





## AGRADECIMIENTOS

En primer lugar, me gustaría agradecer a mis tutores de proyecto Jorge Portilla y Gabriel Noe Mujica. Agradecer su colaboración y su experiencia en redes de sensores inalámbricas.

Dar las gracias también a mi familia por su constante apoyo y ánimo en épocas difíciles.

Gracias a Paula y a José por su amistad y por su apoyo incondicional durante todo este tiempo.



## RESUMEN

Los principales objetivos de este proyecto es la implementación hardware de la tecnología Thread en la plataforma Cookie y el análisis de esta nueva tecnología de comunicaciones. Dicha implementación consistirá en un diseño de esquemáticos y de enrutado de una PCB integrando el módulo KTWM102 desarrollado por Kirale Technologies.

Previamente a la implementación hardware, se prueba y se familiariza con la tecnología Thread con los Dongle de Evaluación desarrollados por Kirale.

Posteriormente a la familiarización con la tecnología, se realiza el diseño hardware incorporando el módulo KTWM102.

Una vez realizado el diseño hardware, se valida su diseño mediante la realización de pruebas de comunicación tanto con un PC, a través del puerto USB, como con la plataforma Cookie, a través del puerto UART. Para esta validación se formaban diferentes topologías de red con los nodos disponibles, incluyendo los Dongle de Evaluación, permitiendo evaluar no solo el correcto diseño hardware, si no también se evalúa el funcionamiento del protocolo Thread implementado.

Finalmente, tras la validación de la integración del hardware en la plataforma, se adquiere el dispositivo Border Router de Kirale. Con ese nuevo dispositivo, se logra una conectividad entre la red Thread, formada por los diferentes nodos y el Border Router, con otros tipos de redes, incluyendo conexión a Internet.

**Palabras clave:** Implementación, Thread, Nodos, Red, Validación, Análisis.



## ABSTRACT

The aim of this project is the hardware implementation of Thread technology on the Cookie platform and the analysis of the Thread communications technology. The hardware implementation consist on an entire PCB design, integrating the KTWM102 module developed by Kirale Technologies.

The firsts steps in the project are testing and getting used to the Thread Technology using de Evaluation Dongles modules developed by Kirale.

Once achieving a certain knowledge about the technology, the hardware design incorporating the module KTWM102 is started.

Next, once the design is achieved, it is validated its design through communications tests both with a PC, through the USB port, and with the Cookie Platform, through the UART port. Different networks topologies are made with all available nodes, including both Evaluation Dongles available, allowing to assess, not only the correct operation of the hardware design, but the operation of the implemented Thread protocol.

Finally, when the correct integration of the hardware design in the Cookie Platform is validated, the Border Router developed by Kirale is acquired. With the Border Router, a new connectivity between the Thread network, formed by the different nodes and the Border Router, and other types of networks due to the Border Router, including Internet connection.

**Keywords:** Implementation, Thread, Nodes, Red, Validation, Analysis.





# INDICE

AGRADECIMIENTOS .....	I
RESUMEN.....	III
ABSTRACT .....	V
INDICE .....	VII
ABREVIATURAS Y ACRÓNIMOS .....	XI
ILUSTRACIONES.....	XIII
TABLAS.....	XV
1. INTRODUCCIÓN Y OBJETIVOS DEL TRABAJO .....	1
1.1. MOTIVACIÓN DEL PROYECTO.....	1
1.2. OBJETIVOS DEL PROYECTO .....	1
1.3. GESTIÓN DEL PROYECTO .....	2
2. ESTADO DEL ARTE.....	3
2.1. INTERNET OF THINGS (IoT) .....	3
2.1.1. INTRODUCCIÓN [2].....	3
2.1.2. COMPONENTES [2] .....	4
2.1.2.1. DISPOSITIVO.....	4
2.1.2.2. RED LOCAL .....	4
2.1.2.3. INTERNET .....	5
2.1.2.4. SERVICIOS BACKEND.....	5
2.1.2.5. APLICACIONES .....	5
2.1.3. SENSORES.....	5
2.1.4. ARQUITECTURA [3] .....	6
2.1.4.1. CAPA DE SENSORIZADO.....	6
2.1.4.2. CAPA DE RED .....	6
2.1.4.3. CAPA DE SERVICIO.....	7
2.1.4.4. CAPA DE INTERFAZ .....	8
2.2. 6LoWPAN [4] .....	9
2.2.1. SIGNIFICADO 6LoWPAN [4] .....	9
2.2.2. PILA DE PROTOCOLOS DE 6LoWPAN [4]–[6] .....	9
2.2.3. PRINCIPALES CARACTERÍSTICAS DE 6LoWPAN [7] .....	11
2.2.3.1. COMPRESIÓN DE CABECERA .....	11
2.2.3.2. ENRUTAMIENTO .....	11
2.2.3.3. SEGURIDAD .....	11
2.2.3.4. PROTOCOLOS DE APLICACIÓN .....	12
2.2.4. RETOS DE 6LoWPAN .....	12
2.2.5. IMPLEMENTACIONES Y APLICACIONES PARA 6LoWPAN [5], [7] .....	13
2.2.5.1. CONTIKI .....	13

	TINYOS .....	14
	THREAD .....	14
2.3.	THREAD .....	15
2.3.1.	INTRODUCCIÓN A REDES THREAD [8], [10]–[12].....	15
2.3.2.	TIPOS DE DISPOSITIVOS .....	16
2.3.3.	PROTOCOLO THREAD .....	18
2.2.5.2.		
2.2.5.3.	REDES DE ÁREA PRIVADA (PAN) [12].....	18
	CAPA FÍSICA [12], [14] .....	21
	CAPA MAC (O ENLACE DE DATOS). [12], [14].....	21
2.3.3.1.	CAPA DE ADAPTACIÓN 6LoWPAN [12].....	22
2.3.3.2.	CAPA DE RED [12] .....	23
2.3.3.3.	PROTOCOLO DE ENRUTAMIENTO .....	23
2.3.3.4.	CAPA DE TRANSPORTE [12] .....	24
2.3.3.5.		
2.3.3.6.	SEGURIDAD Y COMMISSIONING DE DISPOSITIVOS [10], [12] .....	24
2.3.3.7.		
3.	ANÁLISIS DE LA TECNOLOGÍA THREAD .....	27
2.3.3.8.		
3.1.	ANÁLISIS INICIAL DE LA TECNOLOGÍA THREAD .....	27
3.1.1.	CARACTERÍSTICAS TÉCNICAS DE LOS MÓDULOS KIRALE.....	28
	CARACTERÍSTICAS MÓDULO RF KTWM102 .....	28
3.1.1.1.	CARACTERÍSTICAS KTDG102 EVALUATION DONGLE.....	28
3.1.1.2.	CARACTERÍSTICAS DEL BORDER ROUTER KTBRN1 .....	29
3.1.1.3.		
3.1.1.4.	CARACTERÍSTICAS KINOS – NETWORK OS.....	29
3.1.2.	EJEMPLOS DE OTROS DISPOSITIVOS .....	30
3.2.	CONFIGURACIONES INICIALES .....	31
3.2.1.	CONFIGURACIÓN DEL BORDER ROUTER .....	31
3.2.1.1.		
3.2.1.2.	REQUISITOS.....	31
	INSTALACIÓN DEL SW .....	31
3.2.1.2.1.	DESCARGA DEL SW REQUERIDO.....	31
3.2.1.2.2.	FLASHEAR LA IMAGEN EN LA TARJETA SD .....	32
3.2.1.2.3.	INSTALACIÓN DE DRIVERS USB .....	33
3.2.1.2.3.1.	CONEXIÓN VÍA PUERTO USB SERIE .....	33
	PANEL DE ADMINISTRACIÓN WEB .....	35
3.2.1.3.1.	CAMBIAR LA CONFIGURACIÓN DE RED .....	36
3.2.1.3.2.	ACTUALIZAR KIBRA.....	38
3.2.1.3.3.	CONFIGURAR BORDER ROUTER.....	39
3.2.1.3.3.1.	UNIRSE O FORMAR UNA RED THREAD .....	39
3.2.1.3.3.2.	BACKBONE ROUTER SERVER (BBR).....	40
3.2.1.3.3.3.	PREFIJO DE RED (NETWORK PREFIX).....	41
3.2.1.3.4.	INICIO DEL BORDER ROUTER (START-UP). .....	42
3.2.1.3.5.	SERVICIOS .....	46
3.2.1.3.5.1.	SERVIDOR BACKBONE ROUTER .....	46

3.2.1.3.5.2.	DHCP .....	47
3.2.1.3.5.3.	NAT64.....	48
3.2.1.3.5.4.	COMMISSIONER .....	49
3.2.1.3.6.	VISUAL NETWORK .....	50
3.2.1.3.7.	LOGS.....	51
	BREVE RESUMEN.....	52
3.2.1.4.1.	SISTEMA DE FICHEROS AVANZADO .....	52
3.2.1.4.2.	SERVICIOS CRÍTICOS.....	52
3.2.1.4.3.	COMUNICACIÓN ENTRE PROCESOS .....	53
3.2.2.	CONFIGURACIÓN INICIAL MÓDULO KTWM102 .....	54
	INSTALACIÓN DE DRIVERS USB Y DEL BOOTLOADER.....	54
3.2.2.1.1.	WINDOWS .....	55
3.2.2.1.2.	LINUX / MAC OS.....	57
	INSTALL “DFU-UTIL” .....	58
	ACTUALIZACIÓN DE FIRMWARE .....	58
3.2.2.2.	RUNTIME – INSTALACIÓN DE DRIVERS USB.....	60
3.2.2.3.		
3.2.2.4.1.	WINDOWS .....	60
3.2.2.4.2.	LINUX .....	62
3.2.2.4.3.	MAC OS .....	63
3.2.2.5.	CONFIGURACIÓN DE TERMINAL COM .....	63
3.2.2.5.1.	WINDOWS .....	63
3.2.2.5.2.	LINUX / MAC OS.....	64
3.3.	CONFIGURACIÓN DE RED PARA KTWM102 .....	65
3.3.1.1.	KIRALE COMMAND-LINE SHELL– COMANDOS KSH .....	65
3.3.1.2.	SINTAXIS DE LOS COMANDOS.....	65
3.3.1.3.	SINTAXIS DE LOS PARÁMETROS.....	66
3.3.2.1.	MENSAJES DE RESPUESTA .....	67
3.3.2.2.	KIRALE BINARY INTERFACE – COMANDOS KBI.....	68
3.3.2.3.	OPERACIÓN DE INTERFAZ .....	68
3.3.2.4.	FORMATO DEL PAQUETE .....	69
3.3.2.5.	REPRESENTACIÓN DE DATOS.....	70
3.3.3.1.	COMANDOS Y RESPUESTAS.....	71
3.3.3.2.	NOTIFICACIONES .....	72
3.3.3.	CONFIGURACIÓN DE RED .....	72
	MODO OUT-OF-BAND COMMISSIONING DESACTIVADO .....	72
	MODO OUT-OF-BAND COMMISSIONING ACTIVADO.....	73
4.	DISEÑO E IMPLEMENTACIÓN HARDWARE .....	75
4.1.	COMPONENTES COMERCIALES UTILIZADOS.....	75
4.1.1.	ELEMENTOS HARDWARE UTILIZADOS .....	75
4.1.2.	ELEMENTOS SOFTWARE UTILIZADOS.....	78

---

4.2.	DISEÑO DE ESQUEMÁTICO PCB .....	79
4.2.1.	JERARQUÍA DEL CIRCUITO .....	79
4.2.2.	CIRCUITO DE ALIMENTACIÓN .....	80
4.2.3.	INTEGRACIÓN DEL MÓDULO KTWM102.....	81
4.2.4.	COOKIE CONNECTOR .....	82
4.2.5.	USB .....	83
4.3.	LAYOUT.....	84
4.3.1.	LAYOUT CAPA TOP .....	85
4.3.2.	LAYOUT CAPA BOTTOM.....	86
4.3.3.	RESULTADO FINAL DEL DISEÑO.....	87
5.	PRUEBAS EXPERIMENTALES .....	89
5.1.	PRIMERA INTERACCIÓN CON DONGLE USB .....	89
5.2.	RED DE DOS NODOS.....	90
5.2.1.	CREACIÓN DE LA RED.....	90
5.2.2.	PING ENTRE NODOS .....	91
5.2.3.	ENVÍO DE MENSAJES UDP ENTRE AMBOS NODOS.....	91
5.3.	PRUEBAS CON EL BORDER ROUTER.....	93
5.3.1.	INTRODUCCIÓN A LA CONFIGURACIÓN DEL ROUTER.....	93
5.3.2.	PRUEBA DE CONECTIVIDAD IP ENTRE RED THREAD Y LAN.....	94
5.3.3.	RED CON EL BR Y DOS NODOS KTDG102 .....	94
5.3.4.	ENVÍO DE MENSAJES UDP POR SOCKETS .....	97
5.4.	PRUEBAS CON PCB COOKIE THREAD COMO CUARTO NODO .....	99
5.4.1.	PRUEBAS DE CONECTIVIDAD CON EL CUARTO NODO .....	100
5.4.2.	ENVÍO / RECIBO DE SOCKETS .....	101
5.5.	PRUEBAS DE ESTABILIDAD CON 5 NODOS .....	102
5.5.1.	PRUEBAS DE CONECTIVIDAD .....	103
5.5.2.	ENVÍO / RECIBO DE SOCKETS .....	104
5.6.	PRUEBAS CON 6 NODOS Y CON ENVÍOS MULTISALTO.....	105
6.	CONCLUSIONES Y LÍNEAS FUTURAS .....	107
6.1.	CONCLUSIONES .....	107
6.2.	LÍNEAS FUTURAS.....	108
7.	BIBLIOGRAFÍA .....	109

## ABREVIATURAS Y ACRÓNIMOS

IoT	Internet of Things
WAN	Wide Area Network
LAN	Local Area Network
MAN	Metropolitan Area Network
PAN	Personal Area Network
LPWAN	Low Power Wide Area Network
WSN	Wireless Sensor Networks
M2M	Machine to Machine
LTE	Long Term Evolution
IPv6	Internet Protocol version 6
IPv4	Internet Protocol version 4
SOA	Service Oriented Architecture
QoS	Quality of Service
IFP	InterFace Profile
SPP	Service Provisioning Process
WSDL	Web Services Description Language
6LoWPAN	IPv6 over Low power Wide Personal Area Network
WPAN	Wide Personal Area Network
RPL	IPv6 Routing Protocol for Low Power and Lossy Networks
DAG	Direct Acyclical Graphic
DODAG	Destination Oriented DAG
AES	Advanced Encryption Standard
DTLS	Datagram Transport Layer Security
TLS	Transport Layer Security
UDP	User Datagram Protocol
CoAP	Constrained Application Protocol
MQTT-SN	Message Queue Telemetry Transport for Sensor Networks

BLIP	Berkeley Low-power IP stack
REED	Router Eligible End Device
FTD	Full Thread Device
MTD	Minimal Thread Device
ED	End Device
FED	Full End Device
MED	Minimal End Device
SED	Sleepy End Device
MeshCoP	Mesh Commissioning Protocol
BR	Border Router
JD	Joiner Device
JR	Joiner Router
JS	Joiner Session
ULA	Unique Local Address
GUA	Global Unique Address
EID	Endpoint Identifier
REST	Representational State Transfer
KEK	Key Encryption Key
IID	Interface Identifier (from the JD)
MLE	Establecimiento del Enlace MAC
DHCPv6	Protocolo de Configuración de Host Dinámico v6
ICMPv6	Protocolo para Control de Mensajes Internet

*Tabla 1 Abreviaturas y Acrónimos*

## ILUSTRACIONES

Ilustración 1 Diagrama de Gantt Parte 2020.....	2
Ilustración 2 Diagrama de Gantt Parte 2021.....	2
Ilustración 3 Estructura Básica 6LoWPAN [6].....	10
Ilustración 4 Arquitectura en Contiki [7].....	13
Ilustración 5 Ejemplo de una topología de Red Thread en malla [9].....	17
Ilustración 6 Pila de Protocolo Thread en un sistema IoT.....	18
Ilustración 7 Ejemplo PAN Thread [12], [13].....	19
Ilustración 8 Balena Etcher.....	32
Ilustración 9 Administrador de Dispositivos Previo a la instalación de Drivers del BR.....	33
Ilustración 10 Instalación Drivers de Border Router con Zadig.....	34
Ilustración 11 MobaXterm.....	34
Ilustración 12 Login Panel Administración Web.....	35
Ilustración 13 Pestaña Network.....	36
Ilustración 14 Pestaña DNS.....	37
Ilustración 15 Actualización KiBRA.....	38
Ilustración 16 Pestaña Settings.....	39
Ilustración 17 Parámetros a configurar con Out-of-band Commissioning Activado.....	40
Ilustración 18 Parámetros a configurar con Out-of-band Commissioning Desactivado.....	40
Ilustración 19 Configuración Prefijo de Red.....	41
Ilustración 20 Menú KiBRA - Inicio de Border Router.....	42
Ilustración 21 Menú de KiBRA - Border Router Iniciado.....	43
Ilustración 22 Pestaña Settings - Export Settings.....	44
Ilustración 23 Ventana Export Commissioning Information.....	45
Ilustración 24 Servidor Backbone Router.....	46
Ilustración 25 Servicio DHCP.....	47
Ilustración 26 Servicio NAT64.....	48
Ilustración 27 Servicio Commissioner.....	49
Ilustración 28 Pestaña Visual Network.....	50
Ilustración 29 Pestaña Logs.....	51
Ilustración 30 Administrador de dispositivos antes de instalar Drivers de KTWM102.....	54
Ilustración 31 Instalación Drivers con Zadig Paso 1.....	55
Ilustración 32 Instalación Drivers con Zadig Paso 2.....	56
Ilustración 33 Instalación Drivers con Zadig Finalizada.....	56
Ilustración 34 Administrador de Dispositivos Después de Instalar Drivers.....	57
Ilustración 35 Instalar libusbk con Zadig - Runtime.....	60
Ilustración 36 Instalación Driver Libusbk con Zadig Finalizada.....	61
Ilustración 37 Instalar USB SERIAL (CDC) con Zadig.....	61
Ilustración 38 Instalación Driver USB Serial (CDC) con Zadig Finalizada.....	61
Ilustración 39 Administrador de Dispositivos después de la instalación.....	62
Ilustración 40 Terminal Termite.....	64
Ilustración 41 Herramienta KiTools.....	66
Ilustración 42 Direcciones IP como parámetros.....	67
Ilustración 43 Esquema de comunicación entre Host Externo y el dispositivo KTWM102.....	68
Ilustración 44 Ejemplo Rutina de Comando - Esperar Respuesta.....	71
Ilustración 45 Kit de Desarrollo STM32F407G-DISC1.....	75
Ilustración 46 KTDG102 Evaluation Dongle.....	76
Ilustración 47 Módulo KTWM102.....	76
Ilustración 48 Border Router.....	77

Ilustración 49 Módulos de Procesamiento y de Alimentación de la Cookie.....	77
Ilustración 50 Jerarquía Circuito .....	79
Ilustración 51 Circuito de Alimentación.....	80
Ilustración 52 Circuito Integración del Módulo .....	81
Ilustración 53 Conectores Verticales .....	82
Ilustración 54 Conector USB .....	83
Ilustración 55 Layout Completo .....	84
Ilustración 56 Layout Capa TOP.....	85
Ilustración 57 Layout Capa BOTTOM.....	86
Ilustración 58 PCB Cookie Thread TOP .....	87
Ilustración 59 PCB Cookie Thread Capa BOTTOM.....	87
Ilustración 60 Diagrama de conexión PC - Dongle - uC.....	89
Ilustración 61 Esquema montaje Red de Dos Nodos.....	90
Ilustración 62 Logs KiTools al REALIZAR un Ping.....	91
Ilustración 63 Logs KiTools al RECIBIR un Ping.....	91
Ilustración 64 Montaje Border Router .....	93
Ilustración 65 Topología de Red 1 nodo con BR .....	94
Ilustración 66 Topología 1 de BR con 2 nodos Dongle .....	95
Ilustración 67 Topología 2 de BR con dos nodos Dongle .....	95
Ilustración 68 Ping desde PC a BR.....	96
Ilustración 69 Ping desde PC a nodo LEADER .....	96
Ilustración 70 Ping desde PC a nodo MED.....	96
Ilustración 71 Topología para envío de mensajes UDP vía Sockets.....	97
Ilustración 72 Topología 4 nodos con un Dongle como LEADER .....	99
Ilustración 73 Topología 4 nodos con BR como Leader.....	100
Ilustración 74 Ping desde PC a nuevo nodo MED. ....	100
Ilustración 75 Topología A de 5 nodos .....	102
Ilustración 76 Topología B con 5 nodos.....	103
Ilustración 77 Topología con 6 nodos .....	105



## TABLAS

Tabla 1 Abreviaturas y Acrónimos .....	XII
Tabla 2 Diferencias M2M – IoT [2] .....	3
Tabla 3 MAC Layer Frame .....	22
Tabla 4 Servicios KBRNT1 .....	52
Tabla 5 Comandos Servicio KiBRA .....	53
Tabla 6 Servicio Ajenti .....	53
Tabla 7 Listar dispositivos con dfu-util .....	58
Tabla 8 Actualizar Firmware en dispositivos KTWM102 .....	59
Tabla 9 Comando para listar Puertos Serie en Sistemas Linux .....	63
Tabla 10 Comando para listar Puertos Serie en Sistemas en MAC OS .....	63
Tabla 11 Abrir terminal Picocom en Linux / MAC Os .....	64
Tabla 12 Sintaxis comandos KSH .....	65
Tabla 13 Formato del Paquete .....	69
Tabla 14 Bits Byte Type .....	69
Tabla 15 Significado Bits Byte Type .....	70



# 1. INTRODUCCIÓN Y OBJETIVOS DEL TRABAJO

## 1.1. MOTIVACIÓN DEL PROYECTO

Este proyecto surge debido a la aparición de las nuevas tecnologías de comunicaciones y su implementación dentro de la tecnología IoT. Este proyecto incluye la implementación en la plataforma Cookie para WSN de la tecnología Thread.

La motivación principal de este proyecto es la implementación hardware de la tecnología THREAD en la plataforma Cookies, dotando a dicha plataforma de una nueva capa Hardware de Comunicaciones, analizando también la viabilidad de esta implementación. Para esto, se comenzará con una primera analítica y/o exploración de la propia tecnología, analizando sus ventajas e inconvenientes frente a otras tecnologías existentes hoy en día, al igual que un primer análisis del protocolo mediante un kit de evaluación que integra la tecnología THREAD y un microcontrolador. Una vez hecha la primera toma de contacto con la tecnología THREAD, se realizará un diseño Hardware para su implementación en la plataforma Cookies. Una vez verificado el Hardware diseñado, se pasará al análisis más en profundidad del protocolo THREAD mediante diferentes pruebas de conectividad, envío de datos y estabilidad.

## 1.2. OBJETIVOS DEL PROYECTO

Los objetivos más en específico de este proyecto han sido:

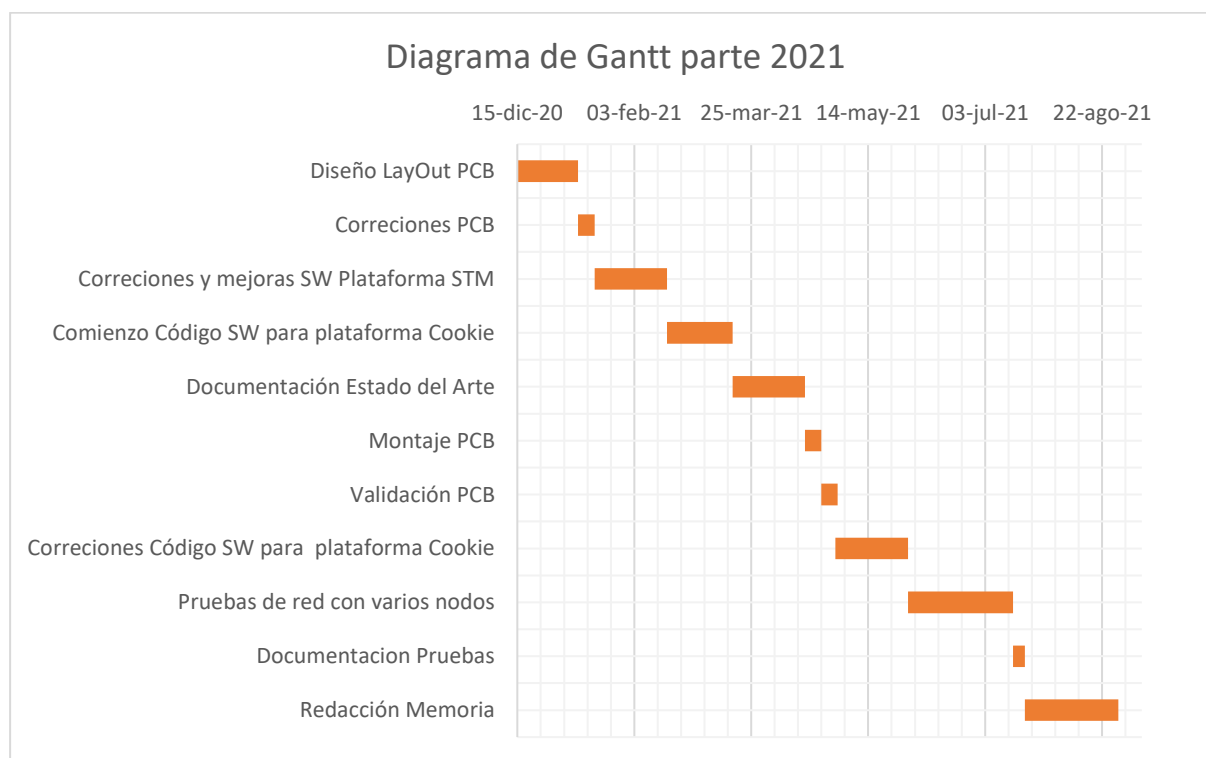
- Manejo de los Evaluation Dongles USB (dispositivos KTDG102) de KIRALE, como primera interacción con el protocolo de comandos utilizado, tanto comandos KSH, a través de la herramienta KiTools, como comandos KBI, a través de puerto UART con un microcontrolador.
- Creación de una red con los módulos KTDG102 disponibles por comandos KBI, probando las diferentes configuraciones posibles, creando el procedimiento de configuración para crear o unirse a una red y el posterior envío de mensajes UDP entre ambos nodos a través de Sockets.
- Diseño de PCB para integrar el dispositivo KTWM102 a la plataforma de la Cookie.
- Configuración y conectividad del nodo Border Router a la red externa.
- Creación de una red THREAD integrando los nodos Dongle USB junto con el BR.
- Conectividad entre un nodo de la red THREAD y un punto de la red externa, como un PC, a través del Border Router.
- Envío de mensajes UDP vía Sockets entre un PC externo y los nodos dentro de la red.
- Validación de la PCB diseñada e integración en la red creada anteriormente.

### 1.3. GESTIÓN DEL PROYECTO

La gestión del proyecto se muestra en dos Diagramas de Gantt, mostrados en Ilustración 1 y en Ilustración 2



*Ilustración 1 Diagrama de Gantt Parte 2020*



*Ilustración 2 Diagrama de Gantt Parte 2021*

## 2. ESTADO DEL ARTE

En los últimos años, se han ido desarrollando cada vez más las tecnologías para Internet de las cosas, o IoT. Entre estas tecnologías están las tecnologías de comunicaciones inalámbricas tanto a redes de áreas pequeñas como de áreas extensas. Estas últimas, conocidas como WAN (wide area network), son usadas como base para la gran mayoría de arquitecturas en proyectos IoT.

Este estudio se centrará en las características de una de ellas: Thread. Para esto conviene conocer mínimamente las tecnologías IoT, y en concreto, el modelo de la tipología red WAN sobre la que mejor se aplica IoT en casos de conexiones inalámbricas: LPWAN.

### 2.1. INTERNET OF THINGS (IoT)

En IoT, un gran número de tecnologías inalámbricas, como el WiFi, el Bluetooth, LoRa, NB-IoT, 2G/3G/4G, etc., han sido usadas en diversas aplicaciones, conectando entre si a millones de dispositivos de manera inalámbrica. 3G y 4G son muy usados en IoT, pero no están totalmente optimizados para aplicaciones de este tipo. [1]

#### 2.1.1. INTRODUCCIÓN [2]

IoT es un término bastante amplio, que incluye muchas tecnologías Wireless Sensor Network, Cloud Computing, Big Data Analytics, Sistemas embebidos, Protocolos de Comunicación, etc.

A la par que IoT, está creciendo otra nueva tecnología, M2M, cuya comunicación se basa en redes de dispositivos intercambiando o enviando datos sin interacción humana. Las diferencias entre M2M e IoT las indicamos a continuación en Tabla 2:

M2M	IoT
Centrada en la capa por debajo de la capa de red.	Centrado en la capa por encima de la capa de red.
Sensado y actuación pueden no estar involucrados.	Sensado y actuación están involucrados.
Énfasis en hardware.	Énfasis en software.
La nube no está involucrada.	La nube está involucrada.

*Tabla 2 Diferencias M2M – IoT [2]*

A nivel de hardware hay ciertos requisitos que llevan tanto al desarrollo de una infraestructura IoT como el de redes de comunicación:

- 1) Fuente de alimentación y Manejo de la alimentación.
- 2) Sensores o Actuadores.
- 3) Procesador y Espacio de memoria.
- 4) Comunicaciones Inalámbricas.
- 5) UI/UX.

### 2.1.2. COMPONENTES [2]

Los principales componentes, según su función, de las tecnologías IoT son:

- Componente para la interacción y comunicación con otros IoT.
- Componente para operaciones de procesamiento y análisis de los datos.
- Componente para la interacción con internet.
- Componente para el manejo de servicios Web de la aplicación.
- Componentes para la integración de los servicios de la aplicación.
- Interfaz de usuario para acceder al IoT.

#### *DISPOSITIVO*

##### 2.1.2.1.

Cada dispositivo IoT es indistinguible por la identidad que recibe dentro de la red. Estos dispositivos cuentan con sistemas HW y SW y se conectan a una red donde interactúan con otros dispositivos. Estos dispositivos realizan tareas como sensorizado, actuación o monitorización de manera remota. Los datos son intercambiados con otros dispositivos y aplicaciones, incluso enviando esos datos a servidores centralizados o aplicaciones basadas en procesamiento de datos en la nube.

En primer lugar, están los dispositivos pequeños, con características limitadas por lo que necesitan de otro dispositivo, con mayores prestaciones, que realice de Gateway para poder conectarse a servidores externos a la red.

Por otro lado, los dispositivos estándar son similares a pequeños ordenadores y dirigen los datos directamente a la nube de la red sin necesidad de un Gateway.

#### *RED LOCAL*

Las redes IoT se utilizan para transmitir las señales recogidas en los sensores con el resto de diferentes componentes, como los routers, puentes, etc, por lo que se genera una gran cantidad de datos. Para la elección de la tecnología de conectividad dentro de las redes, se deben seguir los siguientes criterios:

- Alta tasa de datos.
- Bajos precios en uso de datos.
- Despliegue IPv6

## INTERNET

El Internet es el sistema de redes de ordenadores interconectados globalmente usando el protocolo de internet para unir los diferentes dispositivos.

### 2.1.2.3.

## SERVICIOS BACKEND

Los Servicios Backend se definen como un conjunto de servicios basados en la nube, los cuales ayudan a construir la aplicación IoT o aplicación Backend. Tienen una alta tasa de almacenamiento de datos y una fácil gestión de usuarios, proporcionando así una solución muy viable a empresas y/o usuarios dedicados a IoT. Una vez los datos son recolectados, se envían a la nube, donde se guardarán en servidores que permitirán al usuario la gestión y/o computación de dichos datos.

Los principales servicios Backend principales existentes son:

- Monitorización de dispositivos.
- Servicios de control de dispositivos.
- Servicios de publicación de datos.
- Servicios de búsqueda de datos.

### 2.1.2.5.

## APLICACIONES

Las principales aplicaciones para IoT son aplicaciones como Smart Home, Coches conectados, Smart Cities, IoT en Agricultura o accesorios, como pueden ser los Smart Watch o las Smart Scales (Básculas Inteligentes).

## 2.1.3. SENSORES

Los sensores son el componente más importante en IoT. Es el dispositivo que se opera con baja potencia, menos energía y recursos limitados de almacenamiento. Hay dos tipos de clases de sensores, según nos fijemos en **el tipo de salida** o **el tipo de dato**.

Según el **tipo de salida** tenemos las siguientes clases de sensores:

- 1) **Sensores Analógicos:** Estos sensores generan como salida una señal continua. Captan cantidades analógicas y que son continuas en la naturaleza, como pueden ser los acelerómetros, barómetros, sensores de sonido, temperatura, etc.
- 2) **Sensores Digitales:** Producen una salida binaria, un “1” lógico o un “0” lógico, son valores discretos que pueden ser un solo “bit” (transmisión serie) o un conjunto de bits formando un único “byte” de salida (transmisión paralela). Por esto, se usan por lo general para medidas analíticas.

Por otro lado, según el **tipo de dato** tenemos las siguientes clases de sensores:

- 1) **Sensores escalares:** La señal de salida generada es proporcional a la magnitud que se está midiendo. Medidas físicas como temperatura, presión, etc., son magnitudes escalares y su valor es suficiente información. Dichas medidas también variarán respondiendo proporcionalmente a los cambios en la medida realizada.
- 2) **Sensores vectoriales:** La señal de salida generada es proporcional tanto a la magnitud medida como la orientación de lo que se mide. Ejemplos de esto pueden ser imágenes, sonido, velocidad, aceleración, orientación, etc., medidas de las cuales solo el valor o magnitud no da una información completa. Por ejemplo, un cuerpo con una aceleración, puede tener una aceleración en sus tres ejes, por lo que una información completa sería la aceleración de los 3 ejes, no solo la magnitud del vector resultante de los 3 ejes.

### 2.1.4. ARQUITECTURA [3]

La arquitectura de las tecnologías IoT es una arquitectura orientada al servicio (SOA por sus siglas en inglés). A continuación, se explicará una arquitectura SOA de cuatro capas, la cual se ha pensado desde el punto de vista de las funcionalidades.

Este diseño se compone de las capas de Sensorizado, Red, Servicio e Interfaz.

#### 2.1.4.1. *CAPA DE SENSORIZADO*

Cada vez más dispositivos cuentan con equipamiento de RFID o sensores inteligentes, por lo que la conectividad cada vez es más sencilla. En esta capa, los sistemas inalámbricos inteligentes con sensores, toman las medidas e intercambian la información con diferentes dispositivos. Esto ayuda a identificar cambios en el ambiente.

#### 2.1.4.2.

#### *CAPA DE RED*

La función de esta capa es la de interconectar todo entre sí y permitir el envío de información con otros dispositivos. Además, es capaz de añadir información sobre las infraestructuras IT existentes. En la arquitectura SOA de IoT, los servicios, aportados por los dispositivos, son desplegados en una red heterogénea y se provee de servicio de Internet a todos los dispositivos.

Este proceso puede involucrar el manejo y control de QoS (Quality of Service) para así cumplir con los requisitos de los usuarios y/o aplicaciones. Por otro lado, en las redes dinámicas, es importante la automatización de la búsqueda y mapeamiento de dispositivos.

Estos dispositivos necesitan que se les asigne un rol automáticamente y ser capaces de cambiar de rol en cualquier momento según se necesite. Esto permite a los dispositivos colaborar en la realización de las distintas tareas, según el rol asignado en cada momento.



Para diseñar esta capa, los diseñadores necesitan:

- Tecnologías de gestión de redes para redes heterogéneas.
- Eficiencia energética.
- Requerimientos QoS.
- Procesamiento de datos y señales.
- Seguridad.
- Privacidad.

### *CAPA DE SERVICIO*

La capa de Servicio se apoya en la tecnología del Middleware y aporta las funcionalidades para integrar de manera correcta servicios y aplicaciones en IoT. Una capa de servicio bien diseñada será capaz de identificar los requisitos comunes de aplicación y dar APIs y protocolos para soportar los servicios, aplicaciones y necesidades de usuario que se requieran. Además, se procesan todos los problemas orientados al servicio, incluyendo el intercambio y almacenamiento de información, gestión de datos, comunicación y motor de búsqueda. Las diferentes partes que componen esta capa son:

- **Servicio de Descubrimiento:** Encontrar objetos que puedan aportar los servicios requeridos y la información deseada de manera eficiente.
- **Servicio de Composición:** Habilita la interacción entre los dispositivos conectados. Esta fase es para hacer el scheduling, u organización, o recrear servicios más ajustados de cara a conseguir la manera más fiable de lograr los requerimientos.
- **Gestión de confianza:** Buscando un mecanismo de confianza que pueda evaluar y usar la información aportada por los otros servicios para crear un sistema de confianza.
- **Servicio APIs:** Apoyando la interacción entre los servicios requeridos en IoT.

### *CAPA DE INTERFAZ*

En las redes IoT los dispositivos conectados pueden ser de diferentes proveedores y no siempre se usan los mismos estándares o protocolos, por lo que suele haber problemas en las comunicaciones e intercambios de información entre los dispositivos. Va surgiendo la necesidad de una capa de Interfaz que simplifique la gestión y la interconexión de los distintos dispositivos, ya que el problema se agrava al crecer de manera rápida la cantidad de dispositivos. Un perfil de interfaz (InterFace Profile, IFP) puede ser visto con un subconjunto de los estándares de servicios que ayudan a la interacción con aplicaciones dentro de la red. Los perfiles de interfaz son usados para describir las especificaciones entre las aplicaciones y los servicios. Los servicios, en su respectiva capa, corren en una limitada red de infraestructuras para encontrar eficientemente nuevos servicios para una aplicación y conectarlos a la red.

Tras los nuevos resultados de investigaciones sobre SOA-IoT, se ha visto que el Proceso de provisionamiento de servicios (SPP) puede también dar una interacción efectiva entre aplicaciones y servicios. El SPP empieza haciendo una “consulta de tipos” que envía una petición a los servicios con un formato WSDL genérico y usan un mecanismo de búsqueda para encontrar servicios potenciales. Todas las instancias de servicios se clasifican y un mecanismo de provisionamiento identificará los servicios que encajen con los requisitos de la aplicación, basándose en el contexto de la aplicación y de la aplicación QoS. Finalmente se evalúa el proceso.

## 2.2. 6LoWPAN [4]

Muchas de las soluciones patentadas, como ZigBee o Thread, están vinculadas a una capa de enlace y los perfiles de aplicación solamente resuelven una pequeña parte de las múltiples aplicaciones para redes inalámbricas integradas. Estas también se encuentran con problemas a la hora de evolucionarlas, integración a Internet y de escalabilidad.

**6LoWPAN** consigue vencer estos problemas gracias al uso de IPv6, en dispositivos integrados de bajo consumo y procesamiento limitado, sobre redes inalámbricas de bajo ancho de banda.

### 2.2.1. SIGNIFICADO 6LoWPAN [4]

6LoWPAN es la abreviación de IPv6 over Low power Wireless Personal Area Networks. Desglosando por partes:

#### 1) 6 en 6LoWPAN

El **6** se debe a que este sistema está basado en IPv6. IPv6 es el nuevo Protocolo de Internet, y será el que termine reemplazando a IPv4, debido a que este se está llegando al límite de su rango de direcciones IP posibles.

