Tablas 5.2 y 5.3 no ha sido mayor puesto que había un elemento que podía causar una interferencia en la señal, no obstante y como se puede comprobar en el resultado de dichas tablas este no afecta de manera crítica a la comunicación entre módulos.

Como conclusión se puede comprobar como a medida que aumenta la potencia de TX el nivel de señal recibida es mayor y que para una misma potencia el número de pérdidas aumenta conforme lo hace la distancia.

Se destaca la gran atenuación entre dispositivos (sobre todo en la Tabla 5.4), al realizarse las pruebas en interiores no se puede justificar las pérdidas de una manera tan exacta como si se hubiesen realizado en exteriores, ya que este último caso se podrían razonar mediante pérdidas en espacio libre, si hay visión directa o no, etc… Por tanto para inferir en las mismas hay que acudir a estudios [42] ya realizados donde se comprueba que para pruebas realizadas en interiores y con módulos XBee la atenuación sufrida es bastante considerable y que aumenta en función de la distancia.

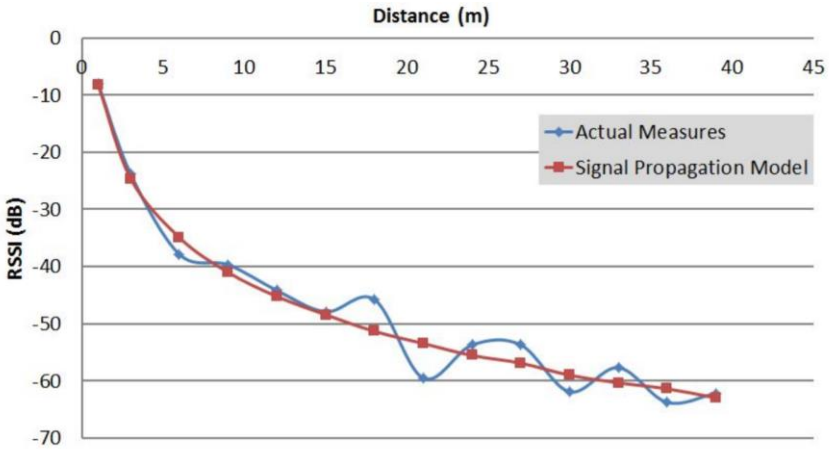


Figura 5.10.- Estudio sobre el nivel de señal recibida en módulos XBee en interiores [42].

A partir de la Figura anterior se puede comprobar como los resultados obtenidos en dicho estudio se asemejan a los resultados obtenidos en la Tablas anteriores. La principal diferencia y por la cual se justifica que el nivel de señal sea menor en los casos analizados que en el del estudio es porque en el caso actual se trata de un entorno industrial que siempre es más crítico.

6.- BALANCE DE BLANCOS

El término “balance de blancos” [37] proviene del mundo audiovisual donde un dispositivo semejante a un osciloscopio (pero especializado en señales de televisión) se usa para igualar los colores rojo, verde y azul (RGB) de los canales de una cámara para así lograr colores blancos precisos bajo varias condiciones de luz.

Por ello los valores obtenidos por el sensor TCS34725 se podrían usar para realizar un correcto balance de blancos y así evitar errores en fotografías tomadas por ejemplo en situaciones temporales donde el nivel de luz es crítico, como pudiesen ser el amanecer o el anochecer.

En el presente capítulo se hará mención a los métodos de balanceado de blancos más comunes en la actualidad.

6.1.- Métodos de balance de blancos

Si un balance de blancos está realizado de una manera incorrecta, se produce una distorsión entre lo que se visualiza en la imagen y lo que se percibe mediante los ojos. Esta distorsión se puede ver ya que la imagen adquiere tonos azulados, verdosos o anaranjados.

Todo ello está relacionado con la temperatura de color (medida en grados kelvin) y la cual es necesaria para realizar un correcto balance de blancos. Así pues, la temperatura de color de la luz de una vela es mucho menor que la de por ejemplo un cielo muy nublado. Algunos ejemplos de temperatura de color estándar se pueden ver en la Tabla 6.1 [37].

Así, una cámara digital busca un punto de referencia en un determinado paisaje que represente el color blanco y a partir de este se calcula el resto de colores basado en el rango de colores conocidos por el ser humano. Esto se realiza mediante sensores RGB que posee la propia cámara y de determinados algoritmos y ecuaciones predeterminadas.

Una cámara suele tener varias opciones de balance de blancos predefinidas, evitando así el proceso de obtención mencionado en el párrafo anterior. Estas opciones suelen ser: tungsteno (pensado para espacios interiores nocturnos), fluorescente (deportes, parkings…), luz de día (nivel de luz de un estudio), nublado (paisajes y retratos), sombra (zonas sombrías) y flash (semejante a la opción nublado).

Temperaturas de color típicas	
Elemento	Temperatura de color (K)
Luz de una vela	1500
Bombillas de 60W	2800
Amanecer y atardecer	3200
Lámpara de tungsteno	3400
Bombillas fluorescentes	4000-4500
Luz de mediodía	5200
Flash de una cámara fotográfica	5600
Cielo muy nublado	6500
Cielo azul profundo	10000-15000

Tabla 6.1.- Temperaturas de color típicas.

6.1.1.- Auto White Balance (AWB)

Es la opción [38] más sencilla y a medida que aumenta el precio de las cámaras aumentan las prestaciones del método. Cuanto mejor se la exposición de una cámara (equilibrio entre la apertura, el tiempo y la sensibilidad de los sensores) mejor será el AWB, así si una imagen se encuentra subexpuesta el resultado será un retrato en tonos azulados, por el contrario si está sobreexpuesta aparecerán en tonos blancos y negros. Los pasos del método son los siguientes:

1. La cámara mide cuanta luz hay gracias al fotómetro del que dispone.
2. Con la medida anterior se ajusta cuanta luz se deja pasar por el objetivo de la cámara.
3. Se ajusta el tiempo que la luz va a estar alcanzando el sensor con el tiempo de exposición.
4. Se ajusta la sensibilidad del sensor.

La ventaja de este método es que se pueden realizar diferentes tipos de fotografía durante una sesión sin necesidad de variar la configuración. Como inconvenientes se destaca que el tiempo dedicado a la edición de la fotografía mediante programas pensados para tal fin es mucho mayor.

6.1.2.- Gray Card (Tarjeta Gris)

Método [38] basado en el uso de una placa de plástico duro que pueden encontrarse en diferentes tamaños (la medida estándar es de 8 x 10 cm).

Figura 6.1.- Tarjeta gris [38].

Se usan para establecer una correcta exposición y por tanto realizar un correcto balance de blancos. El proceso se puede realizar de dos maneras:

1. Situar la *Gray Card* en las condiciones de iluminación que se van a utilizar y tomar una foto rápida donde los límites de la misma los delimite la tarjeta. Una vez realizado, ya se pueden sacar fotos desde una distancia más lejana pero siempre en las mismas condiciones de luz. En caso de que el nivel de luz se vea modificado se deberá de repetir el proceso.
   Después, en el programa de edición de fotos, mediante la herramienta de balance de blancos de la que disponen se puede elegir una de las imágenes predefinidas que más se asemeje a la tarjeta gris para mejorar el resultado final.

2. Se establece un balance de blancos personalizado (CWB) en la cámara y por tanto ya no será necesario realizar un trabajo de procesado y edición de imágenes a tan alto nivel como en el caso anterior. Al realizar el CWB el balance de blancos ya se aplica en la misma cámara en el momento de realizar fotografías.

6.1.3.- Kelvin

Es uno de los métodos [38] para establecer el balance de blancos más utilizado en la actualidad. Se suele usar especialmente cuando se toman fotografías en exteriores. El proceso se basa en determinar el color del ambiente donde se va a realizar la sesión fotográfica y a partir de este configurar la cámara a dicha temperatura. Una vez realizada la configuración los posibles cambios que se pueden hacer en la imagen son mínimos.

Básicamente para aplicar el método Kelvin se debe de pensar en qué condiciones de luz se va a trabajar, es decir fijarse si la luz tiende a ser más amarilla, más azul o simplemente neutral. Para las personas inexpertas se recomienda establecer la temperatura inicial en 2800K y posteriormente ir aumentándola o disminuyéndola en función de la apariencia de la fotografía. Por ejemplo, una imagen azulada indica que se necesitaría fijar una temperatura de color mayor.

Figura 6.2.- Efecto de la variación de la temperatura de color [38].

La principal ventaja de este método es que no se necesita de un elemento extra para realizar el balance de blancos. Como inconveniente se destaca que no es un método preciso para realizar en interiores ya que en ese caso suele haber distintas fuentes de luz.

6.1.4.- ExpoDisc

Método usado en lugar de la *Gray Card* (y que la mejora) ya que permite obtener la temperatura de color en condiciones donde la luz no tiene por qué incidir de manera directa sobre el sujeto a fotografiar sino que puede provenir de, por ejemplo, reflejos.

Medir mediante una tarjeta gris puede diferir dependiendo del ángulo con el que se esté enfocando a la misma y la luz con la que esta esté siendo incidida, mientras que el *ExpoDisc* mide y balancea la luz que proviene de todos los ángulos que rodean al susodicho.

Este método se suele usar en situaciones donde prima de una manera elevada la reflexión de la luz o los contrastes con zonas sombrías. La principal ventaja de este método es que es fácil de usar y mantiene la consistencia en los colores ya que neutraliza todos los efectos indeseados y proporciona un punto de partida equilibrado para las imágenes.

6.1.5.- Hoja blanca de papel

Este método no es el más usado y solamente se suele considerar como última opción ya que es el peor de los mencionados en el presente capítulo. Es fácil de usar y barata y mejora el resultado (aunque no tanto como los otros métodos). El modo de configuración es exactamente idéntico a los dos que se mencionaban con la *Gray Card*.

Como principal ventaja se destaca el precio ínfimo de una hoja de papel, Como desventaja que de un folio a otro suele variar la tonalidad de blanco.

Figura 6.3.- Comparativa de los distintos métodos de balance de blancos [38].

6.2.- Exposímetro

El exposímetro [17] es un tipo concreto de fotómetro que está diseñado para medir la luz y/o la temperatura de color. A partir de estas mediciones proporciona unos valores de exposición determinados (velocidad de saturación y diafragma dependiendo del modo de disparo en el que se esté trabajando).

