# AGRADECIMIENTOS

En mi primer lugar, me gustaría agradecer a mis tutores de proyecto Jorge Portilla y Gabriel 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

El objetivo principal de este proyecto es la implementación hardware de la tecnología Thread en la plataforma Coockie 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 y con el entorno formado por los módulos de Kirale con los Dongle de Evaluación disponibles.

Posteriormente a la familiarización con la tecnología y a los comandos que posteriormente se utilizarán en la plataforma Coockie, 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 Coockie, a través del puerto UART. Para esta validación se formaban diferentes tipos de redes 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.

**Palabras clave:** Implementación, Thread, validación, análisis

# ABSTRACT

El objetivo principal de este proyecto es la implementación hardware de la tecnología Thread en la plataforma Coockie 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 y con el entorno formado por los módulos de Kirale con los Dongle de Evaluación disponibles.

Posteriormente a la familiarización con la tecnología y a los comandos que posteriormente se utilizarán en la plataforma Coockie, 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 Coockie, a través del puerto UART. Para esta validación se formaban diferentes tipos de redes 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.

**Palabras clave:** Implementación, Thread, validación, análisis

# INDICE

[AGRADECIMIENTOS I](#_Toc81761707)

[RESUMEN III](#_Toc81761708)

[ABSTRACT V](#_Toc81761709)

[INDICE VI](#_Toc81761710)

[ABREVIATURAS Y ACRÓNIMOS XI](#_Toc81761711)

[ILUSTRACIONES XIII](#_Toc81761712)

[ECUACIONES XV](#_Toc81761713)

[TABLAS XVI](#_Toc81761714)

[CÓDIGOS XVII](#_Toc81761715)

[1. INTRODUCCIÓN Y OBJETIVOS DEL TRABAJO 2](#_Toc81761716)

[2. ESTADO DEL ARTE 3](#_Toc81761717)

[2.1. INTERNET OF THINGS (IoT) 3](#_Toc81761718)

[2.1.1. INTRODUCCIÓN [2] 3](#_Toc81761719)

[2.1.2. COMPONENTES [2] 4](#_Toc81761720)

[2.1.2.1. DISPOSITIVO 4](#_Toc81761721)

[2.1.2.2. RED LOCAL 4](#_Toc81761722)

[2.1.2.3. INTERNET 5](#_Toc81761724)

[2.1.2.4. SERVICIOS BACKEND 5](#_Toc81761725)

[2.1.2.5. APLICACIONES 5](#_Toc81761726)

[2.1.3. SENSORES 5](#_Toc81761727)

[2.1.4. ARQUITECTURA [3] 6](#_Toc81761728)

[2.1.4.1. CAPA DE SENSORIZADO 6](#_Toc81761729)

[2.1.4.2. CAPA DE RED 6](#_Toc81761730)

[2.1.4.3. CAPA DE SERVICIO 7](#_Toc81761731)

[2.1.4.4. CAPA DE INTERFAZ 7](#_Toc81761732)

[2.2. 6LoWPAN [4] 8](#_Toc81761733)

[2.2.1. SIGNIFICADO 6LoWPAN [4] 8](#_Toc81761734)

[2.2.2. PILA DE PROTOCOLOS DE 6LoWPAN [4]–[6] 9](#_Toc81761735)

[2.2.3. PRINCIPALES CARACTERÍSTICAS DE 6LoWPAN [7] 10](#_Toc81761736)

[2.2.3.1. COMPRESIÓN DE CABECERA 10](#_Toc81761737)

[2.2.3.2. ENRUTAMIENTO 10](#_Toc81761738)

[2.2.3.3. SEGURIDAD 10](#_Toc81761739)

[2.2.3.4. PROTOCOLOS DE APLICACIÓN 11](#_Toc81761740)

[2.2.4. RETOS DE 6LoWPAN 11](#_Toc81761741)

[2.2.5. IMPLEMENTACIONES Y APLICACIONES PARA 6LoWPAN [5], [7] 12](#_Toc81761742)

[2.2.5.1. Contiki 12](#_Toc81761743)

[2.2.5.2. TINYOS 12](#_Toc81761744)

[2.2.5.3. THREAD 13](#_Toc81761745)

[2.3. THREAD 13](#_Toc81761746)

[2.3.1. INTRODUCCIÓN A REDES THREAD [8], [10]–[12] 13](#_Toc81761747)

[2.3.2. TIPOS DE DISPOSITIVOS 15](#_Toc81761748)

[2.3.3. PROTOCOLO THREAD 16](#_Toc81761749)