Mientras que **IPv4** ofrece  $2^{32}$  direcciones (4,294,967,296 direcciones posibles IP en Internet), **IPv6** ofrece  $2^{128}$  direcciones posibles ( $3.4 \times 10^{38}$  direcciones).

#### 2) Lo en 6LoWPAN

**Lo** se debe a Low Power o baja potencia. Usualmente las comunicaciones IP van contrarias a un bajo consumo de potencia. Pero se ha logrado gracias a los sensores inalámbricos y un gran desarrollo, llegar a reducir bastante el consumo.

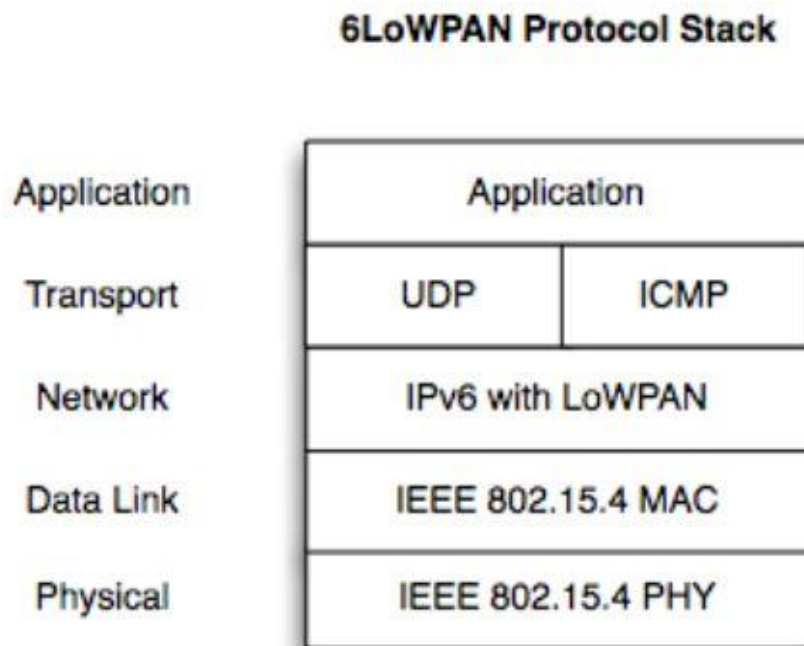
#### 3) WPAN en 6LoWPAN

**WPAN** corresponde a Wireless Personal Area Networks. Una WPAN es la red personal en área que conecta los dispositivos dentro de un rango de distancia muy limitada. Un ejemplo de este tipo de redes son las redes Bluetooth, usadas para conectar, por ejemplo, los smartphones con cascos o auriculares Bluetooth, o incluso con un sistema de manos libres.

En 6LoWPAN se pueden crear redes en mallas, incluyendo una mayor distancia. Debido al uso de 868/915 MHz en vez de 2400 MHz, la cobertura dentro de edificios es mucho mayor.

### 2.2.2. PILA DE PROTOCOLOS DE 6LoWPAN [4]–[6]

El concepto básico de la pila del 6LoWPAN, se muestra a continuación en Ilustración 3:



*Ilustración 3 Estructura Básica 6LoWPAN [6]*

LoWPAN es una capa de adaptación. Aquí el sistema operativo es una parte clave del desarrollo en 6LoWPAN, en términos de la huella en memoria y en funcionalidad.

Gracias a desarrolladores de la suite con el protocolo IPv6 basada en el estándar IEEE 802.15.4, se consigue crear una red 6LoWPAN autoorganizada con protocolo de enrutamiento.

La tecnología de las capas bajas de 6LoWPAN se apoya en el estándar IEEE 802.15.4 de las capas física (PHY) y MAC. Uno de los problemas se encuentra en el tamaño de la Payload de la capa MAC en IPv6, puesto que es mayor de lo que la capa baja de 6LoWPAN puede soportar. Para una buena conectividad entre las capas MAC y de red se recomienda añadir una capa de adaptación (LoWPAN) entre medias de las dos capas. Esta capa de adaptación se encargaría de la compresión de la cabecera, la fragmentación, el reensamblaje y el reenvío de la ruta de la red.

### 2.2.3. PRINCIPALES CARACTERÍSTICAS DE 6LoWPAN [7]

Como se ha indicado tanto en 2.2.1 como en Ilustración 3, 6LoWPAN es una capa de adaptación IoT IPv6, la cual está sobre el estándar IEEE 802.15.4.

El estándar IEEE 802.15.4 es un estándar de WPAN de baja potencia y baja velocidad que usa CSMA/CA y con una configuración típica con un rango de 10 a 100 m y una tasa de rango de datos sin procesar de 2 a 250 KBits/s, estando en la banda de 2.4 GHz ISM. Por último, IEEE 802.15.4 también permite trabajar en la banda de 900MhHz (banda sub-G), aportando hasta unos pocos kilómetros de rango, pero bajando la tasa de datos.

#### COMPRESIÓN DE CABECERA

Para una transmisión eficiente de paquetes IPv6 a través de los enlaces IEEE 802.15.4, se <sup>2.2.3.1.</sup> necesitará una capa de adaptación, la cual comprimirá la cabecera de 40 a 7 bytes. Se comprimirán tanto cabeceras IPv6 como cabeceras UDP de la misma manera.

#### ENRUTAMIENTO

##### 2.2.3.2.

Debido a las limitaciones de las redes 6LoWPAN, la mayoría de los protocolos de enrutamiento en IPv6 no son compatibles, por lo que se ha desarrollado un nuevo protocolo de enrutamiento para 6LoWPAN: **IPv6 Routing Protocol for Low Power and Lossy Networks (RPL)**.

Debido a que las redes con pérdidas (Lossy Networks) no tienen topologías predefinidas, RPL organiza una topología como un Gráfico Acíclico Directo (DAG) particionado en uno o más DAGs Orientados a Destino (DODAGs).

También, RPL está pensado para ser usado en redes con bastantes nodos, los cuales tendrán recursos limitados y las redes estarán gestionadas por uno o unos pocos “supernodos” o Border Routers en el caso de 6LoWPAN.

##### 2.2.3.3.

#### SEGURIDAD

Al ser IEEE 802.15.4, un protocolo de capas de enlace y física inalámbricas, convendrá disponer de un protocolo de seguridad en la capa de enlace. IEEE 802.15.4 usa el Estándar de Encriptación Avanzado (AES por sus siglas en inglés) en el Contador con modo CBC-MAC. El problema está en que esto solo cubre el segmento del enlace que lo usa, pero no tiene seguridad end-to-end. Por lo que, para asegurar dicha seguridad end-to-end, como en Internet, es necesario añadir medidas en las capas altas como en las capas de red y de aplicación.

Para la capa de red, se ha propuesto el protocolo IPSec adaptado para soportar 6LoWPAN y prometiendo seguridad end-to-end.

En la capa de aplicación, el protocolo principal como candidato es el DTLS o *Datagram Transport Layer Security*. Al contrario de TLS, que trabaja en TCP, DTLS trabaja en UDP, protocolo adecuado para redes limitadas como es el caso de 6LoWPAN.

### PROTOCOLOS DE APLICACIÓN

Actualmente los dos protocolos de aplicación, para 6LoWPAN, más populares son Constrained Application Protocol (CoAP) y Message Queuing Telemetry Transport-Sensor Network (MQTT-SN).

Debido a la popularidad de los servicios web, se desarrolló CoAP como un HTTP ligero para 6LoWPAN. Al ser dispositivos limitados, se usa UDP con un sistema propio de retransmisión basado en mensajes. Se usa UDP en vez de TCP debido al menos uso de recursos. CoAP tiene arquitectura REST, debido a que requiere muchos menos recursos y permite a los desarrolladores Web programar para IoT. Entre CoAP y HTTP es fácil traducir mediante dispositivos proxy. CoAP también permite trabajar con medidas de seguridad como DTLS.

MQTT-SN, usa igualmente UDP, pero al contrario que CoAP, MQTT es un protocolo ligero de publicador-suscriptor. Este sistema desengancha productores y consumidores, por lo que es necesario un intermediario.

#### 2.2.4. RETOS DE 6LoWPAN

Debido a las limitaciones de recursos que tienen los nodos y a la implementación de IPv6, hay varios retos a la hora del diseño e implementación de 6LoWPAN. Algunos de los principales retos son:

- **Sobrecarga de cabecera:** La capa de enlace tiene una MTU de 127 bytes, mientras que la MTU de IPv6 es de al menos 1280 bytes con una cabecera de 40 bytes. Esto implica que transmitir directamente los paquetes IPv6 es ineficiente por lo que se generan frecuentes fragmentaciones y desfragmentaciones.
- **Neighbour Discovery Protocol (NDP):** IPv6 usa este protocolo NDP, encargado de reconocer a los diferentes dispositivos cercanos en la red, para configurarse solo combinando la información del prefijo de la red con el mensaje de anuncio del router y el ID del host de la dirección de su capa de enlace, formando así una dirección de 128 bytes. NDP ayuda, en gran medida, en 6LoWPAN debido a que simplifica enormemente la asignación de direcciones IP a muchos dispositivos. A pesar de esto, el protocolo está aún en proceso de adaptación para redes limitadas.
- **Redes con pérdidas:** Debido a la movilidad, las interferencias, etc, estas limitadas redes, con enlaces inalámbricos, no son muy fiables. Esto implica un reto en el enrutamiento debido a que se requiere una relativamente estable y lentamente variable topología de red.
- **Seguridad:** En redes estándar TCP/IP, se usa seguridad end-to-end, como es el caso de IPSec en la capa de red o en Transport Layer Security (TLS) en la capa de aplicación.

### 2.2.5. IMPLEMENTACIONES Y APLICACIONES PARA 6LoWPAN [5], [7]

Actualmente, hay numerosas implementaciones de código abierto para 6LoWPAN. Las más populares son: Contiki, TinyOs, Linux y Thread (en especial el software OpenThread desarrollado por Google).

#### CONTIKI

Contiki es un sistema operativo híbrido basado en Unix. Es de código abierto, muy portable y con capacidad de multitarea y control de eventos. Ha sido diseñado para una eficiencia en memoria en sistemas de redes embebidos y redes de sensores inalámbricos. Puede estar en nodos con capacidades de memoria tan baja como 20 KB en RAM y 100 KB en ROM. También permite comunicaciones IP, pudiendo tanto IPv4 e IPv6.

Gracias a la multitarea y gestión de eventos, se soporta una carga dinámica y reemplazamiento de servicios o programas individualmente. La arquitectura utilizada en Contiki es la mostrada a continuación en Ilustración 4:

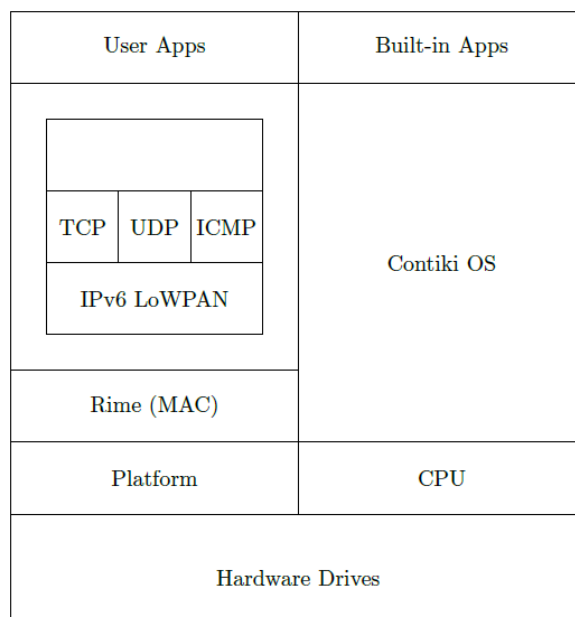


Ilustración 4 Arquitectura en Contiki [7]

### TINYOS

TinyOS es otro sistema basado en Unix, como es Contiki, pensado para dispositivos limitados y de baja potencia. TinyOs 2.X soporta 6LoWPAN gracias a BLIP (*Berkeley Low-power IP stack*). BLIP es la pila de 6LoWPAN más avanzada en cuanto a funcionalidades. Implementa las características básicas definidas en RFC4844, como compresión de la cabecera, fragmentación y direccionamiento, incluyendo también ICMPv6 y paquetes UDP. BLIP implementa también *Descubrimiento de vecinos o NDP*, en una versión reducida, configurando una dirección local de enlace en el arranque de los nodos y una dirección global si se recibe el anuncio del enrutador. En comparación con versiones anteriores, BLIP soporta redes de malla.

### THREAD

Este protocolo es un estándar abierto para las comunicaciones inalámbricas dispositivo a dispositivo desarrollado por empresas como Samsung, Google y otras empresas. Este protocolo se basa en 6LoWPAN y en IPv6 [8].

Al ser el protocolo elegido para el desarrollo de este proyecto, se explicará y analizará más en profundidad en 2.3 THREAD



## 2.3. THREAD

Thread es una especificación IoT gratuita, con redes en malla, basada en 6LoWPAN y en IPv6 [8]. Esta ha sido desarrollada por empresas como Samsung, Google y muchas otras grandes compañías. Como se indica en la especificación THREAD [9], este protocolo es un estándar abierto para comunicaciones inalámbricas dispositivo a dispositivo, económico, de baja potencia y fiable. [7]

### 2.3.1. INTRODUCCIÓN A REDES THREAD [8], [10]–[12]

Thread está específicamente diseñado para ambientes donde es necesario o deseado una red basada en comunicaciones IP y, además, se permite una variedad de capas de aplicación a ser usadas.

Las características generales del protocolo y las redes Thread son [10]:

- **Simplicidad.** La instalación de la red, la puesta en marcha y la operativa son muy simples. Esto se debe a que los protocolos para formar, unir y mantener una red Thread permiten una autoconfiguración y arreglar problemas de enrutado en tiempo real.
- **Seguridad.** No se permite unirse a dispositivos a una red Thread si no están autorizados. Todas las comunicaciones están encriptadas y seguras.
- **Redes pequeñas y grandes.** La cantidad de nodos conectados puede variar entre unos pocos y cientos de dispositivos comunicándose sin problemas. Esto se debe a que la capa de red está diseñada para optimizar el uso de la red en función del uso esperado.
- **Rango.** Los dispositivos comunes dispuestos en una red en malla pueden dar rango suficiente para cubrir una casa. La tecnología de propagación del espectro, se usa en la capa física para tener una buena inmunidad a interferencias.
- **Sin fallos.** La pila del protocolo está diseñada para operar segura y fiablemente incluso con fallos o pérdidas en dispositivos de manera individual.
- **Baja potencia.** Los dispositivos suelen operar durante años con baterías tipo AA gracias al uso de un ciclo duty ajustado.

### 2.3.2. TIPOS DE DISPOSITIVOS

En estas redes, hay 2 clasificaciones de tipos de nodos que puede haber en la propia red según veamos por topología o por funcionalidad.

Para una topología básica tendríamos los siguientes tipos de dispositivos: [8], [10]

- **Routers.** Estos se encargan de dar servicios de enrutamiento a los distintos dispositivos de red y de proporcionar los servicios de seguridad. Estos routers, no están pensados para un modo de bajo consumo. A su vez, pueden degradar su funcionalidad y convertirse en REEDs (Router-Eligible End Devices).
- **Border Routers.** Aparte de tener las funcionalidades de un router, da conectividad a las redes 802.15.4 a redes adyacentes en otras capas físicas, como Wi-Fi o Ethernet, pudiendo dotar de conectividad a Internet a la red Thread creada. Da servicio a los dispositivos dentro de la red, incluyendo servicio de enrutamiento para operaciones fuera de red.
- **Router-Eligible End Devices.** Estos dispositivos pueden convertirse en Routers. Estos dispositivos, por lo general, no reenvían mensajes ni dan servicios de unión y seguridad para otros dispositivos dentro de la red Thread mientras no se conviertan a routers. Este cambio a Routers es manejado por la propia red automáticamente sin necesidad de ser manejado por ningún usuario.
- **Sleepy End Devices.** Son los dispositivos más sencillos, ya que solo se pueden comunicar a través de su nodo o router Padre, y no pueden reenviar mensajes para otros dispositivos. Pueden operar con muy bajo consumo debido a que receptor está desactivado y solo se despierta periódicamente para comunicarse con su padre.

Desde el punto de vista de funcionalidad, se podrían agrupar los dispositivos Thread en dos diferentes tipos, y dentro de cada tipo, cada dispositivo puede adoptar ciertos roles. Estos dos principales tipos son:

- **Full Thread Device (FTD).** Puede comunicarse con otros dispositivos bidireccionalmente y con los nodos hijos (MTD) unidos a él. Estos dispositivos pueden adquirir los siguientes roles:
  - **Leader:** Es el dispositivo Thread responsable del manejo de la asignación de ID de router. Es el único dispositivo en una partición de red Thread que actúa como árbitro central y distribuidor del estado de configuración de la red.
  - **Border Router.** Misma funcionalidad descrita en **Border Routers** en la clasificación anterior.
  - **Active Router.** Misma funcionalidad descrita en **Routers** en la clasificación anterior.

- **REEDs**. Misma funcionalidad descrita en **Router-Eligible End Devices** en la clasificación. Pueden tener varios nodos hijos y mantener enlaces con los Routers vecinos.
- **Full End Device (FED)**. Es un FTD normal, que siempre actuará como un Dispositivo Final (End Device – ED). Un dispositivo FED nunca pedirá convertirse en Router, como haría un REED.
- **Minimal Thread Device (MTD)**. Estos dispositivos solo pueden comunicar con el padre FTD al que están unidos. Los roles que pueden adquirir son:
  - **Minimal End Device (MED)**. Es un dispositivo MTD cuyo receptor está habilitado todo el tiempo y puede comunicarse con su nodo padre en cualquier momento.
  - **Sleepy End Device**. Funcionalidad descrita en **Sleepy End Devices**.

Un ejemplo de topología básica es la mostrada en Ilustración 5.

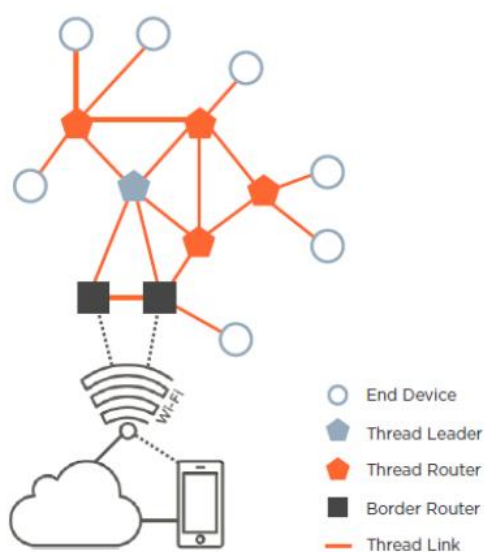


Ilustración 5 Ejemplo de una topología de Red Thread en malla [9]

Un Router o un Border Router podrá asumir el rol de líder para ciertas funciones dentro de la red Thread. Este líder deberá tomar decisiones dentro de la red, como asignar las direcciones de los Routers y permitir nuevas solicitudes de Routers. Si el nodo Líder fallara, un nodo Router o Border Router asumirá ese rol de manera automática, asegurando así ningún punto singular de fallo.

### 2.3.3. PROTOCOLO THREAD

En este apartado se describirá la pila del protocolo Thread, detallando las diferentes capas que la integran, las cuales se muestran en la Ilustración 6:

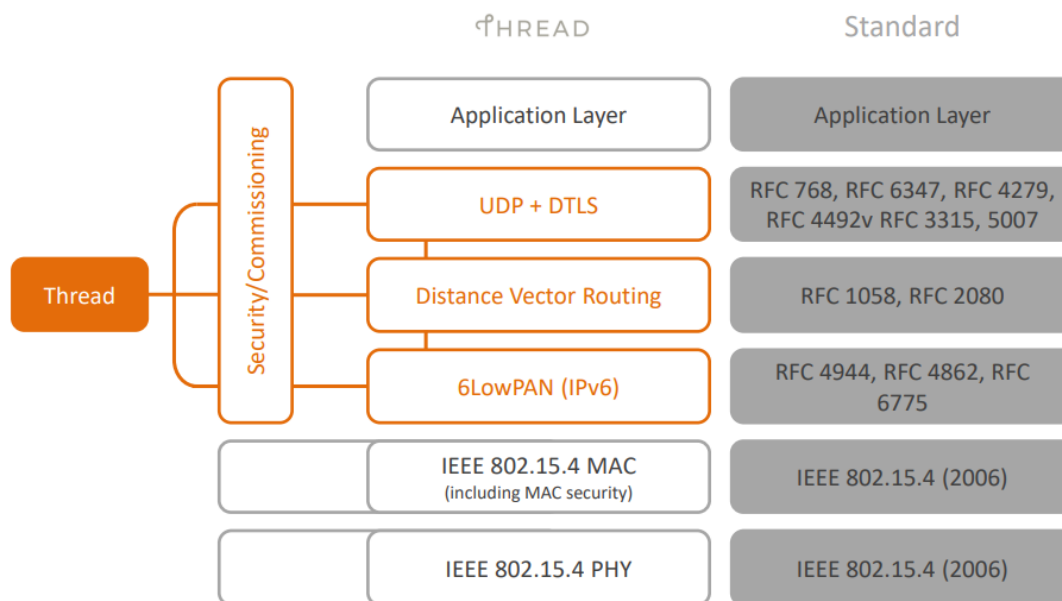


Ilustración 6 Pila de Protocolo Thread en un sistema IoT

#### 2.3.3.1.

### REDES DE ÁREA PRIVADA (PAN) [12]

Las redes de área privada (PAN) son redes de área local, las cuales incluyen hardware de comunicaciones y de computación, junto con sensores y actuadores. Estas redes se conectan al resto de Internet gracias a los Gateways, también conocidos como Border Routers.

Antes de que un dispositivo pueda unirse a una PAN, este deberá estar previsto dentro del servicio o ser “oficial” dentro de la red. En caso del protocolo Thread, para este proceso de puesta en servicio o *commissioning*, se usa *MeshCoP* (Mesh Commissioning Protocol). Este protocolo MeshCoP permite a una red de malla una segura autenticación, una puesta en servicio y la unión de nuevos dispositivos desconocidos. Más adelante se entrará más en detalle de este protocolo.

Un ejemplo de una PAN Thread se muestra a continuación en Ilustración 7:

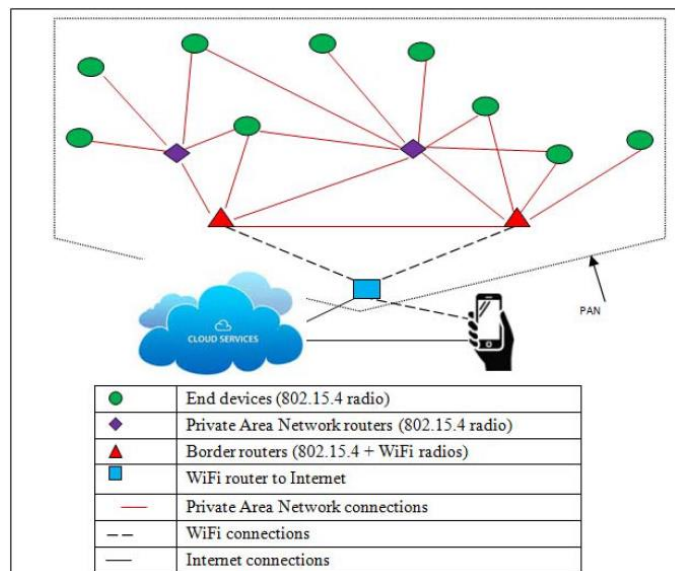


Ilustración 7 Ejemplo PAN Thread [12], [13]

Las redes PAN que usen el protocolo Thread tendrán las siguientes características:

- **Nodos.** Cada dispositivo o nodo tendrá unas determinadas características, unos darán conectividad a Internet a la PAN, otros solo podrán transmitir a su vecino más cercano, otros podrán mandar datos de un nodo a otro. Los distintos tipos de dispositivos o nodos posibles son los vistos la sección 2.3.2. Tipos de dispositivos.
- **Topología.** Uno de los requerimientos para las PAN Thread es no tener un punto singular de fallo. Para esto, Thread ha escogido una topología de red enmallada o de malla para sus redes PAN. El motivo de esta topología es su alta resiliencia, escalabilidad, robustez y una habilidad de auto reparación en caso de fallo de algún componente.
- **Enlaces de comunicación.** Para estos enlaces de la red, las redes PAN Thread usan radios inalámbricas, las cuáles trabajan en la frecuencia de 2450 MHZ y se basan en el estándar IEEE 802.15.4. Se detallará más en profundidad posteriormente.
- **Enrutamiento.** Cada nodo puede transmitir al nodo vecino más cercano, por lo que, si el destinatario no es ese nodo cercano, se enviarán los datos a través de múltiples nodos. Estos enlaces se configuran automáticamente por el software Thread. Se detallará más en profundidad posteriormente.
- **Identificación de dispositivos.** Cada nodo en una red PAN Thread tiene una dirección IPv6, la cual es comprimida con 6LoWPAN. Se detallará más en profundidad posteriormente.

- **Transmisión de datos.** Todos los enlaces de datos son encriptados en la capa de Enlace y/o en la capa de Transporte.
- **Recursos informáticos.** Todos los nodos necesitan alguna forma de computación, para formar los paquetes de datos, encriptación/descriptación de datos, realizar la transmisión, etc.
- **Servicio ubicuo.** Se ofrece a sus usuarios un servicio siempre disponible.
- **Semántica.** Se analiza vía software, en la nube, los sensores y los datos de entrada, junto con el resto de información y se da una respuesta basada en contexto para el sistema IoT.
- **Gateways.** Las redes PAN Thread, pueden disponer de múltiples Gateways, como los Border Routers, para conectarse a Internet. Añaden redundancia y eliminan puntos de fallo. Estos Border Routers pueden conectarse tanto vía radio al resto de nodos como vía cable o inalámbrica a Internet o al servidor en la nube.

Todas estas características están diseñadas bajo las siguientes cinco restricciones:

- **Bajo consumo.** La mayor parte de los dispositivos están conectados a baterías, por lo que, para evitar un mantenimiento de batería cada poco tiempo, un bajo consumo será importante para alargar la duración de dicha batería. Incluso si estuvieran conectados a la red eléctrica, si no son de bajo consumo, puede generarse un considerable gasto debido a la gran cantidad de componentes.
- **Bajo coste.** Es importante que el sistema tenga dispositivos bajo coste y de bajo coste de mantenimiento, si lo que se quiere es un sistema ubicuo.
- **Seguridad.** Es importante mantener la seguridad de la PAN y de la nube.
- **Interfaz de usuario.** Los dispositivos IoT suelen carecer de interfaz de usuario, por lo que será de gran utilidad que estos dispositivos tengan la facilidad de conectarse a dispositivos con una buena GUI, como un ordenador.
- **Retos de comunicación.** Las redes PAN suelen trabajar en interiores, donde suele haber bastantes obstáculos para la comunicación, como paredes, otros equipos electrónicos, etc. Tanto los obstáculos como los dispositivos pueden variar su posición, al igual que pueden aparecer nuevos dispositivos y los viejos desaparecen. Los dispositivos pueden desconectarse y reconectarse a la red sin ser notificados. Para solucionar este tipo de comunicaciones poco fiables, Thread usa DTLS debido a que DTLS asume una capa de Transporte poco fiable. Debido al bajo consumo, la señal de comunicaciones es débil, complicando más las comunicaciones, por lo que una de las medidas tomadas es la transmisión de paquetes pequeños y con redundancia para tener cierta tolerancia a errores.

### *CAPA FÍSICA [12], [14]*

Los dispositivos Thread soportan una interfaz cumpliendo las especificaciones definidas en IEEE 802.15.4 relativos a los 2450 MHz. Thread también usa modulación O-QPSK y tiene una tasa de datos de 250 kbps. Las señales de radio de Thread pueden alcanzar un salto máximo de 30 metros entre nodo y nodo, pudiendo dar hasta un máximo de 36 saltos y conectando hasta más de 250 dispositivos en una PAN.

### *CAPA MAC (O ENLACE DE DATOS). [12], [14]*

La siguiente capa por implementar de la especificación IEEE 802.15.4 en los dispositivos Thread es el conjunto de Control de Acceso a los Medios (capa MAC). Dependiendo de si el dispositivo tiene o no capacidad de enrutamiento, los dispositivos tendrán que funcionar como un dispositivo MAC completo (MAC Full Function Device – FFD) o como un dispositivo MAC reducido (MAC Reduced Function Device – RFD).

Las capacidades MAC que debe de tener un dispositivo Thread son:

- Uso de Carrier Sense Multiple Access with Collision Avoidance (CSMA-CA).
- Tener tipos de escaneo activo y de detección de Energía (ED).
- Ser capaz de generar y recibir los siguientes paquetes / Frames MAC.
  - Paquetes de datos MAC y de Acknowledgement.
  - Paquetes de solicitud de datos y de solicitud del Beacon.
  - Paquete Beacon cuando se recibe solicitud de FFD.
- Dar un enlace fiable entre dos entidades MAC.
- Implementar paquete de seguridad MAC basado en la configuración de la PAN.

Por otro lado, los dispositivos Thread MAC no se les permite:

- Operar con un modo periódico de Beacon activado.
- Garantizar mecanismos de espacios de tiempo.
- Generar o implementar los siguientes paquetes de comandos:
  - Solicitud de Asociación.
  - Respuesta a Asociación.
  - Solicitud de Desasociación, conflicto con PAD ID.
  - Notificación de huérfanos.
  - Realineación de Coordinadores.
  - Modo Coordinador de PAN FDD.
- Direccionamiento 00 para paquetes de Datos o comandos MAC.
- Cualquier otro paquete de opciones de seguridad, excepto las mencionadas en la configuración de la PAN.

El datagrama que hace el Establecimiento del Enlace MAC (MLE) puede ser o no seguro. Para un MLE seguro, se indica con un byte inicial de valor 0, seguido de un byte de cabecera auxiliar, mientras que cuando no es seguro, el valor del byte inicial es 255 y no tiene ningún byte de cabecera auxiliar.

En Thread hay 18 MLE datagramas de comandos genéricos disponibles, los cuales tratan varios aspectos de establecer enlaces, solicitudes de datos, enrutamiento y descubrimiento de nodos vecinos. Estos comandos MLE pueden ser secuenciados para tareas de alto nivel.

La estructura del paquete MAC de IEEE 802.15.4-2006 (sección 7.2.1) [15] es la mostrada en Tabla 3:

FC	SQ	DstPID	Dst	SrcPID	Src	AuxSec	UD	FCS
----	----	--------	-----	--------	-----	--------	----	-----

*Tabla 3 MAC Layer Frame*

Las diferentes partes que forman el paquete son:

- **FC** → Frame Control (2 bytes).
- **SQ** → Sequence number (1 byte).
- **DstPID** → Destination PID (0/2 bytes).
- **Dst** → Destination Address (0/2/8 bytes).
- **SrcPID** → Source PAN ID (0/2 bytes).
- **Src** → Source Address (0/2/8 bytes).
- **AuxSec** → Auxiliary Security Header (0/5/6/10/14 bytes). Contiene también los campos de Control de Seguridad (Security Control), un contador de paquete (Frame Counter) y un Identificador de Clave (Key Identifier).
- **UD** → User data (variable bytes).
- **FCS** → ITU-T-CRC-16 (2 bytes).

### 2.3.3.4.

#### *CAPA DE ADAPTACIÓN 6LoWPAN [12]*

Thread usa 6LoWPAN, según se especifica en RFC 4944 [16] y RFC 6282 [17], por lo que es posible el uso de IPv6 dentro de las PAN, a pesar de una implementación menos restrictiva. Esta capa permite la fragmentación y el reensamblaje de los paquetes IPv6 procedentes y destinados a la capa MAC.

Esta capa está diseñada para llevar el datagrama con enlaces restringidos, como el estándar IEEE 802.15.4, con anchos de banda limitados, pequeñas memorias, bajas potencias de consumo y transmisiones de un máximo de 250 kbps.

IEEE 802.15.4 especifica una MTU, unidad máxima de transmisión, de 127 bytes, de los cuales 80 octetos de los habilitados en la payload de la MAC están disponibles. 6LoWPAN también permite un encabezado de dirección de la malla admitiendo el reenvío sub-IP.

Debido a la compresión de cabecera en datagramas IPv6, 6LoWPAN reduce bastante las cabeceras IPv6 y UDP a tan solo unos bytes [17].



---

## CAPA DE RED [12]

Los dispositivos Thread deben implementar la especificación IPv6, RFC 2460 [18], al igual que la arquitectura de direccionamiento IPv6, RFC 4291 [19]. Thread usa dos campos para el direccionamiento.

2.3.3.5. Enlace Local, el cuál es accesible en una transmisión de radio.

- Ámbito Local, el cuál es accesible dentro de la PAN mediante saltos.

Los dispositivos Thread deben admitir una dirección de enlace local y al menos dos direcciones de Ámbito Local para uso de comunicaciones internas a la PAN. Si hay disponibilidad de recursos, los dispositivos Thread podrían admitir direcciones IPv6 adicionales como la Unique Local Address (ULA) y la Global Unique Address (GUA). También deberán soportarse también direcciones Multicast de Enlace Local y de internas de la red.

Las direcciones Unicast IPv6 son usadas para el uso interno de la red. Las Unicast pueden ser localizadores de enrutamientos, localizador de cualquier tipo (Leader, DHCPv6, Servicio, Commissioning, NDP) o incluso Identificador de Punto Final (EID – Endpoint Identifier).

Las redes Thread no dependen del protocolo DHCPv6, por lo que los dispositivos no tendrán que implementarlo. Lo que sí que deberán implementar es el protocolo ICMPv6, RFC 4443 [20].

## PROTOCOLO DE ENRUTAMIENTO

2.3.3.6.

Para el enrutamiento, Thread usa un simple protocolo de enrutamiento por vector de distancia. El enrutamiento dentro de la PAN está basado en RIPng (estándares RFC 2080 y RFC 1058). Los routers de la PAN envían periódicamente sus “costes” de enrutamiento a todos los demás routers junto con la calidad de enlace de un salto. Los costes de enrutamiento para un mensaje se sacarán de este último dato.

Cada router crea y mantiene una base de datos con los enrutamientos por cada interfaz que usa enrutamientos por distancias. Estas bases contienen:

- Id del rúter.
- Enlaces
- Rutas

Los routers de la PAN realizan un seguimiento de la versión Thread de los enlaces de sus nodos hijos y, de manera proactiva, manda actualizaciones de versiones más nuevas. Además, si el router es un nodo Leader, este tiene una base de datos con las asignaciones de id. Los routers Líderes recogen, comparan y distribuyen la información sobre los Border Routers y otros servidores disponibles para la red Thread vía MLE.

Thread tiene el concepto de set completo y de set estable de nodos de red. Las redes Thread se pueden dividir en diferentes sets para romper las comunicaciones. Por lo tanto, se incluye una provisión para que cada set actúe como una red diferente, pudiendo unirse luego como una sola red. Si no hay un nodo Líder alcanzable o disponible, un REED o un Border Router pueden generar una nueva partición en la red.

### *CAPA DE TRANSPORTE [12]*

Esta capa de transporte en Thread está basada en User Datagram Protocol (UDP), descrito en RFC 768 [21], y en Requerimientos para Host De Internet, RFC 1122 [22].

Para el protocolo de mensajería, Thread usa Constrained Application Protocol (CoAP) [23], [24] debido a las limitaciones de baja memoria y de baja capacidad de procesamiento. El protocolo usado por CoAP es un UDP, que obliga a usar DTLS (Datagram Transport Layer Security), RFC 6347 [25].

CoAP tiene un conjunto de modos de seguridad y de cifrados de implementación obligatoria. CoAP toma prestado algunos conceptos del protocolo Representational State Transfer (REST).

Los modos en los que opera CoAP son:

- Modo sin seguridad ("no security").
- Modo Clave Pre-Compartida ("Pre-shared key").
- Modo Certificado ("certificate").

CoAPs (CoAP seguro) puede también configurarse en el modo "raw public key", definido en RFC 7250 [26]. En el modo de Clave Pre-Compartida, CoAP junta claves Pre-Compartidas con una lista de los correspondientes nodos de comunicación. En cuanto al modo Certificado, está muy bien establecido, pero se desaconseja su uso debido a las limitaciones de recursos.

El modo sin seguridad es solo recomendable cuando en la capa de Enlace de Datos hay encriptación disponible y en uso.

Thread usa encriptación DTLS en la capa de Transporte en caso de que el hardware no soporte la encriptación de los datos. Otra posibilidad es una doble encriptación, tanto en la capa de transporte como en la de enlace de datos (capa MAC), lo cual reduce el ratio de compresión, dejando paso a su vez a más transmisiones de paquetes y potencias consumidas mayores.

2.3.3.8.

### *SEGURIDAD Y COMMISSIONING DE DISPOSITIVOS [10], [12]*

Para dicho Commissioning Thread usa MeshCoP (Mesh Commissioning Protocol), el cual permite una autenticación, un control y una unión a la red de nuevos dispositivos desconocidos a la red de manera segura.

Una PAN Thread es una malla de dispositivos autónoma que se autoconfigura con la interfaz y la capa de enlace a nivel de seguridad de IEEE 802.15.4. Cada dispositivo de la PAN debe poseer una clave maestra secreta compartida y actualizada.

Para añadir a la PAN un dispositivo, primero habrá que autenticarlo manualmente como un dispositivo aceptado para unirse a la red, prosiguiendo con el proceso de commissioning del dispositivo y compartiéndole la clave maestra de la PAN a través de un canal seguro.

Los dos principales métodos de commissioning disponibles en Thread son:

- Usando el método Out-of-band, configurar directamente sobre el dispositivo toda la información necesaria para el proceso de commissioning. Esta información permitirá a los dispositivos que quieran unirse, que se unan a la red Thread correcta. Este método se desarrollará más detalladamente en 3.3.3 CONFIGURACIÓN DE RED de manera aplicada a nuestro proyecto.
- Establece una sesión de commissioning entre una aplicación propia de commissioning, ya sea en web, teléfono móvil, etc y el dispositivo que se quiere unir. Esta aplicación de comisión le da la información necesaria al dispositivo para que se conecte a la red adecuada.

Antes de que un dispositivo sea autenticado, el propio Commissioner deberá autenticarse también. Para ello se establece una conexión socket cliente/servidor, la cual se le llamará Sesión de Commissioning y sucede entre el Commissioner y el propio Border Router (BR) usando DTLS (RFC 6347 [25]) o TLS (RFC 5246 [27]). Esta sesión abierta usa un determinado puerto UDP y se usa una credencial conocida como Pre-Shared Key for Commissioner. Tras terminar esta autenticación, el Commissioner se registra con el BR, y este último comparte la información del Commissioner al Nodo Leader, el cuál procederá a aceptar o rechazar al Commissioner. En caso de ser aceptado el Commissioner, el Leader manda su información a los Joiner Routers (JR) de la PAN.

Cuando un dispositivo desea unirse a la PAN, se le llama Joiner Device (JD). Para unirse a la PAN, el JD debe enviar mensajes de Descubrimiento Solicitud/Respuesta usando MLE en modo inseguro a un JR. Antes de que el tiempo de unión expire y si la unión de dispositivos a la red está activada, el JR deberá enviar una Respuesta de Descubrimiento con un puerto UDP de unión. Si no está activada la unión de dispositivos a la red, no se enviará ninguna respuesta. Tras recibir este mensaje de Respuesta, la unión del JD se dará por validada.