En la actualidad existen tres tipos de exposímetros: de luz reflejada, de luz incidente y los que pueden medir ambas. Se suelen fabricar en Selenio, Sulfuro de Cadmio y Silicio.

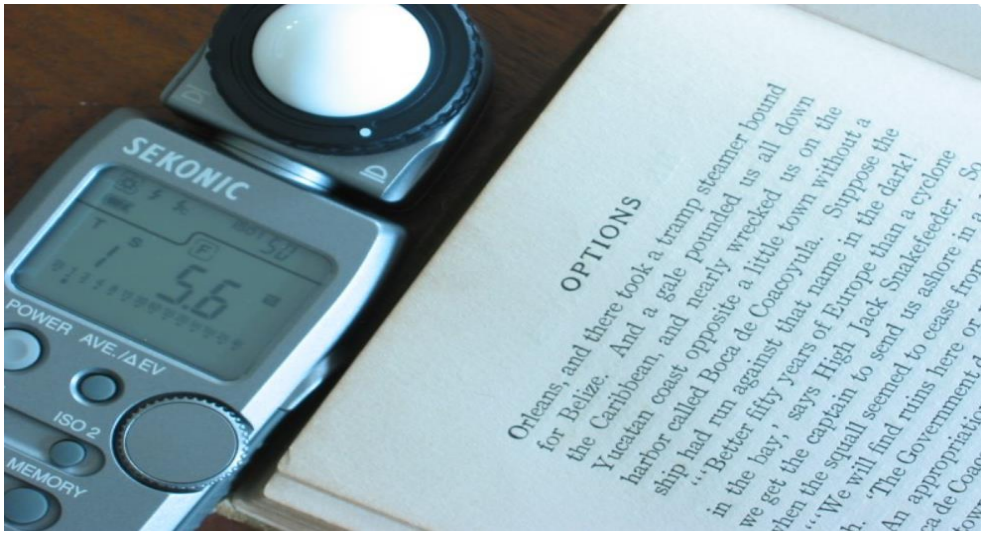


Figura 6.4.- Ejemplo de exposímetro [17].

6.2.1.- Exposímetro de luz reflejada

Fotómetro [17] que mide la luz reflejada de personas u objetos que están siendo fotografiados. Los valores de exposición dependerán de los tonos que tengan los objetos ya que aquellos de tonalidades más claras tienden a reflejar más luz.

Este sensor tiene distintos métodos de obtener una medición de luz óptima. Para elegir uno u otro hay que conocer muy bien la escena donde se quiere realizar la fotografía. Los métodos son:

- Matricial: toma referencias de exposición de diferentes puntos repartidos en el cuadre y propone una exposición media en base a diferentes luces.
- Puntual: mide exactamente en un punto concreto de la escena y se centra en el mismo descartando el resto de luces.
- Parcial: semejante al puntual pero abarcando algo más que solamente un punto concreto.
- Ponderada al centro: realiza la medición del centro de la imagen pero también considera el resto de luces del entorno.

Como ventaja se destaca que este sensor ya viene integrado en las cámaras actuales y que por tanto no se tienen que cargar y si se conocen sus características se pueden trabajar en un gran número de situaciones.

Como desventajas se destaca que no reflejan la luz real sino la que reflejan los objetos y por tanto es muy fácil que se pierdan situaciones donde hay varias entradas de luz y una de ellas predomine sobre el resto haciendo que se pierda calidad en la imagen.

6.2.2.- Exposímetro de luz incidente

Sensor [17] que mide la intensidad de luz real de la escena que incide en el sujeto y proporciona unos valores bastante precisos. Permite conocer exactamente el nivel de luz que hay en una escena ya que diferencia entre las desigualdades exactas que hay entre los distintos tipos de luz. Más preciso que el de luz reflejada.

Su funcionamiento se basa en medir la intensidad de luz, independientemente del tono que posea todo aquello que aparece en la imagen. Para ello:

- Escoger una sensibilidad ISO de forma manual, la misma para el sensor que para la cámara.
- Elegir una velocidad de obturación e indicársela tanto a la cámara como al fotómetro.
- Medir la luz de la que se quieren conocer los valores de exposición. El fotómetro proporcionará un resultado en forma de apertura del diafragma.

7.- INDUSTRIALIZACIÓN DEL PROTOTIPO

En el presente capítulo se analizarán las opciones reales que tiene el prototipo de ser implantado en una planta real. Como se puede comprobar a lo largo del documento se puede demostrar la viabilidad técnica del prototipo desarrollado puesto que la comunicación en un entorno industrial parece funcionar de manera correcta (ver Capítulo 5.-Pruebas Realizadas).

La industrialización de un prototipo consta de las siguientes etapas [41]:

- **Prototipo industrial**: elección de la tecnología y requisitos.
- **Verificación del diseño**: comprobación del correcto funcionamiento del prototipo.
- **Revisión del diseño**: a partir de los resultados de la verificación se realiza una revisión del diseño y se sugieren reenfoques o cambios.
- **Validación del producto**: realización de los protocolos de validación del nuevo producto de acuerdo a las normas de calidad.
- **Moldes**: se determina la necesidad de un molde para la fabricación del nuevo producto, buscando siempre las características que mejor se adecuen al producto.
- **Diseño de embalajes**: búsqueda del embalaje adecuado del prototipo en caso de que este necesite ser transportado para la realización de nuevas pruebas.
- **Transporte de productos**: relacionado con el punto anterior, búsqueda del mejor medio de vehículo para trasladar el producto.
- **Certificación de productos**: analizar las necesidades normativas de acuerdo a la regulación europea (CE), regulación americana (FDA) u otras regulaciones internacionales (Esquema CB). Búsqueda de empresas que se encarguen de certificar dicha normativa.
- **Equipos médicos**: en caso de tratarse de un equipo médico se debe de realizar una certificación más exhaustiva.
- **Gestión de calidad**: implantación de un sistema de calidad, normalmente gestión general ISO9001.

A partir de las etapas mencionadas se puede comprobar como las partes del "prototipo industrial" y "verificación preliminar del diseño" (ya que habría que realizar un número mayor de pruebas) ya han sido realizadas. La siguiente etapa a realizar sería la "revisión del diseño" de cara a realizar mejoras en el mismo (como pudiese ser el cambio del Arduino por un microcontrolador de propósito mucho más específico para el caso del prototipo que funciona mediante batería) y realizar alguna pequeña modificación, tras finalizar esta, siguen el resto fases como son la "validación del mismo", "obtención de certificados", "gestión de calidad", etc… La única etapa que no habría que tener en cuenta sería la relacionada con

"equipos médicos", puesto que el prototipo diseñado no está relacionado con el mundo médico.

Se destaca que dentro de las etapas ya se incluyen pruebas como resistencia a descargas eléctricas, influencia de los distintos tipos de ruido, etc… En caso de cumplirlas el prototipo ya estaría listo para su comercialización y su implantación en un entorno industrial.

8.- CONCLUSIONES Y LÍNEAS DE TRABAJO FUTURAS

A lo largo del desarrollo de este proyecto se han ido presentando diferentes imprevistos que se han solucionado de la manera más eficiente posible. Un buen ejemplo es el caso de posibilitar la comunicación entre el coordinador de la red con los módulos XBee y con el servidor *KEPServerEX*, donde en un principio se intentó realizar dicha comunicación mediante el protocolo Modbus (como puede comprobarse en el Anexo 8.1), no obstante al comprobar que esto no era posible se decantó por un método más común como es el de los *sockets* TCP/IP.

También se presentó algún inconveniente al realizar la comunicación entre el Arduino NANO y el módulo XBee ya que en un principio se intentó realizar la configuración mediante el software *XCTU*, pero al comprobar que esto no era posible se decidió controlar el envío de manera directa desde el software del Arduino y usar comunicación UART.

Con respecto a la configuración del servidor, pese a tener numerosas alternativas, la elección del método U-CON permitió la visualización de los datos provenientes de más de un dispositivo y ver cómo estos se actualizaban en tiempo real sin ningún inconveniente.

La parte relacionada con el diseño de la placa se realizó de manera satisfactoria, pese a que podría haber algún problema con el trazado de las pistas (debido a que alguna se hubiese trazado mal) o con la creación de alguna huella de algún componente.

Con respecto a las líneas futuras de trabajo. La mejora más inmediata sería la de continuar el diseño de la PCB alimentada por batería y comprobar que esta funciona de igual manera que la que se alimenta mediante toma de red.

Se podría completar la parte de la red que se sale de los ámbitos de este proyecto. Para ello se debería de desarrollar la interacción con actuadores y almacenamiento de datos en la nube (*Big Data*).

Se podrían realizar pruebas de alcance de la red DigiMesh en exteriores (una vez se hubiese comprobado que la versión con batería funciona de manera correcta) con un mayor número de dispositivos XBee y ver como todos direccionan los datos de manera correcta al coordinador de la red.

Realizar mejoras de seguridad en el código implementado en el ConnectPort X4IA (eliminar la necesidad de incluir un puerto nuevo por cada dispositivo XBee de la red) e incluso intentar hacerlo más eficiente.

También se podría diseñar un software que permitiese la corrección de fotografías a partir de los datos que proporcionar el sensor RGB tomando como referencia algunos de los métodos explicados en el capítulo “Balance de blancos”.

9.- ANEXOS

9.1.- Modbus

Este apartado pretende ilustrar como sería el proceso de configuración en caso de que el protocolo de comunicación elegido hubiese sido Modbus. Este protocolo es muy usado en entornos industriales ya que es libre y por tanto gratuito, no tiene restricciones sobre envíos de datos y su integración es sencilla.

No obstante, en el caso del presente proyecto el uso de este protocolo no ha sido posible, principalmente debido a no existir un driver compatible con los dispositivos XBee con los que se ha trabajado.

Como gran parte del proceso de configuración es semejante al explicado a lo largo del proyecto, solo se hará énfasis en aquellas que difieran. Entre los programas usados destaca el *Digi ESP Python IDE*.