[2.3.3.1. REDES DE ÁREA PRIVADA - PRIVATE AREA NETWORK (PAN) [12] 17](#_Toc81761750)

[2.3.3.2. CAPA FÍSICA [12], [14] 19](#_Toc81761751)

[2.3.3.3. CAPA MAC (O ENLACE DE DATOS). [12], [14] 19](#_Toc81761752)

[2.3.3.4. CAPA DE ADAPTACIÓN 6LoWPAN [12] 21](#_Toc81761753)

[2.3.3.5. CAPA DE RED [12] 21](#_Toc81761754)

[2.3.3.6. PROTOCOLO DE ENRUTAMIENTO 22](#_Toc81761755)

[2.3.3.7. CAPA DE TRANSPORTE [12] 22](#_Toc81761756)

[2.3.3.8. SEGURIDAD Y COMMISSIONING DE DISPOSITIVOS [10], [12] 23](#_Toc81761757)

[3. ANALISIS DE LA TECNOLOGÍA THREAD 25](#_Toc81761758)

[3.1. ANÁLISIS INICIAL DE LA TECNOLOGÍA THREAD 25](#_Toc81761759)

[3.1.1. CARACTERÍSTICAS TÉCNOLOGICAS DE LOS MÓDULOS KIRALE 26](#_Toc81761760)

[3.1.1.1. CARACTERÍSTICAS MÓDULO RF KTWM102 26](#_Toc81761761)

[3.1.1.2. CARACTERÍSTICAS KTDG102 EVALUATION DONGLE 26](#_Toc81761762)

[3.1.1.3. CARACTERÍSTICAS DEL BORDER ROUTER KTBRN1 26](#_Toc81761763)

[3.1.1.4. CARACTERÍSTICAS KINOS – NETWORK OS 27](#_Toc81761764)

[3.1.2. EJEMPLOS DE OTROS DISPOSITIVOS 27](#_Toc81761765)

[3.2. CONFIGURACIONES INICIALES 28](#_Toc81761766)

[3.2.1. CONFIGURACIÓN DEL BORDER ROUTER 28](#_Toc81761767)

[3.2.1.1. REQUERIMIENTOS 28](#_Toc81761768)

[3.2.1.2. GUÍA DE INSTALACIÓN 28](#_Toc81761769)

[3.2.1.2.1. DESCARGA DEL SOFTWARE REQUERIDO 28](#_Toc81761770)

[3.2.1.2.2. FLASHEAR LA IMAGEN EN LA TARJETA SD 29](#_Toc81761771)

[3.2.1.2.3. PRIMERA INSTALACIÓN 30](#_Toc81761772)

[3.2.1.2.3.1. CONEXIÓN VÍA PUERTO USB SERIE 30](#_Toc81761773)

[3.2.1.3. PANEL DE ADMINISTRACIÓN WEB 32](#_Toc81761774)

[3.2.1.3.1. CAMBIAR LA CONFIGURACIÓN DE RED 33](#_Toc81761775)

[3.2.1.3.2. ACTUALIZAR KiBRA 34](#_Toc81761776)

[3.2.1.3.3. CONFIGURAR BORDER ROUTER 35](#_Toc81761777)

[3.2.1.3.3.1. UNIRSE O FORMAR UNA RED THREAD 35](#_Toc81761778)

[3.2.1.3.3.2. BACKBONE ROUTER SERVER (BBR). 36](#_Toc81761779)

[3.2.1.3.3.3. PREFIJO DE RED (NETWORK PREFIX) 37](#_Toc81761780)

[3.2.1.3.4. INICIO DEL BORDER ROUTER (START-UP). 38](#_Toc81761781)

[3.2.1.3.5. SERVICIOS 42](#_Toc81761782)

[3.2.1.3.5.1. SERVIDOR BACKBONE ROUTER 42](#_Toc81761783)

[3.2.1.3.5.2. DHCP 43](#_Toc81761784)

[3.2.1.3.5.3. NAT64 44](#_Toc81761785)

[3.2.1.3.5.4. COMMISSIONER 45](#_Toc81761786)

[3.2.1.3.6. VISUAL NETWORK 46](#_Toc81761787)

[3.2.1.3.7. LOGS 47](#_Toc81761788)

[3.2.1.4. BREVE RESUMEN 48](#_Toc81761789)

[3.2.1.4.1. SISTEMA DE FICHEROS AVANZADO 48](#_Toc81761790)

[3.2.1.4.2. SERVICIOS CRÍTICOS 48](#_Toc81761791)