Una vez en la red, el JD mandará un saludo vía DTLS al JR, estableciendo con él una Joiner Session (JS). El JR mandará mensajes UDP al BR, el cual los transmitirá al Commissioner. A pesar de manejar estos mensajes, ni el BR ni el JR tienen acceso al contenido de los mensajes, solo los transmiten al puerto UDP indicado. Para afianzar una confianza entre dispositivos, hay un intercambio de mensajes entre el JD y el Commissioner.

El Commissioner revisa el Identificador de Interfaz del JD (IID) y sus credenciales y si todo está correcto, el JD recibirá con los datos y los servicios apropiados, a la vez que comparte su Key Encryption Key (KEK) y finalmente se cierra la JS.

Finalmente, el Commissioner enviará una señal al JR que al JD se le compartirán las credenciales de la red. Finalmente, el Commissioner le proporciona un secreto compartido entre JD y JR, completando así la unión del JD.



## 3. ANÁLISIS DE LA TECNOLOGÍA THREAD

### 3.1. ANÁLISIS INICIAL DE LA TECNOLOGÍA THREAD

A la hora de realizar la selección de los módulos, se ha realizado un primer análisis de todos los fabricantes de dispositivos de tecnología THREAD. Se ha visto que hay bastantes desarrolladores en los diferentes ámbitos, ya sean módulos, hardware o entornos de desarrollo, etc, siendo alguno de los más destacados Texas Instruments, Kirale, Google con Openthread y Qualcomm.

Entre las diferentes empresas que desarrollan tecnología THREAD, se ha visto que Kirale Technologies es la única entidad certificada Thread con módulos, hardware de desarrollo y Border Router disponibles, por lo que se vio como potencial elección para los dispositivos a adquirir, a pesar de no contar con un entorno de desarrollo software.

Otras ventajas que se han visto en la tecnología de Kirale son:

- **Módulos RF** con una huella de PCB optimizada para un pequeño tamaño y con una alta eficiencia energética. También permiten todos los roles de Thread.
- **Primer Backbone Border Router.** Tan solo Kirale ha conseguido desarrollar un dispositivo Border Router, capaz de enlazar las redes Thread con otras redes IPv4 o IPv6. La única alternativa es la ofrecida por OpenThread, con un Border Router vía Software con Raspberry Pi 3B o BeagleBone. Esto sería interesante en caso de querer aprovechar las diferentes entradas del hardware para diferentes tipos de sensorizados. En el caso de este proyecto, el hardware de sensorizado está integrado en la plataforma Cookie donde irán integrados los módulos RF.
- **Dongle de Evaluación.** Dispositivo de evaluación que permite un prototipado rápido y a bajo coste. También destaca por ser de los kit de evaluación Thread de menor tamaño.
- **KiNOS.** Stack con Certificación Thread que se caracteriza por versatilidad al soportar todos los roles Thread y por habilitar diferentes interfaces para una fácil integración y un bajo consumo.

Una posible desventaja es la falta de sensorizado o de entradas Analógicas y/o Digitales para la integración de sensores, de cara a posibles pruebas de envíos de datos que se estén leyendo en tiempo real. Esto obliga a elaborar una plataforma integrando los sensores a un microcontrolador que se encargue tanto de la lectura de los sensores como del control del Dongle de evaluación vía UART.

Tras el primer análisis realizado, se ha elegido la tecnología de Kirale, debido a que en este proyecto la implementación de tecnología THREAD será hardware y la plataforma final integra un pequeño sistema de sensorizado, por lo que las principales desventajas no afectarían al desarrollo del proyecto.

#### 3.1.1. CARACTERÍSTICAS TÉCNICAS DE LOS MÓDULOS KIRALE

##### *CARACTERÍSTICAS MÓDULO RF KTWM102*

Los módulos RF KTWM102 es un Co-Procesador que integra en su totalidad el protocolo de red IEEE 802.15.4. El KTWM102 permite integrar en distintos dispositivos la conectividad Thread de manera sencilla y a bajo coste. Cuenta con el sistema KiNOS, una Stack con Certificado Thread, segura, robusta y de altas prestaciones. KiNOS puede ejecutarse tanto de manera autónoma como bajo la acción de un host que se comunique con el módulo KTWM102.

KTWM102 ofrece una conectividad serie de alta velocidad a Thread a través de una interfaz UART, permitiendo así una integración de las comunicaciones con un host con un microcontrolador. A su vez, integra conectividad nativa USB 2.0 para cumplir con los requisitos de diseño del Border Router.

Finalmente, KTWM102 combina una arquitectura de microprocesador de gran eficiencia energética con técnicas de ahorro energético integradas en el sistema KiNOS.

##### *CARACTERÍSTICAS KTDG102 EVALUATION DONGLE*

###### 3.1.1.2.

En primera instancia, el módulo KTDG102 integra el módulo RF KTWM102 comentado en 3.1.1.1 CARACTERÍSTICAS MÓDULO RF KTWM102 incluyendo todas sus características y tecnologías.

Dispone de un conector UART permitiendo así enlazar con cualquier host, como un microcontrolador. A la vez dispone de un conector USB, permitiendo así ser conectado a un PC.

Dispone de un multiplexor de alimentación de dos entradas, permitiendo que el Dongle sea alimentado tanto por el conector USB como por el conector UART a través de un host.

Integra una señalización lumínica para conocer el estado del dispositivo en todo momento.

El Dongle KTDG102 tienen una gran versatilidad, puesto que permite ser configurado como cualquiera de los tipos de nodos posibles dentro de una red Thread, pudiendo así analizar los comportamientos de todos los tipos de nodos dentro de la red.

El sistema KiNOS proporciona la interfaz KSH, una interfaz sencilla y con breve curva de aprendizaje, a través de la cual el usuario puede controlar sobre los dispositivos.

El KTDG102 permite actualizar el firmware a los usuarios, pudiendo así cambiar incluso la funcionalidad del dispositivo, ya sea a un nodo Thread, un sniffer 802.15.4 o un dispositivo Golden Reference, el cual ejecutar la certificación Thread interna.

---

### *CARACTERÍSTICAS DEL BORDER ROUTER KTBRN1*

El módulo Border Router, tal como se ha explicado, es el módulo que proporcionará la conectividad a la red THREAD con otras capas físicas como Ethernet o Wi-Fi. El KTBRN1 es el primer Border Router que incluye el servicio de Thread Backbone Border Router. A su vez, implementa la mayoría de las funcionalidades y características indicadas en la especificación V1.2 del protocolo Thread.

El Border Router es un equipo basado en la placa NanoPi NEO, al que se le ha añadido una placa HAT incorporando el módulo RF KTWM102. Se incorpora un procesador ARM Cortex A7 de cuatro núcleos con 512 MB de RAM.

KBRN1 admite tanto protocolo IPv4 como IPv6, enrutando el tráfico según sea necesario. Aparte de este enrutamiento, KTBRN1, ofrece servicios como servicios DNS, DHCP, NAT, etc.

KTBRN1, incluye un paquete software formado por un conjunto de Scripts en Python, llamado KiBRA, a la vez que un sistema operativo basado en Debian y Ubuntu, Armbian, sobre el que corre el paquete de KiBRA. KiBRA se encarga de las tareas necesarias para proporcionar los servicios del Border Router KTBRN1, con la ventaja de ser de código abierto y customizable.

Por último, KTBRN1 incluye, gracias a KiBRA, un servicio de Administración Web desde el que manejar y configurar la red Thread del nodo KTBRN1 y todos los servicios proporcionados.

#### *3.1.1.4. CARACTERÍSTICAS KINOS – NETWORK OS*

KiNOS proviene de las siglas Kirale Real-Time Network Operating System y es un Stack Certificado Thread que se caracteriza por sus altas prestaciones, seguridad y robusted para comunicaciones inalámbricas de bajo consumo.

Permite actualizar el firmware a través USB-DFU y gracias a una encriptación punto a punto AES-EAX. A su vez, protege al dispositivo de firmware no oficial.

Por lo que más se destaca KiNOS es por el soporte de todos los roles de Thread y por las interfaces que permiten una fácil integración y desplegar modos de bajo consumo o de ahorro de batería.

#### 3.1.2. EJEMPLOS DE OTROS DISPOSITIVOS

Como ya se ha mencionado en el análisis realizado en 3.1 ANÁLISIS INICIAL DE LA TECNOLOGÍA THREAD, existe una gran variedad de desarrolladores en la tecnología Thread. A continuación se hará mención de algunas de las diferentes alternativas con certificación Thread que existen actualmente.

En cuanto a Border Routers, como ya se ha mencionado, solo se dispone de la alternativa ofrecida por OpenThread, el cuál consiste en una implementación en sistemas basados en Linux o en Raspberry Pi 3B.

En caso de necesitar un entorno de desarrollo software, existen entornos desarrollados por NXP ([MCUXpresso SDK, IDE, and Configuration tools](#)), Nordic Semiconductor ([nRF Connect SDK](#)), Silicon Labs ([Simplicity Studio](#)) y MMB Networks ([RapidConnect Desktop](#)), por lo que habría que ver cual se adapta más a los requerimientos del proyecto.

En cuanto a hardware de desarrollo, Nordic Semiconductor (el equivalente al Dongle de Kirale sería: [nRF52840 Dongle](#)) y NXP ([OM15080-K32W](#)) cuentan con mayor variedad de dispositivos hardware, pero también existen otros dispositivos desarrollados por empresas tecnológicas como Qualcomm ([QCA 4020/ 4024/ 4025](#)) o Texas Instruments ([SimpleLink LaunchPad™ Development Kits](#)).

En módulos RF con certificado Thread, el desarrollador con mayor variedad es Silicon Labs (teniendo un equivalente al módulo de Kirale: [MGM210L / MGM210P](#)), pero también se cuenta con desarrolladores como MMB Networks ([RapidConnect Module](#)) y NXP ([Azurewave NXP Platform Wireless Modules](#)).

Para sistemas operativos, tenemos desarrolladores como Silicon Labs ([Micrium OS](#)) o Zephyr ([Zephyr RTOS](#)).

Finalmente, aparte de Kirale, hay también más desarrolladores de Stack Software, como son OpenThread ([OpenThread](#)) y Silicon Labs ([Silicon Labs Thread Stack](#)).



## 3.2. CONFIGURACIONES INICIALES

### 3.2.1. CONFIGURACIÓN DEL BORDER ROUTER

#### *REQUISITOS*

Para la instalación inicial se necesitarán los siguientes componentes o elementos:

- Un dispositivo Border Router KTBRN1.
- 3.2.1.1. • Una tarjeta micro SD, de al menos 2GB y de clase 10 A1.
- Cable USB de tipo A a micro USB tipo B.
- Cable Ethernet.
- Un ordenador personal

#### *INSTALACIÓN DEL SW*

##### 3.2.1.2.

Para la instalación del software es necesario descargar varios ficheros, algunos de los cuales pueden no ser necesarios si ya se disponen de ellos en el PC. Para instalar y configurar todo el entorno se deben de seguir los pasos descritos a continuación. Dependiendo de si ya se disponen, instaladas, de algunas de las herramientas requeridas, no será necesario realizar el paso correspondiente.

##### 3.2.1.2.1. DESCARGA DEL SW REQUERIDO

El software necesario se divide en dos partes. Por un lado el software que se debe de instalar en el PC y por otro el SW que se instalará en el dispositivo KTBRN1.

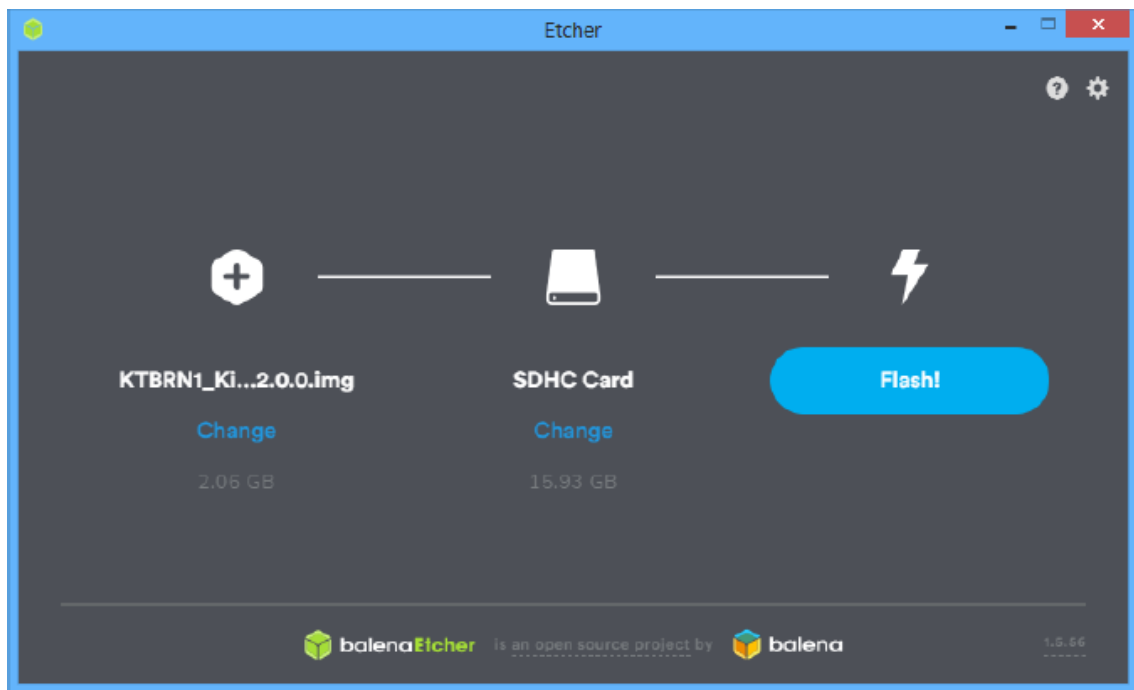
- **Software KTBRN1:** Este SW será instalado en el módulo KTBRN1. Dicho SW se puede encontrar en la página oficial de Kirale. Siguiendo el siguiente enlace se podrá descargar el SW necesario. [KTBRN1 + KiBRA image file](#).  
***Nota:** Debe asegurarse la descarga de la última versión. El fichero descargado incluye el software de KiBRA. Una vez dentro del enlace, dentro del apartado de software, podremos acceder tanto a la descarga del SW con extensión .gz como a la descarga del fichero KiBRA-v2.x.x.zip, el cual se usará posteriormente.*
- **Balena Etcher:** Software utilizado para la flashear o actualizar la imagen descargada previamente en la tarjeta SD. Se podrá descargar en el siguiente enlace [Balena Etcher](#).
- **MobaXterm:** Programa usado tanto para la conexiones por terminal Serie como cliente SSH con el dispositivo KTBR1. Para su descarga, acceder en el siguiente enlace [MobaXterm free](#) y seleccionar “Get MobaXterm Now” para elegir la versión deseada.
- **Zadig:** Programa utilizado para la instalación de los drivers de USB Serie. Se podrá descargar en la página oficial, siguiendo el siguiente enlace: [Zadig](#)

#### 3.2.1.2.2. FLASHEAR LA IMAGEN EN LA TARJETA SD

Como se ha indicado anteriormente, para el flasheo de la memoria se usará el programa de Balena Etcher. En caso de no estar instalado en el PC, ejecutar el fichero .exe descargado para su instalación.

Si es necesario actualizar o flashear una imagen en una tarjeta SD, se deberán seguir las siguientes instrucciones:

- Ejecutar el programa, la Ilustración 8 muestra la ventana del programa.
- Inicialmente, se debe seleccionar el fichero con extensión .gz (la imagen) que se desea grabar (campo izquierdo de la ventana).
- Seguidamente, se selecciona la unidad donde se encuentra la SD.
- Posteriormente, se clicá en Flash y esperar a que termine.
- Una vez finalizado el proceso de flasheado, se expulsará la tarjeta SD y se introducirá en la ranura para micro SD del módulo KTBRN1.



*Ilustración 8 Balena Etcher*

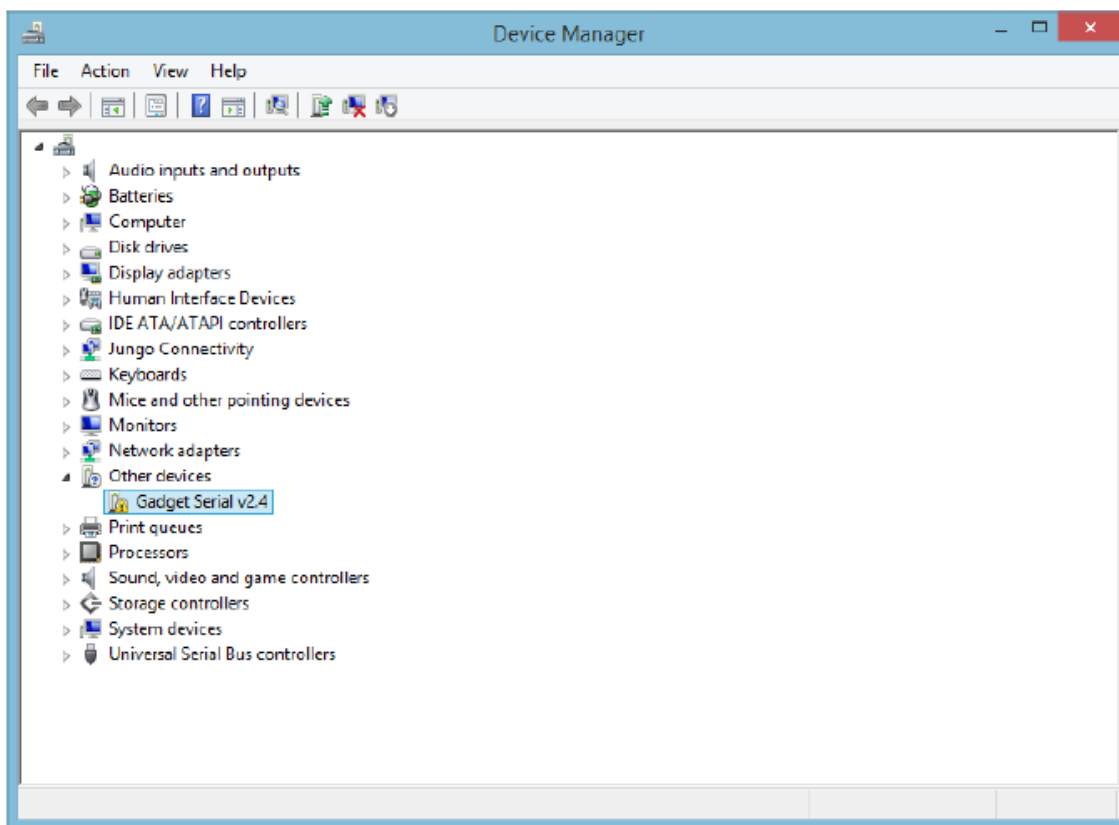
### 3.2.1.2.3. INSTALACIÓN DE DRIVERS USB

Se conectará con un cable USB el dispositivo KTBRN1 al PC. La primera vez que se encienda el dispositivo, tardará unos minutos en estar listo para aceptar conexiones. **No apagar** el sistema hasta que el proceso de primera instalación haya terminado.

Una vez terminado, se seguirán los pasos descritos en 3.2.1.2.3.1 CONEXIÓN VÍA PUERTO USB SERIE para acceder al dispositivo KTBRN1 a través de puerto USB Serie.

#### 3.2.1.2.3.1. CONEXIÓN VÍA PUERTO USB SERIE

Al conectarse, deberá detectarse y listarse un nuevo dispositivo Serie (USB a Serie), dependiendo del sistema operativo del ordenador. Quizás se requiera que la instalación del driver para el puerto USB a Serie, para ello se comprobará si el ordenador lo reconoce, abriendo el Administrador de dispositivos.



*Ilustración 9 Administrador de Dispositivos Previo a la instalación de Drivers del BR*

En caso de encontrarse en la situación mostrada en la Ilustración 9, se instalan los driver usando la herramienta Zadig. Al abrir la herramienta Zadig, se seleccionará el dispositivo *Gadget Serial v2.4* y el driver *USB Serial (CDC)*, tal como se muestra en la Ilustración 10. Una vez se hayan seleccionado las opciones comentadas se clicará en *Upgrade Driver*.

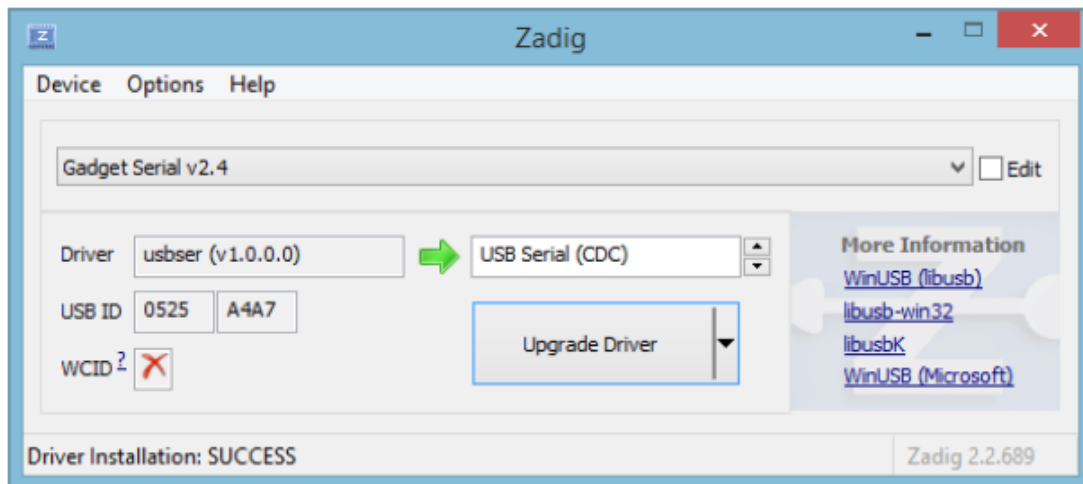


Ilustración 10 Instalación Drivers de Border Router con Zadig

Una vez se tenga acceso al KTBRN1 vía USB Serie, se abrirá MobaXterm con una nueva sesión Serie. Se seleccionará el puerto asignado al KTBRN1 con un BaudRate de 115200. Tras la configuración de la terminal, aparecerá una consola de inicio de sesión, en la cual se deberá iniciar sesión con el usuario *root* y la contraseña *kirale123*. Una vez iniciada la sesión, saldrá por la consola la pantalla de bienvenida, tal como se muestra en Ilustración 11.

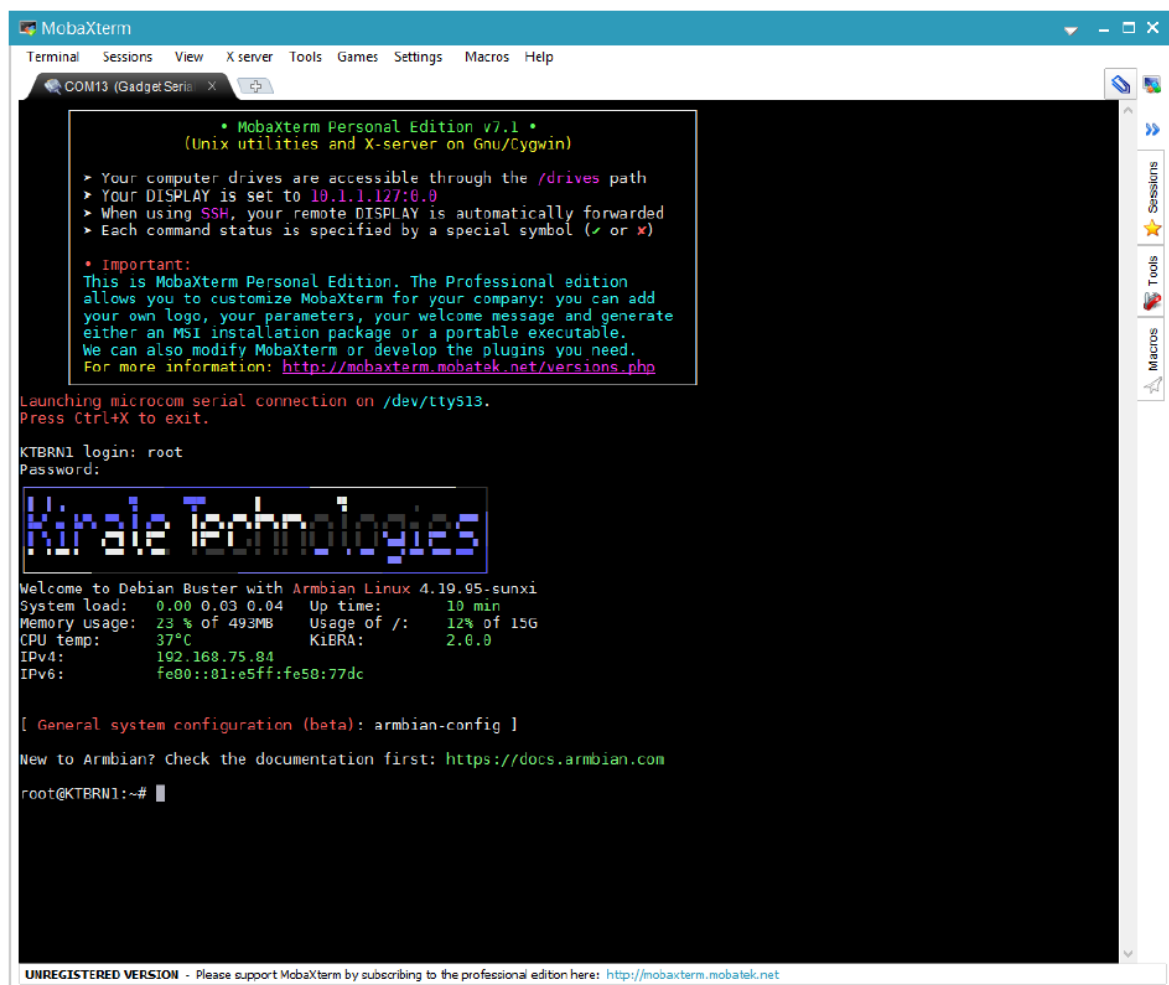


Ilustración 11 MobaXterm

La pantalla de bienvenida mostrará información sobre las direcciones IP configuradas en el KTBRN1 y que versión de KiBRA está instalada, tal como se ve en Ilustración 11. Por defecto, la imagen instalada viene configurada con una dirección IPv4 estática para la interfaz Ethernet, la cuál es: 192.168.75.84/24

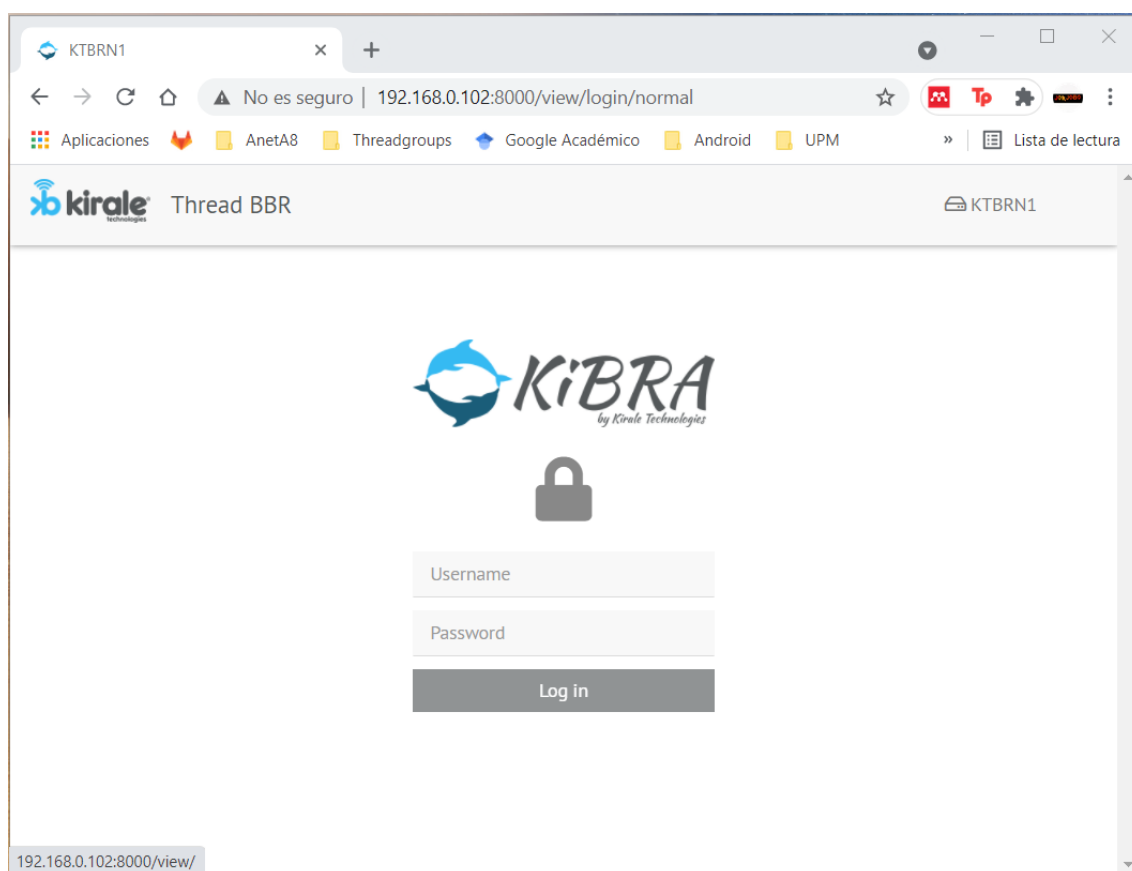
Es posible que esta dirección IPv4 no esté dentro de la red local en la que se vaya a trabajar, por lo que deberá cambiarse desde la terminal Serie mediante comandos Unix, para que el dispositivo KTBRN1 sea accesible desde el resto de equipos. Posteriormente a realizar el cambio en la dirección IPv4 se podrá acceder desde puerto SSH.

Además, KTBRN1 viene con el protocolo IPv6 habilitado para la interfaz Ethernet, por lo que es posible acceder tanto al Panel de Administración Web como a puerto SSH usando las direcciones IPv4 e IPv6.

### *PANEL DE ADMINISTRACIÓN WEB*

El dispositivo KTBRN1 dispone de un Panel de Administración Web, el cual nos permitirá configurar posteriormente los diferentes servicios del Border Router.

Para acceder al Panel de Administración Web, está habilitado el puerto 8000 del KTBRN1. Se accederá introduciendo `http://[IPv4]:8000` o `http://[IPv6]:8000` en el navegador (preferiblemente Google Chrome o Mozilla Firefox debido a razones de compatibilidad). Una vez introducida la dirección web, se mostrará la página de acceso / login, tal como se visualiza en Ilustración 12.



*Ilustración 12 Login Panel Administración Web*

Las credenciales son las mismas a las mencionadas en 3.2.1.2.3.1 CONEXIÓN VÍA PUERTO USB SERIE. Acceder con usuario *root* y contraseña *kirale123*.

#### 3.2.1.3.1. CAMBIAR LA CONFIGURACIÓN DE RED

En esta pestaña será posible modificar la configuración de la interfaz Ethernet, tanto en la correspondiente a IPv4 como a IPv6. Este cambio podrá realizarse accediendo al menú “Network” en la administración Web, mostrado en Ilustración 13.

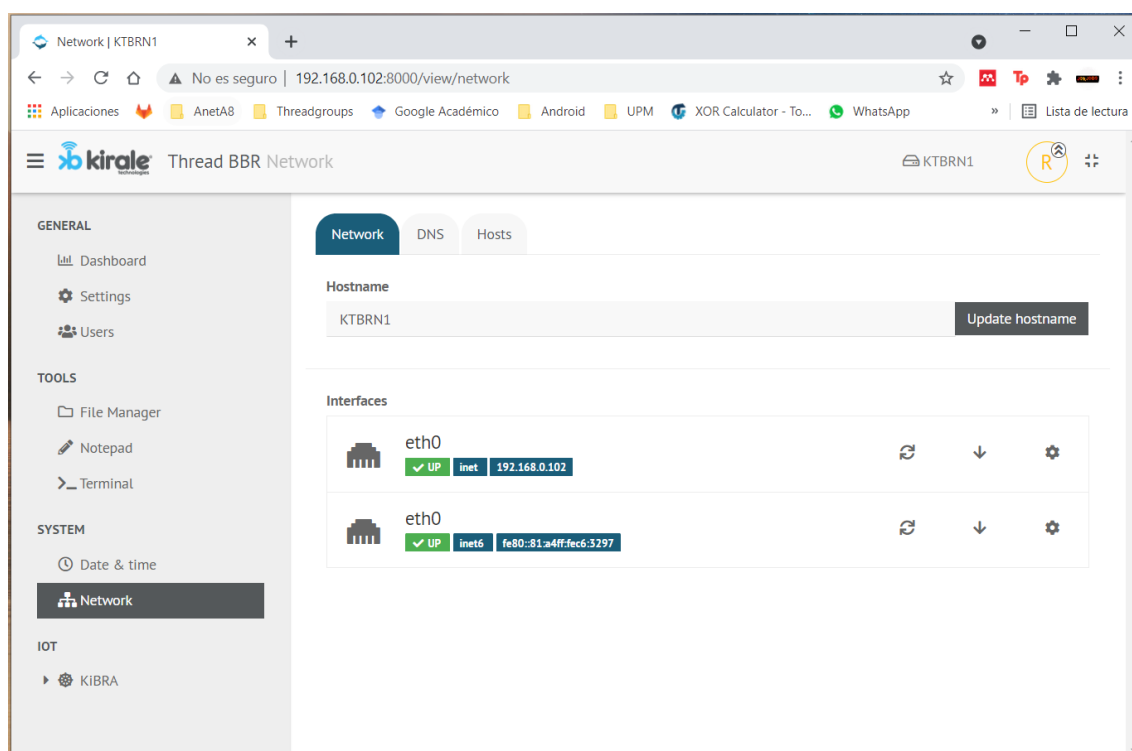


Ilustración 13 Pestaña Network

**Nota:** Cada vez que se realice un cambio en alguna de las interfaces de ethernet, ya sea en IPv4 como en IPv6, deberá reiniciarse el dispositivo KTBRN1 para asegurarse de que se guarden las configuraciones nuevas.

Aparte de la correcta configuración de las direcciones IPv4 e IPv6, se deberá configurar correctamente los servidores DNS. Para esta configuración, en mismo menú de Network, se deberá acceder a la pestaña DNS, donde se deberán añadir los servidores DNS correspondientes de la red. La pestaña se mostrará como se ilustra en Ilustración 14:

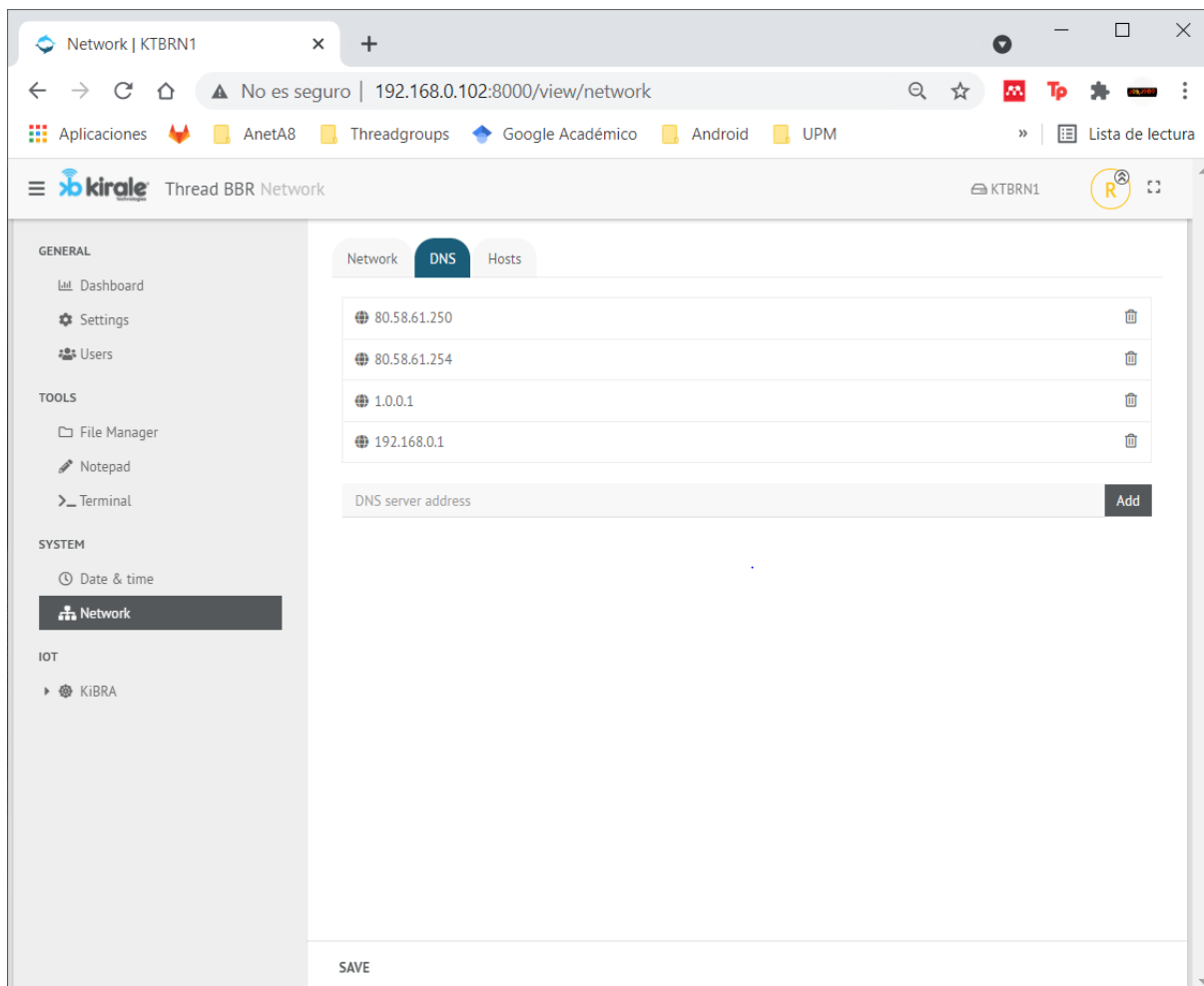


Ilustración 14 Pestaña DNS

#### 3.2.1.3.2. ACTUALIZAR KiBRA

Para actualizar la versión de KiBRA, se deberá acceder al menú de KiBRA y se clicará en el icono de “Upgrade”, situado al lado de la imagen del KTBRN1 en la sección de “System”, tal como se muestra en Ilustración 15

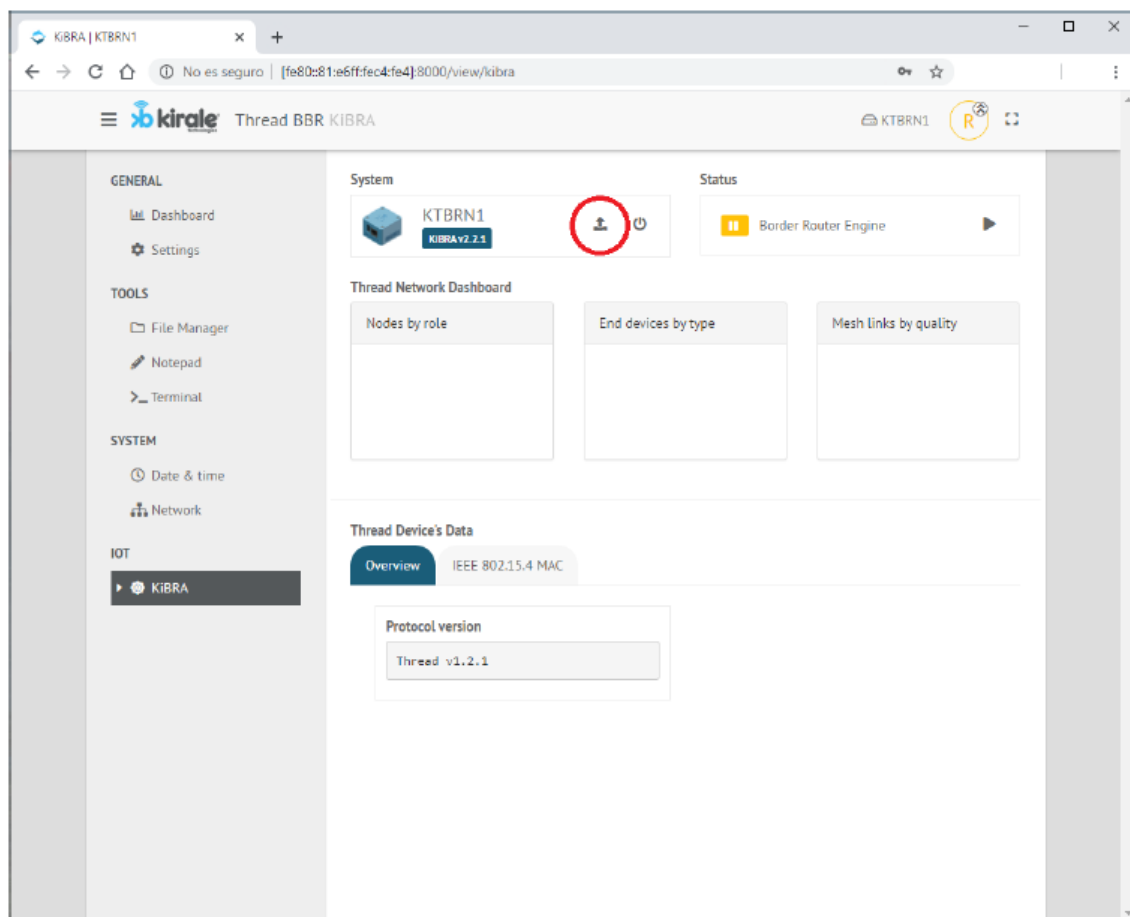


Ilustración 15 Actualización KiBRA

Después, se seleccionará el fichero KiBRA-v2.x.x.zip descargado anteriormente, y se clicará en el botón “Install” y se seguirán las instrucciones que aparecerán en pantalla.