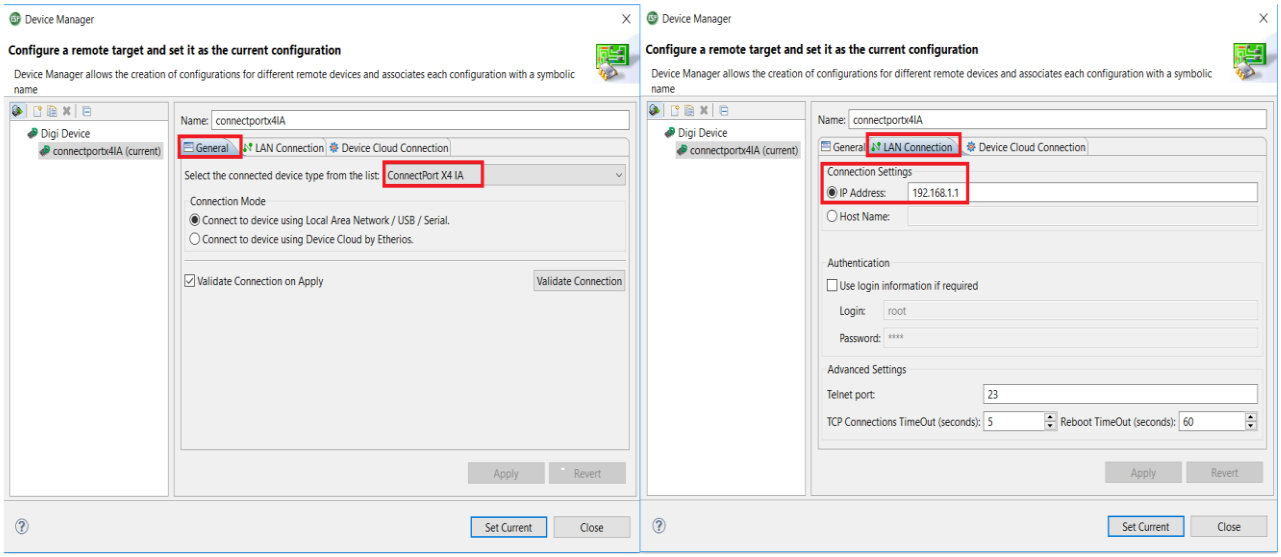


Figura 8.1.- Configuración manager en DigiEsp.

En la Figura 8.1 (a) se ve cómo se elige el ConnectPort X4IA como manager y en la Figura 8.1 (b) se comprueba cómo se pone la dirección IP de la misma. Posteriormente, si se ejecutase el código que se muestra en este apartado, se generarían unos archivos Python de manera automática en el dispositivo que posibilitarían la comunicación.

El código a implementar debería de ser semejante al que se muestra a continuación, el problema es que en la línea donde se selecciona el *driver* se ha usado el que corresponde a

otro dispositivo que sí es compatible con Modbus, en el caso de los dispositivos XBee elegidos el no hay *driver* disponible.

```
#Se definen los dipositivos que conforman la red así como las características
de los mismos
devices:
  - name: digimesh_device_manager0
    driver:
devices.xbee.xbee_device_manager.digimesh_device_manager:DigiMeshDeviceManager
  - name: XBEEA
    driver: devices.modbus.mbdia_xbee_aio:MBusXBeeAIO
    settings:
        xbee_device_manager: "digimesh_device_manager0"
        extended_address: "00:13:a2:00:41:77:22:1e!"
        sample_rate_ms: 1000
        power: true
        sleep: false
        raw_value: false
        enable_low_battery: false

  - name: XBEEB
    driver: devices.modbus.mbdia_xbee_aio:MBusXBeeAIO
    settings:
        xbee_device_manager: "digimesh_device_manager0"
        extended_address: "00:13:a2:00:41:77:22:43!"
        sample_rate_ms: 1000
        power: true
        sleep: false
        raw_value: false
        enable_low_battery: false

#Se define el uso del protocolo Modbus y los dispositivos que lo usarían.
presentations:
  - name: mbus_srv
    driver: presentations.modbus.mbdia_pres:MbDiaPresentation
    settings:
        mapping: "((1,'XBEEA'),(2,'XBEEB'))"
```

Seguidamente se muestra la configuración que habría que realizar en el ConnectPort X4IA. Como el proceso general ya ha sido explicado, solamente se mostrará cómo se debe de configurar el protocolo Modbus en el dispositivo.

En primer lugar hay que ir a la parte de “Automatización Industrial” y seguidamente como puede verse en la Figura 8.2 (a) seleccionar el protocolo Modbus, posteriormente se selecciona el puerto por el que provendrán los mensajes así como la capa de transporte (en este caso puerto 502 y se ha seleccionado tanto transporte por UDP como por TCP). Se realiza una configuración semejante para el caso del destino de los mensajes.

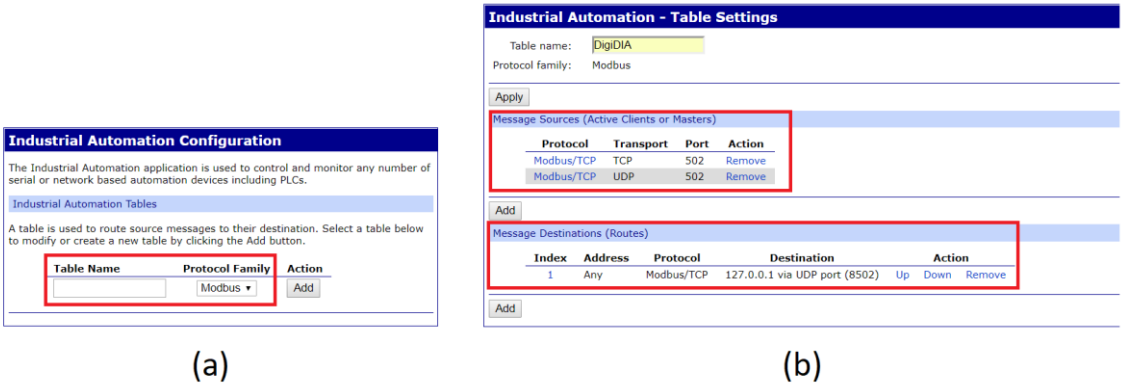


Figura 8.2.- Configuración Modbus en ConnectPort X4IA.

Al final, los principales objetivos de esta configuración son: que todos los mensajes de un dispositivo remoto lleguen a la *Gateway*, direccionar todos los mensajes entrantes Modbus TCP/UDP hacia la *Gateway* a través del puerto 502 y direccionar todo el tráfico Modbus hacia la máquina local a través del puerto 8502.

9.2.- Arduino Genuino: código

[34]

```
#include <Wire.h> //Adición de las librerías Wire.h y la propia del
sensor TCS34725
#include <Adafruit_TCS34725.h>

/* Initialise with default values (int time = 2.4ms, gain = 1x) */
// Adafruit_TCS34725 tcs = Adafruit_TCS34725();

//Tiempo de adquisición de datos de 700ms (el máximo posible para ser lo
más precisos) y ganancia x1.
Adafruit_TCS34725 tcs = Adafruit_TCS34725(TCS34725_INTEGRATIONTIME_700MS,
TCS34725_GAIN_1X);

//Inicialización del programa
void setup(void) {
  Serial.begin(9600); //Se fija la tasa de transmisión a 9600 en Serial y
se inicializa.

  if (!tcs.begin()) //En caso de que el sensor no se inicie se envía un
mensaje de error.
  {
    Serial.println("Error al iniciar TCS34725");
    while (1) {
      delay(1000); //Retardo de 1 segundo
      Serial.println("error");

    }
  }
```

```
}
//Ejecución principal del programa (bucle principal)
void loop(void) {
  uint16_t r, g, b, c, colorTemp, lux; //Se definen las variables
  //Obtención de los valores del sensor
  tcs.getRawData(&r, &g, &b, &c);
  //Calculo de la temperatura de color a partir de los componentes rojo,
//verde y azul
  colorTemp = tcs.calculateColorTemperature(r, g, b);
  //Calculo de lux (lumincancia) a partir de la componentes rojo, verde y
//azul
  lux = tcs.calculateLux(r, g, b);

  //Datos enviados: temperatura de color, lux, rojo, verde, azul y clear
(sin filtro es decir abarca todo el espectro).

  Serial1.print("Temperatura color: "); Serial.print(colorTemp, DEC);
Serial.println(" K");
  Serial.print("Lux: "); Serial.print(lux , DEC);
  Serial.print("Rojo: "); Serial.print(r, DEC);
  Serial.print("Verde: "); Serial.print(g, DEC);
  Serial.print("Azul: "); Serial.print(b, DEC);
  Serial.print("Clear: "); Serial.print(c, DEC);
  Serial.println(" ");
  delay(1000); //Retardo de 1 segundo.
}
```

9.3.- ConnectPort X4IA: código

En este sub-apartado se mostrarán los códigos usados por el ConnectPort X4IA para así hacer posible la comunicación con el XBee y con el servidor.