[3.2.1.4.3. COMUNICACIÓN ENTRE PROCESOS 49](#_Toc81761792)

[3.2.2. CONFIGURACIÓN INICIAL MÓDULO KTWM102 50](#_Toc81761793)

[3.2.2.1. INSTALACIÓN DE DRIVERS USB Y DEL BOOTLOADER 50](#_Toc81761794)

[3.2.2.1.1. WINDOWS 51](#_Toc81761795)

[3.2.2.1.2. LINUX / MAC OS 53](#_Toc81761796)

[3.2.2.2. INSTALL “DFU-UTIL” 53](#_Toc81761797)

[3.2.2.3. ACTUALIZACIÓN DE FIRMWARE 54](#_Toc81761798)

[3.2.2.4. RUNTIME – INSTALACIÓN DE DRIVERS USB 55](#_Toc81761799)

[3.2.2.4.1. WINDOWS 55](#_Toc81761800)

[3.2.2.4.2. LINUX 58](#_Toc81761801)

[3.2.2.4.3. MAC OS 59](#_Toc81761802)

[3.2.2.5. CONFIGURACIÓN DE TERMINAL COM 59](#_Toc81761803)

[3.2.2.5.1. WINDOWS 59](#_Toc81761804)

[3.2.2.5.2. LINUX / MAC OS 60](#_Toc81761805)

[3.3. CONFIGURACIÓN DE RED PARA KTWM102 60](#_Toc81761806)

[3.3.1. KIRALE COMMAND-LINE SHELL– COMANDOS KSH 61](#_Toc81761808)

[3.3.1.1. SINTAXIS DE LOS COMANDOS 61](#_Toc81761809)

[3.3.1.2. SINTAXIS DE LOS PARÁMETROS 62](#_Toc81761810)

[3.3.1.3. MENSAJES DE RESPUESTA 63](#_Toc81761812)

[3.3.2. KIRALE BINARY INTERFACE – COMANDOS KBI 64](#_Toc81761813)

[3.3.2.1. OPERACIÓN DE INTERFAZ 64](#_Toc81761814)

[3.3.2.2. FORMATO DEL PAQUETE 64](#_Toc81761815)

[3.3.2.3. REPRESENTACIÓN DE DATOS 66](#_Toc81761818)

[3.3.2.4. COMANDOS Y RESPUESTAS 67](#_Toc81761819)

[3.3.2.5. NOTIFICACIONES 68](#_Toc81761820)

[3.3.3. CONFIGURACIÓN DE RED 68](#_Toc81761821)

[3.3.3.1. MODO OUT-OF-BAND COMMISSIONING DESACTIVADO 68](#_Toc81761824)

[3.3.3.2. MODO OUT-OF-BAND COMMISSIONING ACTIVADO 69](#_Toc81761825)

[4. DISEÑO E IMPLEMENTACIÓN HARDWARE 70](#_Toc81761826)

[4.1. COMPONENTES COMERCIALES UTILIZADOS 70](#_Toc81761827)

[4.1.1. ELEMENTOS HARDWARE UTILIZADOS 70](#_Toc81761828)

[4.1.2. ELEMENTOS SOFTWARE UTILIZADOS 73](#_Toc81761829)

[4.2. DISEÑO DE ESQUEMÁTICO PCB 74](#_Toc81761830)

[4.2.1. JERARQUÍA DEL CIRCUITO 74](#_Toc81761831)

[4.2.2. CIRCUITO DE ALIMENTACIÓN 75](#_Toc81761832)

[4.2.3. INTEGRACIÓN DEL MÓDULO KTWM102 76](#_Toc81761833)

[4.2.4. COOCKIE CONNECTOR 77](#_Toc81761834)

[4.2.5. USB 78](#_Toc81761835)

[4.3. LAYOUT 79](#_Toc81761836)

[4.3.1. LAYOUT CAPA TOP 80](#_Toc81761837)

[4.3.2. LAYOUT CAPA BOTTOM 81](#_Toc81761838)

[4.3.3. RESULTADO FINAL DEL DISEÑO 82](#_Toc81761839)

[5. PRUEBAS EXPERIMENTALES 83](#_Toc81761840)

[5.1. PRIMERA INTERACCIÓN CON DONGLE USB 83](#_Toc81761841)

[5.2. RED DE DOS NODOS 84](#_Toc81761843)

[5.2.1. CREACIÓN DE LA RED 84](#_Toc81761844)

[5.2.2. PING ENTRE NODOS 85](#_Toc81761846)