Deberá mantenerse KiBRA actualizado, ya que, como se detalló en 3.1.1.3 CARACTERÍSTICAS DEL BORDER ROUTER KTBRN1, es el sistema encargado de los servicios y funcionalidades del KTBRN1.



### 3.2.1.3.3. CONFIGURAR BORDER ROUTER

Se clicará en el submenú “Settings” por debajo del menú KiBRA, para acceder a la página de configuración. En esta pestaña, será donde se configurará los diferentes parámetros para unirse a la red, tal como se muestra en Ilustración 16.

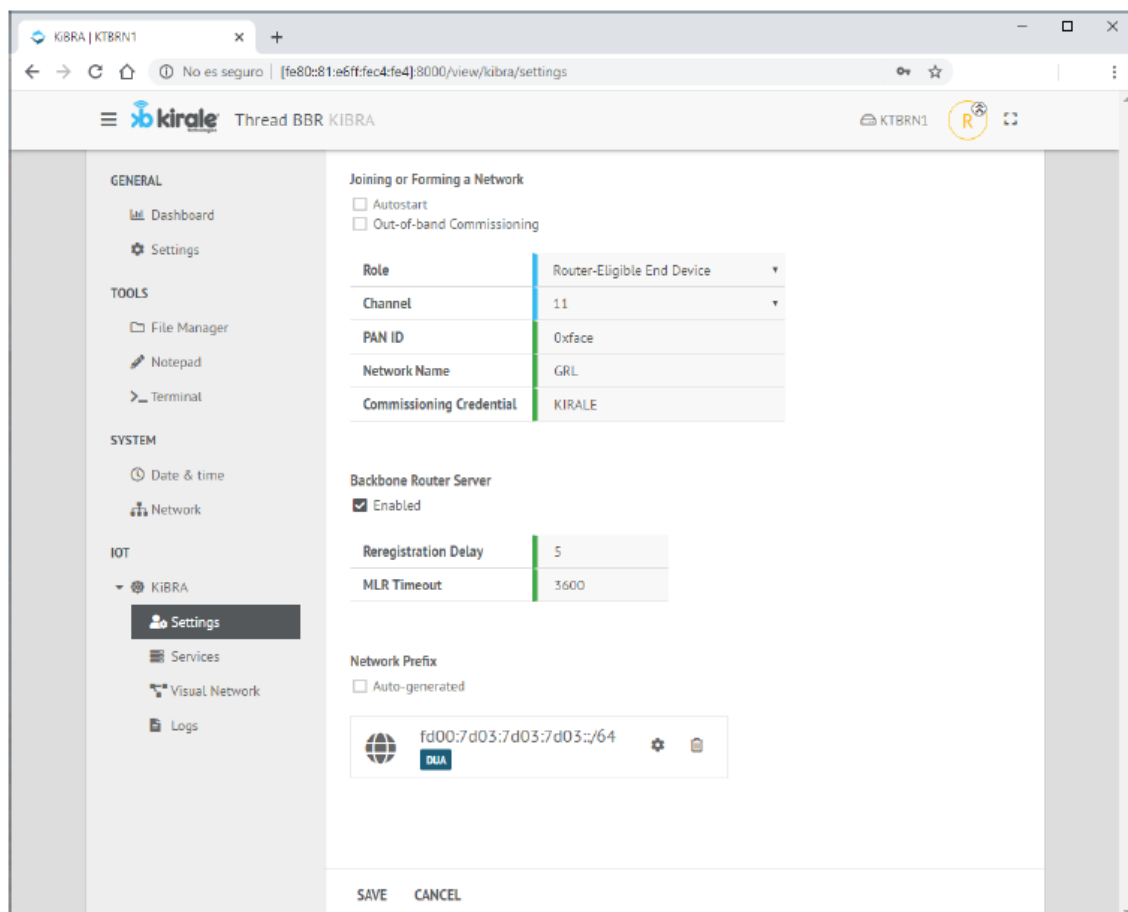


Ilustración 16 Pestaña Settings

#### 3.2.1.3.3.1. UNIRSE O FORMAR UNA RED THREAD

En esta sección se mostrará la configuración para que un dispositivo pueda unirse a una red y los parámetros que necesitarán para el proceso.

- 1) **Autostart:** Si esta opción está activada, el BR se intentará unir a la red Thread después del siguiente reinicio.
- 2) **Out-of-band Commissioning:** Permite activar o desactivar este tipo de conexión a la red cuando el sistema arranca. Este servicio de red se explicará más detalladamente en 3.3.3 CONFIGURACIÓN DE RED

- a. **Activando** este modo los parámetros a configurar de la red serán los mostrados a continuación en Ilustración 17, siendo todos de obligatoria configuración, tal como se detallará en 3.3.3.2 MODO OUT-OF-BAND COMMISSIONING ACTIVADO:

Joining or Forming a Network

☐ Autostart

☒ Out-of-band Commissioning

Role	Router-Eligible End Device
Channel	11
PAN ID	0x1234
Network Name	MyNetwork
Master Key	0x00112233445566778899aabbccddeeff
Mesh-Local Prefix	fd12:3400:3800::
Extended PAN ID	0x1122334455667788
Thread Domain Name	DefaultDomain

Ilustración 17 Parámetros a configurar con Out-of-band Commissioning Activado

- b. **Desactivando** esta opción, el Border Router buscará la red de la manera detallada en ,los parámetros a configurar serán los mostrados en Ilustración 18, siendo el Rol, la única configuración indispensable para el arranque del Border Router, tal como se detallará en 3.3.3.1 MODO OUT-OF-BAND COMMISSIONING DESACTIVADO:

Joining or Forming a Network

☐ Autostart

☐ Out-of-band Commissioning

Role	Router-Eligible End Device
Channel	11
PAN ID	0x1234
Network Name	MyNetwork
Joining Credential	8404D20000000709
Thread Domain Name	DefaultDomain

Ilustración 18 Parámetros a configurar con Out-of-band Commissioning Desactivado

#### 3.2.1.3.3.2. BACKBONE ROUTER SERVER (BBR).

Da la posibilidad de habilitar o deshabilitar la función BBR, dotando al BR del servicio DUA, el cuál detecta duplicaciones de direcciones en los dispositivos. De igual manera, el administrador de la red podrá configurar los parámetros específicos que usará el Servidor del Border Router.

### 3.2.1.3.3.3. PREFIJO DE RED (NETWORK PREFIX)

Esta opción activa la configuración manual del prefijo de la red, el cuál servirá como identificador de la red en la dirección IPv6 de un nodo interno.

En la configuración del prefijo de red, tal como se muestra en Ilustración 19, el usuario podrá decidir entre diferentes opciones el tipo de direccionamiento IPv6 de los nodos dentro de la red, como DHCP o SLAAC (desactivando DHCP). Si el BBR está activado, la opción de prefijo DUA será activada.

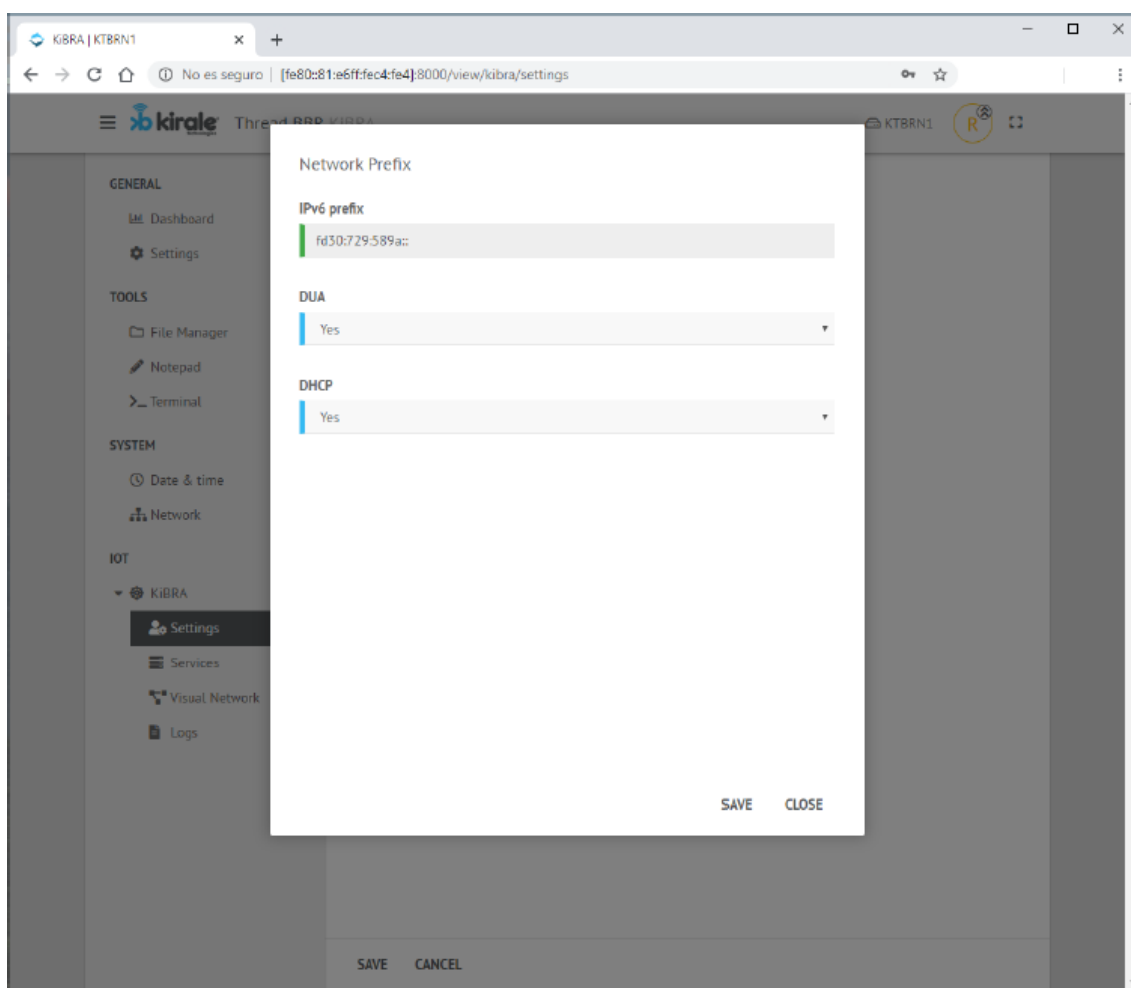


Ilustración 19 Configuración Prefijo de Red

Una vez configurado el Border Router con los parámetros deseados, **“Guardar”** los cambios.

#### 3.2.1.3.4. INICIO DEL BORDER ROUTER (START-UP).

Para activar el Border Router y sus servicios, se accederá al menú KiBRA, mostrado en Ilustración 20, y se clicará en el botón de Start.

Tras encender el sistema, el Border Router se unirá a la red seleccionada o formará una nueva, según los ajustes configurados por el administrador en 3.2.1.3.3.1 UNIRSE O FORMAR UNA RED THREAD.

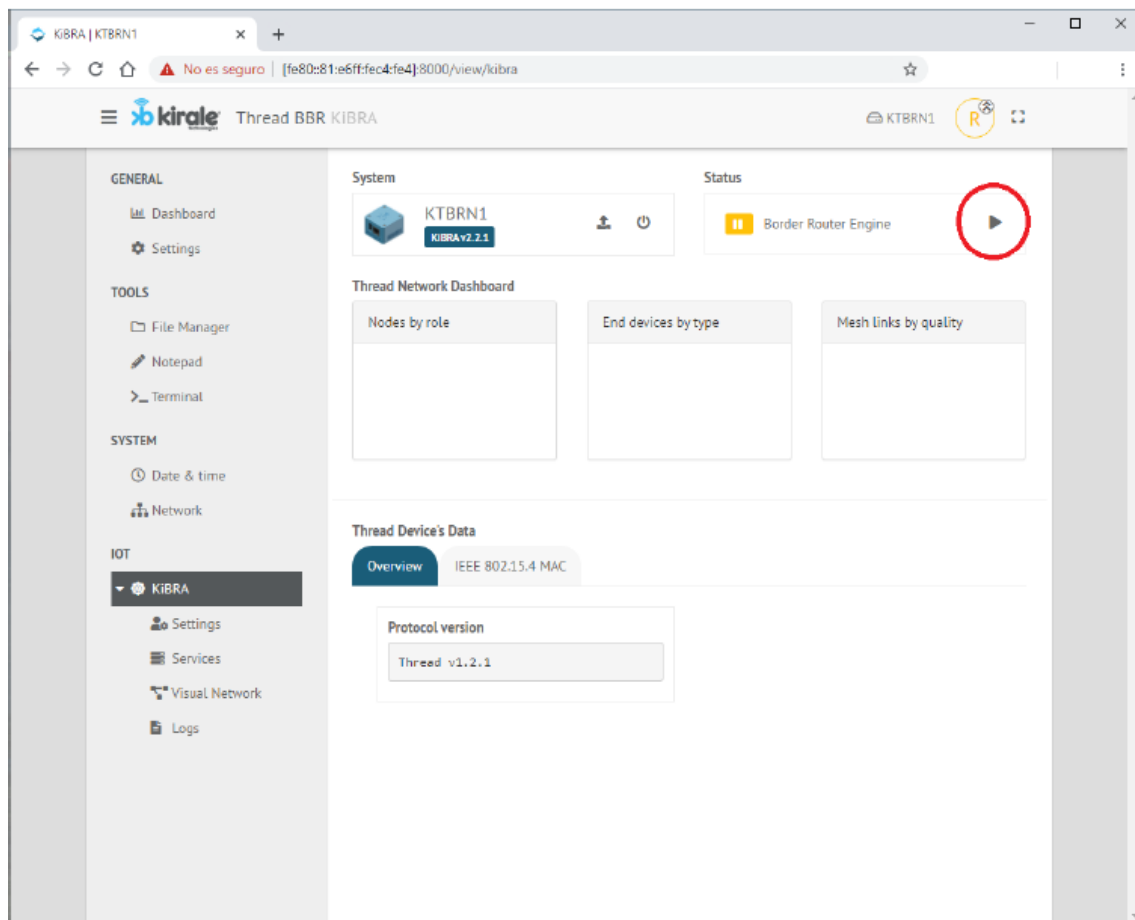


Ilustración 20 Menú KiBRA - Inicio de Border Router

Tras formar parte de la red, el BR procederá a activar los servicios configurados. Una vez arrancados todos los servicios, el menú KiBRA se indicará tanto que el BR está arrancado, los roles de los nodos y el número de nodos en la red, la calidad de los enlaces dentro de la red y, finalmente, la información Thread del propio dispositivo BR. En Ilustración 21 se indica como se mostrará esta pestaña cuando el BR haya arrancado siendo configurado como nodo Leader de la red. Según se unan nodos a la red, esta pestaña se irá actualizando automáticamente añadiendo toda la información de estos.

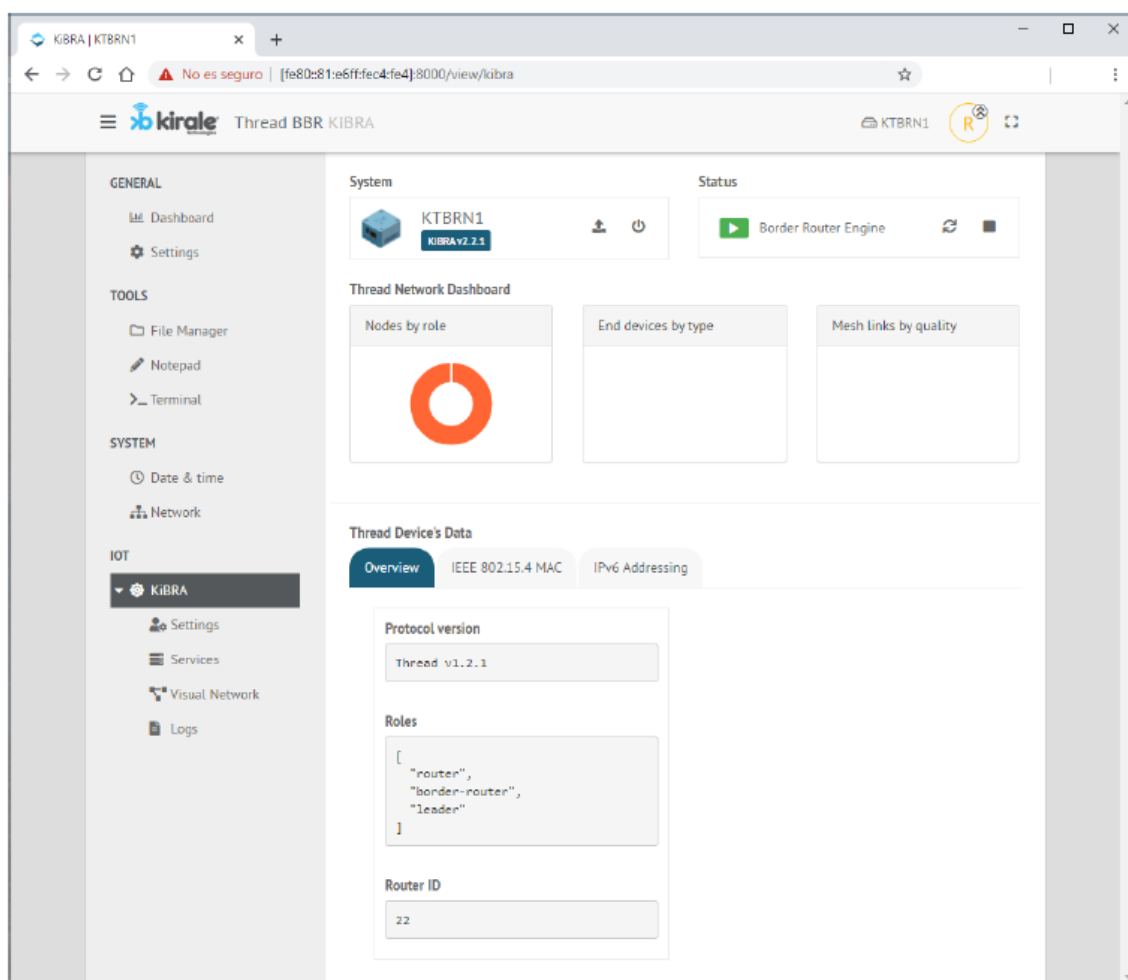


Ilustración 21 Menú de KiBRA - Border Router Iniciado

### 3. ANÁLISIS DE LA TECNOLOGÍA THREAD

Si se procede a volver al submenú de “Settings” (Ajustes), como se muestra en la Ilustración 22, será posible ver el resto de parámetros de la configuración de la red Thread que han sido configurados, ya sea por el administrador o automáticamente en el arranque.

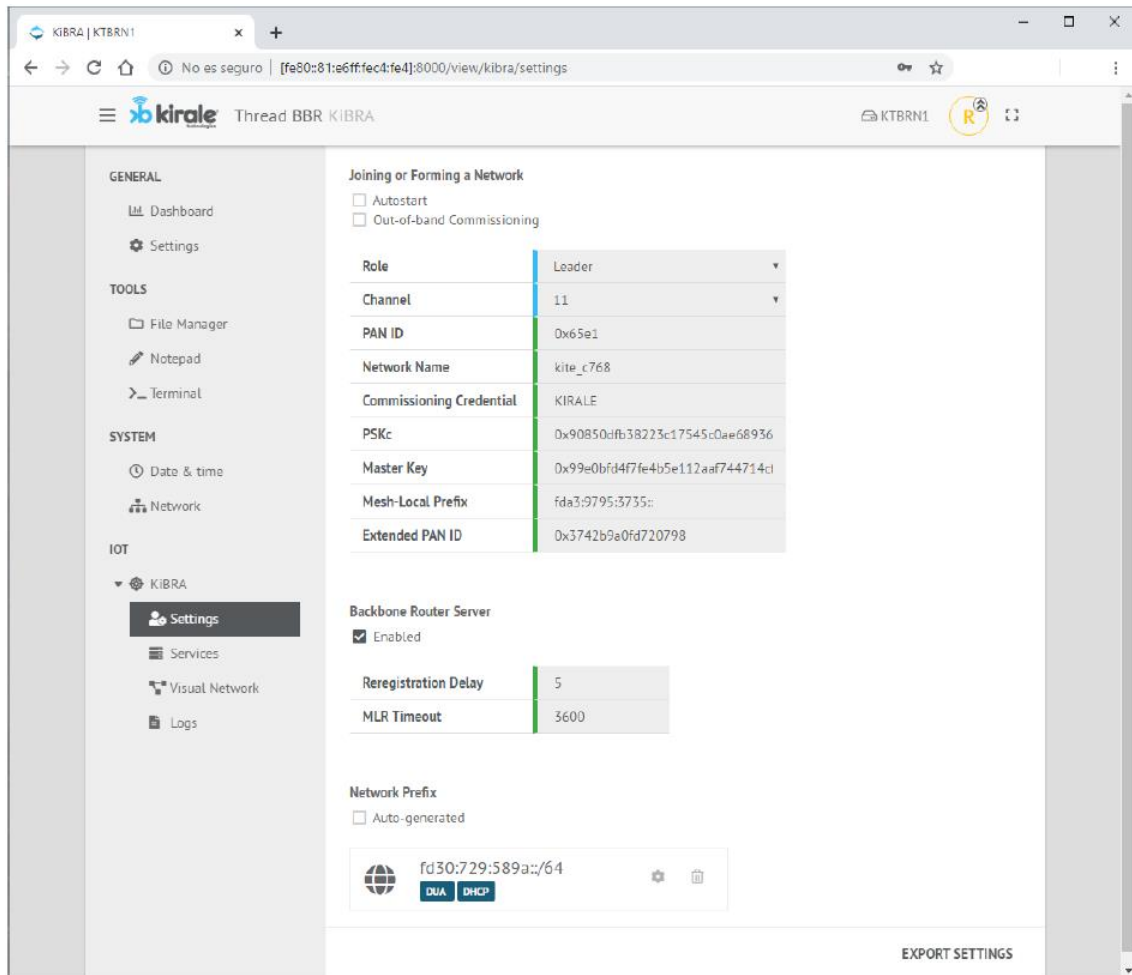


Ilustración 22 Pestaña Settings - Export Settings

En esta pestaña, es posible visualizar y copiar la información necesaria para la configuración de otros dispositivos y su posterior unión a la misma red.

Esto se podrá clicando en el botón situado en la esquina inferior derecha llamado “Export Settings”, abriéndose una ventana emergente.

En la nueva ventana, mostrada en Ilustración 23, permite elegir el rol y copiar la información de “commissioning”, requerida para la configuración del nuevo nodo e introducirlo en la red, utilizando comandos KiNOS.a través de la interfaz de KiTools proporcionada por Kirale.

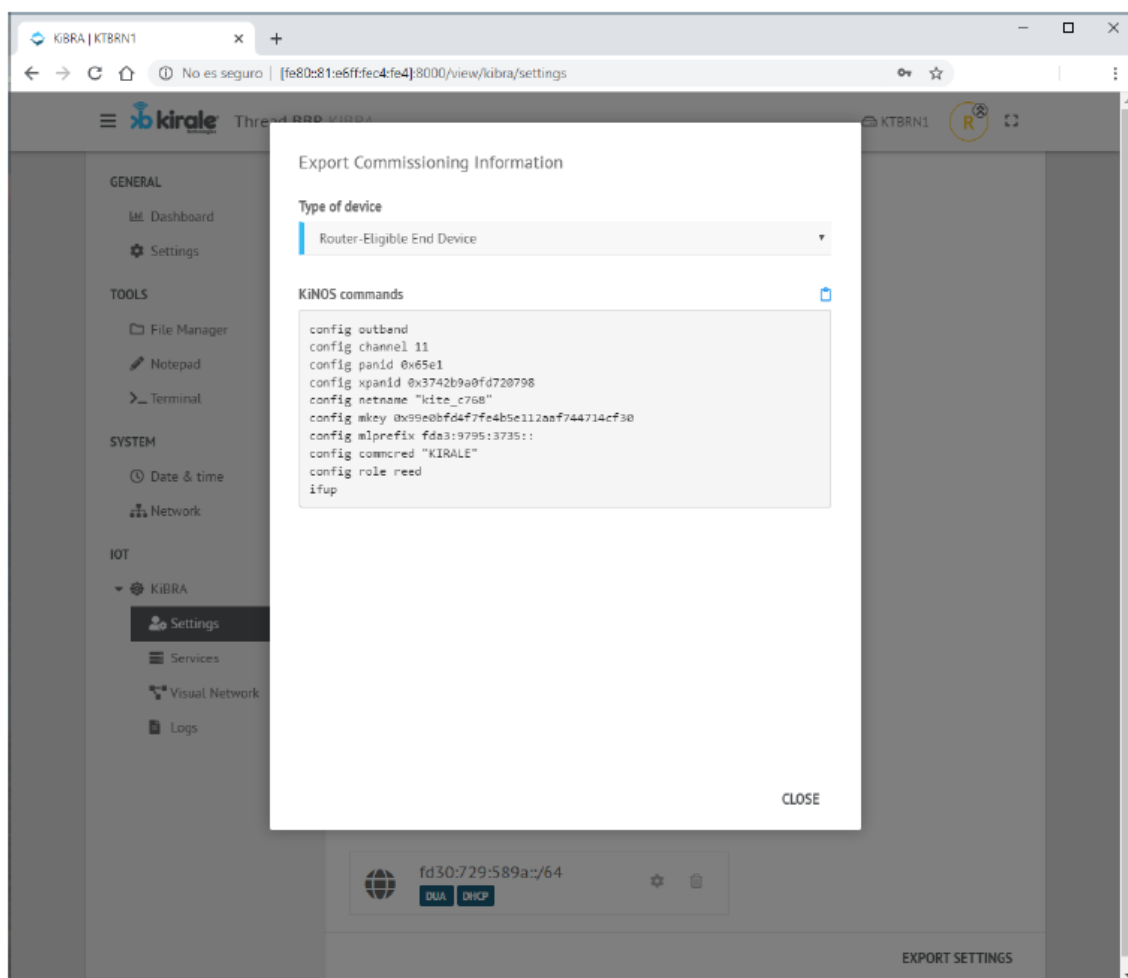


Ilustración 23 Ventana Export Commissioning Information

#### 3.2.1.3.5. SERVICIOS

El administrador de la red podrá ver qué servicios están siendo provistos por el Border Router en cada momento y su estado en el submenú “Services” por debajo del menú “KiBRA”. Hay cuatro posibles servicios que el Border Router es capaz de proveer.

##### 3.2.1.3.5.1. SERVIDOR BACKBONE ROUTER

Cuando la opción “Backbone Router Server” esté habilitada en el menú de “Settings”, los datos relacionados aparecerán en esta pestaña, tal como se muestra a continuación en Ilustración 24.

Activando este servicio, se podrán conectar diferentes redes locales Thread dentro de una gran red troncal.

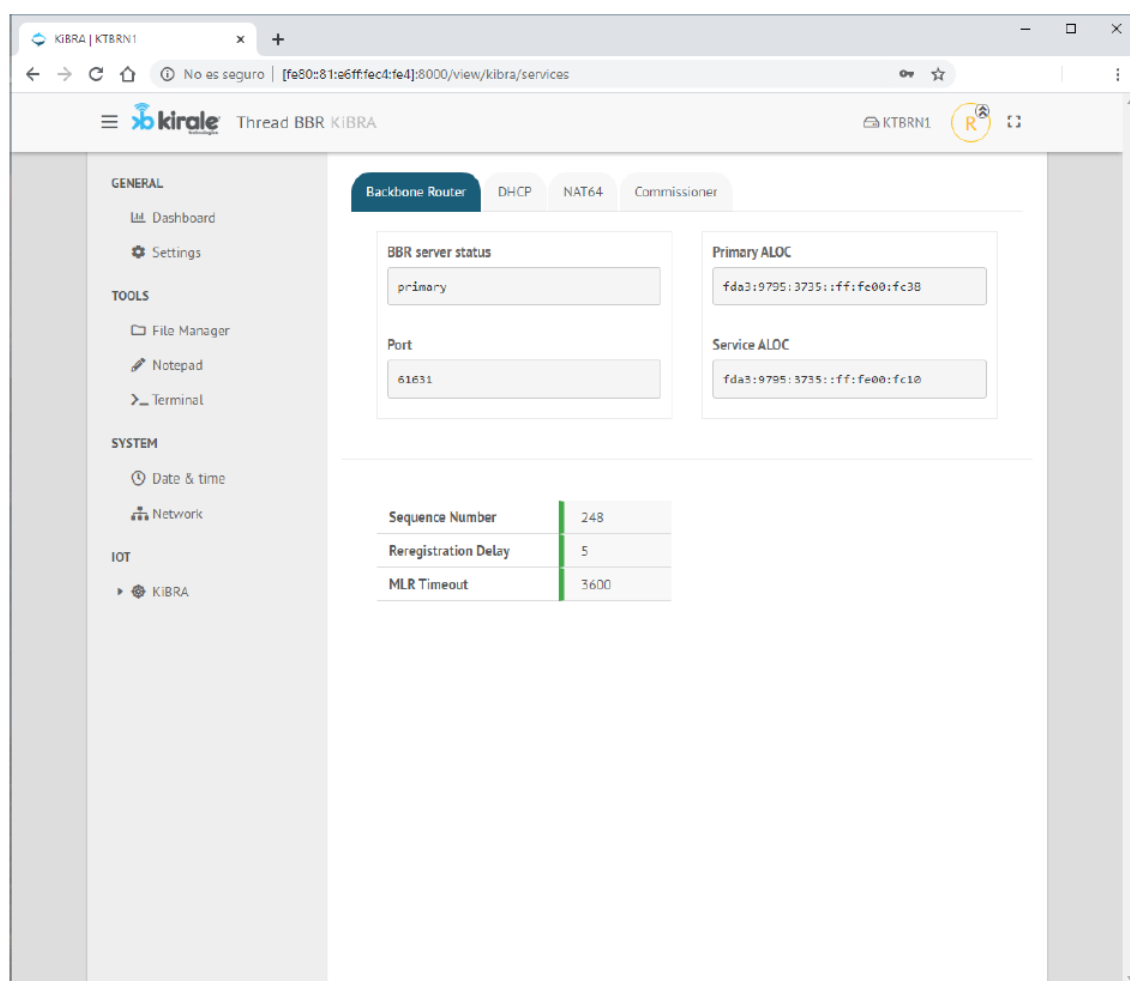


Ilustración 24 Servidor Backbone Router



### 3.2.1.3.5.2. DHCP

Si la opción DHCP se activa al configurar el prefijo de red, la siguiente página mostrará tanto la información del servidor DHCP como una lista de los nodos que han adquirido una dirección IPv6 vía DHCP y cuál es. Toda esta información se muestra con la disposición que se expone en Ilustración 25.

Esta pestaña será útil de cara a saber las direcciones IPv6 de los diferentes nodos pudiendo así realizar transmisiones de datos entre ellos. Esto se debe a que desde un propio nodo, no podrá consultarse las diferentes direcciones IP en la red salvo las propias del nodo, por lo que convendrá acceder a esta pestaña o configurar expresamente los nodos, tras estar conectados en la red, con una IPv6 en específico, conocida por el usuario, consiguiendo así conocer las direcciones IP que serán asignadas a los nodos antes de formar la red.

The screenshot shows the KIBRA web interface for a Thread BBR. The left sidebar contains navigation menus for GENERAL, TOOLS, SYSTEM, and IOT. The main content area is titled 'Thread BBR KIBRA' and has tabs for 'Backbone Router', 'DHCP', 'NAT64', and 'Commissioner'. The 'DHCP' tab is active, showing the following configuration:

- Server status:** on
- Service ALOC:** fda3:9795:3735::ff:fe00:fc01
- DNS server:** fda3:9795:3735::ff:fc00:4800
- NTP server:** fda3:9795:3735::ff:fe00:4800

Below the configuration is the 'DHCPv6 Lease Table' with the following data:

IPv6 address	RLOC16	MAC extended address
fd30:729:589a:0:204c:eceb:ba66:5f8b	local	02-17-8a-3b-f6-43-be-09

At the bottom of the table, there are navigation buttons: First, Previous, 1 (selected), Next, Last.

Ilustración 25 Servicio DHCP

#### 3.2.1.3.5.3. NAT64

El servicio NAT64 será el servicio encargado de realizar de interfaz entre la red Thread creada con IPv6 y la red IPv4. La dirección IPv4 que usará el servidor NAT64 del Border Router será la configurada previamente en 3.2.1.2.3.1 CONEXIÓN VÍA PUERTO USB SERIE . La tabla de la sesión NAT se mostrará en esta pestaña con la disposición indicada en Ilustración 26.

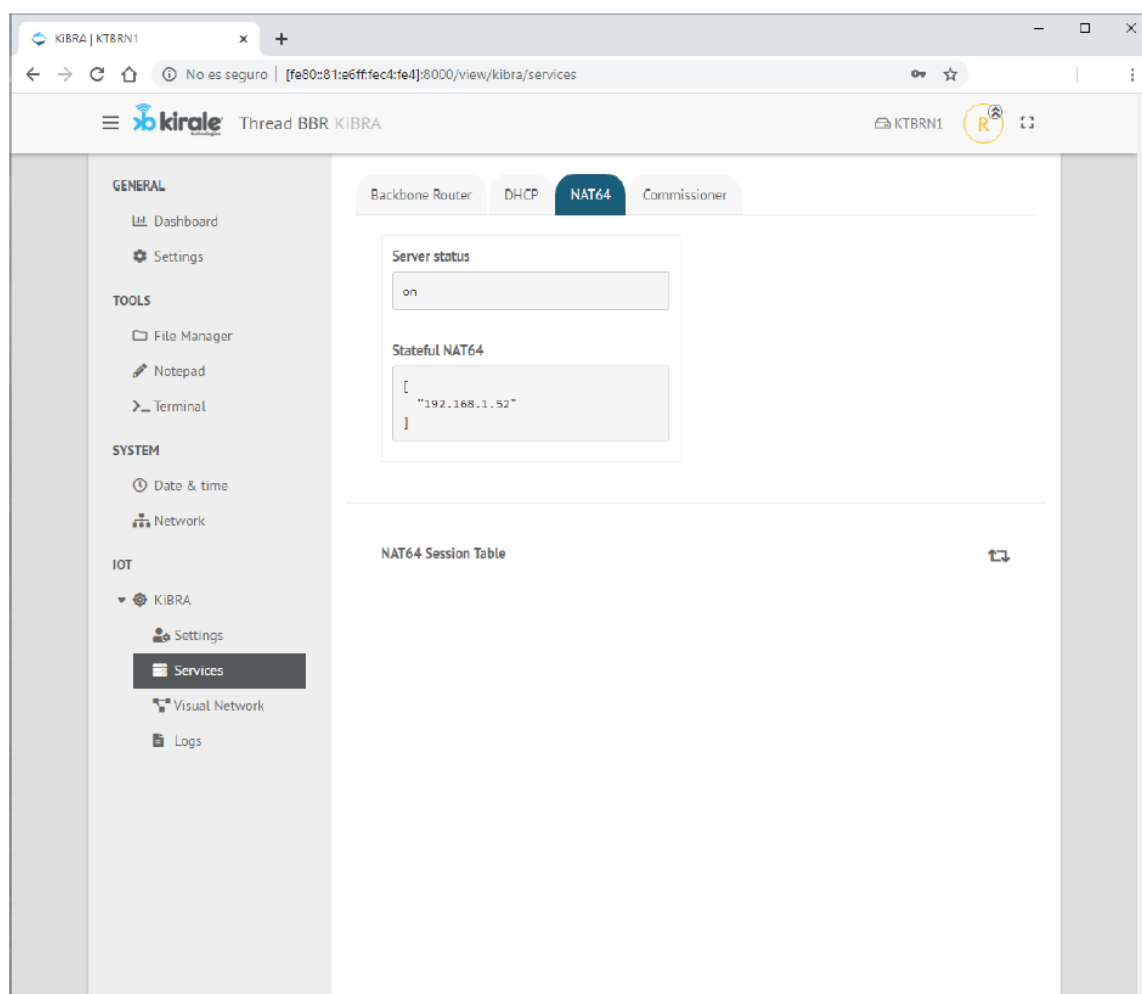


Ilustración 26 Servicio NAT64

### 3.2.1.3.5.4. COMMISSIONER

El Border Router puede hacer también de commissioner dentro de la red Thread. Una vez esta función está activada, el administrador de la red, puede activar la dirección de datos para permitir que se puedan unir nuevos dispositivos a la red, tal como se explica en 2.3.3.8 SEGURIDAD Y COMMISSIONING DE DISPOSITIVOS [10], [12].

En esta pestaña, también se podrá definir las configuraciones de commissioning, ya sea con Joining Credential, como es el caso mostrado en Ilustración 27, o con la dirección EUI-64 MAC del propio nodo y la Joiner Credential.