9.3.1.- Config_cpx4.py

```
############################################################
# ARCHIVO DE CONFIGURACIÓN: DIGI CONNECTPORT X4IA 868 MHZ
############################################################
# Carlos: Este fichero trata de recopilar toda la información y la
configuración que deberían
# replicarse en otros desarrollos de proyectos

## KEPSERVER IP -- Servidor OPC para recibir los datos
KS_ipAddr = "192.168.1.100"

## Carlos: Lista de nodos XBee representados por su dirección MAC. Todos
los nodos que estén usados en el proyecto deberían
##listarse en este archivo, el cual es un diccionario python que asigna a
cada MAC un puerto TCP.


# Carlos: puertos internos para cada módulo XBee.
node_list = {
       "[00:13:a2:00:41:4f:a0:d5]!":4000,
```

```
        "[00:13:a2:00:41:77:22:43]!":4001,
        "[00:13:a2:00:41:77:22:1e]!":4002,
        "[00:13:a2:00:41:77:22:40]!":4003

}

# Carlos: diccionario para dar cierto orden a los dispositivos. Se usará
solo para dar consistencia al orden del puerto
# y la selección.
device_order = {
        "[00:13:a2:00:41:4f:a0:d5]!":0,
        "[00:13:a2:00:41:77:22:43]!":1,
        "[00:13:a2:00:41:77:22:1e]!":2,
        "[00:13:a2:00:41:77:22:40]!":3

}

# Carlos: Puerto TCP/IP para conectar cada XBee con el KEPServerEX.
KS_port = {
        "[00:13:a2:00:41:4f:a0:d5]!":50010,
        "[00:13:a2:00:41:77:22:43]!":50011,
        "[00:13:a2:00:41:77:22:1e]!":50012,
        "[00:13:a2:00:41:77:22:40]!":50013

    }
```

9.3.2.- 868_TCP_send.py

```
## 868 SOFTWARE DE PAQUETIZACIÓN DE DATOS
## ======================================
# Autor: Carlos González
# Última modificación: 09.04.2018

## La secuencia de comandos de configuración es el archivo asociado
"config_cpx4.py".
## Cualquier modificación sujeta al código debe realizarse previamente en
este archivo
## config_cpx4.py
##    -node_list: Dirección MAC y puertos internos ZigBee para CPX4
##    -KS_port: Dirección MAC y  puerto Ethernet para enviar datos hacia
el Kepserver

"""
Este script está destinado a actuar como una aplicación de túnel de datos
entre la red TCP y la red zigbee.
Asigna múltiples sockets TCP para escuchar en puertos específicos en el
dispositivo de puerta de enlace, esto se define en
bind_table.py, que puede ser hecho a mano o generado. Cada uno de los
sockets TCP corresponde a un Zigbee específico
dirección en la red zigbee. Esto permite que una aplicación envíe por
ejemplo datos al puerto 4000 y siempre lo tenga
 enviado a la misma dirección de zigbee.

En este programa, los dispositivos serie 2 soportan una carga máxima de
72 bytes de datos, sin capacidad interna para
comprender datos fragmentados. Los dispositivos de la serie 1 tienen una
carga máxima de 100 bytes, pero al igual que el
```

```
anterior, sin capacidad interna para comprender datos fragmentados.

"""
#Se importan las librerias que se van a usar así como los archivos python
de los que haga uso este programa
#como es el caso de config_cpx4.
import encodings
# En este archivo se encuentran: la dirección mac de los dipositivos
usados xbee + connectport
# y se distinguen los puertos internos de los dispositivos y los de
salida hacia el servidor, además de un número
# identificatorio del nivel de orden.
import config_cpx4
import socket
import select
import sys
import zigbee
import errno
import struct
import traceback
from time import clock

tcp_conn_dict = {}
tcp_port_dict = {}
tcp_data_dict = {}  ##Datos asociados con los puertos TCP

# Carlos: configurar el puerto interno (desde el CPX4) para hacer un
socket desde el ZigBee,
#sin malentendidos con el puerto TCP utilizado para comunicarse con
Kepserver
# Carlos: referenciado al archivo "config_cpx4.py"

zig_data_queue = []
listen_list = []
client_list = []

MAX_ZIGBEE_PACKET_SIZE = 100 ##Carlos: se configura durante el bucle
MAX_TCP_PACKET_SIZE    = 8192
segment    = 0
end_point  = 0
profile_id = 0
cluster_id = 0

"""
Este valor debe modificarse para adaptarse a las necesidades de túnel de
datos.
Factores  como la velocidad en baudios del dispositivo remoto,
el tiempo de paquetización y el tamaño de los datos se deben de
considerar para elegir dicho factor.
"""

TCP_BUFFERING_TIME = .300 #300 milisegundos es el tiempo de espera fijado

##############################################################################
#########################
# Introducir el TCP_BUFFERING_TIME, por defecto se tomaría 300
milisegundos
##############################################################################
#########################
```

```
#Se introduce solamente un argumento, TCP_BUFFERING_TIME
try:
  if len(sys.argv) > 1:
    TCP_BUFFERING_TIME = float(sys.argv[1])
except Exception, e:
  print get_exception_info()
  print "TCP_BUFFERING_TIME requiere de un numero flotante, por defecto
es .3 (300 ms)"
  sys.exit(0)

##############################################################################
##########################
# Detecta el tipo de radio (serie) y asigna la información de dirección
en consecuencia
## Carlos: Detecta el módulo de radio instalado en la puerta de enlace y
asigna MAX_ZIGBEE_PACKET_SIZE
##############################################################################
##########################

"""
Debido a las diferencias entre las radios serie 1 y serie 2, se puede
determinar
que tipo de radio es el que se comunica con la gateway gracias a los
parámetros
end_point, profile_id y cluster_id. Para ello,se usa ddo_get_param
(local_radio, 'HV'),
que es el comando AT para obtener la versión de hardware.

Cuando se obtiene la versión de hardware, se descomprime la información
en una tupla,
 y se toma el primer (y único) elemento de esa tupla, y se asigna a
hw_version.
 Si es más de 6400, es un radio serie 2, si es menos de 6400, es serie 1.
"""

try:
  hw_version = zigbee.ddo_get_param(None, "HV")
  hw_version = struct.unpack("=H", hw_version)[0]
except Exception, e:
  hw_version = None
  print get_exception_info()
  print "Failed to retrieve hardware version from local radio"
  print "Assuming it's a series 1 device"

if hw_version != None:
  if hw_version > 6400: #Si la version de hardware es mayor de 6400 es un
serie 2
    print "Detectado Series 2 radio en la gateway, configurando el socket
zigbee de manera correcta"
    end_point  = 0xE8 #232
    profile_id = 0xC105
    cluster_id = 0x11 #Fuera del UART
    MAX_ZIGBEE_PACKET_SIZE = 100   #El tamaño de los paquetes es de 72
bytes máximo

  else:
    print "Detectado Series 1 radio en la gateway, configurando el socket
zigbee de manera correcta"
```

```
##########################################################################
#########################
#  cleanUpConn() Función que sirve para ir eliminando clientes y todo lo
asociado a estos como conexiones
##########################################################################
#########################
def cleanUpConn(conn):
  global client_list
  global tcp_conn_dict
  global tcp_data_dict

  try:
    client_list.remove(conn)    #Se elimina de la lista de escucha
    sock = tcp_conn_dict[conn] #Se localiza el listener
    tcp_conn_dict[sock] = None #Se configura el listener a None
    tcp_data_dict[sock] = []    #Se resetea la cola de datos
    del tcp_conn_dict[conn]     #Se elimina el par de valores clave para
la búsqueda inversa
    conn.close()                #Se cierra la instancia
    conn = None                 #y se pone a valor None
  except Exception, e:
    print get_exception_info()

def _format_exception_info(max_tb_level=20):

    e_type, e_value, e_traceback = sys.exc_info()
    e_name = e_type.__name__
    e_args = []

    try:
        for item in e_value.args:
          e_args.append(str(item))
    except AttributeError:
        pass

    try:
        for item in e_value.message:
          e_args.append(str(item))
    except AttributeError:
        pass

    tb_string = traceback.format_tb(e_traceback, max_tb_level)
    return (str(e_name), e_args, str(tb_string))

def get_exception_info():
    """Formatea la última excepción planteada y devuelve una cadena que
contiene el nombre de la excepción,
      cualquier información asociada con la excepción y un seguimiento de
llamada para la excepción."""

    e_name, e_args, e_tb = _format_exception_info()
    e_string = "".join(["Exception ", e_name, ":\n",
                        "".join(e_args), "\n",
                        "Traceback:\n", "".join(e_tb)])

    return e_string
```

```
################################################################################
#########################
# Inicio
# Carlos: a partir de aquí se debería de ajustar el pograma
################################################################################
#########################
"""
 Se busca la dirección actual de zigbee en diccionario de puertos tcp, se
crea un socket por entrada,
 se vincula el socket al puerto tcp especificado y se mapean los
diferentes diccionarios para
 proporcionar búsquedas rápidas
"""

##Al usar keys (), solo se genera la lista una vez al inicio,
##de lo contrario cambiará debido a la asignación de puertos -> búsquedas
de direcciones

for addr in config_cpx4.node_list.keys():

  port = config_cpx4.node_list[addr]

  ##Se crea un socket, se enlaza con localhost:port, se pone a escuchar y
se añade a la lista de escucha
  sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
  sock.bind(("", port))
  sock.listen(1)
  listen_list.append(sock)

  tcp_conn_dict[sock] = None  ##Inicia el listener al diccionario de
connexión

  tcp_port_dict[sock] = port  ##Asignar listener a la búsqueda del puerto
  tcp_port_dict[port] = sock  ##Asignar puerto a la búsqueda de listeners

  tcp_data_dict[sock] = []    ##Asignar listener a la búsqueda
tcp_data_queue

  config_cpx4.node_list[port] = addr  ##Asignar puerto a la busqueda
zigbee_address

##La búsqueda entre la dirección y el puerto ya existe en este punto

#finaliza la key en config_cpx4.node_list


## Se crea el socket zigbee, se vincula a un punto final local,
configurado para ser no bloqueante

zig_sock = socket.socket(socket.AF_ZIGBEE, socket.SOCK_DGRAM,
socket.ZBS_PROT_TRANSPORT)
zig_sock.bind(('', end_point, profile_id, cluster_id))
zig_sock.setblocking(0)


## Crear un socket TCP por dispositivo para comunicarse con Kepserver
## Carlos: utilizar un bucle for para iterar en la lista de nodos y
generar directamente el número de sockets TCP
## igual que los dispositivos. La relación será node_list -> socket
```

```
n_TCPsockets = len(config_cpx4.node_list)
socket_kepserver_xbee= [None]*n_TCPsockets
for device in range(n_TCPsockets):
    socket_kepserver_xbee[device]=socket.socket(socket.AF_INET,
socket.SOCK_DGRAM)



##############################################################################
############################
# Bucle principal
##############################################################################
############################

print "Inicio bucle principal"
while 1:

  """
  Se escucha en los sockets de escucha para nuevas conexiones. Los
sockets clientes para los
  nuevos datos TCP y los sockets zigbee para los datos que vienen de los
XBee.

  Se escucha la capacidad de escritura de los sockets de los clientes
para que se pueda escribir
  los datos zigbee recibidos y en cola sobre ellos, y tambien se escucha
la capacidad de escritura en el
  socket zigbee para escribir en él los datos tcp recibidos y en cola.
  """

  rl, wl, el = select.select(listen_list + client_list + [zig_sock],
client_list + [zig_sock],[],.5)

#Sockets de lectura
#Sockets de escritura
#No usado (lista de errores)


##############################################################################
############################
  # Lectura de Zigbee
  # Carlos: Parte importante, gestiona la conexión y recibe datos de
zigbees que tiene en la
  #red y luego retransmite esta información a la puerta de enlace
(Kepserver) a través de
  #una conexión TCP / IP

##############################################################################
############################

  if zig_sock in rl:
    data, addr = zig_sock.recvfrom(MAX_ZIGBEE_PACKET_SIZE)
    print "Zigbee read: %d bytes from address " %(len(data)), addr
    try:
        port_ord = config_cpx4.device_order[addr[0]]
            #envío de datos al keepserver
```

```
socket_kepserver_xbee[port_ord].sendto(data,(config_cpx4.KS_ipAddr,config
_cpx4.KS_port[addr[0]]))
    except:
        print("Problema enviando los datos al KEPSERVER")
```

10.- PLANIFICACIÓN

En el presente capítulo se mostrarán las tareas llevadas a cabo en el proyecto actual. Se explicará brevemente en que ha consistido cada una de ellas así como el tiempo estimado de las mismas.