[5.2.3. ENVÍO DE MENSAJES UDP A TRAVÉS DE SOCKETS ENTRE AMBOS NODOS 85](#_Toc81761847)

[5.3. PRUEBAS CON EL BORDER ROUTER 86](#_Toc81761848)

[5.3.1. INTRODUCCIÓN A LA CONFIGURACIÓN DEL ROUTER. 86](#_Toc81761849)

[5.3.2. PRUEBA DE CONECTIVIDAD IP ENTRE RED THREAD Y LAN 87](#_Toc81761850)

[5.3.3. RED CON EL BR Y DOS NODOS KTDG102 88](#_Toc81761851)

[5.3.4. ENVÍO DE MENSAJES UDP POR SOCKETS 90](#_Toc81761852)

[5.4. PRUEBAS CON PCB COOCKIE THREAD COMO CUARTO NODO 92](#_Toc81761853)

[5.4.1. PRUEBAS DE CONECTIVIDAD CON EL CUARTO NODO 93](#_Toc81761854)

[5.4.2. ENVÍO / RECIBO DE SOCKETS 94](#_Toc81761855)

[5.5. PRUEBAS DE ESTABILIDAD CON 5 NODOS 95](#_Toc81761856)

[5.5.1. PRUEBAS DE CONECTIVIDAD 96](#_Toc81761857)

[5.5.2. ENVÍO / RECIBO DE SOCKETS 96](#_Toc81761858)

[5.6. PRUEBAS CON 6 NODOS Y CON ENVÍOS MULTISALTO 98](#_Toc81761859)

[6. CONCLUSIONES Y LÍNEAS FUTURAS 100](#_Toc81761860)

[7. BIBLIOGRAFÍA 101](#_Toc81761861)

# 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 |

Tabla 1 Abreviaturas y Acrónimos

# ILUSTRACIONES

[Ilustración 1 Estructura Básica 6LoWPAN [6] 9](#_Toc81762019)

[Ilustración 2 Arquitectura en Contiki [7] 12](#_Toc81762020)

[Ilustración 3 Ejemplo de una topología de Red Thread en malla [9] 16](#_Toc81762021)

[Ilustración 4 Pila de Protocolo Thread en un sistema IoT 17](#_Toc81762022)

[Ilustración 5 Ejemplo PAN Thread [12], [13] 17](#_Toc81762023)

[Ilustración 6 Balena Etcher 29](#_Toc81762024)

[Ilustración 7 Border Router en Administrador de Dispositivos Previo a la instalación de Drivers 30](#_Toc81762025)

[Ilustración 8 Instalación Drivers de Border Router con Zadig 31](#_Toc81762026)

[Ilustración 9 MobaXterm 31](#_Toc81762027)

[Ilustración 10 Login Panel Administración Web 32](#_Toc81762028)

[Ilustración 11 Pestaña Network 33](#_Toc81762029)

[Ilustración 12 Actualización KiBRA 34](#_Toc81762030)

[Ilustración 13 Pestaña Settings 35](#_Toc81762031)

[Ilustración 14 Parámetros a configurar con Out-of-band Commissioning Activado 36](F:\\Jorge\\UPM\\master\\TFM\\Documentos\\TFM_V2.docx" \l "_Toc81762032)

[Ilustración 15 Parámetros a configurar con Out-of-band Commissioning Desactivado 36](F:\\Jorge\\UPM\\master\\TFM\\Documentos\\TFM_V2.docx" \l "_Toc81762033)

[Ilustración 16 Configuración Prefijo de Red 37](#_Toc81762034)

[Ilustración 17 Menú KiBRA - Inicio de Border Router 38](#_Toc81762035)

[Ilustración 18 Menú de KiBRA - Border Router Iniciado 39](#_Toc81762036)

[Ilustración 19 Pestaña Settings - Export Settings 40](#_Toc81762037)

[Ilustración 20 Ventana Export Commissioning Information 41](#_Toc81762038)

[Ilustración 21 Servidor Backbone Router 42](#_Toc81762039)

[Ilustración 22 Servicio DHCP 43](#_Toc81762040)

[Ilustración 23 Servicio NAT64 44](#_Toc81762041)

[Ilustración 24 Servicio Commissioner 45](#_Toc81762042)

[Ilustración 25 Pestaña Visual Network 46](#_Toc81762043)

[Ilustración 26 Pestaña Logs 47](#_Toc81762044)

[Ilustración 27 Administrador de dispositivo antes de instalar los Drivers de KTWM102 50](#_Toc81762045)