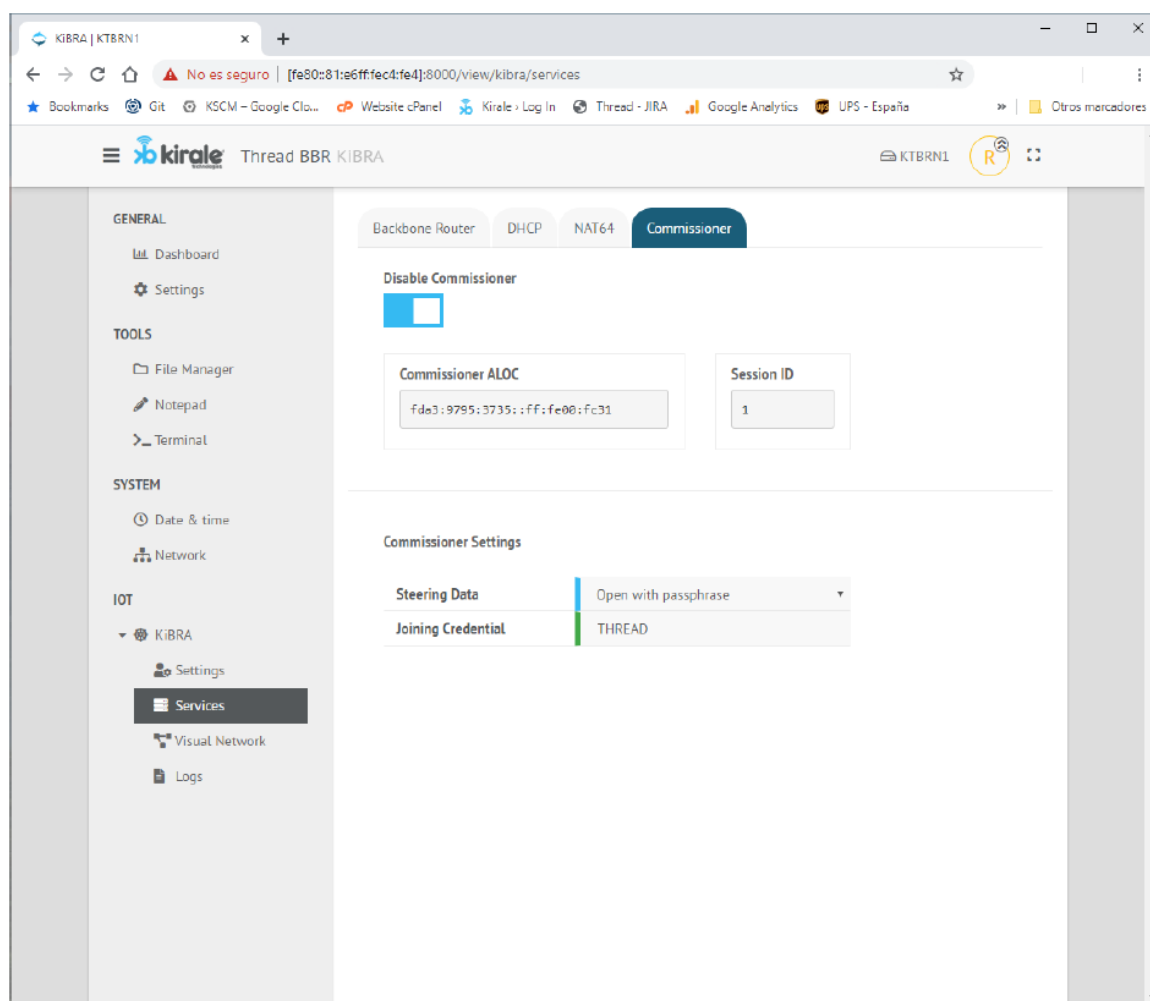


Ilustración 27 Servicio Commissioner

#### 3.2.1.3.6. VISUAL NETWORK

El mapa de topología de red es un mapa que permite al administrador de la red, o a los usuarios habilitados, ver el Layout de los dispositivos conectados. Este mapa con la topología de la red es muy útil para entender cómo se han conectado los dispositivos unos a otros y así entender las mejores técnicas para resolver los problemas que puedan surgir. En la Ilustración 28 se muestra un ejemplo de una topología de red que incluye todos los roles de nodos posibles.

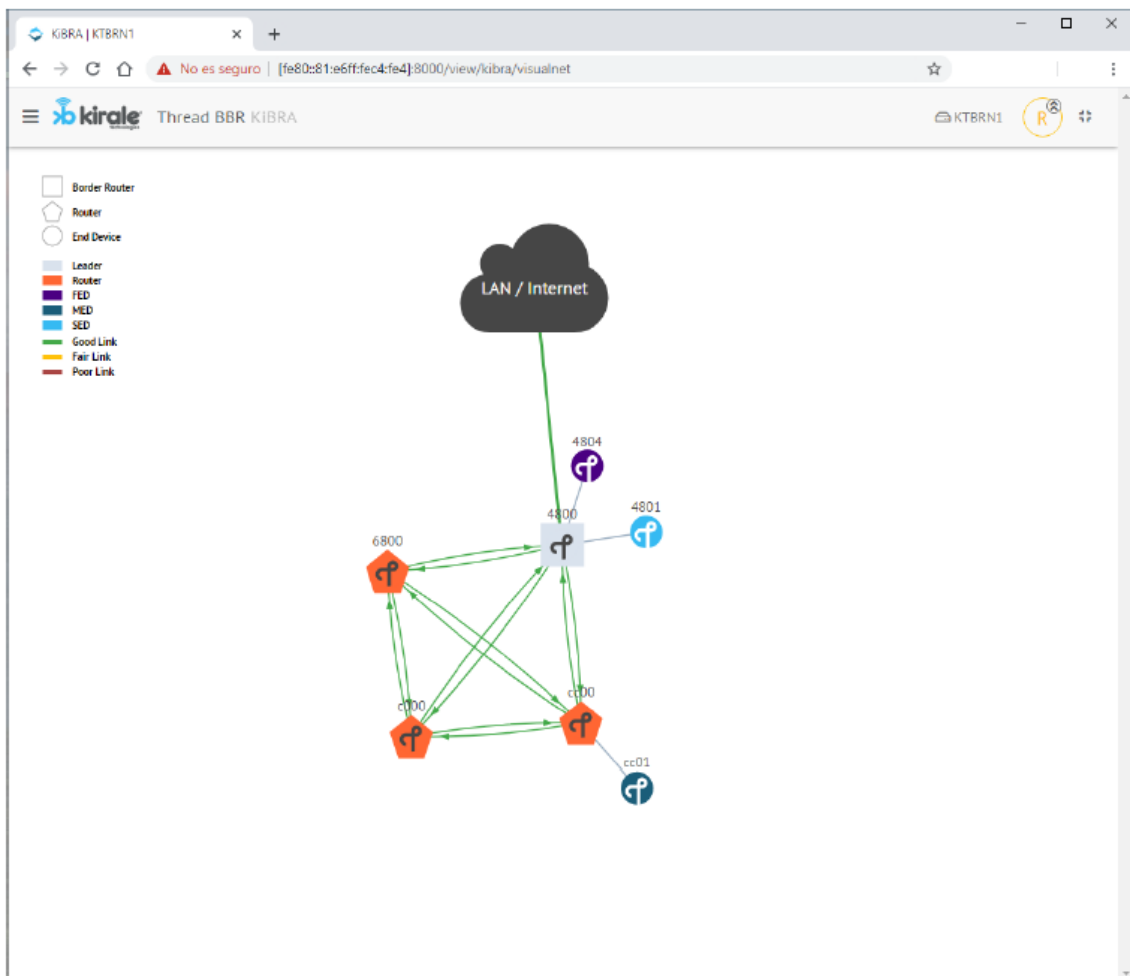
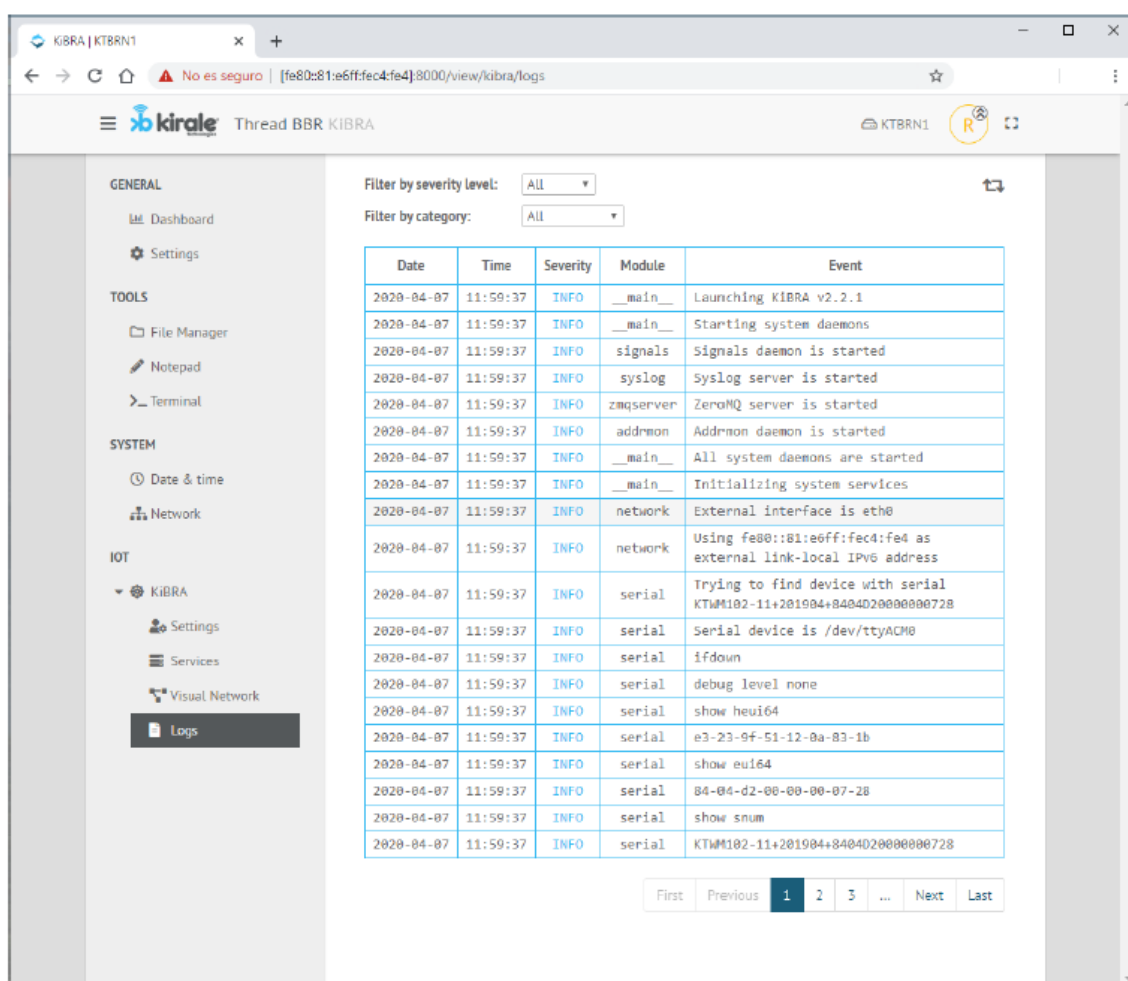


Ilustración 28 Pestaña Visual Network

### 3.2.1.3.7. LOGS

El administrador puede ver dentro de los logs del sistema en el submenú “Logs” por debajo del menú “KiBRA”. Se permite filtrar logs por nivel de gravedad/severidad y categoría. En esta pestaña es posible ver desde el inicio o parada de los distintos servicios del Border Router, hasta la dirección EUI-64 MAC de los nodos que se conectan o se desconectan, pudiendo ver así, analizar problemas mediante el análisis de los Logs.

En la Ilustración 29, se muestra los Logs del proceso de arranque de servicios del Border Router.



Date	Time	Severity	Module	Event
2020-04-07	11:59:37	INFO	__main__	Launching KIBRA v2.2.1
2020-04-07	11:59:37	INFO	__main__	Starting system daemons
2020-04-07	11:59:37	INFO	signals	Signals daemon is started
2020-04-07	11:59:37	INFO	syslog	Syslog server is started
2020-04-07	11:59:37	INFO	zmqserver	ZeroMQ server is started
2020-04-07	11:59:37	INFO	addrmon	Addrmon daemon is started
2020-04-07	11:59:37	INFO	__main__	All system daemons are started
2020-04-07	11:59:37	INFO	__main__	Initializing system services
2020-04-07	11:59:37	INFO	network	External interface is eth0
2020-04-07	11:59:37	INFO	network	Using fe80::81:e6ff:fe4:fe4 as external link-local IPv6 address
2020-04-07	11:59:37	INFO	serial	Trying to find device with serial KTW102-11+201904+8404020000000728
2020-04-07	11:59:37	INFO	serial	Serial device is /dev/ttyACM0
2020-04-07	11:59:37	INFO	serial	ifdown
2020-04-07	11:59:37	INFO	serial	debug level none
2020-04-07	11:59:37	INFO	serial	show heui64
2020-04-07	11:59:37	INFO	serial	e3-23-9f-51-12-0a-83-1b
2020-04-07	11:59:37	INFO	serial	show eui64
2020-04-07	11:59:37	INFO	serial	84-04-d2-00-00-00-07-28
2020-04-07	11:59:37	INFO	serial	show snum
2020-04-07	11:59:37	INFO	serial	KTW102-11+201904+8404020000000728

Ilustración 29 Pestaña Logs

#### *BREVE RESUMEN*

De cara a tener un mayor conocimiento sobre cómo funciona KTBRN1, a continuación se dará una detallada descripción del sistema, que herramientas hay disponibles y algunos consejos para solucionar problemas.

##### 3.2.1.4.

##### 3.2.1.4.1. SISTEMA DE FICHEROS AVANZADO

El dispositivo KTBRN1 es un sistema basado en Linux, el cual tiene la peculiaridad que su sistema de ficheros está corriendo desde una tarjeta SD. Esto supone un gran desafío a la hora de garantizar la fiabilidad y el rendimiento del sistema.

Las tarjetas SD son propensas a dañarse o corromperse, implicando a la pérdida de ficheros almacenados y otros datos. Además, tienen una vida útil, tiempo a partir del cual pueden dañarse.

Kirale Technologies ha diseñado un avanzado sistema de ficheros para superar estos inconvenientes de manera eficaz. Este diseño monta la partición de “solo lectura” y todos los ficheros que son escritos no son realmente escritos en el disco pero permanecen en la RAM. De esta manera el sistema de ficheros no se corromperá porque al reiniciarse vuelve a tener la imagen antigua. Por otro lado, una sincronización software escribirá estos ficheros que deben de permanecer actualizados y reflejar así los cambios después del reinicio.

##### 3.2.1.4.2. SERVICIOS CRÍTICOS

Hay dos servicios críticos corriendo en el dispositivo KTBRN1. Por un lado el servicio “kibra”, el cual se encarga de todas las funcionalidades de Border Router, mientras que por otro lado está el servicio de “ajenti”, el cual gestiona el Panel de Administración Web. Ambas son aplicaciones Python instaladas en entorno virtual.

Usando comandos comunes de Linux, el administrador podrá saber el estado de ambos servicios y reiniciarlos si es necesario. En Tabla 4 Servicios KBRNT1 se muestran los dos posibles servicios con sus respectivos argumentos.

```
root@KTBRN1:~# service kibra (status | start | stop | restart)
root@KTBRN1:~# service ajenti (status | start | stop | restart)
```

*Tabla 4 Servicios KBRNT1*

El administrador puede iniciar manualmente la aplicación “kibra” usando los comandos indicados a continuación en Tabla 5:

```
root@KTBRN1:~# service kibra stop
root@KTBRN1:~# source /opt/kirale/py3env/bin/activate
(py3env) root@KTBRN1:~# python -m kibra—log debug
```

*Tabla 5 Comandos Servicio KiBRA*

En el caso de la aplicación “ajenti”, los comandos a utilizar son los indicados a continuación en Tabla 6:

```
root@KTBRN1:~# service ajenti stop
root@KTBRN1:~# source /opt/kirale/py2env/bin/activate
(py2env) root@KTBRN1:~# ajenti-panel —dev
```

*Tabla 6 Servicio Ajenti*

#### 3.2.1.4.3. COMUNICACIÓN ENTRE PROCESOS

El Panel de Administración Web y KiBRA se comunican constantemente entre ellos a través de un puerto local TCP.

#### 3.2.2. CONFIGURACIÓN INICIAL MÓDULO KTWM102

En esta sección se abordará la configuración los módulos KTWM102, tanto la configuración para poder acceder a él desde el ordenador como los parámetros a configurar para poder conectar el dispositivo a una red.

**Nota:** Este procedimiento es para los dispositivos KTDG102 Evaluation Dongles, los cuales añaden al KTWM102 el circuito necesario para poder conectarlos a un ordenador vía USB. En caso de usar un módulo KTWM102, se deberá diseñar un circuito con conector USB para poder conectarlo. Una vez realizado el circuito, seguir el mismo procedimiento explicado a continuación.

#### INSTALACIÓN DE DRIVERS USB Y DEL BOOTLOADER

3.2.2.1. Para la instalación de los drivers necesarios, se comenzará por instalar los drivers del Bootloader. Se conectará el dispositivo a un puerto USB disponible del ordenador. Si aún no hay una imagen firmware valida grabada en el dispositivo, el Led del Dongle, empezará a parpadear rápidamente. Esto indica que el dispositivo ha entrado en modo DFU y está esperando una actualización de firmware. El administrador de dispositivos mostrará el dispositivo como “KiNOS Boot DFU” en la pestaña de “Otros dispositivos”, como se muestra en Ilustración 30.

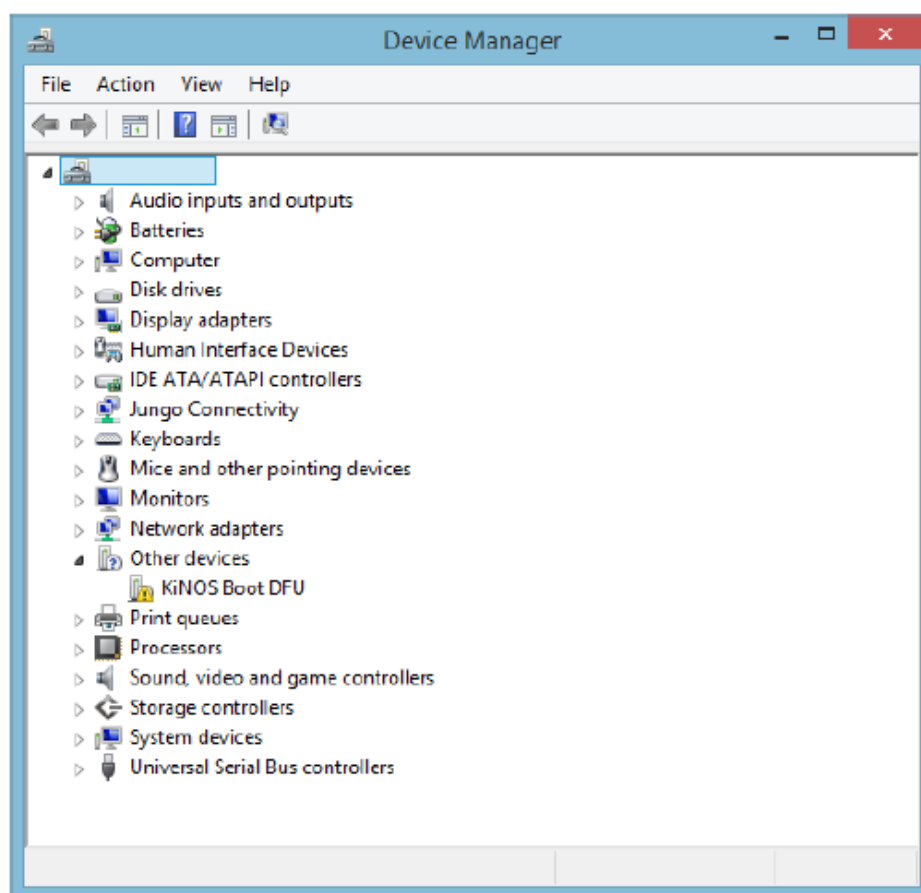


Ilustración 30 Administrador de dispositivos antes de instalar Drivers de KTWM102



### 3.2.2.1.1. WINDOWS

Los sistemas Windows requieren instalar manualmente los drivers USB para los KTDG102 (solo será necesario la primera vez). En algunos casos, el dispositivo KiNOS Boot DFU puede instalarse automáticamente con los drivers genéricos de Windows, pero se necesitará reemplazarlos.

Para instalar o reemplazar los drivers, se necesitará la herramienta **Zadig**, previamente descargada. Esta aplicación es para instalar una “libusb” compatible con el dispositivo.

Abrir Zadig (no necesita de instalación). En caso de que salte el aviso de una ventana UAC (User Account Control), seleccionar “Yes” o “Sí”. Este aviso es mostrado en Ilustración 31.

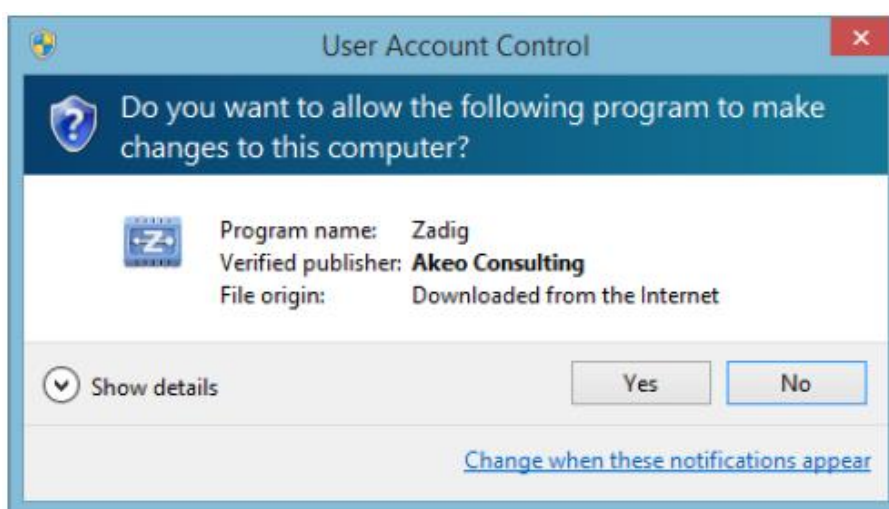


Ilustración 31 Instalación Drivers con Zadig Paso 1

Una vez esté Zadig corriendo, deberán aparecer las interfaces KiNOS en la lista desplegable. Es posible conectar el dispositivo incluso después de haber abierto Zadig, la lista se actualizará automáticamente. En caso de que no aparezca, probablemente sea que ya haya algún driver instalado. Para verlo, ir al menú de “**Options**” y seleccionar “**List All Devices**”.

Seleccionar KiNOS Boot DFU en la lista desplegable, y seleccionar el driver “libusbK”, mostrándose la información como se ilustra en Ilustración 32 y clicar en “**Install / Replace Driver**”.

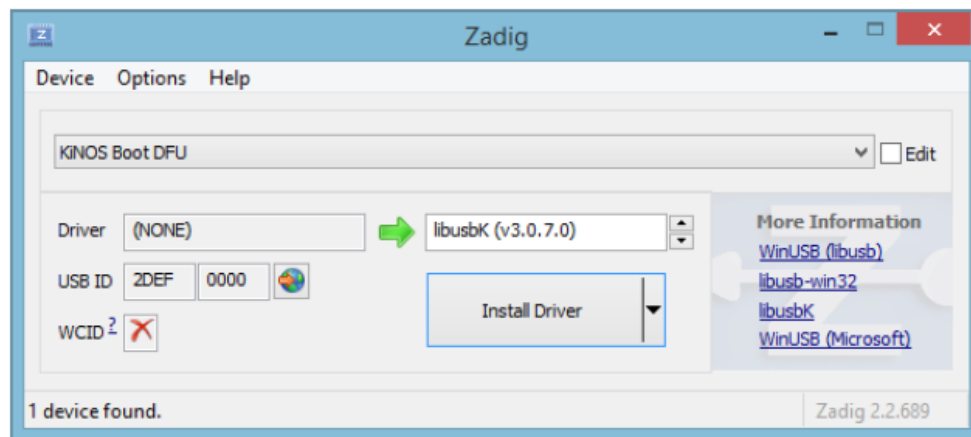


Ilustración 32 Instalación Drivers con Zadig Paso 2

El proceso tomará alrededor de un segundo y el resultado será un mensaje de éxito como el indicado en la Ilustración 33, teniendo ya el primer driver instalado.

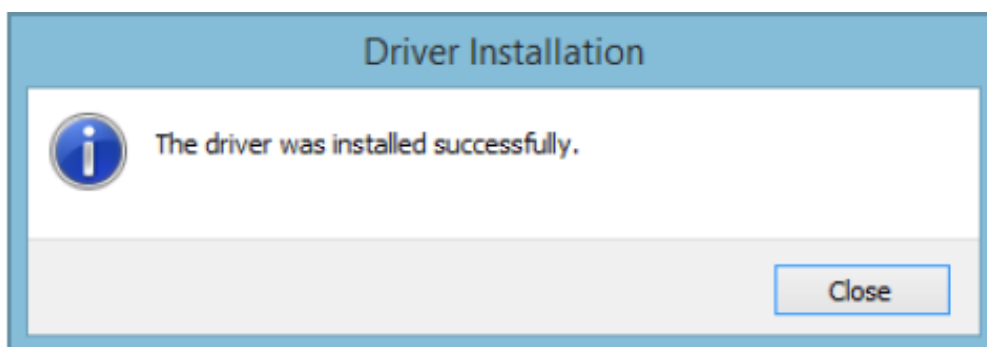
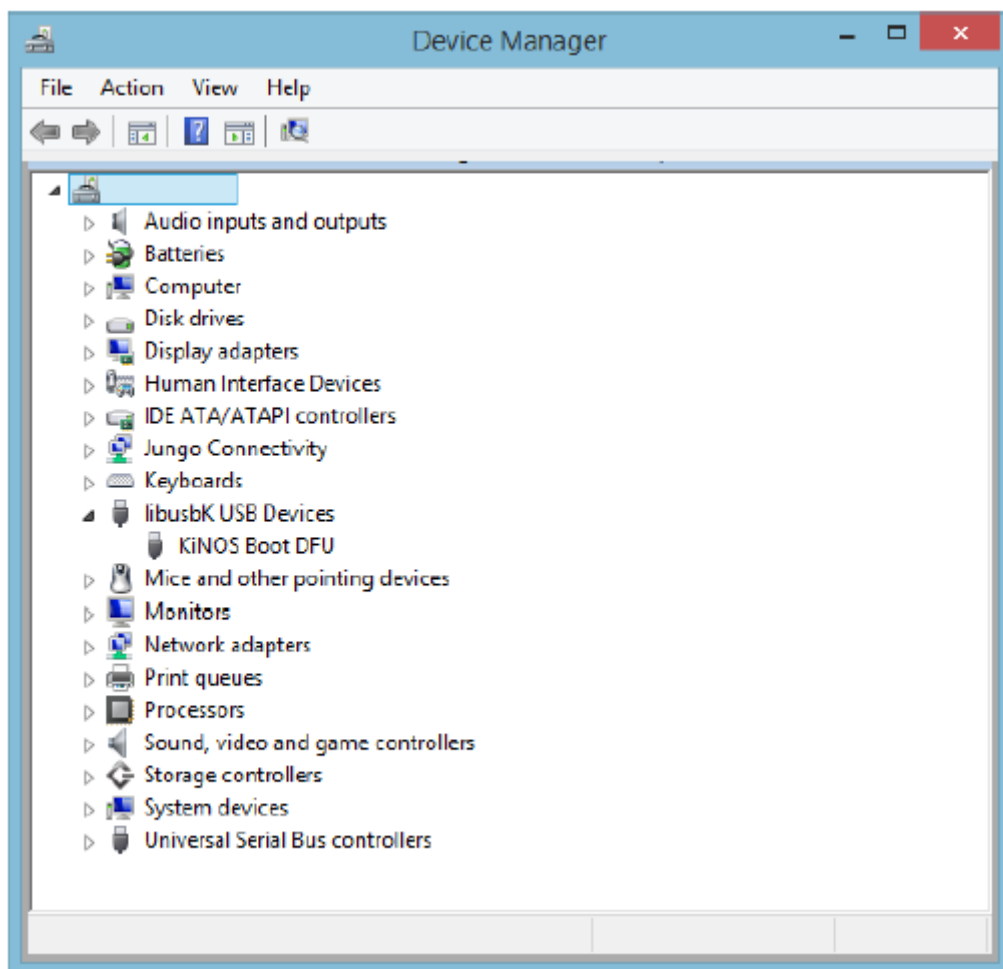


Ilustración 33 Instalación Drivers con Zadig Finalizada

Ahora, en el *Administrador de Dispositivos*, el KiNOS Boot DFU deberá salir como se muestra en la Ilustración 34:



*Ilustración 34 Administrador de Dispositivos Después de Instalar Drivers*

#### 3.2.2.1.2. LINUX / MAC OS

En Linux o MAC OS, no se necesita instalación de ningún driver específico para sistemas basados en Linux con versión de Kernel superior a 2.6.22 ni para sistemas MAC OS X desde la versión 10.4 (Tiger).

#### *INSTALL “DFU-UTIL”*

El protocolo usado para cargar el Firmware a los dispositivos Kirale KTDG USB Dongle a través de interfaz USB es el estándar **DFU 1.1**. Descargar “dfu-util” e instalarlo en caso de requerirlo.

3.2.2.a) En Windos: descargar de <http://dfu-util.sourceforge.net/> y extraer el .zip descargado en la carpeta deseada. No requiere instalación.

b) En MAC OS:

```
$ brew install dfu-util
```

(Get Brew)

c) En Linux:

```
$ sudo apt install dfu-util
```

#### *ACTUALIZACIÓN DE FIRMWARE*

3.2.2.3.

Abrir dfu-util desde la ventana de comandos (o Windows PowerShell en caso de PC con sistema Windows) y listar, con el comando mostrado en Tabla 7, en los dispositivos conectados para encontrar el dispositivo deseado. El USB Product ID para un dispositivo KiNOS DFU en modo bootloader es **0000**.

```
$ dfu-util—list
```

```
Found DFU: [2def:0000] ver=0100, devnum=8, cfg=1, intf=0, path="1-1.4.3", alt=0,
name="KiNOS DFU", serial="8404D2000000045B"
```

```
Found Runtime: [2def:0102] ver=0100, devnum=9, cfg=1, intf=0, path="1-1.4.4", alt=0,
name="KiNOS DFU", serial="8404D2000000045C"
```

*Tabla 7 Listar dispositivos con dfu-util*

De los diferentes dispositivos listados, ya sea en modo Bootloader o en modo Runtrime (el USB Product ID es 0102), flashear el fichero del firmware al dispositivo deseado (especificando el número de serie) con el comando mostrado en Tabla 8. Esta transferencia del archivo puede tardar varios segundos.

```
$ dfu-util—download      KiNOS-GEN-KTWM102-1.1.6533.62822.dfu—serial
8404D20000000045B
Match vendor ID from file: 2def
Match product ID from file: 0000
Opening DFU capable USB device...
ID 2def:0000
Run-time device DFU version 0110
Claiming USB DFU Interface...
Setting Alternate Setting #0 ...
Determining device status: state = dfuIDLE, status = 0
dfuIDLE, continuing
DFU mode device DFU version 0110
Device returned transfer size 64
Copying data from PC to DFU device
Download [=====] 100% 245628 bytes
Download done.
state(6) = dfuMANIFEST-SYNC, status(0) = No error condition is present
unable to read DFU status after completion
```

*Tabla 8 Actualizar Firmware en dispositivos KTWM102*

Una vez dfu-util ha terminado la transferencia del firmware, el KTDG USB Dongle se reiniciará y empezará a aplicar el nuevo firmware en la memoria flash interna (parpadeo rápido del led). Esto puede tardar varios segundos. Cuando el led empiece a parpadear lentamente, de manera estable, el flasheo del firmware ha terminado y el firmware KiNOS empieza a operar en el modo de runtime.

El fichero DFU puede conseguirse desde el siguiente link:

<https://www.kirale.com/support/#downloads>

#### RUNTIME – INSTALACIÓN DE DRIVERS USB

En modo run-time el KTDG102 Dongle es un dispositivo USB Compuesto que combina tres tipos de interfaces USB:

- Device Firmware Upgrade (DFU).
- 3.2.2.4• Virtual Serial (CDC – ACM).
- Ethernet over USB (CDC-ECM).

Windows no soporta el modelo USB-ECM de manera nativa, por lo que se requiere un driver de terceros que está fuera del alcance de Kirale Technologies.

Para los otros dos interfaces USB, se necesitará la instalación de los drivers para sistemas Windows siguiendo las siguientes instrucciones:

##### 3.2.2.4.1. WINDOWS

En algunos casos el dispositivo KiNOS DFU puede instalar automáticamente un driver genérico de Windows y aparecerá por debajo de “Virtual COM Ports”. En caso de haberse instalado el driver genérico, se necesitará reemplazar. Se usará Zadig para instalar o reemplazar los drivers USB.

Seleccionar *KiNOS DFU (Interface 0)* en la lista desplegable, seleccionar el driver “*libusbK*” y seleccionar “*Install/Replace Driver*”, mostrándose la información como en la Ilustración 35.

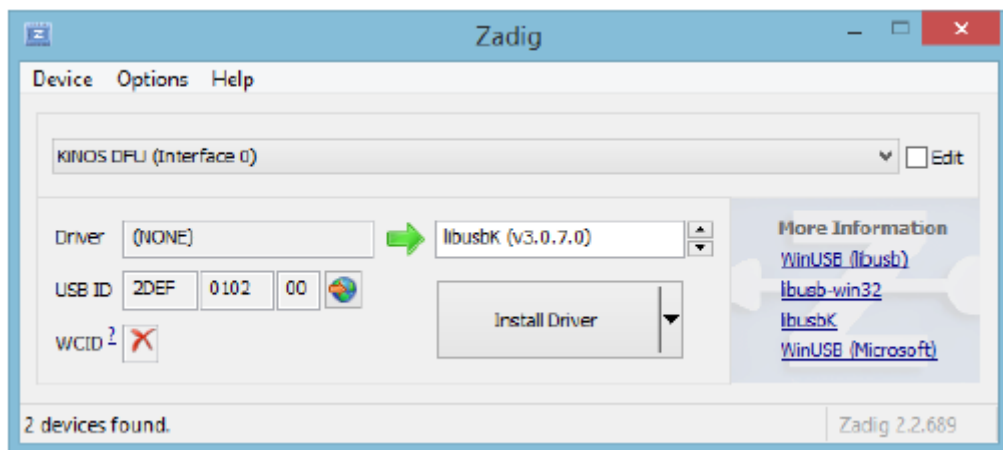


Ilustración 35 Instalar libusbK con Zadig - Runtime

El proceso tardará alrededor de un segundo y saldrá el mensaje de éxito mostrado en Ilustración 36.

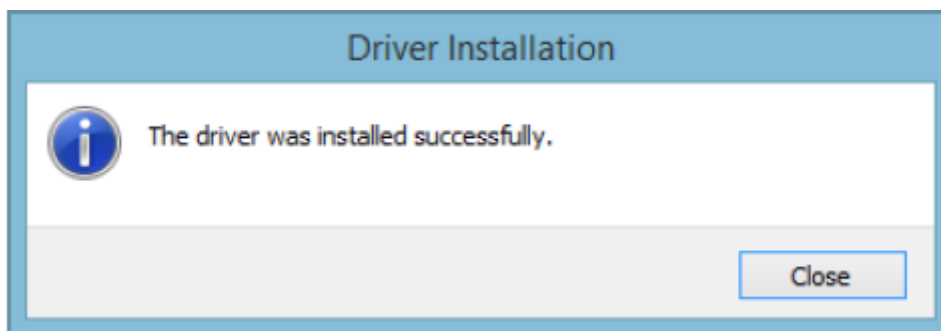


Ilustración 36 Instalación Driver Libusbk con Zadig Finalizada

Después, se selecciona *KiNOS Virtual COM (Interface 1)* en la lista desplegable, seleccionar el driver “**USB Serial (CDC)**”, tal como se muestra en Ilustración 37 y clicar en “*Install/Replace Driver*”.

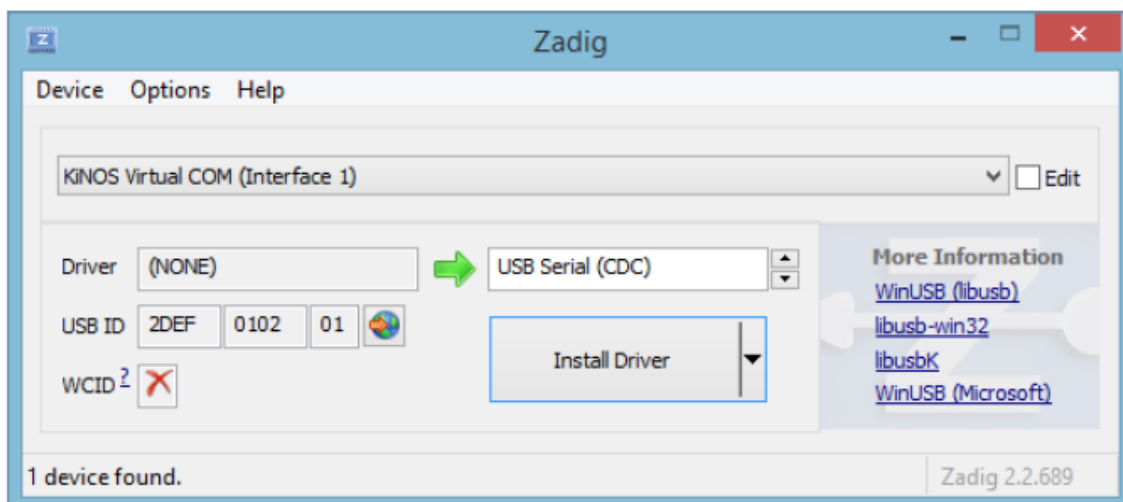


Ilustración 37 Instalar USB SERIAL (CDC) con Zadig

El proceso tardará alrededor de un segundo y saldrá el mensaje de éxito mostrado en Ilustración 38.