Para finalizar se mostrará un Diagrama de Gantt donde se podrán ver reflejadas las tareas a las que se hace mención. Se destaca que para todas ellas solo se dispone de un recurso humano, por lo que cuando se indique que se están desarrollando dos tareas a la vez ambas están siendo realizadas por la misma persona.

10.1.- Tareas y Diagrama de Gantt

Las tareas de las que consta el Diagrama de Gantt y por tanto el proyecto realizado son las siguientes:

- **Estudio del estado del arte**

En esta tarea se ha realizado un análisis sobre los distintos tipos de redes y protocolos usados a día de hoy en diferentes entornos de trabajo, pero centrándose sobre todo en el ámbito industrial. La finalidad es la de saber los principales elementos de los que consta una red IoT y los protocolos en los que se basan para posteriormente poder llevar a cabo las tareas que siguen.

- **Elección de componentes**

Se ha analizado el mercado actual para comprobar que elementos podrían converger en una red que fuese fiable y funcional al 100%.

- **Configuración de los componentes**

Una vez escogidos los componentes se ha realizado una configuración de tal manera que todos ellos formen una única red y puedan identificarse entre sí.

- **Implementación de la comunicación**

Tarea que consta en la creación de códigos de programación en Python y que permitan que los datos enviados desde un módulo XBee puedan llegar al *Gateway* y esta pueda reenviarlos hacia el servidor.

- **Configuración del servidor**

Basada en lograr que los datos enviados por un módulo inalámbrico se puedan visualizar en el servidor y que se actualicen los datos de manera automática.

- **Sensor RGB y Arduino NANO**

Tarea basada en lograr obtener los datos que genera el sensor RGB y poder enviarlos hacia un Arduino NANO y después desde este último hacia un módulo XBee.

- **Diseño placa de circuito impreso**

Una vez que se ha logrado realizar la comunicación entre los elementos: sensor, Arduino y XBee, se ha diseñado una PCB con la finalidad de realizar un prototipo real.

- **Fabricación placa de circuito impreso**

Esta tarea indica el periodo temporal donde se realiza físicamente la placa, es decir, el tiempo entre que se ha diseñado la placa y esta se recibe para ser construida.

- **Construcción prototipo**

Una vez recibida la placa de circuito impreso se procede a la construcción de la misma para que adquiera la forma de un prototipo que se podría comercializar.

- **Verificación y realización de pruebas**

Comprobación del prototipo y ver como este forma parte de la red sin ningún problema y el proceso de comunicación se realiza de manera correcta en distintas condiciones.

En la Figura 9.1 se puede ver el Diagrama de Gantt realizado.

Id.	Nombre de tarea	Inicio	Finalizar	Duración
1	Estudio del estado del arte	01/02/2018	21/02/2018	15d
2	Elección de componentes	22/02/2018	02/03/2018	7d
3	Configuración de los componentes	05/03/2018	09/03/2018	5d
4	Implementación de la comunicación	12/03/2018	13/04/2018	25d
5	Configuración del servidor	16/04/2018	19/04/2018	4d
6	Sensor RGB y Arduino NANO	20/04/2018	27/04/2018	6d
7	Diseño placa de circuito impreso	30/04/2018	04/05/2018	5d
8	Fabricación placa de circuito impreso	07/05/2018	18/05/2018	10d
9	Construcción de prototipo	21/05/2018	22/05/2018	2d
10	Verificación y realización de pruebas	23/05/2018	28/05/2018	4d

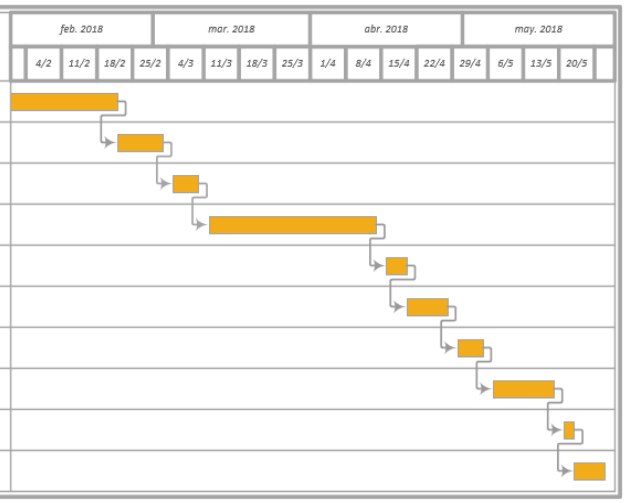


Figura 9.1.- Diagrama de Gantt del presente proyecto.

11.- REFERENCIAS

[1] [Online] Lawrence Jones, "El auge del Internet de las Cosas", http://www.think-progress.com/es/movilidad/el-auge-del-internet-de-las-cosas-2/, [Accedido: 14/05/2018].

[2] [Online] Wikipedia, "Internet de las Cosas", https://es.wikipedia.org/wiki/Internet_de_las_cosas, [Accedido: 14/05/2018].

[3] [Online] Altonivel.com, "7 áreas que pueden aprovechar el Internet de las Cosas", https://www.altonivel.com.mx/empresas/negocios/52406-7-areas-para-desarrollar-el-internet-de-las-cosas/ , [Accedido: 14/05/2018].

[4] [Online] James Whittaker, "The Internet of Things", https://dcps.instructure.com/courses/9338/pages/2-slash-5-the-internet-of-things, [Accedido: 15/05/2018].

[5] [Online] Bruno Cendón, "El origen del IoT", http://www.bcendon.com/el-origen-del-iot/ , [Accedido: 15/05/2018].

[6] [Online] Mark Albert, "Siete cosas sobre el Internet de las Cosas y la Industria 4.0", https://www.mms-mexico.com/art%C3%ADculos/siete-cosas-sobre-el-internet-de-las-cosas-y-la-industria-40- , [Accedido: 15/05/2018].

[7] [Online] Ministerio de Industria, Energía y Turismo, "La transformación digital de la industria española: informe preliminar", http://www6.mityc.es/IndustriaConectada40/informe-industria-conectada40.pdf , [Accedido: 16/05/2018].

[8] [Online] Pera Ponsa y Antoni Granollers, "Diseño y automatización industrial". Universidad Politécnica de Cataluña, https://www.epsevg.upc.edu/hcd/material/lecturas/interfaz.pdf , [Accedido: 16/05/2018].

[9] [Online] "Conectividad en el Internet de las Cosas — Topología de Redes", http://aprender.tdrobotica.co/cursos/conectividad-en-el-internet-de-las-cosas/lessons/topologia-de-redes/ , [Accedido: 16/05/2018].

[10] [Online] Logicbus, "Tecnología Zigbee, Boletín Técnico", http://www.logicbus.com.mx/News_Mail/2014/v17e010914.html , [Accedido: 16/05/2018].

[11] [Online] JJ Velasco, "¿En qué consiste el BLE?", https://hipertextual.com/2013/12/que-es-bluetooth-le , [Accedido: 16/05/2018].

[12] [Online] Daniel García García, "Estudio de 6loWPAN para su aplicación al IoT", https://riull.ull.es/xmlui/bitstream/handle/915/945/Estudio%20de%206loWPAN%20para%20su%20aplicacion%20a%20Internet%20de%20las%20Cosas.pdf?sequence=1 , [Accedido: 16/05/2018].

[13] [Online] "Las tecnologías WiFi y WiMax", http://www.dip-badajoz.es/agenda/tablon/jornadaWIFI/doc/tecnologias_wifi_wmax.pdf , [Accedido: 17/05/2018].

[14] [Online] "Estándar GSM, Sistema Global de Comunicaciones móviles", https://es.ccm.net/contents/681-estandar-gsm-sistema-global-de-comunicaciones-moviles , [Accedido: 17/05/2018].

[15] [Online] "¿Qué es y cómo funciona SigFox?", http://blog.330ohms.com/2017/05/11/que-es-sigfox-y-como-funciona/ , [Accedido: 17/05/2018].

[16] [Online] Ramón Pérez Hernández, "Desarrollo de prototipo de sensor Internet of Things usando la red SigFox", http://bibing.us.es/proyectos/abreproy/90269/fichero/tfgRamonPerezHernandez.pdf , [Accedido: 17/05/2018].

[17] [Online] Alexa de Blois, "Fotómotro o Exposímetro: qué es y para qué se utiliza", https://www.blogdelfotografo.com/fotometro-exposimetro/ [Accedido: 09/06/2018].

[18] [Online] "WirelessHART", https://www.pepperl-fuchs.com/global/es/10028.htm [Accedido: 21/05/2018].

[19] [Online] "WirelessHART: Tecnología de radio en ingeniería de procesos", https://www.pepperl-fuchs.com/spain/es/3869.htm [Accedido: 21/05/2018].

[20] [Online] Ignacio Ordoñez Monfort, "Estudio de la arquitectura y el nivel de desarrollo de la red LoRaWAN y de los dispostivos LoRa", http://openaccess.uoc.edu/webapps/o2/bitstream/10609/64365/6/iordonezTFM0617mem%C3%B2ria.pdf, [Accedido: 21/05/2018].

[21] [Online] "ABB WiMon Condition Monitoring System", http://new.abb.com/oil-and-gas/products/rotating-machines/wimon [Accedido: 21/05/2018].

[22] [Online] "WISE-4000 Series: IoT Ethernet I/O Module", http://www.mcielectronics.cl/website_MCI/static/documents/Manual_WISE-4000.pdf [Accedido: 21/05/2018].