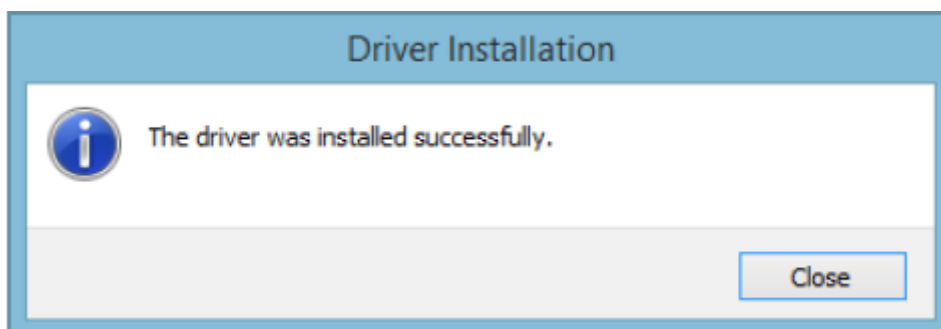
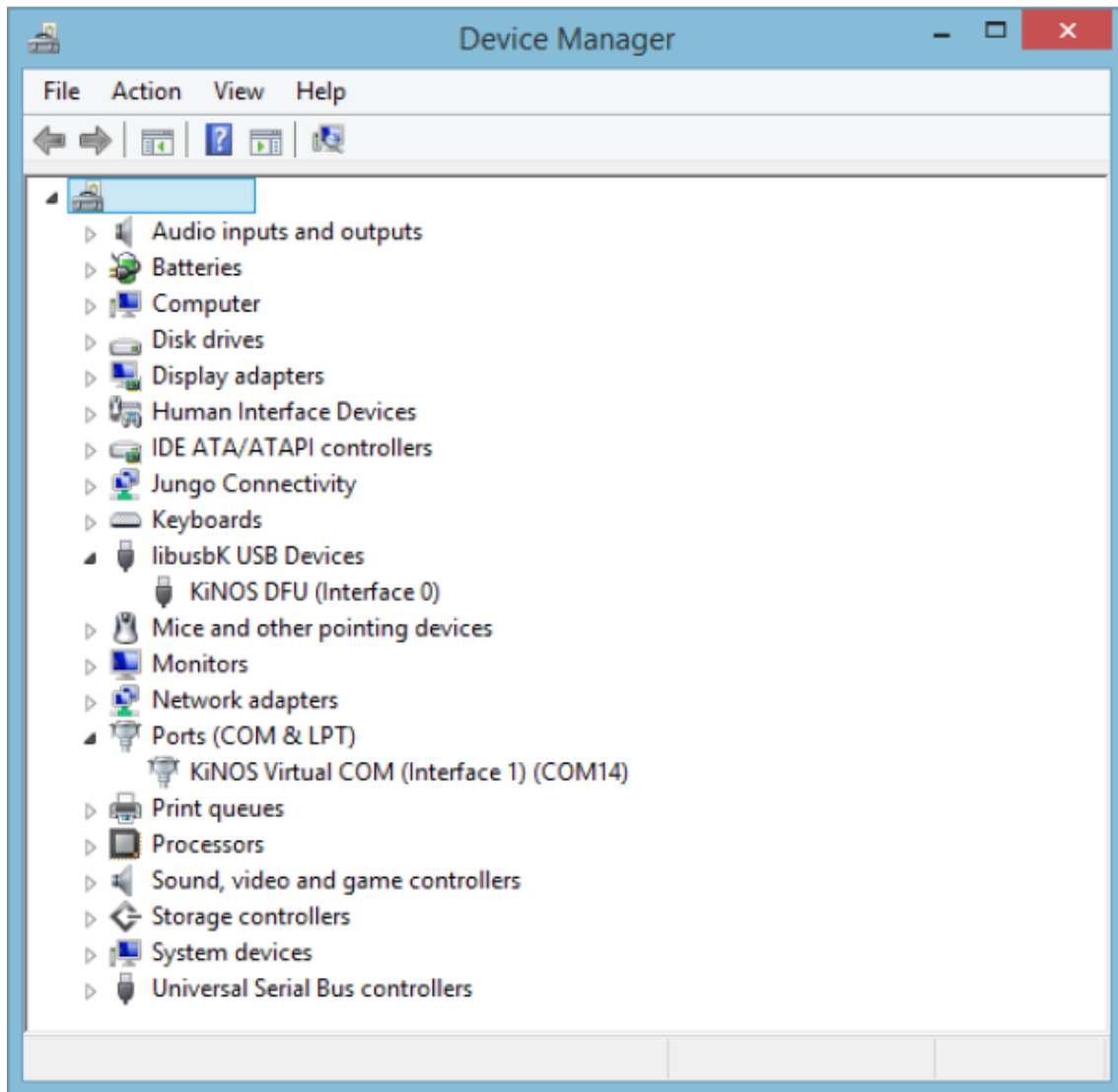


Ilustración 38 Instalación Driver USB Serial (CDC) con Zadig Finalizada

Ahora, en *Administrador de Dispositivos*, la interfaz KiNOS debería aparecer nombrada como se muestra en la Ilustración 39.



*Ilustración 39 Administrador de Dispositivos después de la instalación*

#### 3.2.2.4.2. LINUX

No se necesita ninguna instalación de un driver en específico para sistemas basados en Linux con versiones de Kernel superiores a 2.6.

**Nota:** La interfaz USB CDC-ECM está deshabilitada por defecto. Puede habilitarse por línea de comandos. Para más información detallada, mirar la **KSH Reference Guide**.



Para encontrar Puertos de Serie e interfaces Ethernet, conectar el Dongle a un USB del ordenador y después abrir una terminal e introducir el comando indicado a continuación en Tabla 9:

```
~$ dmesg | tail
```

Tabla 9 Comando para listar Puertos Serie en Sistemas Linux

### 3.2.2.4.3. MAC OS

No se necesita ninguna instalación de un driver en específico para sistemas Mac OS X desde la versión 10.4 (Tiger).

**Nota:** La interfaz USB CDC-ECM está deshabilitada por defecto. Puede habilitarse por línea de comandos. Para más información detallada, mirar la **KSH Reference Guide**.

Para encontrar Puertos de Serie e interfaces Ethernet, conectar el Dongle a un USB del ordenador y después abrir una terminal e introducir el comando mostrado a continuación en Tabla 10:

```
~$ networksetup -listallhardwareports
```

3.2.2.5.

Tabla 10 Comando para listar Puertos Serie en Sistemas en MAC OS

## CONFIGURACIÓN DE TERMINAL COM

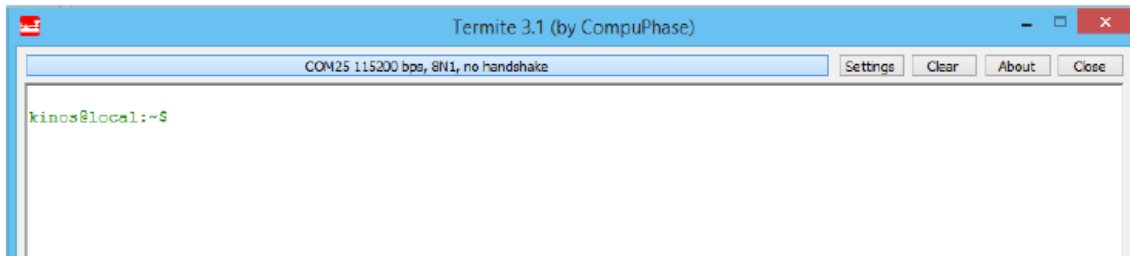
### 3.2.2.5.1. WINDOWS

Una vez instalado los drivers, podrá accederse tanto a módulos KTDG102 como KTWM102 desde una terminal Serie. Actualmente hay gran variedad de terminales serie COM disponibles para sistemas Windows. En esta guía se usa una genérica llamada "Termite", pero es posible utilizar cualquier otra terminal configurándose de igual manera.

Abrir la terminal serie y configurar con las siguientes configuraciones para comunicaciones serie con dispositivos de Kirale: 115200 bauds, 8 bits, 1stop bit, no parity.

**Nota:** La terminal USB serie, debe configurarse para añadir un carácter "CR" cuando se pulsa la tecla Enter.

Una vez configurado, seleccionar el puerto serie que aparece en el *Administrador de dispositivos* para el KiNOS Virtual COM. Después, para testear si la interfaz de USB de Kirale está activa y corriendo, presionar “Enter” y deberá devolverse el indicador de KiNOS, tal y como se muestra en Ilustración 40.



*Ilustración 40 Terminal Termit*

**Nota:** Recomendado la activación del Echo Local para la lectura de los comandos enviados.

#### 3.2.2.5.2. LINUX / MAC OS

Si no se tiene ninguna terminal serie para puerto serie COM, se deberá instalar una, como puede ser “Picocom”.

La configuración es la misma necesaria para la comunicación serie con los Sistemas Windows. Para abrir la terminal de Picocom, ejecutar el comando indicado a continuación en Tabla 11:

```
~$ picocom -c—omap lfcr /dev/ttyACM0
```

*Tabla 11 Abrir terminal Picocom en Linux / MAC Os*

c → Activación Local Echo.

omap lfcr → Asignar el avance de línea de salida al retorno de carro

/dev/ttyACM0 → El dispositivo serie asignado por Linux. Usar **dmesg** | **tail** cuando se haya conectado el dispositivo para comprobar el nombre del dispositivo.

Para irse del programa utilizar **Ctrl+a** y **Ctrl+x**

### 3.3. CONFIGURACIÓN DE RED PARA KTWM102

A la hora de enviar comandos tanto para configurar los parámetros del módulo KTWM102 como para ver los parámetros ya configurados se disponen de dos posibles vías para ello.

- 1) La primera vía es comandos KSH, los cuáles son con una sintaxis amigable para una ejecución manual y sencilla para el usuario desde un PC. Se conecta el módulo por USB.
- 2) La segunda vía es por comandos KBI. Estos comandos se enviarán por vía UART al módulo desde microcontroladores u otros sistemas similares.

#### 3.3.1. KIRALE COMMAND-LINE SHELL– COMANDOS KSH

##### SINTAXIS DE LOS COMANDOS

Los comandos KSH tienen una sintaxis amigable para las personas. Esta sintaxis está basada en palabras ASCII separadas, las cuales especifican cada una las palabras clave o parámetros del comando. Se representan como se indica a continuación en Tabla 12:

```
kinos@local:~$ command < arg > < key > < subkey > [ param 1 ] ... [ param N ]
```

Tabla 12 Sintaxis comandos KSH

Donde **Command** es el nombre del comando a ejecutarse de primer nivel. Este puede tener un argumento a ejecutar a bajo nivel e incluso claves y subclaves y otros parámetros que sean datos variables requeridos para la correcta ejecución del comando.

Para el envío de estos comandos se necesitará descargar la herramienta KiTools a su última versión de la siguiente dirección:

<https://www.kirale.com/support/#downloads>

Una vez descargado, conectar el módulo KTWM102 al PC por conexión USB abrir el ejecutable descargado y se abrirá una ventana de comandos similar a la de Windows, tal como se muestra a continuación en Ilustración 41 Herramienta KiTools.

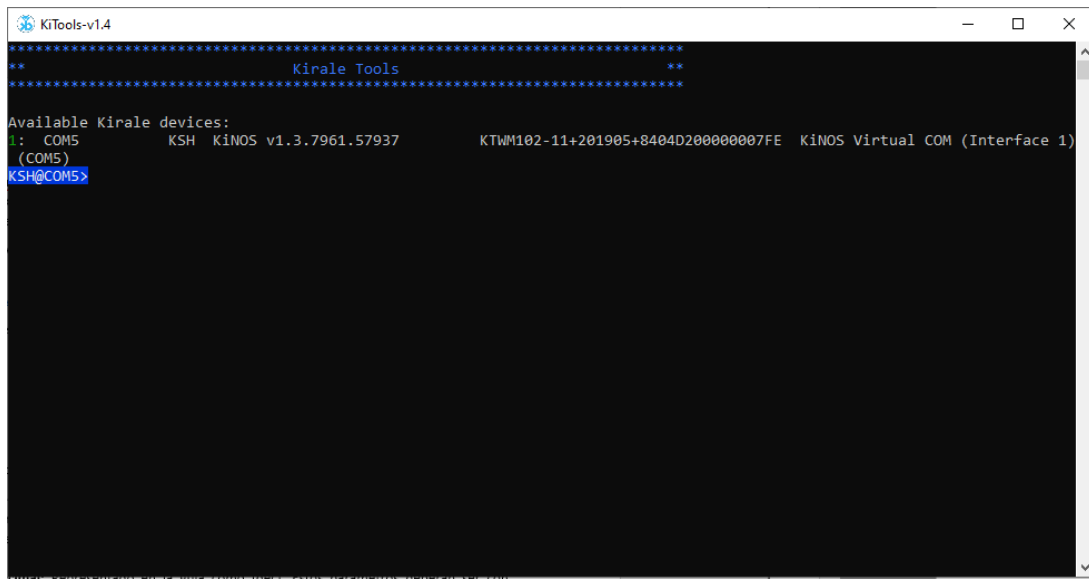


Ilustración 41 Herramienta KiTools

Para ver más en detalle la sintaxis de cada comando, descargar y ver la [KSH Reference Guide](#).

#### 3.3.1.2. SINTAXIS DE LOS PARÁMETROS

Hay seis tipos de formatos posibles de parámetros:

- **Hexadecimal:** Representado en la guía como [hex]. Estos parámetros deberán ser completamente en notación hexadecimal, como “0xABCD” o “0xabcd”, siempre con el prefijo “0x”.
- **Decimal:** Representado en la guía como [dec]. Estos parámetros deberán ser con notación decimal, como “64”.
- **Cadena de caracteres:** Representado en la guía como [str]. Estos parámetros deberán ir entre comillas dobles y pueden estar limitados en tamaño. Este límite de tamaño está especificado como número de caracteres admitidos.
- **Dirección IP:** Este parámetro se usa para especificar direcciones IPv6, rutas, o prefijos y debe ir con el formato especificado en la **RFC 4291 section 2** (16 bytes). En la se muestran algunas recomendaciones a la hora de usar direcciones IP como parámetros de los comandos KSH.

1. The preferred form is x:x:x:x:x:x, where the 'x's are one to four hexadecimal digits of the eight 16-bit pieces of the address.

Examples:

```
ABCD:EF01:2345:6789:ABCD:EF01:2345:6789
2001:DB8:0:0:8:800:200C:417A
```

2. The compressed form, where the use of "::" indicates one or more groups of 16 bits of zeros. The "::" can only appear once in an address. The "::" can also be used to compress leading or trailing zeros in an address.

For example, the following addresses:

```
2001:DB8:0:0:8:800:200C:417A
FF01:0:0:0:0:0:0:101
```

may be represented as:

```
2001:DB8::8:800:200C:417A
FF01::101
```

3. An alternative form that is sometimes more convenient when dealing with a mixed environment of IPv4 and IPv6 nodes is x:x:x:x:d.d.d.d, where the 'x's are the hexadecimal values of the six high-order 16-bit pieces of the address, and the 'd's are the decimal values of the four low-order 8-bit pieces of the address (standard IPv4 representation).

Example:

```
0:0:0:0:0:0:13.1.68.3
```

*Ilustración 42 Direcciones IP como parámetros*

- **Direcciones MAC:** Representado en la guía como [mac]. Este parámetro se usa para especificar direcciones de 64-bits.
- **Cadena / Array:** Se representa en la guía como [arr]. Este parámetro se usa para especificar payloads que deben enviarse sin ningún cambio. Están hechos de cadena de bytes en notación hexadecimal sin el prefijo "0x".

### 3.3.1.3.

#### MENSAJES DE RESPUESTA

Cuando se ejecuta un comando en la Kirale Command-Line Shell, se pueden reportar algunos errores como respuesta a su ejecución.

- 1) **Invalid Syntax:** Tanto el comando como el argumento son reconocidos pero alguna de las palabras claves o subclaves son erróneas, o hay más parámetros de lo esperado para ese comando.
- 2) **Command not found:** El comando o el argumento no existe.
- 3) **Bad parameter:** Alguno de los parámetros que se han introducido es erróneo o su valor está fuera de rango.
- 4) **Command not allowed:** El comando no puede ejecutarse en ese momento. Quizás se requiera que el dispositivo esté en un específico estado u otra configuración no permita la ejecución del comando.
- 5) **Configuration setting missing:** El comando no puede ejecutarse debido a que se requieren otras configuraciones antes de ejecutarse.
- 6) **Processing – please wait:** El dispositivo está ejecutando otro proceso de mayor prioridad. Solo se podrá usar el comando "**show status**". Otros comandos no podrán usarse hasta que el proceso prioritario termine.

Si no hay mensaje de respuesta, significa que el comando se ha procesado correctamente.

## 3.3.2. KIRALE BINARY INTERFACE – COMANDOS KBI

## OPERACIÓN DE INTERFAZ

Antes de entrar en el funcionamiento de cómo usar los comandos KBI por puerto UART, a continuación, en Ilustración 43, se muestra un esquema de un Host externo, como puede ser un microcontrolador, el cuál incluya una codificación y decodificación COBS usando comandos KBI para comunicarse con el módulo KTWM102 y el sistema KiNOS a través del puerto UART:

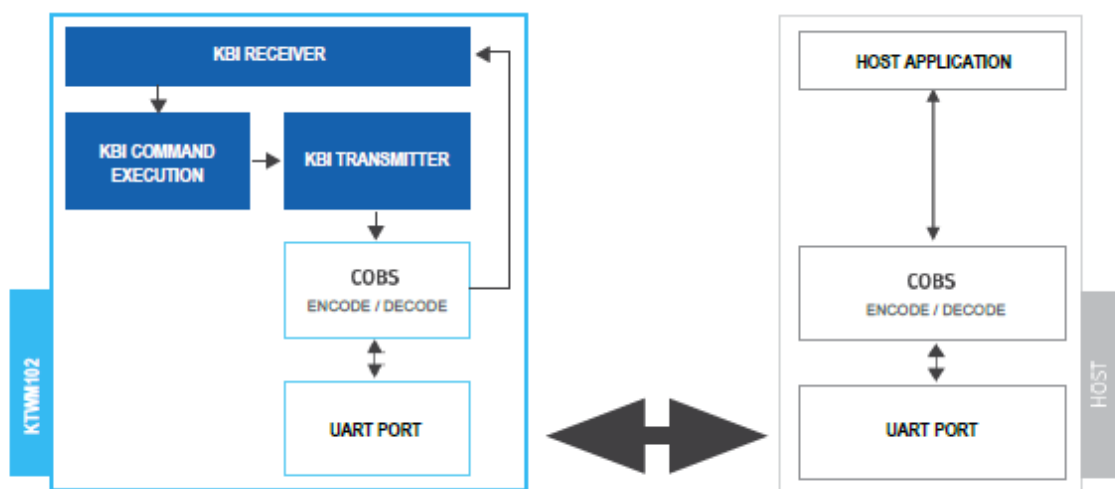


Ilustración 43 Esquema de comunicación entre Host Externo y el dispositivo KTWM102

Una vez conectado el sistema como se indicada en el esquema mostrado en Ilustración 43, se deberá configurar el puerto serie de Host con un Baudrate o tasa de datos, de 115200 bps, 8 data bits, no parity, 1 stop bit, no flow control.

**Nota:** Comprobar que los pines de comunicación serie cumplen con las características eléctricas.

Como se detalla en la Ilustración 43, entre la interfaz UART y el KBI hay un nivel intermedio de codificación/decodificación. Esto es necesario para transmitir un paquete con un carácter delimitador de inicio **0x00**. Cada vez que se transmite un delimitador, se indica el comienzo de un nuevo paquete. Por esta razón, otros caracteres 0x00 del propio paquete necesitarán ser codificados para no confundirlos con un nuevo delimitador. Si se detecta un error en la recepción del paquete UART y el receptor es incapaz de decodificar el mensaje, este enviará una notificación de error (codificada como [0x00 0xFF]) automáticamente al remitente.

Este procedimiento es llevado a cabo por el sistema de codificación **Consistent Overhead Byte Stuffing (COBS)** implementado en KiNOS. Este sistema se explica con más detalle en la [KBI Reference Guide](#) y en [draft-ietf-pppext-cobs-00](#). Por lo que el Host externo deberá implementar este sistema de codificación COBS para poder interactuar con KiNOS vía UART.

### FORMATO DEL PAQUETE

En esta sección se explicará el formato de los paquetes a transmitir o a recibir desde un host externo sin la codificación COBS. Este formato es el mostrado a continuación en Tabla 13:

#### 3.3.2.2.

Header					Payload
L0	L1	TYPE	CMD	CKS	(Optional $\leq 1268$ Bytes)

Tabla 13 Formato del Paquete

En primer lugar, hay una cabecera de cinco bytes de largo donde:

- **L0**: El byte más significativo del largo de la payload.
- **L1**: El byte menos significativo del largo de la payload.
- **TYPE**: Descriptor de tipo.
- **CMD**: Descriptor de Comando
- **CKS**: Byte de Checksum. Se calcula con el XOR de todo el resto de bytes del paquete.

Después de la cabecera puede haber una payload de tamaño variable, el cuál puede presentarse como información opcional para un comando o respuesta específico. El tamaño máximo de la payload es de 1268 bytes. Todos los subcampos de la payload deben ser big endian.

El byte **TYPE** tiene los siguientes bits mostrados en Tabla 14 y en Tabla 15:

TYPE (1 byte)							
FT	FT	FT	FT	FC	FC	FC	FC

Tabla 14 Bits Byte Type

FT: FRAME TYPE					
FC: FRAME CODE	0 - Reserved	1 - Command	2 - Response	3 - Notification	4 to 15 - Reserved
0	-	Write / Execute	OK	Ping Reply	-
1	-	Read	Value	Socket Received Data	-
2	-	Delete	Bad parameter	Named Ping Reply	-
3	-	-	Bad Command	Named Socket Received Data	-
4	-	-	Operation not allowed	Destination Unreachable	-
5	-	-	Memory Allocation Error	-	-
6	-	-	Config. Settings Missing	-	-
7	-	-	Firmware Update Error	-	-
8	-	-	Busy	-	-
9 to 15	-	-	-	-	-

3.3.2.3.

Tabla 15 Significado Bits Byte Type

### REPRESENTACIÓN DE DATOS

La payload de un comando, la respuesta o una notificación, puede consistir en uno o más parámetros separados, cada uno con diferente representación. Los principales tipos son:

- **HEX (n):** Valor hexadecimal genérico de un tamaño variable de hasta n bytes.
- **HEXN (n):** Valor hexadecimal genérico de un tamaño fijo de n bytes.
- **DEC (n):** Valor decimal (entero sin signo) representado en n bytes (tamaño fijo).
- **ENU:** Caso específico de DEC(1) en el cual solo estarán permitidos una enumeración de valores definidos (diferentes para cada comando).
- **STR (a, b):** Cadena ASCII de un tamaño fijado entre a y b caracteres. Se omite el EOS para comandos y se incluye para respuestas o notificaciones.



- **STRN (n):** Cadena ASCII con un tamaño fijado de n caracteres, incluido EOS. En caso de que la cadena sea más corta de lo requerido, está permitido añadir más bytes EOS como padding.
- **MAC:** Caso específico de HEXN(8), representando un identificador de interfaz (dirección MAC), con un tamaño de 8 bytes.
- **ADDR (n):** Caso específico de HEXN(8) o HEXN(16), representando un prefijo IPv6 de tamaño 64 bits o una dirección IPv6 con 128 bits de tamaño.
- **LIST (X):** Se usa para indicar que la payload consiste en la repetición de cierto patrón.

### COMANDOS Y RESPUESTAS

Los dispositivos KiNOS pueden realizar dos tipos de comunicaciones serie vía UART:

3.3.2.4.

- 1) La primera es la repuesta ante la recepción de un comando desde un host externo y su posterior ejecución. Esto implica, que cada comando recibido, genera una respuesta y esta es transmitida de vuelta al host.
- 2) La segunda opción de transmisión desde el dispositivo KiNOS son las notificaciones, o mensajes para informar de eventos asíncronos. (Ver [3.2.5. Notificaciones](#))

Será necesaria una rutina “Comando – Esperar respuesta” en el host externo cuando se transmite un comando al dispositivo KiNOS. Esto implica que el host debe esperar a una respuesta justo después cada comando enviado al dispositivo KiNOS. Se recomienda que en el lado del host haya una coherencia entre el comando enviado y la respuesta recibida.

En algunos casos, puede pasar que el host externo reciba una notificación asíncrona mientras está esperando una respuesta a un comando, por lo que el host tendrá que decidir si guardarla y procesarla después o rechazarla y continuar esperando la respuesta del comando.

Un ejemplo de esta rutina de “Comando – Esperar respuesta” sería lo mostrado a continuación en Ilustración 44:

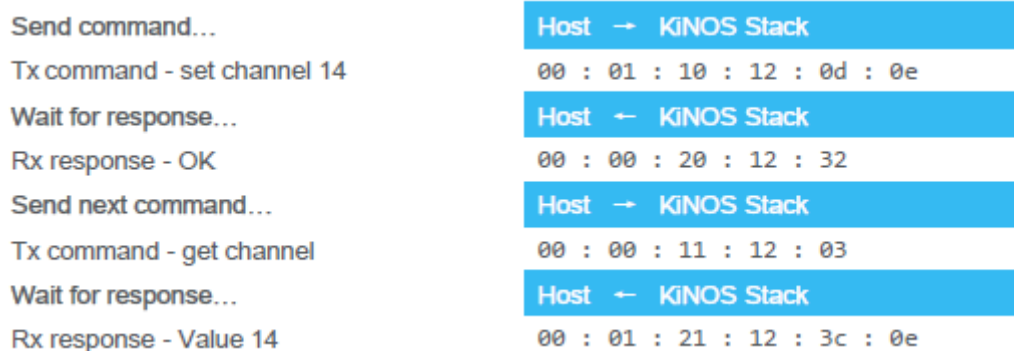


Ilustración 44 Ejemplo Rutina de Comando - Esperar Respuesta

## NOTIFICACIONES

Un dispositivo KiNOS manda una notificación a un host externo para informar de eventos asíncronos. Estos eventos asíncronos pueden ser eventos como la recepción de datos por tráfico serie por UDP, la cual puede darse en cualquier momento por radio y se transmite la notificación por UART. En estas notificaciones, el byte de CMD tiene valor nulo, 0x00.

3.3.2.5.

Para más detalle de los comandos ver en la [KBI Reference Guide](#)

### 3.3.3. CONFIGURACIÓN DE RED

Una vez explicadas los diferentes métodos de enviar los comandos al módulo KTWM102, se procederá a explicar el procedimiento para poder configurar los parámetros para que el nodo se una a la red correctamente.

Previo a configurar los parámetros de la red, se debe considerar si se desea o no realizar el proceso de “In Band”.

- En caso afirmativo, por defecto, se configurará la red de acuerdo con lo especificado en el apartado 3.3.3.1 MODO OUT-OF-BAND COMMISSIONING DESACTIVADO. En este modo, el nodo buscará las redes que haya alrededor y se conectará a la red con mejor conectividad.
- En caso negativo, si se desea unir el nodo a una red en específico, se deberá activar el modo de Commissioning “Out of Band”, y posteriormente introducir todos los parámetros y configuraciones necesarios de la red en concreto. La configuración necesaria se indica en el apartado 3.3.3.2 MODO OUT-OF-BAND COMMISSIONING ACTIVADO.

Para ver la sintaxis de los comandos a enviar para las diferentes configuraciones, ver las guías KBH y KBI anteriormente mencionadas.

**Nota:** En ambos casos, siempre se debe ejecutar el comando **Clear** para borrar posibles configuraciones anteriores. Solo es conveniente **no** ejecutarlo cuando la configuración guardada es la misma que se vaya a utilizar en ese momento.

3.3.3.1.

#### MODO OUT-OF-BAND COMMISSIONING DESACTIVADO

Este modo es el modo por defecto, pero aun así es recomendable ejecutar el comando de desactivar el modo “Out-of-Band Commissioning”.

Una vez se asegure que este modo esté desactivado, la configuración requerirá de pocos parámetros obligatorios.

Antes de la ejecución del comando **Ifup**, deberá configurarse el Role que tendrá el nodo. En caso contrario, se generará el error “*Configuration settings missing*”. Otros parámetros como el *Canal*, *PAN ID*, *Nombre de Red* y la *Credencial de Commissioning*, serán opcionales.

Una vez ejecutado el comando **IfUp**, el módulo, si es configurado como Leader, creará una red y generará automáticamente todos los parámetros de configuración de la red. En caso de que el módulo no sea un nodo Leader, este buscará todas las redes cercanas y se conectará

a la red a la que tenga mejor conectividad y cumpla con los parámetros configurados (en caso de haber configurado). Una vez unido el módulo a la red, completará los parámetros no configurados previamente con los parámetros de la red a la que se haya conectado.

Una vez conectado a la red, se guardará su configuración como configuración válida y ya no podrá modificarse salvo ejecutando previamente el comando Clear.

### *MODO OUT-OF-BAND COMMISSIONING ACTIVADO*

Este modo no está activado por defecto, por lo que se deberá ejecutar el respectivo comando de activación del modo “Out-of-Band Commissioning”. Este modo convendrá activarlo cuando se quiera unir los módulos a una red en concreto, como puede ser una red “privada”.

En este modo se deberá configurar más parámetros para poder especificar correctamente la red a la que se quiera unir o que se quiera crear. Los parámetros a configurar son: *Rol*, *Canal*, *PAN ID*, *Nombre de Red*, *Prefijo Local*, *Master Key*, *PAN ID Extendida* y la *Credencial de Commisioning*.

Sin uno de esos parámetros, se generará el error “*Configuration settings missing*”.



## 4. DISEÑO E IMPLEMENTACIÓN HARDWARE

En este capítulo se mostrará el diseño del esquemático y el respectivo Layout de la PCB realizada para la integración del dispositivo KTWM102 a la vez que los elementos comerciales utilizados para el desarrollo del trabajo.

### 4.1. COMPONENTES COMERCIALES UTILIZADOS

#### 4.1.1. ELEMENTOS HARDWARE UTILIZADOS

Los elementos Hardware comerciales utilizados han sido:

- 1) **Kit de Desarrollo STM32F407G-DISC1:** Kit de microcontrolador, mostrado en Ilustración 45, utilizado en primera instancia para el envío de comandos vía UART a los módulos KTDG102 Evaluation Dongle de Kirale. Se ha utilizado 1 unidad.



*Ilustración 45 Kit de Desarrollo STM32F407G-DISC1*

- 2) **KTDG102 Evaluation Dongle:** Módulo Thread de Evaluación de Kirale, mostrado en Ilustración 46. Se han utilizado 2 unidades. Se han detallado sus características anteriormente en 3.1.1.2 CARACTERÍSTICAS KTDG102 EVALUATION DONGLE.



*Ilustración 46 KTDG102 Evaluation Dongle*

- 3) **Módulo KTWM102:** Módulo Thread de Kirale, Ilustración 47, el cual se integra en la PCB diseñada. Sus características han sido comentadas anteriormente en 3.1.1.1 CARACTERÍSTICAS MÓDULO RF KTWM102.



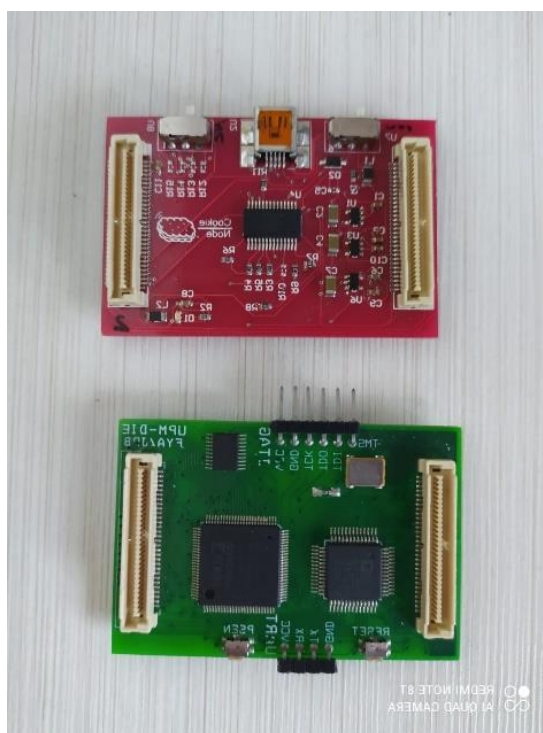
*Ilustración 47 Módulo KTWM102*

- 4) **Border Router:** Border Router propio de Kirale, mostrado en Ilustración 48 y cuyas características se han desarrollado anteriormente en 3.1.1.3 CARACTERÍSTICAS DEL BORDER ROUTER KTBRN1.



*Ilustración 48 Border Router*

- 5) **Cookie:** Plataforma sobre la que se implementa el diseño final de la PCB realizada, compuesto por los dos módulos mostrados en Ilustración 49, en rojo el módulo de alimentación y en verde el módulo de procesamiento.



*Ilustración 49 Módulos de Procesamiento y de Alimentación de la Cookie*

##### 4.1.2. ELEMENTOS SOFTWARE UTILIZADOS

Aparte de diferentes dispositivos hardware comerciales, se han usado diferentes entornos o plataformas software para el correcto desarrollo del proyecto. Estos elementos han sido:

- **Keil uVision 5:** Programación y debug del microcontrolador ARM de STM.
- **Keil uVision 3:** Programación para el ADuC841 usado en la Cookie.
- **Termite:** Terminal serie para poder leer los datos enviados de ambos microcontroladores utilizados y poder leer por pantalla las respuestas de los módulos a los diferentes comandos.
- **WSD (Windows Serial Downloader):** Carga del fichero binario (\*.hex) al microcontrolador ADuC841.
- **KiTools:** Herramienta proporcionada por Kirale Technologies para la comunicación con los módulos KTDG102 y KTM102. Siempre se ejecutará al principio de cada sesión en cada módulo los comandos *debug module all* y *debug level all*, para poder ver a todos los niveles lo que va sucediendo en el módulo, permitiendo debuggear el envío y recibo de datos y/o comandos.
- **Altium Designer:** Programa utilizado para el diseño de esquemáticos y Layout de la PCB.
- **Github:** Plataforma utilizada para el control de versiones del trabajo, tanto de la parte de desarrollo software, hardware y la documentación del proyecto.



## 4.2. DISEÑO DE ESQUEMÁTICO PCB

### 4.2.1. JERARQUÍA DEL CIRCUITO

El circuito se ha compuesto de las siguientes partes:

- **Alimentación:** Se gestiona la alimentación del circuito ya sea desde 5V o 3V3.
- **Módulo:** Se implementa el módulo KTWM102 junto con el circuito necesario.
- **Cookie Connector:** Zona donde se definen los conectores a utilizar para integrar la PCB a la Cookie.
- **USB:** Se introduce un conector micro USB tipo B.

Estas cuatro partes se integran y se conectan entre sí de la como se muestra a continuación en Ilustración 50:

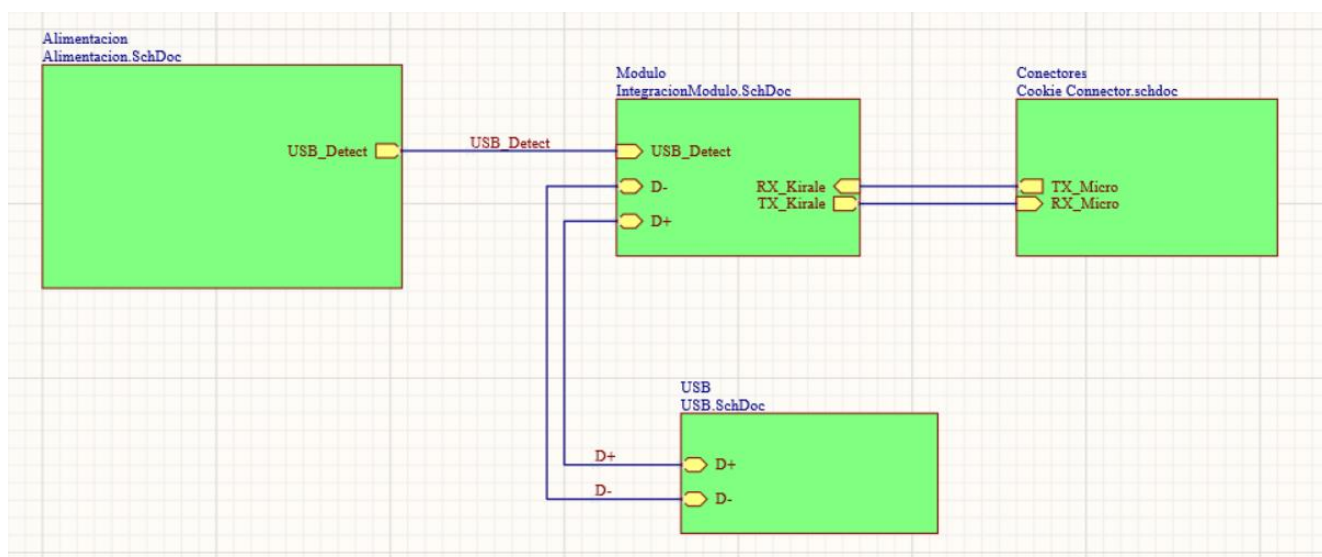


Ilustración 50 Jerarquía Circuito

## 4.2.2. CIRCUITO DE ALIMENTACIÓN

En la parte de alimentación, el circuito realizado ha sido el mostrado a continuación en Ilustración 51:

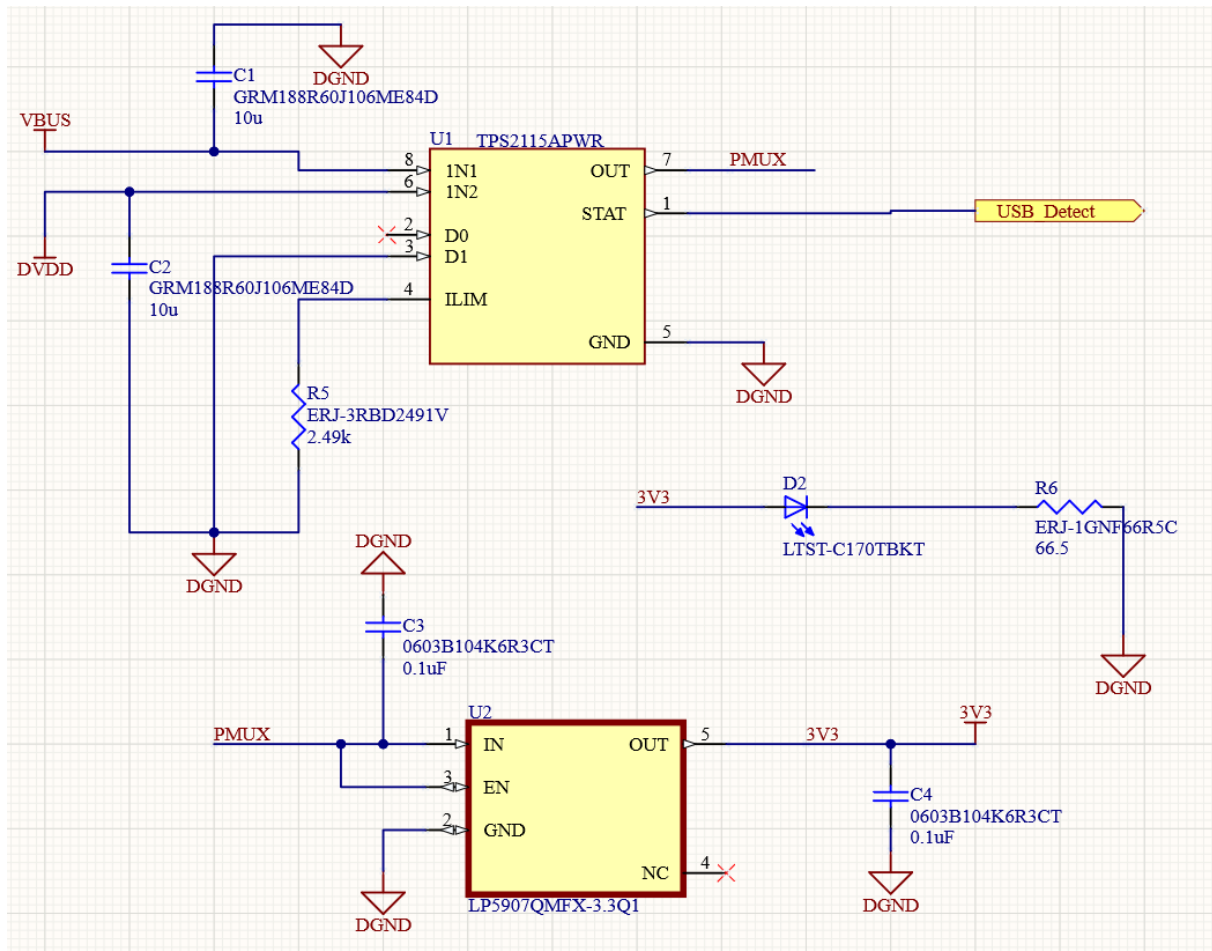


Ilustración 51 Circuito de Alimentación

Para entender el circuito se deberá tener en cuenta que disponemos de dos alimentaciones posibles:

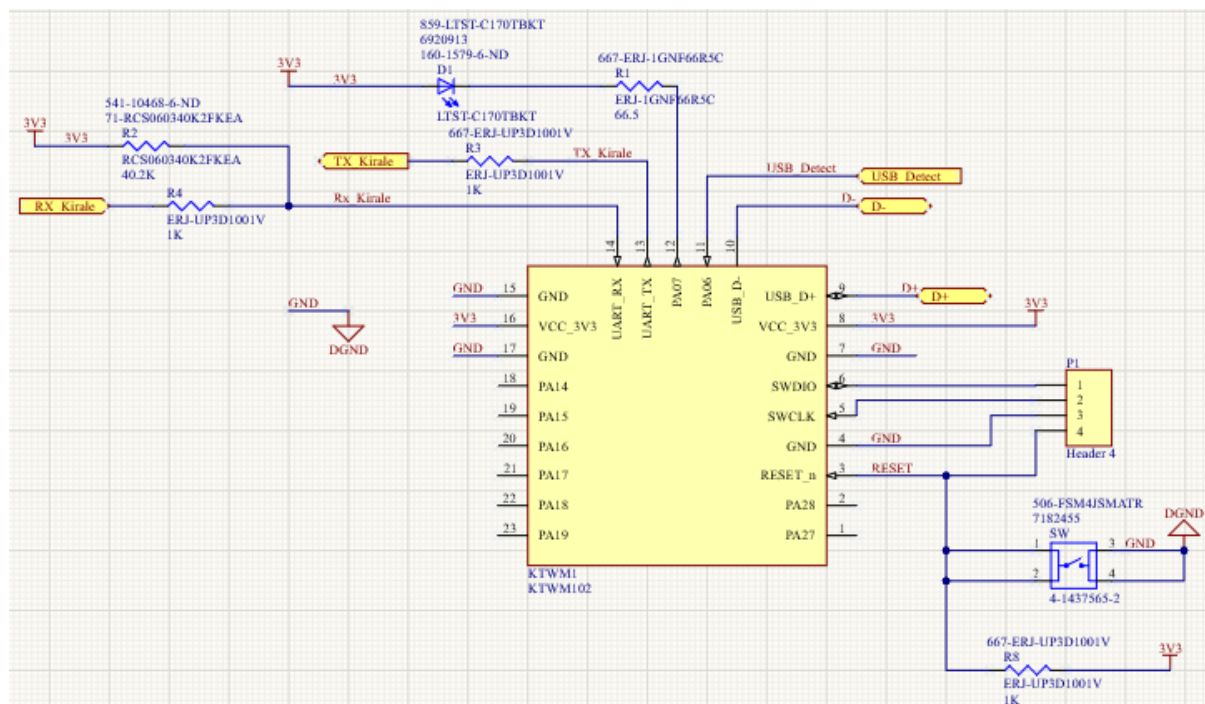
- Alimentación de 5V recibida por el conector USB (definida como **VBUS**).
- Alimentación de 3V3 recibida de la placa de alimentación de la Cookie (definida como **DVDD**).