[23] [Online] "WISE-PaaS", http://wise-paas.advantech.com/en-us/embedded [Accedido: 21/05/2018].

[24] [Online] "Libelium World", http://www.libelium.com/smart_cities/ [Accedido: 21/05/2018].

[25] [Online] Javier Martín Moreno y Daniel Ruiz Fernández, "Informe Técnico: Protocolo ZigBee (IEEE 802.15.4)", https://rua.ua.es/dspace/bitstream/10045/1109/7/Informe_ZigBee.pdf [Accedido: 21/05/2018].

[26] [Online] Jean M.Winter, "WirelessHART: an overview", https://www.slideshare.net/eawareTech/wirelesshart , [Accedido: 22/05/2018].

[27] [Online] "Libelium Integrates Sensors With IBM Bluemix Cloud for Smart Cities", https://www.businesswire.com/news/home/20151201005677/en/Libelium-Integrates-Sensors-IBM-Bluemix-Cloud-Smart , [Accedido: 22/05/2018].

[28] [Online] "Condition Monitoring and Wireless Vibration Transmitter", http://new.abb.com/products/measurement-products/wireless-products-and-solutions/integrated-solutions/condition-monitoring-and-wireless-vibration-transmitter-copy20140702_151332 , [Accedido: 22/05/2018].

[29] [Online] "Redes de malla inalámbricas: ZigBee frente a DigiMesh", https://www.arrow.com/es-mx/research-and-events/articles/wireless-mesh-networking-zigbee-vs-digimesh , [Accedido: 22/05/2018].

[30] [Online] "ConnectPort X4 Family", https://www.digi.com/pdf/ds_connectportx4.pdf , [Accedido: 25/05/2018].

[31] [Online] "Digi XBee SX868", https://www.digi.com/products/xbee-rf-solutions/sub-1-ghz-modules/digi-xbee-sx-868#specifications , [Accedido: 25/05/2018].

[32] [Online] "Aprendiendo Arduino: Aprendiendo a manejar Arduino en profundidad", https://aprendiendoarduino.wordpress.com/2015/03/29/microcontrolador-vs-microprocesador/ , [Accedido: 25/05/2018].

[33] [Online] "Arduino NANO", https://store.arduino.cc/usa/arduino-nano , [Accedido: 29/05/2018].

[34] [Online] Luis Llamas, "Medir Valores RGB con Arduino y Sensor de Color TCS34725", https://www.luisllamas.es/arduino-sensor-color-rgb-tcs34725/ ,[Accedido: 29/05/2018].

[35] [Online] "KEPServerEX", https://www.kepware.com/en-us/products/kepserverex/ , [Accedido: 29/05/2018].

[36] [Online] "Datasheet TSR 1-2433", https://www.mouser.es/datasheet/2/687/tsr1-537631.pdf , [Accedido: 08/06/2018].

[37] [Online] Jennifer Clarkson, "Photography 101: White Balance explained", https://www.picturecorrect.com/tips/photography-white-balance-explained/ ,[Accedido: 09/06/2018].

[38] [Online] Courtney Slazinik, "White Balance: comparing different methods", https://clickitupanotch.com/white-balance-comparing-different-methods/ , [Accedido: 09/06/2018].

[39] [Online] HXX, "Materiales de impresión 3D: PLA", http://hxx.es/2015/03/12/materiales-de-impresion-3d-i-pla-acido-polilactico/ , [Accedido: 19/06/2018].

[40] [Online] 3DNatives, "FDM o modelado por deposición fundida", https://www.3dnatives.com/es/modelado-por-deposicion-fundida29072015/ , [Accedido: 19/06/2018].

[41] [Online] Linking Innovations, "Industrialización de productos", http://www.linkinginnovations.es/servicios/industrializacion-de-productos , [Accedido: 19/06/2018].

[42] [Online] Wen-Hsing Kuo, Yung-Sheng Chen, Kui-Tai Cheng, and Tai-Wei Lu, "Signal Strength Based Indoor and Outdoor Localization Scheme in Zigbee Sensor Networks", https://pdfs.semanticscholar.org/5dd9/95aeb03166ef8bf266d6ef065f0de8261d40.pdf, [Accedido: 23/06/2018].

[43] [Online] John Conway, "The Industrial Internet of Things: An Evolution to a Smart Manufacturing Enterprise", http://www.mhi.org/media/members/15373/131111777451441650.pdf , [Accedido: 27/06/2018].

[44] [Online] Aroha Llanos, "Diferencias entre IIoT e IoT", https://oasys-sw.com/diferencias-iiot-e-iot/ , [Accedido: 27/06/2018].

PLANOS

ÍNDICE DE PLANOS

201809-1 Esquema eléctrico PCB alimentada mediante toma de red.

201809-2 Esquema eléctrico PCB alimentada mediante batería.

201809-3 Fotolito de la capa superior del circuito impreso de la placa alimentada mediante toma de red.

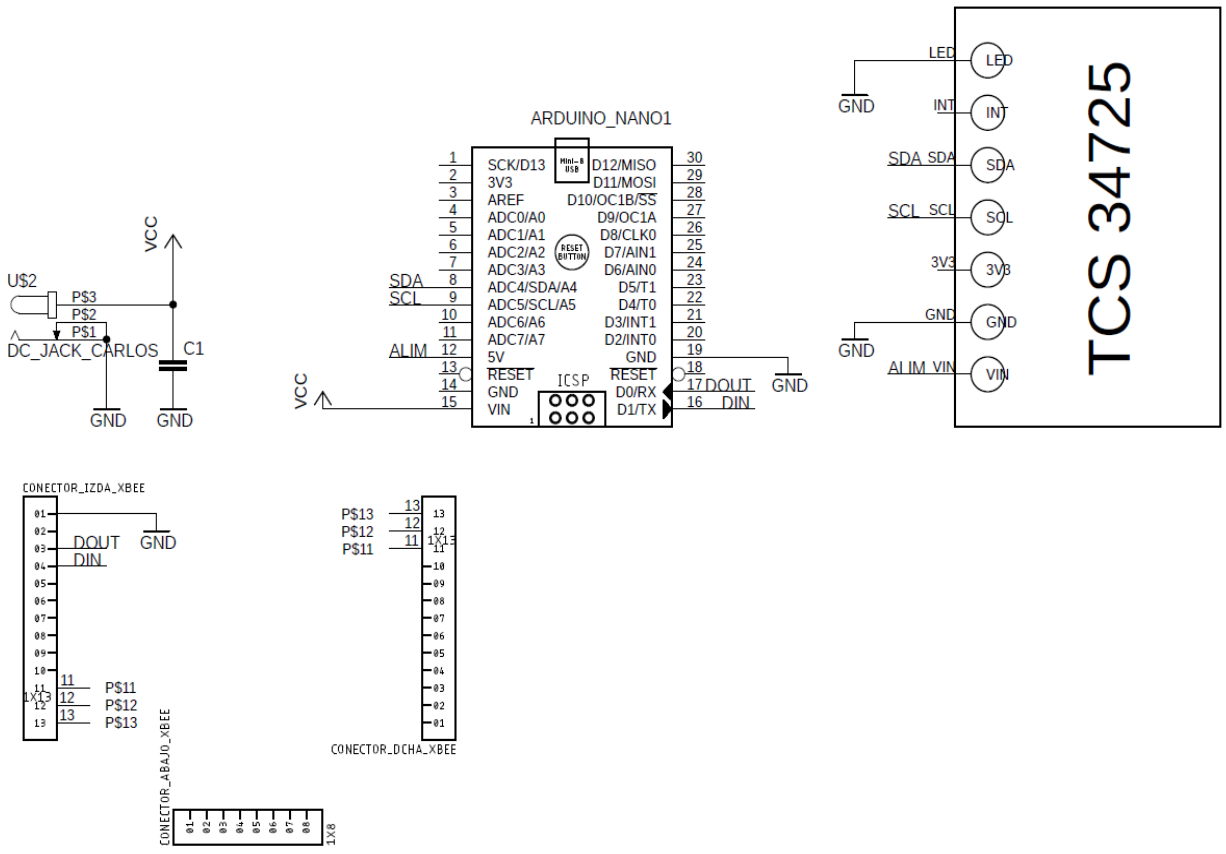
201809-4 Fotolito de la capa inferior del circuito impreso de la placa alimentada mediante toma de red.

201809-5 Fotolito de la capa superior del circuito impreso de la placa alimentada mediante batería.

201809-6 Fotolito de la capa inferior del circuito impreso de la placa alimentada mediante batería.

201809-7 Carcasa para prototipo impresa en 3D.

201809-8 Tapa de carcasa para prototipo impresa en 3D.