En primer lugar, ambas líneas de alimentación pasan por U1, un switch de potencia, el cual se ha configurado para priorizar la alimentación de 5V sobre la de 3V3 como alimentación de la PCB. En caso de que la PCB solo se conecte a la Cookie y no por USB, por lo tanto, no se dispondrá de 5V, la alimentación de la PCB será la alimentación de 3V3 recibida de la placa de alimentación de la Cookie. Esta alimentación seleccionada (**PMUX**) se hará pasar por U2, LDO a 3V3, asegurando así una alimentación estable de 3V3, ya se haya seleccionado previamente 5V o 3V3. Finalmente, ya se pasará esta alimentación al resto del circuito.

Se ha añadido un Led SMD, junto con una resistencia en serie, a la salida de 3V3 del LDO (U2) para confirmar el correcto funcionamiento de esta parte del circuito.

### 4.2.3. INTEGRACIÓN DEL MÓDULO KTWM102

El circuito realizado para integrar el módulo KTWM102 es el mostrado a continuación en Ilustración 52:



*Ilustración 52 Circuito Integración del Módulo*

El módulo KTWM102 es el mostrado en la Ilustración 47, en el apartado 4.1.1 ELEMENTOS HARDWARE UTILIZADOS

Es el dispositivo utilizado para las comunicaciones entre los diferentes nodos. Para el diseño de esta parte del circuito, se han seguido las recomendaciones de KIRALE. Se añaden un pulsador con pull-up para el RESET\_n del propio KTWM102. El Led SMD incluido es utilizado para indicar el estado del nodo (conectado a la red, sin conexión, etc...).

## 4.2.4. COOKIE CONNECTOR

Esta parte del circuito es la interfaz con el resto de PCB integrantes de la Cookie a través de conectores verticales, el esquemático es el mostrado a continuación en Ilustración 53:

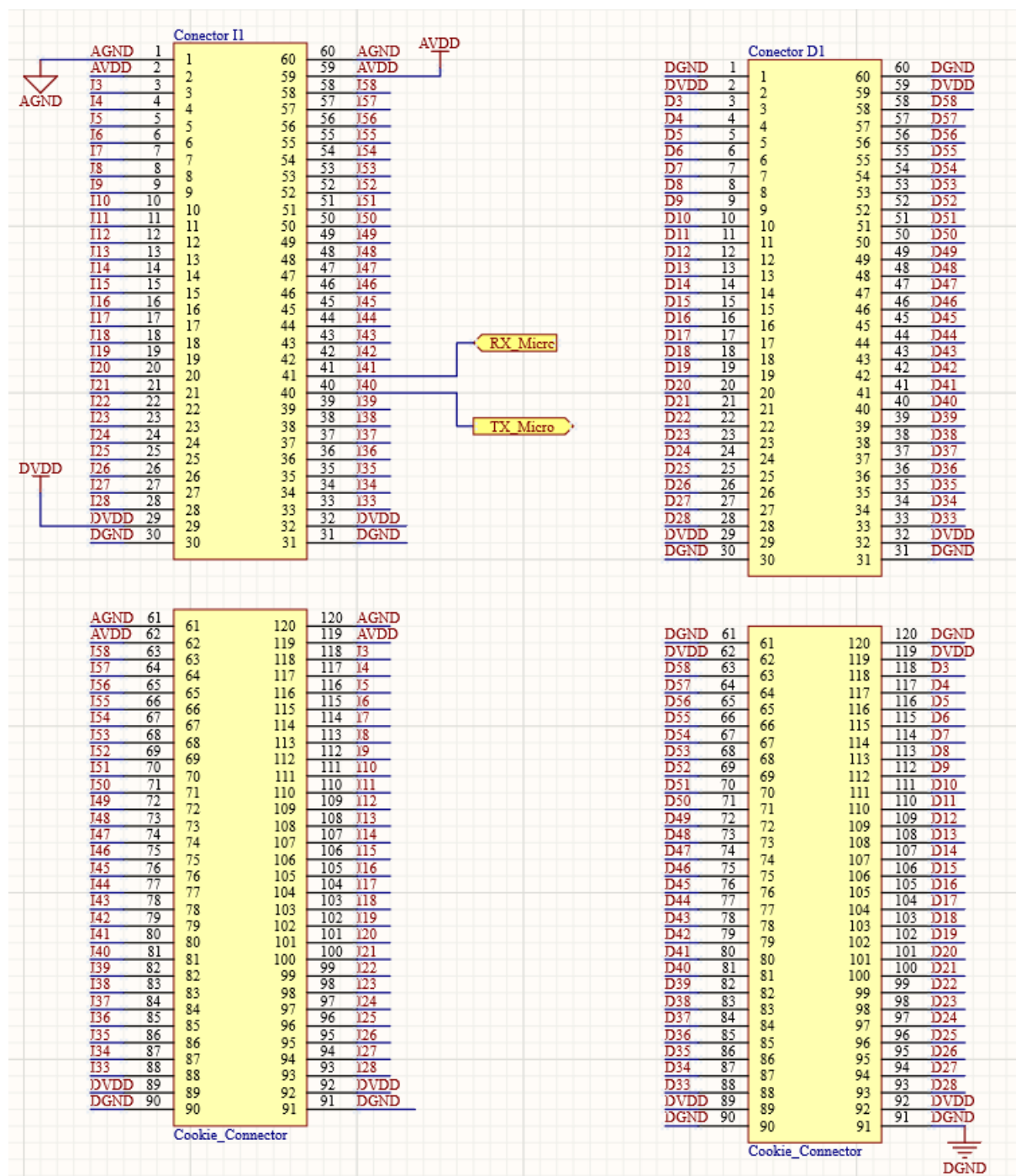


Ilustración 53 Conectores Verticales

### 4.2.5. USB

Esta última parte implementa un conector micro USB tipo B, pudiendo así conectar el dispositivo KTWM102 con un PC a través de la herramienta KiTools. El circuito realizado es el mostrado a continuación en Ilustración 54:

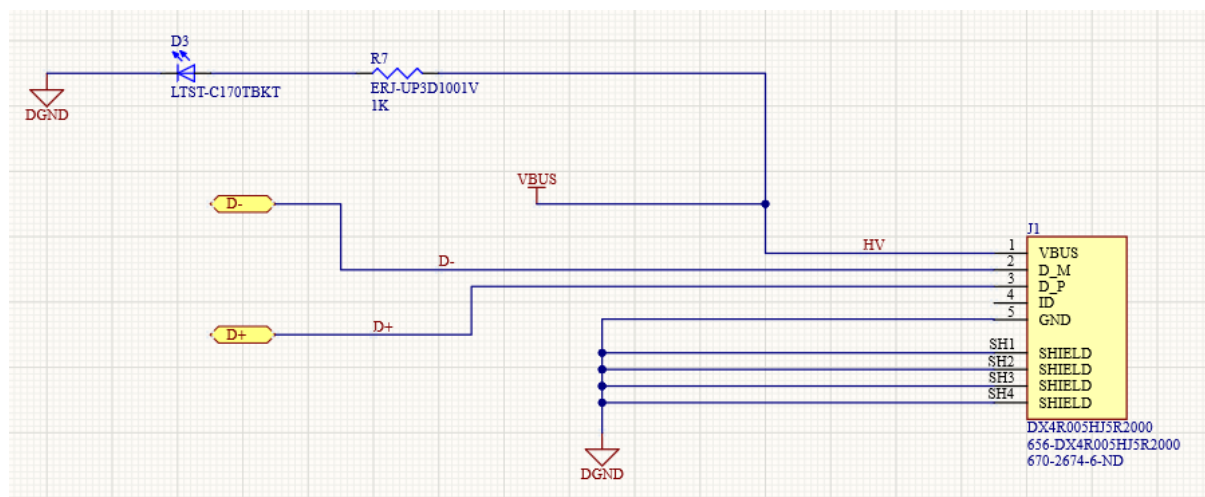
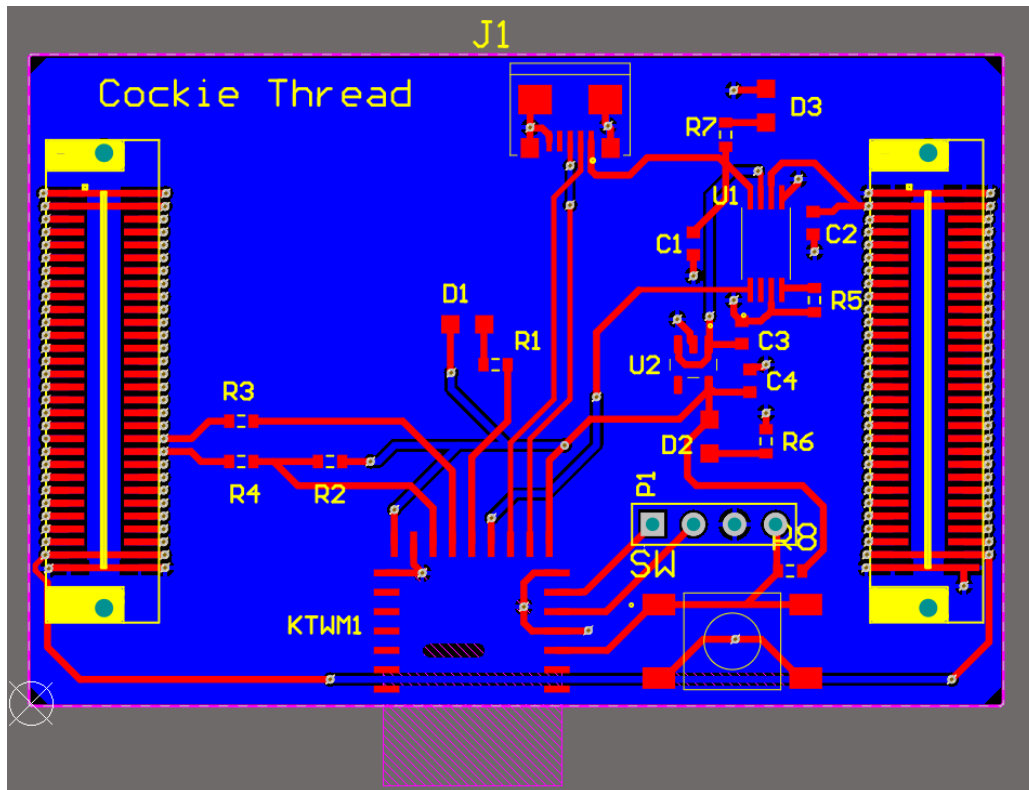


Ilustración 54 Conector USB

Se añade un Led SMD con una resistencia en serie, como comprobación de que lleguen los 5V correctamente al circuito, como símbolo de una buena conexión del conector.

### 4.3. LAYOUT

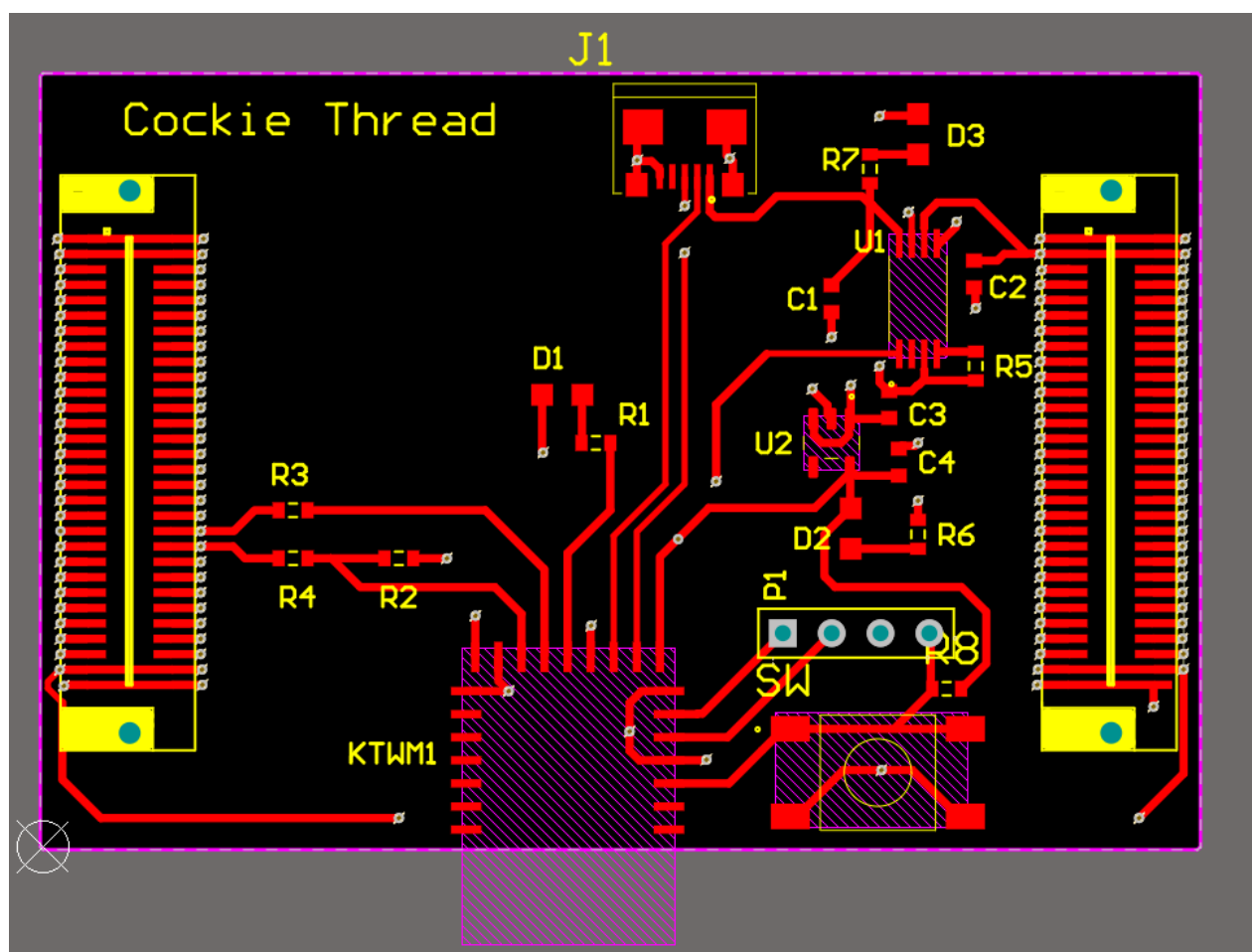
El Layout completo realizado es el mostrado a continuación en Ilustración 55. Posteriormente se mostrará el Layout elegido separando las diferentes capas.





#### 4.3.1. LAYOUT CAPA TOP

Como se muestra a continuación, en Ilustración 56, se han posicionado en la capa TOP la mayoría de las pistas y se ha realizado el posicionado de los componentes exceptuando 2 de los 4 conectores verticales.



*Ilustración 56 Layout Capa TOP*

##### 4.3.2. LAYOUT CAPA BOTTOM

Como se muestra a continuación, en Ilustración 57 , se han posicionado en la capa BOTTOM dos conectores verticales y algunas pistas debido a cambios de cara necesarios. A su vez, se ha usado la capa BOTTOM para añadir un plano de masa.

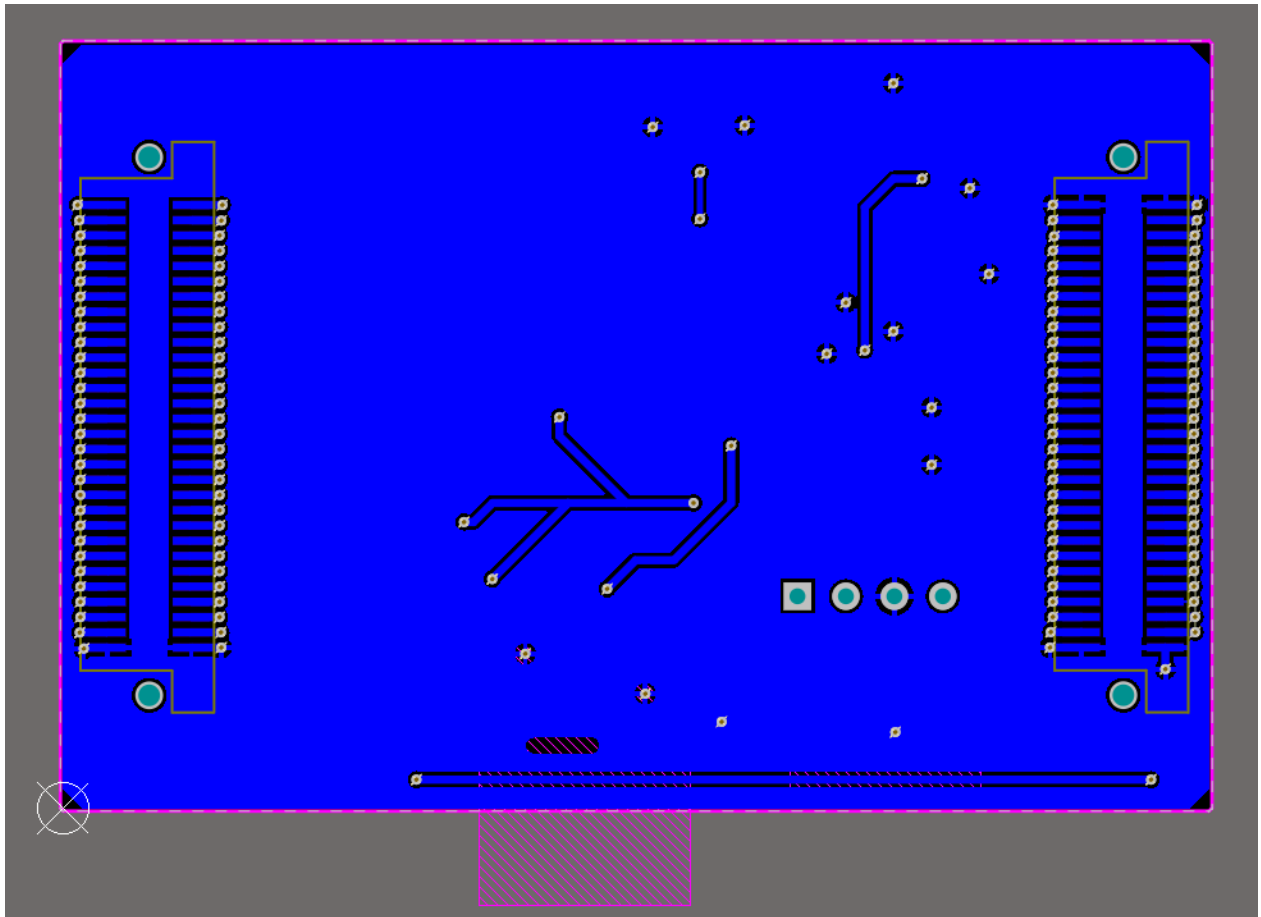


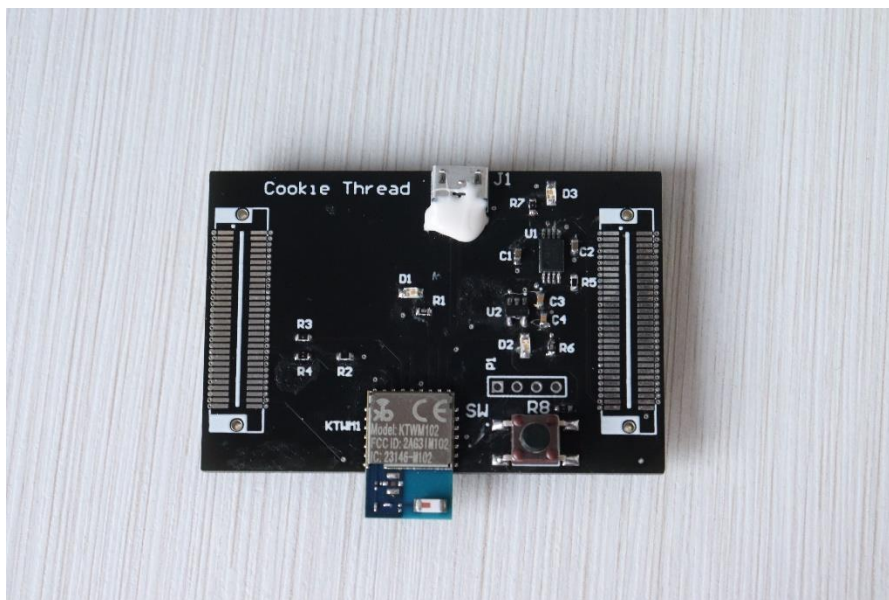
Ilustración 57 Layout Capa BOTTOM



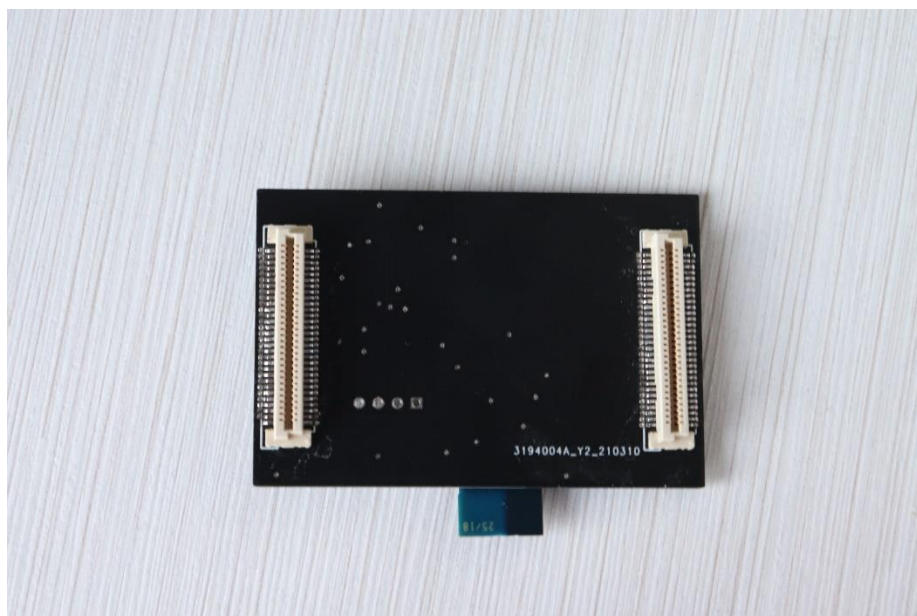
### 4.3.3. RESULTADO FINAL DEL DISEÑO

Una vez finalizado el diseño se mandó fabricar la PCB quedando como se ve en Ilustración 58 y en .

Basándose en proyectos previos, como se ve en Ilustración 51, se ha añadido silicona no conductora al conector USB para aumentar la fijación del mismo y evitar posibles averías durante el proceso de conexión y desconexión del cable.



*Ilustración 58 PCB Cookie Thread TOP*



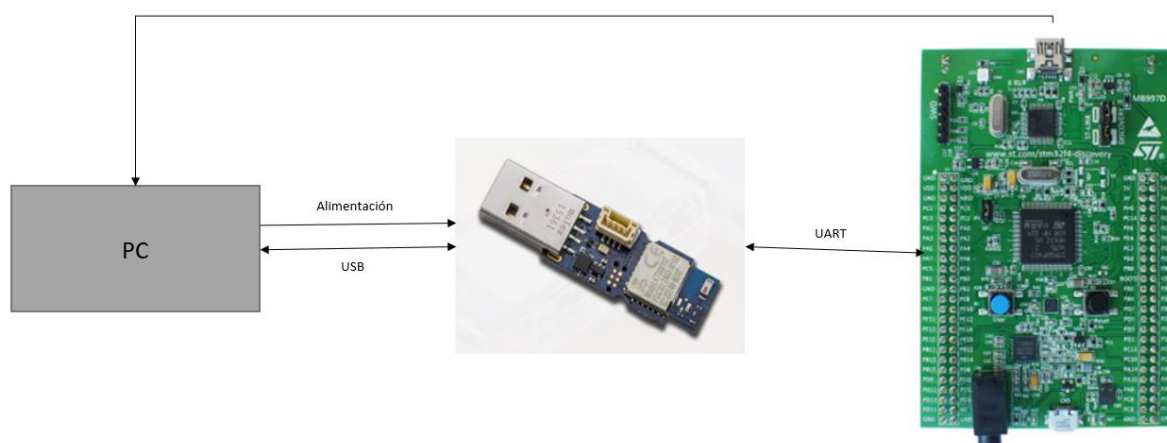
*Ilustración 59 PCB Cookie Thread Capa BOTTOM*



## 5. PRUEBAS EXPERIMENTALES

### 5.1. PRIMERA INTERACCIÓN CON DONGLE USB

Esta primera prueba consiste en una primera interacción con el módulo, tanto desde el PC usando la herramienta proporcionada por Kirale de KiTools y usando el puerto USB, como desde un microcontrolador por vía UART. El diagrama de conexión que se ha utilizado ha sido el mostrado a continuación en Ilustración 60:



*Ilustración 60 Diagrama de conexión PC - Dongle - uC*

Esta primera prueba se divide en dos partes:

- 1) La primera parte se ejecutan los diferentes comandos desde el PC usando la herramienta KiTools con los comandos KSH, comprobando que se ejecutan correctamente según lo esperado.
- 2) Una vez comprobado los comandos KSH, se usa el microcontrolador para el envío de comandos KBI vía puerto UART. Para la comprobación del correcto envío se usan dos maneras:
  - a. La primera es la activación de los modos de debug del módulo en la herramienta KiTools en el PC, pidiéndole al módulo que informe de la actividad en las diferentes capas. Cuando le lleguen mensajes y realice las respectivas respuestas, saltará un log de mensaje por rx y por tx con los bytes recibidos/transmitidos. A su vez, si tenemos un error de Checksum, esta herramienta nos avisará del error.
  - b. Una vez enviada el comando KBI, recogemos la respuesta enviada por el módulo KTDG102 y analizamos su respuesta.

Tras hacerse al envío de comandos de manera correcta, se elabora la secuencia de comandos para la configuración completa de la red para ese módulo KTDG102 con el Rol de LEADER.

## 5.2. RED DE DOS NODOS

El siguiente paso ha consistido en la creación de una red con un nodo LEADER y la unión de un nodo MED a dicha red. Para esto se han usado los dos KTDG Evaluation Dongles conectados al PC con la herramienta KiTools por USB y al microcontrolador ARM por vía UART.

Se han realizado varias pruebas para las cuales el esquema de montaje ha sido el mostrado en Ilustración 61:

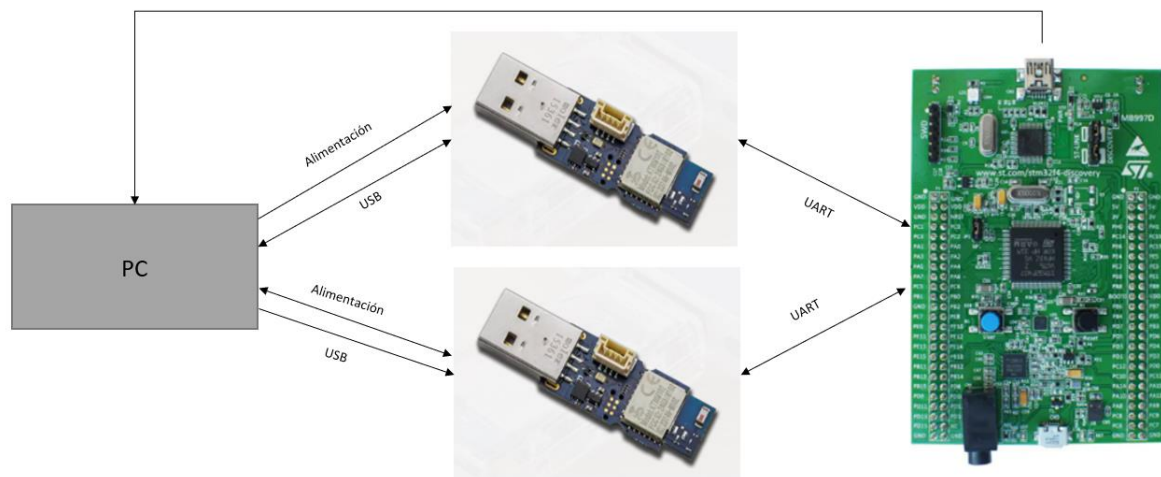


Ilustración 61 Esquema montaje Red de Dos Nodos

### 5.2.1. CREACIÓN DE LA RED

La primera prueba realizada ha sido la creación correcta de la red (de la manera vista en PRIMERA INTERACCIÓN CON DONGLE USB) y la posterior unión del segundo nodo con el rol de MED. La red, para ambos nodos se ha realizado con el modo “Out-of-Band Commissioning” desactivado, pero configurando en ambos casos el PAN ID, el Canal y la Commissioning Credential.

Para esto, se ha usado la secuencia de creación de red con Rol Leader usada anteriormente para el primer nodo, y una vez creada la red, se volvía a usar esta secuencia modificando el Rol a configurar en el nodo para el segundo nodo.

Tras ejecutar el primer nodo su secuencia el comando IfUp, se debe dejar un tiempo de alrededor a 7 segundos para que pueda crearse la red definida. A su vez, se comprueba que debe dejarse entre 7 y 9 segundos en el nodo MED después del comando IfUp para que este se una a la red, ya que al ser en modo “Out-of-Band Commissioning” desactivado, y no tener todos los parámetros, este nodo buscará una red de entre las cercanas que tenga los mismos parámetros a los configurados. Cuando este encuentre la red, saltará un aviso en el nodo LEADER en la herramienta de KiTools, de que se está uniendo un nodo.

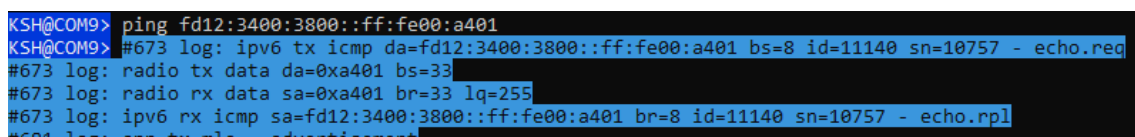
Por otro lado, cuando se ejecutaban los comandos y se enviaba respuesta al microcontrolador, este las imprimía por pantalla en el PC, pudiendo así detectar posibles fallos en alguno de los comandos para poder arreglarlos.

### 5.2.2. PING ENTRE NODOS

Una vez tenemos los nodos en la misma red, una primera comprobación de la comunicación entre ambos módulos. En este caso, al ser una prueba rápida y sencilla de comprobación, se ha usado la herramienta KiTools, abriendo dos sesiones, una por cada módulo. Una vez en red, se ejecuta el comando *show netconfig* desde una de las dos sesiones de KiTools. Con esto podemos ver la dirección IP de los módulos.

Tras ver la dirección de uno de los módulos Dongle, se ejecutará el comando *ping <arg>* en el módulo KTDG102 restante, siendo el argumento una de las direcciones IP que hemos visto del otro módulo KTDG102.

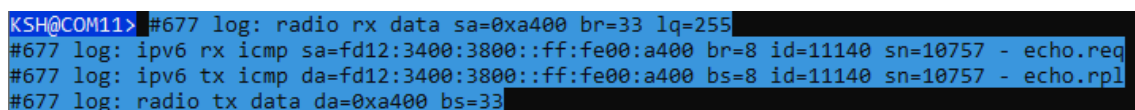
En el lado que se realiza el ping saldrán los Logs en KiTools que se muestran en Ilustración 62:



```
KSH@COM9> ping fd12:3400:3800::ff:fe00:a401
KSH@COM9> #673 log: ipv6 tx icmp da=fd12:3400:3800::ff:fe00:a401 bs=8 id=11140 sn=10757 - echo.req
#673 log: radio tx data da=0xa401 bs=33
#673 log: radio rx data sa=0xa401 br=33 lq=255
#673 log: ipv6 rx icmp sa=fd12:3400:3800::ff:fe00:a401 br=8 id=11140 sn=10757 - echo.rpl
#681 log: app tx mlo advertisement
```

Ilustración 62 Logs KiTools al REALIZAR un Ping

En cambio, en el KiTools del lado que recibimos el ping, saldrán los Logs mostrados a continuación en Ilustración 63:



```
KSH@COM11> #677 log: radio rx data sa=0xa400 br=33 lq=255
#677 log: ipv6 rx icmp sa=fd12:3400:3800::ff:fe00:a400 br=8 id=11140 sn=10757 - echo.req
#677 log: ipv6 tx icmp da=fd12:3400:3800::ff:fe00:a400 bs=8 id=11140 sn=10757 - echo.rpl
#677 log: radio tx data da=0xa400 bs=33
```

Ilustración 63 Logs KiTools al RECIBIR un Ping

### 5.2.3. ENVÍO DE MENSAJES UDP ENTRE AMBOS NODOS

Una vez se ha comprobado que los dos nodos han tenido una primera comunicación básica mediante el comando ping, el siguiente paso es la prueba del envío de mensajes entre ellos vía Sockets UDP.

Antes de poder enviar y/o recibir un socket, se deberán tener en cuenta dos cosas:

- 1) A priori, no se puede saber la dirección IP que tendrá el nodo una vez conectado a la red. Por lo que convendrá ejecutar el comando de asignación de IP (en nuestro código llamado como WriteIP) con una dirección IP a elegida arbitrariamente. Este comando

debe realizarse una vez se haya unido el nodo a la red. Esto convendrá de cara al posterior envío de los mensajes.

- 2) En esta prueba, al no haber un nodo router como tal, deberá ejecutarse desde el nodo LEADER el comando ROUTE con dirección al nodo MED. Este creará el enlace para los propios mensajes. Debe ejecutarse antes del envío de los mensajes. Como se ha visto en la prueba explicada en 5.2.2 PING ENTRE NODOS, en este caso no hizo falta realizar el comando route, pero a la hora del envío de sockets no se consiguió recibir los mensajes sin este comando previo.

Para enviar un mensaje a través de sockets UDP, se debe:

- Abrir un socket en un puerto determinado (por cada nodo). El puerto asociado al socket puede ser diferente en cada nodo.
- Enviar mensaje a través del puerto asignado al socket del propio nodo, al puerto y dirección IPv6 del nodo receptor.

Para recibir un mensaje a través de sockets UDP, bastará con tan solo abrir un socket en un puerto determinado, en caso de no haberlo abierto para un anterior envío.

Para la lectura de los mensajes, se recogerá la notificación enviada por puerto UART al microcontrolador, ya que dispondremos de la información del remitente y del mensaje recibido, mientras que por la herramienta KiTools solo dispondremos de la dirección de origen o de destino del mensaje, según veamos la sesión de un nodo u otro.

### 5.3. PRUEBAS CON EL BORDER ROUTER

Una vez generada estas primeras pruebas con los KTDG102, se introduce en la red el Border Router de Kirale.

#### 5.3.1. INTRODUCCIÓN A LA CONFIGURACIÓN DEL ROUTER.

Esta primera prueba ha consistido en una primera interacción con la interfaz del panel de administración web del Border Router y a su configuración correcta. El esquema del montaje realizado ha sido el mostrado a continuación en Ilustración 64:



Ilustración 64 Montaje Border Router

Para una primera configuración del BR, una vez instalados los drivers USB, se abrió una terminal COM con MobaXterm, para poder acceder al Border Router. Una vez loggados dentro de la terminal se ha configurado su dirección IPv4 a una dirección dentro de la red que se estaba usando. En este caso la dirección IPv4 configurada para esta prueba y para el futuro funcionamiento ha sido: **192.168.0.102**, con una configuración de IP estática.

Una vez configurada la dirección IP estática, se podrá acceder también por puerto SSH. Para comprobar también el correcto funcionamiento del BR, se introduce *[Dirección IP asignada]:8000* en el navegador web y debe mostrarse el Panel de Administración Web del BR.