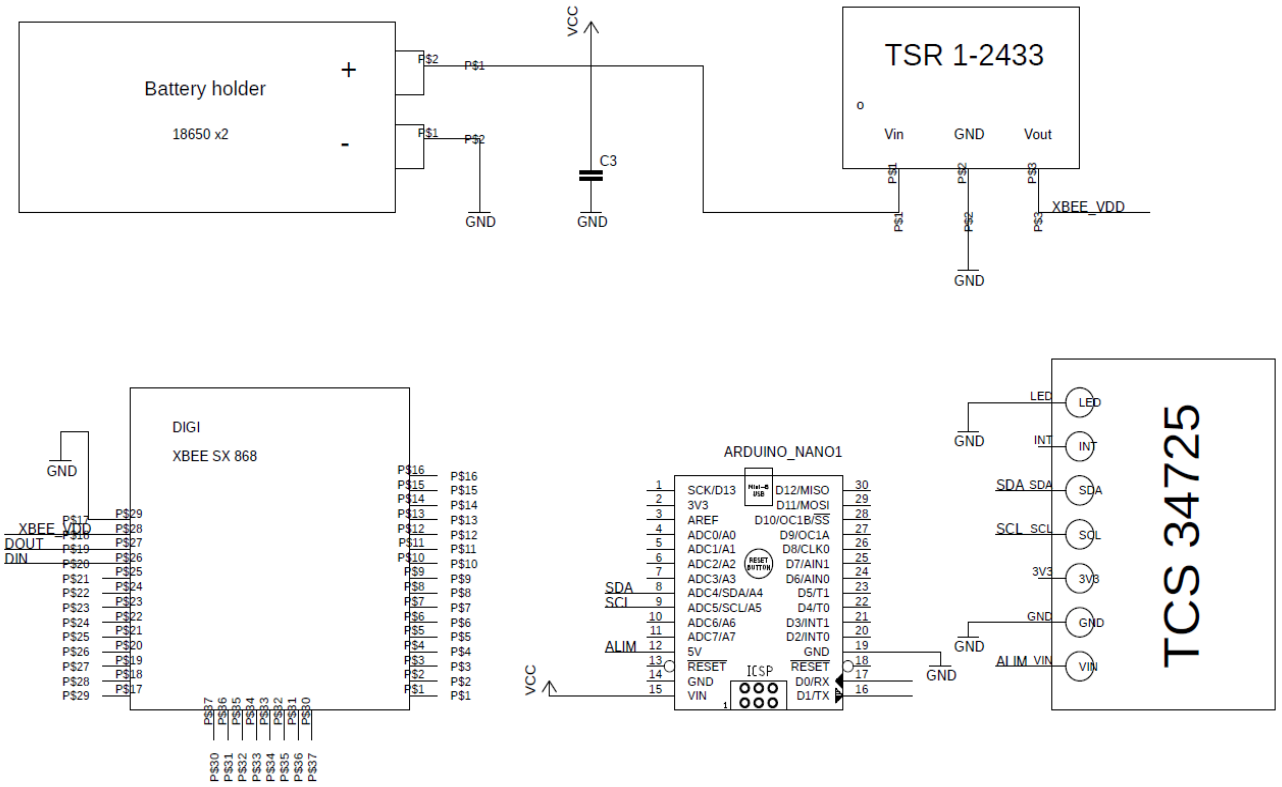


ESCUELA POLITÉCNICA DE INGENIERÍA DE GIJÓN

TRABAJO FIN DE MÁSTER

	FECHA	NOMBRE	FIRMA	
Proyectado	10/06/2018	Carlos González		TECNOLOGÍAS DE REDES IOT APLICABLES A PROCESOS INDUSTRIALES
Dibujado	10/06/2018	Carlos González		
Comprobado	10/06/2018	Andrés García		

ESCALA	ESQUEMA ELÉCTRICO PCB ALIMENTADA MEDIANTE TOMA DE RED	PLANO Nº 201809-1
		Sustituye a
		Sustituido por

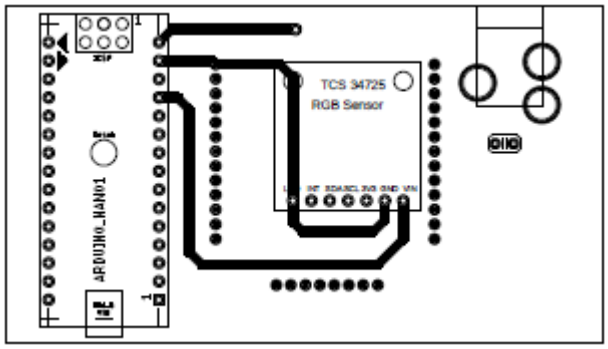


ESCUELA POLITÉCNICA DE INGENIERÍA DE GIJÓN

TRABAJO FIN DE MÁSTER

	FECHA	NOMBRE	FIRMA	TECNOLOGÍAS DE REDES IOT APLICABLES A PROCESOS INDUSTRIALES
Proyectado	10/06/2018	Carlos González		
Dibujado	10/06/2018	Carlos González		
Comprobado	10/06/2018	Andrés García		

ESCALA	ESQUEMA ELÉCTRICO PCB ALIMENTADA MEDIANTE BATERÍA	PLANO Nº 201809-2
		Sustituye a
		Sustituido por

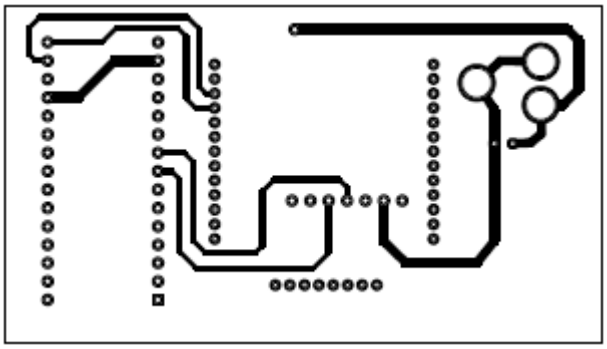


ESCUELA POLITÉCNICA DE INGENIERÍA DE GIJÓN

TRABAJO FIN DE MÁSTER

	FECHA	NOMBRE	FIRMA	TECNOLOGÍAS DE REDES IOT APLICABLES A PROCESOS INDUSTRIALES
Proyectado	10/06/2018	Carlos González		
Dibujado	10/06/2018	Carlos González		
Comprobado	10/06/2018	Andrés García		

ESCALA 1:1	FOTOLITO DE LA CAPA SUPERIOR DEL CIRCUITO IMPRESO DE LA PLACA ALIMENTADA MEDIANTE TOMA DE RED	PLANO Nº 201809-3
		Sustituye a
		Sustituido por

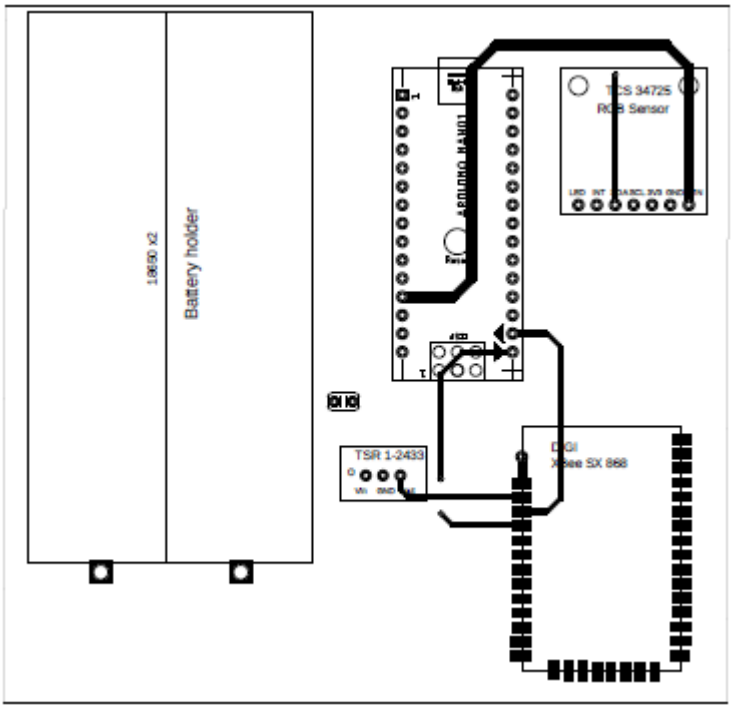


ESCUELA POLITÉCNICA DE INGENIERÍA DE GIJÓN

TRABAJO FIN DE MÁSTER

	FECHA	NOMBRE	FIRMA	TECNOLOGÍAS DE REDES IOT APLICABLES A PROCESOS INDUSTRIALES
Proyectado	10/06/2018	Carlos González		
Dibujado	10/06/2018	Carlos González		
Comprobado	10/06/2018	Andrés García		

ESCALA 1:1	FOTOLITO DE LA CAPA INFERIOR DEL CIRCUITO IMPRESO DE LA PLACA ALIMENTADA MEDIANTE TOMA DE RED	PLANO Nº 201809-4
		Sustituye a
		Sustituido por

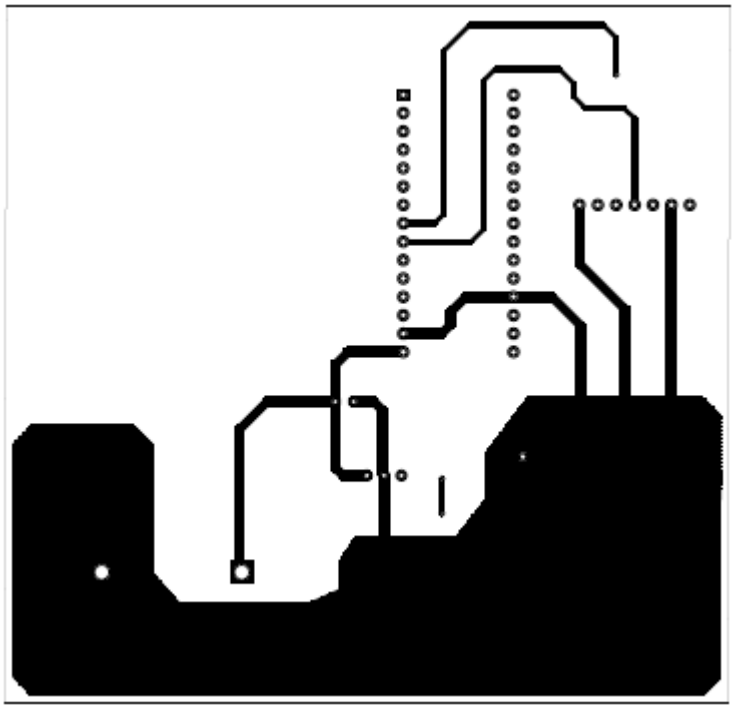


ESCUELA POLITÉCNICA DE INGENIERÍA DE GIJÓN

TRABAJO FIN DE MÁSTER

	FECHA	NOMBRE	FIRMA	TECNOLOGÍAS DE REDES IOT APLICABLES A PROCESOS INDUSTRIALES
Proyectado	10/06/2018	Carlos González		
Dibujado	10/06/2018	Carlos González		
Comprobado	10/06/2018	Andrés García		

ESCALA 1:1	FOTOLITO DE LA CAPA SUPERIOR DEL CIRCUITO IMPRESO DE LA PLACA ALIMENTADA MEDIANTE BATERÍA	PLANO Nº 201809-5
		Sustituye a
		Sustituido por

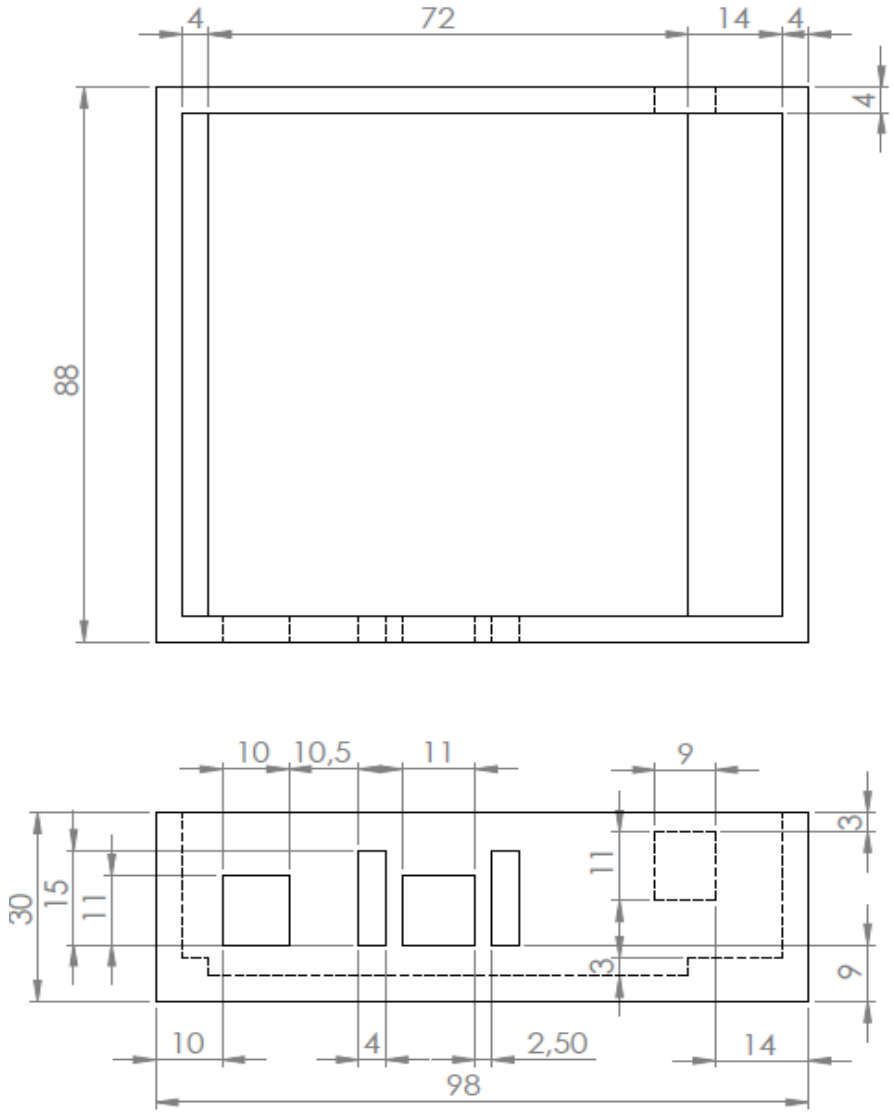


ESCUELA POLITÉCNICA DE INGENIERÍA DE GIJÓN

TRABAJO FIN DE MÁSTER

	FECHA	NOMBRE	FIRMA	TECNOLOGÍAS DE REDES IOT APLICABLES A PROCESOS INDUSTRIALES
Proyectado	10/06/2018	Carlos González		
Dibujado	10/06/2018	Carlos González		
Comprobado	10/06/2018	Andrés García		

ESCALA 1:1	FOTOLITO DE LA CAPA INFERIOR DEL CIRCUITO IMPRESO DE LA PLACA ALIMENTADA MEDIANTE BATERÍA	PLANO Nº 201809-6
		Sustituye a
		Sustituido por

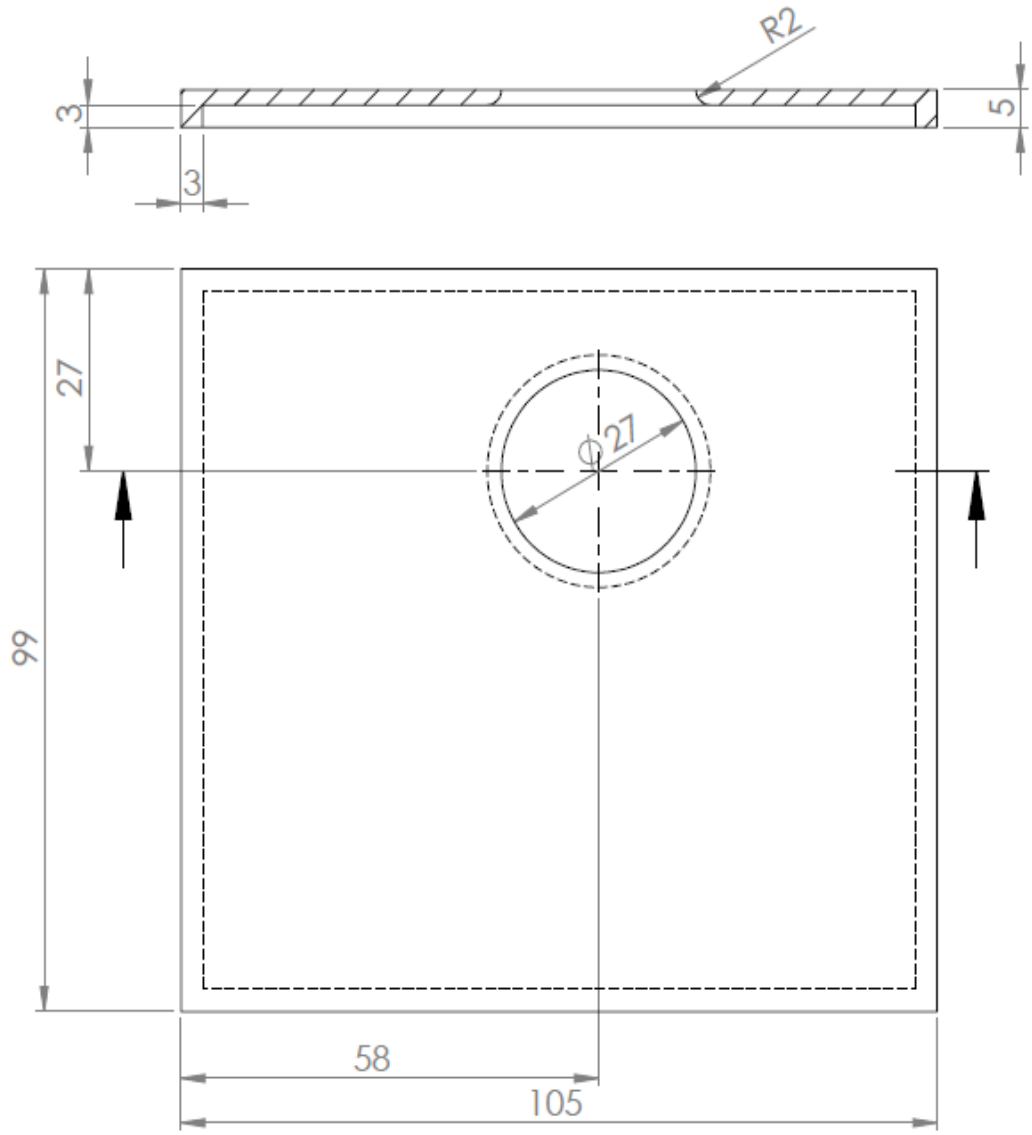


ESCUELA POLITÉCNICA DE INGENIERÍA DE GIJÓN

TRABAJO FIN DE MÁSTER

	FECHA	NOMBRE	FIRMA	TECNOLOGÍAS DE REDES IOT APLICABLES A PROCESOS INDUSTRIALES
Proyectado	10/06/2018	Carlos González		
Dibujado	10/06/2018	Carlos González		
Comprobado	10/06/2018	Andrés García		

ESCALA 1:1	CARCASA PARA PROTOTIPO IMPRESA EN 3D	PLANO Nº 201809-7
		Sustituye a
		Sustituido por



ESCUELA POLITÉCNICA DE INGENIERÍA DE GIJÓN

TRABAJO FIN DE MÁSTER

	FECHA	NOMBRE	FIRMA	TECNOLOGÍAS DE REDES IOT APLICABLES A PROCESOS INDUSTRIALES
Proyectado	10/06/2018	Carlos González		
Dibujado	10/06/2018	Carlos González		
Comprobado	10/06/2018	Andrés García		

ESCALA 1:1	TAPA DE CARCASA PARA PROTOTIPO IMPRESA EN 3D	PLANO Nº 201809-8
		Sustituye a
		Sustituido por

PRESUPUESTO

ÍNDICE DE PRESUPUESTO

1.-COMPONENTES 2
2.-PERSONAL 3
3.-PRESUPUESTO TOTAL 4

1.-COMPONENTES

	Componente	Unidades	Precio unitario (€)	Precio total (€)
1.1	Kit de desarrollo XBee SX868	1	163,18	163,18
1.2	Pack 3 Arduino NANO	1	10,99	10,99
1.3	Carcasa 3D	1	10,00	10,00
1.4	Condensador cerámico	1	0,10	0,10
1.5	Conector JACK hembra	1	1,05	1,05
1.6	Tira de pines 2mm (x10)	1	2,26	2,26
1.7	Lumiesfera	1	10,04	10,04
1.8	Sensor TCS34725	1	7,95	7,95
1.9	Placa de circuito impreso	1	6,10	6,10
1.10	ConnectPort X4IA	1	596,40	596,40

Subtotal componentes (€)	808,07

2.-PERSONAL

	Componente	Unidades	Precio unitario (€)	Precio total (€)
2.1	Horas de investigación y desarrollo	200	20,00	4.000,00
2.2	Horas de montaje y ajustes	75	20,00	1.500,00
2.3	Horas de pruebas	50	20,00	1.400,00

Subtotal personal (€)	6.900,00

3.-PRESUPUESTO TOTAL

Total componentes (€)	808.07
Total personal (€)	6.900,00

Total presupuesto de ejecución (€)	7.708,70
Costes indirectos (10%)	770,87

Total parcial (€)	8.479,57
IVA (21%)	1.780,71

Total Presupuesto de Ejecución por Contrata (€)	10.260,28

Asciende el Presupuesto de Ejecución por Contrata a la cantidad de **DIEZ MIL DOSCIENTOS SESENTA EUROS CON VENTIOCHO CÉNTIMOS.**