Una vez se ha accedido al sistema, se configura todo el sistema con las DNS y los Gateways a los que se conecta la red local. En este caso se ha configurado los DNS de la red de Movistar en la pestaña de DNS, dentro de menú **Network** y se configura en la interfaz eth0 dentro de la pestaña **Network**: como Gateway el Router de la red local de la casa, la dirección 192.168.0.1 y con máscara de red 255.255.255.0.

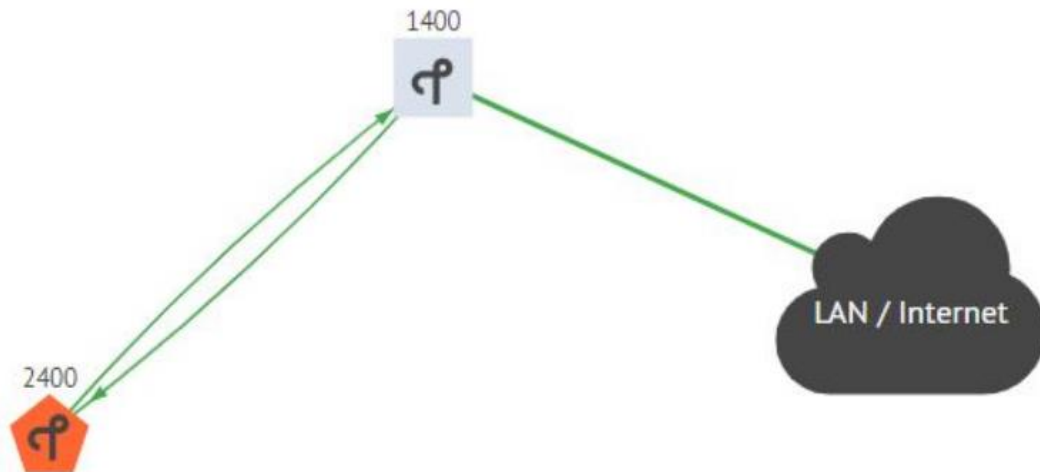
Finalmente se configura la red Thread a crear y se ejecuta su creación. Se comprueba que desde la CMD de Windows se puede hacer ping a la IPv4 y ping -6 a la dirección IPv6 del Border Router, dando en ambos casos una latencia de 3 ms aproximadamente.

También desde la sesión SSH o COM abierta desde MobaXTerm, se puede probar a hacer el ping a una dirección web como la propia página de Kirale o de Google. En ambos casos daba una latencia mínima de 20 ms.



### 5.3.2. PRUEBA DE CONECTIVIDAD IP ENTRE RED THREAD Y LAN

Para esta prueba se ha usado el BR y uno de los Dongle KTDG102 formando la topología de red mostrada a continuación en Ilustración 65:



*Ilustración 65 Topología de Red 1 nodo con BR*

El Dongle se usará como REED o router, mientras que el BR hará de Leader. En esta prueba se siguen los pasos indicados por Kirale para esta comprobación de conectividad. Los pasos a seguir son los indicados en:

1. [Descripción Genérica y Condiciones Previas](#)
2. [Conectividad IPv6](#)
3. [Conectividad IPv4 a IPv6](#)
4. [Conectividad IPv6 a IPv4](#)

Una vez todos estos pasos han salido correctamente, se confirma el correcto funcionamiento del BR.

### 5.3.3. RED CON EL BR Y DOS NODOS KTDG102

Una vez configurado correctamente el BR se pasa a la configuración de una red juntando los dos Dongles KTDG102.

Como primera prueba de creación de esta red, se configuró un Dongle como Leader, otro como Med y el BR como REED. Primero se ejecutaban las secuencias de inicio de ambos Dongles y posteriormente se lanzaba el nodo del BR unirse a la red.

Al principio de unirse el BR, se observa en el menú de Visual Network la topología de red mostrada a continuación en Ilustración 66:



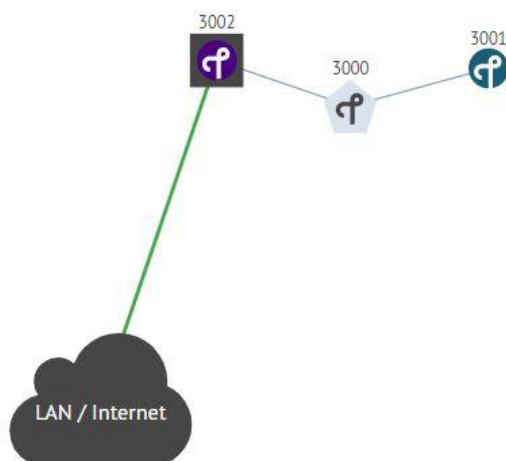


Ilustración 66 Topología 1 de BR con 2 nodos Dongle

Siendo el Dongle Leader el nodo de color gris, el MED el azul circular y finalmente el BR el nodo cuadrado morado. El BR tardará un rato en configurarse como router y en tener activos todos los servicios de interfaz con la red LAN o el internet. Una vez pasa un rato, de aproximadamente 1 o 2 minutos, se observa que el BR cambia a color naranja y hay una reestructuración en la Topología, quedando como se muestra a continuación en Ilustración 67:

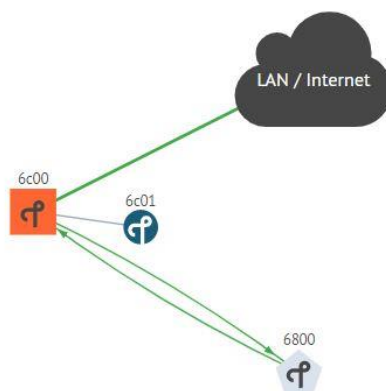


Ilustración 67 Topología 2 de BR con dos nodos Dongle

Se observa el cambio de padre del nodo MED.

Una vez el Border Router inicia todos sus servicios y la red esta estable, se comprueba la accesibilidad a los nodos desde el PC, haciendo *PING -6* a las direcciones IPv6 asignadas a los dos Dongle.

**Nota:** Para ver las direcciones IP de los Dongle KTDG102, abrir la herramienta KiTools en el PC y ejecutar el comando `show netconfig`, de las diferentes direcciones IPv6, se debe coger la que empieza con el prefijo configurado en BR y termina con `::a2e:XX`, siendo XX la parte variable de cada módulo.

Los resultados de estos PING han sido los mostrados en Ilustración 68, Ilustración 69 e Ilustración 70:

```
Haciendo ping a fd00:7d03:7d03:7d03::a2e:1e con 32 bytes de datos:
Respuesta desde fd00:7d03:7d03:7d03::a2e:1e: tiempo=4ms
Respuesta desde fd00:7d03:7d03:7d03::a2e:1e: tiempo=2ms
Respuesta desde fd00:7d03:7d03:7d03::a2e:1e: tiempo=2ms
Respuesta desde fd00:7d03:7d03:7d03::a2e:1e: tiempo=2ms

Estadísticas de ping para fd00:7d03:7d03:7d03::a2e:1e:
  Paquetes: enviados = 4, recibidos = 4, perdidos = 0
    (0% perdidos),
  Tiempos aproximados de ida y vuelta en milisegundos:
    Mínimo = 2ms, Máximo = 4ms, Media = 2ms
```

*Ilustración 68 Ping desde PC a BR*

```
Haciendo ping a fd00:7d03:7d03:7d03::a2e:9f con 32 bytes de datos:
Respuesta desde fd00:7d03:7d03:7d03::a2e:9f: tiempo=23ms
Respuesta desde fd00:7d03:7d03:7d03::a2e:9f: tiempo=23ms
Respuesta desde fd00:7d03:7d03:7d03::a2e:9f: tiempo=23ms
Respuesta desde fd00:7d03:7d03:7d03::a2e:9f: tiempo=23ms

Estadísticas de ping para fd00:7d03:7d03:7d03::a2e:9f:
  Paquetes: enviados = 4, recibidos = 4, perdidos = 0
    (0% perdidos),
  Tiempos aproximados de ida y vuelta en milisegundos:
    Mínimo = 23ms, Máximo = 23ms, Media = 23ms
```

*Ilustración 69 Ping desde PC a nodo LEADER*

```
Haciendo ping a fd00:7d03:7d03:7d03::a2e:20 con 32 bytes de datos:
Respuesta desde fd00:7d03:7d03:7d03::a2e:20: tiempo=22ms
Respuesta desde fd00:7d03:7d03:7d03::a2e:20: tiempo=21ms
Respuesta desde fd00:7d03:7d03:7d03::a2e:20: tiempo=21ms
Respuesta desde fd00:7d03:7d03:7d03::a2e:20: tiempo=22ms

Estadísticas de ping para fd00:7d03:7d03:7d03::a2e:20:
  Paquetes: enviados = 4, recibidos = 4, perdidos = 0
    (0% perdidos),
  Tiempos aproximados de ida y vuelta en milisegundos:
    Mínimo = 21ms, Máximo = 22ms, Media = 21ms
```

*Ilustración 70 Ping desde PC a nodo MED*

Se observa que ambos Dongles a 1 salto de distancia del BR están alrededor de 20 ms de latencia respecto al BR, estando los 2 a una distancia similar del BR. Se comprueban los ping en dirección contraria, desde los Dongles al PC y desde los Dongles a la web de [www.kirale.com](http://www.kirale.com), comprobando que en ambos nodos recibimos la respuesta a los ping realizados.

#### 5.3.4. ENVÍO DE MENSAJES UDP POR SOCKETS

En esta prueba, se ha probado el envío de Sockets entre ambos Dongles y entre PC y Dongles. La topología de red utilizada se muestra en Ilustración 71:

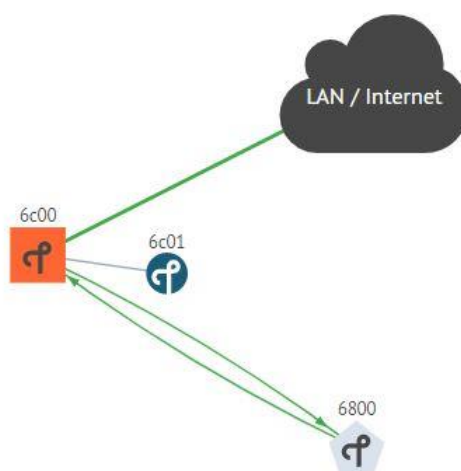


Ilustración 71 Topología para envío de mensajes UDP vía Sockets

Para esto, como se ha mencionado en 5.2.3 ENVÍO DE MENSAJES UDP ENTRE AMBOS NODOS se deberá tener en cuenta el asignar una dirección IP conocida previamente, la cual debe empezar con el prefijo configurado en BR y terminar con **::a2e:XX**. En cuanto al comando `Route()`, ya no hará falta hacerlo, debido a que se dispone de un router (el BR) que hace automáticamente este enrutado.

Siguiendo el mismo proceso anteriormente realizado, se comprueba que los sockets llegan correctamente de un nodo a otro pasando por BR como punto intermedio, por lo que se darían dos saltos.

Finalmente, con un script de Python se ha probado a enviar mensajes UDP vía Sockets. Gracias a la herramienta KiTools, se comprueba el correcto funcionamiento envío a los dos Dongles y que estos los reciben correctamente. Para la lectura por vía UART debe implementarse un código de lectura por interrupciones para poder detectar estas notificaciones. Aunque por interrupciones sea el método más idóneo y correcto, para las pruebas se puede probar dicha recogida con la actual función `receive(huart X)` implementada en el microcontrolador ARM.

Como última prueba de envío de mensajes, se añade al código del microcontrolador ARM que maneja los dos Dongles, una rutina para enviar mensajes UDP a través de Sockets desde el nodo MED al nodo Leader cada 10 segundos, mientras que desde el PC se ejecuta un script de Python el cuál envía mensajes UDP por Sockets a ambos nodos cada 2 segundos. Esta prueba se hace con una duración de 2 horas y media, observando así la estabilidad de la red ante un largo periodo de envío de mensajes. Tras realizar esta prueba con la misma duración varias veces, se observa que la red suele mantener una estabilidad alta, a pesar de la cual hay ciertos problemas en algunas ocasiones, en las que la red se satura y se dejan de recibir mensajes del nodo MED en el nodo Leader durante un tiempo.

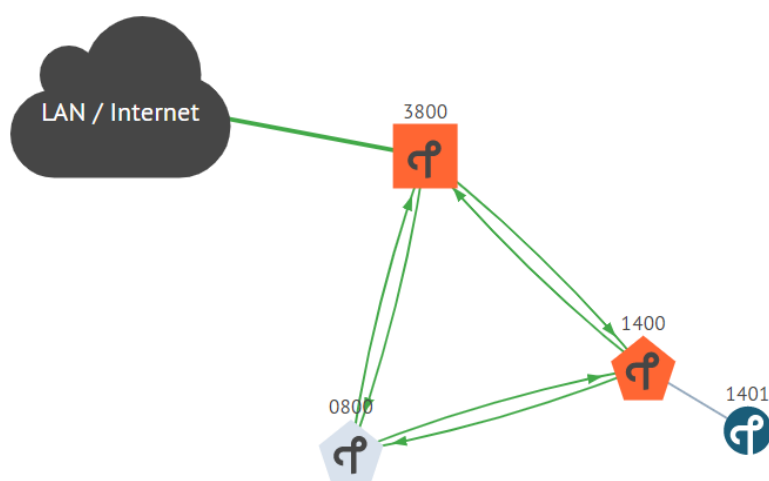
También durante esta última prueba, se va observando la temperatura de ambos Dongles con el comando *show uptime* con la herramienta KiTools, se observa que la temperatura del módulo KTWM102 integrado en el Dongle se mantiene constante alrededor de los 33°C durante toda la prueba.

## 5.4. PRUEBAS CON PCB COOKIE THREAD COMO CUARTO NODO

Una vez las primeras pruebas con solo los 2 Dongles y el BR, se integra el módulo KTWM102 integrado en la PCB Cookie Thread diseñada. Para unas primeras pruebas del correcto funcionamiento se prueba este nuevo nodo con la herramienta KiTools conectándolo por USB al PC, como los Dongles.

Para estas pruebas se han usado dos topologías:

- 1) La primera, configurando 1 Dongle como Leader, el otro como REED o router y el BR como REED o router. Finalmente, el nuevo nodo se configura como MED. Quedando la topología mostrada en Ilustración 72



*Ilustración 72 Topología 4 nodos con un Dongle como LEADER*

Se configura el BR como Leader de la red, un Dongle se configura como REED o Router y el otro Dongle como MED. Finalmente, el nuevo nodo se configura como MED. La topología resultante se muestra en Ilustración 73.

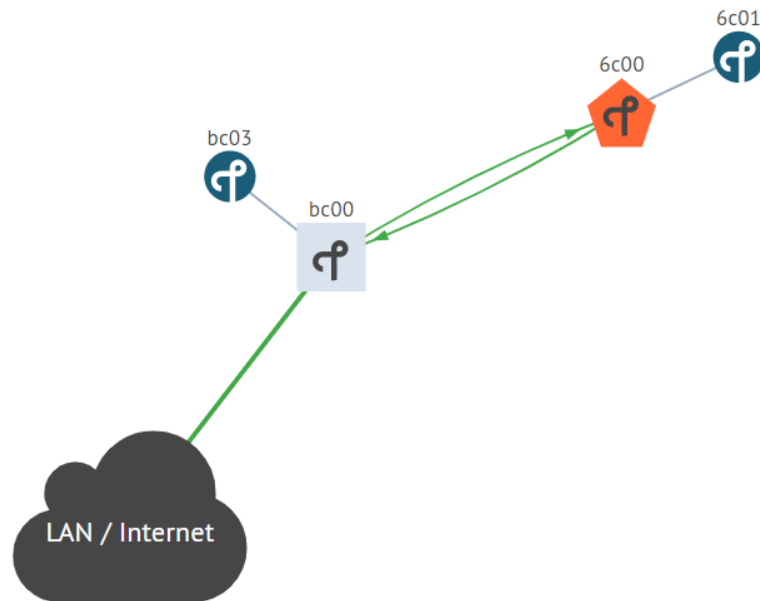


Ilustración 73 Topología 4 nodos con BR como Leader

#### 5.4.1. PRUEBAS DE CONECTIVIDAD CON EL CUARTO NODO

Una vez formada la red en cualquiera de sus dos topologías con 4 nodos, queda el nodo nuevo, formado por la PCB diseñada con el KTWM102, a 2 saltos de distancia del BR. Para probar la conectividad de este nodo, se realiza un ping desde el PC a este nuevo nodo y comprobamos que el resultado medio es de aproximadamente 46 ms:

```
C:\Users\j_con>ping -6 fd00:7d03:7d03:7d03::a2e:1e

Haciendo ping a fd00:7d03:7d03:7d03::a2e:1e con 32 bytes de datos:
Respuesta desde fd00:7d03:7d03:7d03::a2e:1e: tiempo=46ms
Respuesta desde fd00:7d03:7d03:7d03::a2e:1e: tiempo=46ms
Respuesta desde fd00:7d03:7d03:7d03::a2e:1e: tiempo=46ms
Respuesta desde fd00:7d03:7d03:7d03::a2e:1e: tiempo=47ms

Estadísticas de ping para fd00:7d03:7d03:7d03::a2e:1e:
    Paquetes: enviados = 4, recibidos = 4, perdidos = 0
    (0% perdidos),
    Tiempos aproximados de ida y vuelta en milisegundos:
    Mínimo = 46ms, Máximo = 47ms, Media = 46ms
```

Ilustración 74 Ping desde PC a nuevo nodo MED.

Teniendo en cuenta que para nodo a un solo salto del BR hay una media de 21-22 ms de ping, podemos ver, que para este nuevo salto se incrementa alrededor de 24 ms.

**Nota:** Tener en cuenta que los dos saltos realizados son entre nodos separados la misma distancia de 30 – 35 cm de distancia.

Desde el propio nodo se hace ping a [www.kirale.com](http://www.kirale.com), a [www.google.com](http://www.google.com), al PC y a los diferentes nodos, y desde todos los puntos se recibe correctamente la respuesta al ping.

### 5.4.2. ENVÍO / RECIBO DE SOCKETS

Siguiendo con el uso de las dos topologías mencionadas en 5.4 PRUEBAS CON PCB COOKIE THREAD COMO CUARTO NODO se prueba la interacción de Sockets. Para el recibo de Sockets se usan los modos de debug de la herramienta KiTools, de la misma manera que se ha usado anteriormente.

En las primeras pruebas se ha probado al envío de Socket entre los diferentes nodos y entre PC y nodos, probando las capacidades multisalto de las redes THREAD. En esta primera parte se ha dejado el nodo de PCB solo recibiendo Sockets, sin ningún envío, comprobándose que se recibían estos Sockets desde la herramienta KiTools.

En estas pruebas se ve que los envíos de sockets entre distintos nodos llegan correctamente sin pérdidas, o muy ínfimas.

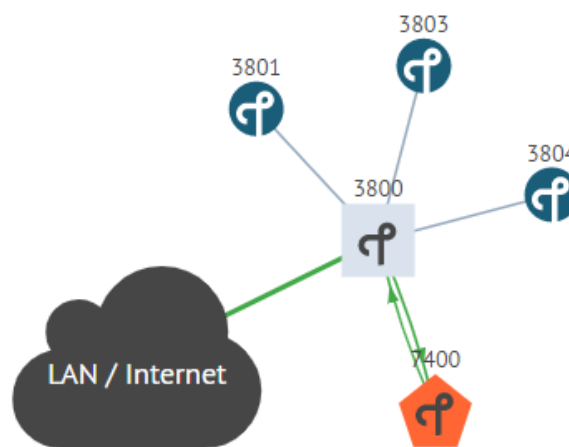
Como segunda prueba, con la segunda topología de red indicada en 5.4 PRUEBAS CON PCB COOKIE THREAD COMO CUARTO NODO, se ejecuta desde el microcontrolador ARM una rutina para enviar mensajes UPD a través de Sockets desde el Dongle con Rol MED al Dongle con Rol REED cada 10 segundos mientras que desde el PC se ejecuta un script de Python para enviar mensajes UDP a través de Sockets al Dongle REED y al módulo MED de la PCB Cookie cada 2 segundos. Esta prueba se realiza con una duración de 2 horas y media, varias veces, observando, al igual que 5.3.4 ENVÍO DE MENSAJES UDP POR SOCKETS, que la red es estable salvo en ciertos momentos en los que la comunicación entre nodos se cae. En algunas ocasiones, se observa en la pestaña de Visual Network, dentro del Panel de Administración Web del sistema KiBRA del BR, que, en algunos momentos de la prueba, algún nodo, se desconectaba de la red durante un rato, a partir de lo cual dicho nodo dejaba de enviar y/o recibir mensajes UDP. Para decantar posibles fallos, se prueba la estabilidad de red sin envío de mensajes, y se observa, que todos los nodos permanecen conectados el 100% del tiempo, por lo que puede que se esté saturando la red al enviar información cada tan poco tiempo, provocando que la red se vuelva algo inestable.

### 5.5. PRUEBAS DE ESTABILIDAD CON 5 NODOS

Tras validar el funcionamiento de la primera PCB Cookie Thread, se crea el código para el manejo de los módulos KTWM102 desde el kit de la Cookie y se añade como 5 nodo a la red otra PCB de Cookie Thread, controlada a través de KiTools.

En este caso se han probado dos topologías diferentes, en las cuales los nodos se han configurado de la misma manera: El nodo BR como líder, un nodo Dongle como Router/REED, y el resto como MED. Solo ha cambiado el padre al que se conectaban los diferentes nodos.

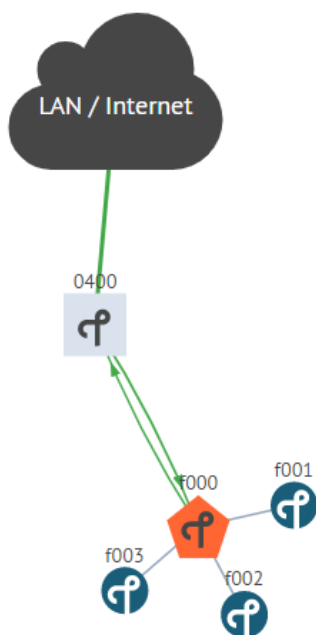
- 1) Todos los nodos se han conectado al BR como nodo padre. Ver Ilustración 75.



*Ilustración 75 Topología A de 5 nodos*



- 2) El nodo REED se conecta al BR como nodo hijo, y los MED se conectan al REED. Ver Ilustración 76.



*Ilustración 76 Topología B con 5 nodos*

### 5.5.1. PRUEBAS DE CONECTIVIDAD

Para verificar los datos obtenidos en las pruebas con 4 nodos, se vuelve a verificar la latencia entre el PC y los diferentes nodos.

En el caso de la primera topología, al estar todos a un salto de distancia del BR, se verifica que, para todos los nodos, la latencia aproximada entre el PC y los nodos MED, es de alrededor de 22 o 23 ms.

En el caso de la segunda topología, para el nodo Router, se sigue observando esa latencia de 22 o 23 ms, mientras que a los nodos MED se ha visto que la latencia sigue siendo alrededor de los 45-46 ms, como en las primeras pruebas.

### 5.5.2. ENVÍO / RECIBO DE SOCKETS

En esta prueba se mandan mensajes UDP por Sockets de la siguiente manera:

- Los 2 nodos Dongles (1 REED y 1 MED) envían mensajes a los dos nodos MED formados por las PCBs.
- Nodo formado por la Cookie Completa (PCB Thread + Cookie) envía mensajes al nodo REED formado por el Dongle.
- PC envía mensajes a todos los nodos salvo al BR.
- El último nodo MED, formado por la PCB Cookie Thread, al no disponer de un microcontrolador que envíe los comandos a ejecutar, solo recibirá mensajes.

En el caso de la primera topología, todos los mensajes entre nodos darán dos saltos, el primero al BR y el segundo al nodo correspondiente, mientras que los mensajes recibidos desde el PC, solo darán un único salto dentro de la red Thread, desde el BR al nodo correspondiente. Al no poderse debuguear el BR desde las herramientas KiTools, no se ha podido ver forma sencilla de ver esa gestión de los mensajes que le llegan de un nodo a otro, solo podemos ver el aviso de recibo en el nodo receptor.

En cuanto a la segunda topología, todos los mensajes que se envíen a los nodos MED darán dos saltos, el primero al REED y el segundo al nodo correspondiente. Los mensajes destinados al REED solo darán un salto dentro de la propia red Thread. En los casos de los mensajes con dos saltos dentro de la red Thread, podemos ver que en el nodo intermedio, el nodo REED en este caso, se dan dos mensajes de debug en la herramienta KiTools:

- El primero es un mensaje de RX con la dirección RLOC16 del nodo emisor. En caso de haber más saltos anteriores, la dirección RLOC16 que se muestra en este aviso es la del anterior nodo al actual.
- El segundo es un mensaje de TX con la dirección RLOC16 del siguiente nodo receptor al que manda el mensaje, ya sea para gestionar otro salto o su recepción final.

Con ambas topologías se dejó la prueba de envío y recibo de mensajes durante 2 horas y media cada una, viendo que siempre había una buena estabilidad de red sin ningún cambio en las uniones. Los mensajes enviados eran de 6 bytes desde los Dongles o la PCB o de 13 bytes en el caso de los mensajes enviados a través del PC. Al estar los nodos relativamente cercanos, no había ninguna pérdida de bytes en los diferentes saltos.

## 5.6. PRUEBAS CON 6 NODOS Y CON ENVÍOS MULTISALTO

Viendo que con 5 nodos ya hay buena estabilidad en cuanto a la red, se incrementa el número de nodos a 6 añadiendo una PCB Cookie Thread controlada desde el PC. Se ha configurado la red con el BR como nodo Líder, un Dongle KTDG102 y una PCB Cookie Thread como REEDs o Routers, y las otras dos PCB (una de ellas con la Cookie Completa) y el Dongle restantes como MED.

Para buscar el mayor número de saltos entre nodos MED, se distanciaron tanto el BR, como los dos Routers lo máximo posible entre sí y posteriormente se configuraron los nodos MED con una potencia de transmisión de -17 dBm, para que, al conectarse a la red, solo viera como potencial nodo padre a uno de todos los posibles.

Haciendo esto en los 3 nodos MED, se creó la topología mostrada en Ilustración 77:

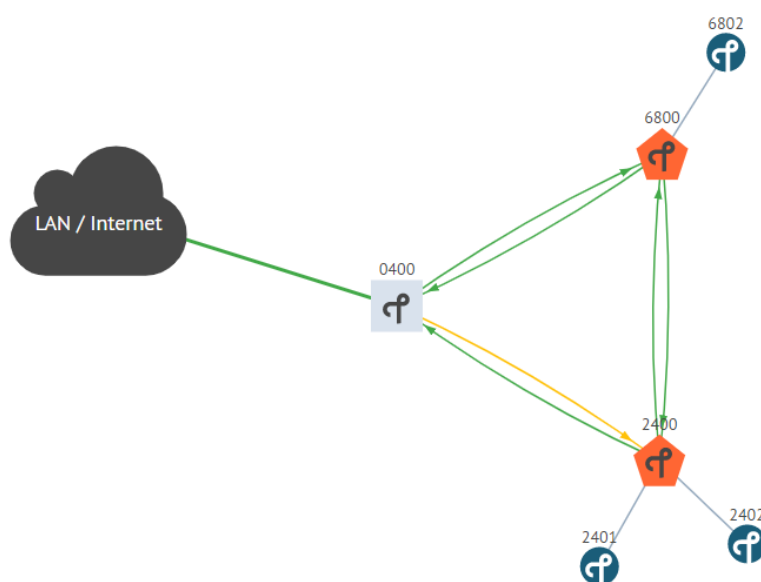


Ilustración 77 Topología con 6 nodos

**Nota:** Las direcciones RLOC16 indicadas en esta topología serán usadas para una mejor identificación de los nodos en el desarrollo y documentación de la prueba. Estas direcciones cambiarán cada vez que se vuelva a iniciar la red, por lo que no se deberán tomar como valores de referencia.

En este caso los dos nodos MED formados por las PCB se unieron a la red como nodos hijo del Router con RLOC16 0x2400, el nodo Router formado por la tercera PCB, mientras que el nodo MED Dongle es el nodo hijo del otro nodo Router Dongle con dirección RLOC16 0x6800.

En esta prueba, las diferentes rutas de envío de mensajes UDP a través de sockets han sido:

- **Nodo MED Dongle (0x6802):** Envía a los 3 nodos de PCB Cookie, teniendo que dar 2 saltos en caso de enviar al nodo REED (0x2400), y 3 saltos a los otros dos nodos MED de PCB (0x2401 y 0x2402).
- **Nodo REED Dongle (0x6800):** Envía a los 3 nodos de PCB, teniendo 1 y 2 saltos respectivamente.
- **Nodo MED Cookie Completa (0x2402):** Envía al nodo REED formado por el Dongle.
- **Nodos MED (0x2401) y REED (0x2400)** de PCB Cookie restantes solo reciben.
- **PC:** Envía a todos los nodos, teniendo 1 y 2 saltos dentro de la red Thread.

Los mensajes enviados han sido de 6 bytes, los enviados desde los nodos, y de entre 13 y 100 bytes en el caso de los enviados desde el PC. Esto se debe a la mayor facilidad del cambio en el tamaño de los mensajes en el caso de los enviados desde el PC, debido a que se evita que al modificar el programa del microcontrolador, el nodo deje no solo de enviar mensajes si no que pueda desconectarse de la red por algún envío erróneo desde el microcontrolador.

Esta prueba se realizó durante 5 horas varias veces. En una de las ocasiones se observa que en ocasiones alguno de los REED cambia su dirección RLOC16. Al cambiar la dirección RLOC16, el enlace con el nodo hijo se pierde y este último se pasa a ser nodo hijo del otro nodo REED, que era más cercano que el BR. Alrededor de los 5 minutos, los nodos que habían cambiado de nodo padre vuelven a conectarse al nodo padre REED inicial. En los dos cambios de nodo padre, los mensajes UDP enviados, se siguen recibiendo correctamente sin ningún problema, por lo que se puede apreciar como la red Thread es capaz de reajustarse correctamente cuando un nodo se cae o se reconecta.

Con el cambio del tamaño de los mensajes enviados desde el PC, se observa que, incluso para mensajes largos de 100 bytes, la red sigue siendo estable durante largo tiempo sin problemas de pérdidas de datos en los nodos receptores.

Al igual que en 5.4.2 ENVÍO / RECIBO DE SOCKETS, se observa que, cuando un mensaje UDP pasa por uno de los nodos intermedios, este nodo intermedio da un aviso por la herramienta KiTools indicando el nodo del que proviene el mensaje y el nodo al que se envía. Por ejemplo:

En el caso de los mensajes enviados desde en nodo MED (0x6802) al nodo MED 0x2401, el mensaje daría 3 saltos para llegar desde el emisor al receptor, teniendo 2 puntos medios, los dos REED.

- En el primer REED por donde pasa el mensaje, nodo 0x6800, indicará que se reciben datos por el RX provenientes del nodo 0x6802, y que se envían por el TX al nodo 0x2400, el siguiente REED.
- En el segundo REED por donde pasa el mensaje, nodo 0x2400, indicará que se reciben datos por el RX provenientes del nodo 0x6800 y se envían por el TX al nodo 0x2401.
- El nodo MED 0x2401, al recibir el mensaje UDP, y al ser el receptor, dará un aviso por UART, en caso de tener un host conectado por puerto UART, con los datos de la IP del emisor (nodo 0x6802) y con el mensaje recibido. También se mostrará por la herramienta KiTools que se han recibido datos por el RX de la dirección IP del emisor y el tamaño en bytes del mensaje recibido.

## 6. CONCLUSIONES Y LÍNEAS FUTURAS

En este capítulo se sacarán las conclusiones del trabajo a la vez que se comentarán futuros posibles desarrollos de este proyecto.

### 6.1. CONCLUSIONES

En este proyecto se ha buscado la viabilidad de la implementación de las comunicaciones THREAD en la plataforma Cookie. Aparte de su viabilidad, se buscaba un diseño simple para integrar el dispositivo KTWM102, para implementar este tipo de comunicaciones y estudiar las diferentes formas de configuración de redes, buscando la configuración idónea para una red THREAD adaptada a las necesidades del proyecto.

Tras las diferentes pruebas realizadas, se ha visto que:

- A pesar de haberse realizado pruebas con topologías de red con una cantidad baja de nodos conectados, se ha buscado una disposición de los dispositivos de manera que, se maximizarán las distancias entre nodo y nodo y con obstáculos físicos dificultando, en parte, la conectividad y las comunicaciones, y se ha observado que las redes THREAD son redes bastante estables y con una buena capacidad multisalto entre nodos con muy escasa pérdida de la información enviada.
- Variedad de opciones en las configuraciones disponibles a la hora de configurar tanto la red como los diferentes nodos que la compondrán.
- Sencillez a la hora de crear la red y de configurar los nodos para su unión gracias a la herramienta KiTools y los comandos KSH, los cuáles son con una interfaz sencilla con lenguaje humano.
- Estabilidad térmica de los Dongles USB (dispositivos KTDG102) y en la PCB diseñada con el KTWM102. Se aumenta 1 o 2 grados la temperatura del KTWM102 a lo largo de pruebas de 2 horas con un alto funcionamiento en los nodos y en ambientes con un impacto directo del sol sobre dichos nodos. Por lo que, en ambientes normales sin mucha carga de trabajo sobre el nodo, no habrá problemas térmicos .

### 6.2. LÍNEAS FUTURAS

Un posible desarrollo futuro para este proyecto es la integración de la capa de sensores a los nodos desarrollados, pudiendo implementar el envío de esa información real entre diferentes nodos.

A su vez podría desarrollarse una plataforma Web donde se puedan enviar, a través de mensajes UDP mediante Sockets, las medidas tomadas por los diferentes nodos y pudiéndose gestionar la red mediante esta Web, ya sea desde la configuración de los tiempos de transmisión de las medidas desde los nodos, el poder ver el estado de cada nodo, etc.

Esta propuesta de añadir una plataforma Web de gestión, ayudaría a una gestión mucho más fácil de los nodos, ya que el panel de Administración Web proporcionado por el Border Router solo aporta como datos el rol y la dirección IP de los nodos conectados, a la vez que la topología de red generada, mientras que con una plataforma Web creada en específico podría mostrar datos como las medidas de los sensores, el estado de la batería de cada nodo, etc.

## 7. BIBLIOGRAFÍA

- [1] S. Li, L. Da Xu, and S. Zhao, "5G Internet of Things: A survey," *J. Ind. Inf. Integr.*, vol. 10, no. February, pp. 1–9, 2018, doi: 10.1016/j.jii.2018.01.005.
- [2] R. M. Gomathi, G. H. S. Krishna, E. Brumancia, and Y. M. Dhas, "A Survey on IoT Technologies, Evolution and Architecture," *2nd Int. Conf. Comput. Commun. Signal Process. Spec. Focus Technol. Innov. Smart Environ. ICCSP 2018*, no. Iccsp, pp. 1–5, 2018, doi: 10.1109/ICCSP.2018.8452820.
- [3] L. Da Xu, W. He, and S. Li, "Internet of things in industries: A survey," *IEEE Trans. Ind. Informatics*, vol. 10, no. 4, pp. 2233–2243, 2014, doi: 10.1109/TII.2014.2300753.
- [4] A. J. Albarakati, J. Qayyum, and K. a. Fakeeh, "A Survey on 6LoWPAN & its Future Research Challenges," *Int. J. Comput. Sci. Mob. Comput.*, vol. 3, no. 10, pp. 558–570, 2014.
- [5] Y. Chen *et al.*, "6LoWPAN stacks: A survey," *7th Int. Conf. Wirel. Commun. Netw. Mob. Comput. WiCOM 2011*, pp. 6–9, 2011, doi: 10.1109/wicom.2011.6040344.
- [6] Z. Shelby and C. Bormann, *6LoWPAN: The Wireless Embedded Internet*, no. c. 2009.
- [7] Z. Yang and C. H. Chang, "6LoWPAN overview and implementations," *Proc. 2019 Int. Conf. Embed. Wirel. Syst. Networks*, pp. 357–361, 2019, [Online]. Available: <https://dl.acm.org/doi/10.5555/3324320.3324409>.
- [8] W. Rzepecki, L. Iwanecki, and P. Ryba, "IEEE 802.15.4 thread mesh network - Data transmission in harsh environment," *Proc. - 2018 IEEE 6th Int. Conf. Futur. Internet Things Cloud Work. W-FiCloud 2018*, pp. 42–47, 2018, doi: 10.1109/W-FiCloud.2018.00013.
- [9] Thread Group, "Thread 1.1. 1 Specification." Thread Group, 2017.
- [10] Kirale, "KTWM102 Thread NCP module." pp. 1–17, [Online]. Available: [www.kirale.com](http://www.kirale.com).
- [11] Kirale, "Kirale Command-Line Shell," 2019.
- [12] I. Unwala, Z. Taqvi, and J. Lu, "Thread: An IoT protocol," *IEEE Green Technol. Conf.*, vol. 2018-April, pp. 161–167, 2018, doi: 10.1109/GreenTech.2018.00037.
- [13] J. Gopaluni, I. Unwala, J. Lu, and X. Yang, "Implementation of GUI for open thread," *Proc. - 2018 Int. Conf. Comput. Sci. Comput. Intell. CSCI 2018*, pp. 1015–1018, 2018, doi: 10.1109/CSCI46756.2018.00196.
- [14] I. Unwala, Z. Taqvi, and J. Lu, "IoT security: ZWave and thread," *IEEE Green Technol. Conf.*, vol. 2018-April, pp. 176–182, 2018, doi: 10.1109/GreenTech.2018.00040.
- [15] IEEE Std 802.15.4, "IEEE Standards, Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs)," 2006.
- [16] "RFC 4944, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks"," 2007. <https://tools.ietf.org/html/rfc4944>.
- [17] "RFC 6282 'Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks,'" 2019. <https://tools.ietf.org/html/rfc6282>.
- [18] "RFC 2460 'Internet Protocol, Version 6 (IPv6) Specification,'" 1998. .
- [19] "RFC 4291 'IP Version 6 Addressing Architecture,'" 2006.

- <https://www.ietf.org/rfc/rfc4291>.
- [20] “RFC 4443 ‘Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification,’” 2006. <https://tools.ietf.org/html/rfc4443>.
  - [21] “RFC 768 ‘ User Datagram Protocol,’” 1980. <https://tools.ietf.org/html/rfc768>.
  - [22] “RFC 1122 ‘Requirements for Internet Hosts -- Communication Layers,’” 1989. <https://tools.ietf.org/html/rfc1122>.
  - [23] M. B. Tamboli and D. D. Ambawade, “Secure and efficient CoAP based authentication and access control for internet of things (IoT),” *2016 IEEE Int. Conf. Recent Trends Electron. Inf. Commun. Technol. RTEICT 2016 - Proc.*, pp. 1245–1250, 2017, doi: 10.1109/RTEICT.2016.7808031.
  - [24] C. Bormann, A. P. Castellani, and Z. Shelby, “CoAP: An application protocol for billions of tiny internet nodes,” *IEEE Internet Comput.*, vol. 16, no. 2, pp. 62–67, 2012, doi: 10.1109/MIC.2012.29.
  - [25] “RFC 6347 ‘Datagram Transport Layer Security Version 1.2,’” 2012. <https://tools.ietf.org/html/rfc6347>.
  - [26] “RFC 7250 ‘Using Raw Public Keys in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS),” 2014. <https://tools.ietf.org/html/rfc7250>.
  - [27] “RFC 5246 ‘The Transport Layer Security (TLS) Protocol Version 1.2,’” 2008. <https://tools.ietf.org/html/rfc5246>.