[Ilustración 28 Instalación Drivers con Zadig Paso 1 51](#_Toc81762046)

[Ilustración 29 Instalación Drivers con Zadig Paso 2 52](#_Toc81762047)

[Ilustración 30 Instalación Drivers con Zadig Finalizada 52](#_Toc81762048)

[Ilustración 31 Administrador de Dispositivos Después de Instalar Drivers 53](#_Toc81762049)

[Ilustración 32 Instalar libusbk con Zadig 56](#_Toc81762050)

[Ilustración 33 Instalación Driver Libusbk xon Zadig Finalizada 56](#_Toc81762051)

[Ilustración 34 Instalar USB SERIAL (CDC) con Zadig 57](#_Toc81762052)

[Ilustración 35 Instalación Driver USB Serial (CDC) con Zadig Finalizada 57](#_Toc81762053)

[Ilustración 36 Administrador de Dispositivos después de la instalación 58](#_Toc81762054)

[Ilustración 37 Terminal Termite 60](#_Toc81762055)

[Ilustración 38 Herramienta KiTools 62](#_Toc81762056)

[Ilustración 39 IP como parámetros 63](#_Toc81762057)

[Ilustración 40 Esquema de comunicación entre Host Externo y el dispositivo KTWM102 64](#_Toc81762058)

[Ilustración 41 Ejemplo Rutina de Comando - Esperar Respuesta 67](#_Toc81762059)

[Ilustración 42 Kit de Desarrollo STM32F407G-DISC1 70](#_Toc81762060)

[Ilustración 43 KTDG102 Evaluation Dongle 71](#_Toc81762061)

[Ilustración 44 Módulo KTWM102 72](#_Toc81762062)

[Ilustración 45 Border Router 72](#_Toc81762063)

[Ilustración 46 Módulos de Procesamiento y de Alimentación de la Coockie 73](#_Toc81762064)

[Ilustración 47 Jerarquía Circuito 74](F:\\Jorge\\UPM\\master\\TFM\\Documentos\\TFM_V2.docx" \l "_Toc81762065)

[Ilustración 48 Circuito de Alimentación 75](#_Toc81762066)

[Ilustración 49 Circuito Integración del Módulo 76](#_Toc81762067)

[Ilustración 50 Conectores Verticales 77](#_Toc81762068)

[Ilustración 51 Conector USB 78](#_Toc81762069)

[Ilustración 52 Layout Completo 79](#_Toc81762070)

[Ilustración 53 Layout Capa TOP 80](#_Toc81762071)

[Ilustración 54 Layout Capa BOTTOM 81](#_Toc81762072)

[Ilustración 55 PCB Coockie Thread TOP 82](#_Toc81762073)

[Ilustración 56 PCB Coockie Thread Capa BOTTOM 82](#_Toc81762074)

[Ilustración 57 Diagrama de conexión PC - Dongle - uC 83](F:\\Jorge\\UPM\\master\\TFM\\Documentos\\TFM_V2.docx" \l "_Toc81762075)

[Ilustración 58 Esquema montaje Red de Dos Nodos 84](F:\\Jorge\\UPM\\master\\TFM\\Documentos\\TFM_V2.docx" \l "_Toc81762076)

[Ilustración 59 Logs KiTools al REALIZAR un Ping 85](#_Toc81762077)

[Ilustración 60 Logs KiTools al RECIBIR un Ping 85](#_Toc81762078)

[Ilustración 61 Montaje Border Router 86](#_Toc81762079)

[Ilustración 62 Topología de Red 1 nodo con BR 87](#_Toc81762080)

[Ilustración 63 Topología 1 de BR con 2 nodos Dongle 88](#_Toc81762081)

[Ilustración 64 Topología 2 de BR con dos nodos Dongle 89](#_Toc81762082)

[Ilustración 65 Ping desde PC a BR 89](#_Toc81762083)

[Ilustración 66 Ping desde PC a nodo LEADER 89](#_Toc81762084)

[Ilustración 67 Ping desde PC a nodo MED 90](#_Toc81762085)

[Ilustración 68 Topología para envío de mensajes UDP vía Sockets 90](#_Toc81762086)

[Ilustración 69 Topología 4 nodos con un Dongle como LEADER 92](#_Toc81762087)

[Ilustración 70 Topología 4 nodos con BR como Leader 93](#_Toc81762088)

[Ilustración 71 Ping desde PC a nuevo nodo MED. 93](#_Toc81762089)

[Ilustración 72 Topología A de 5 nodos 95](#_Toc81762090)

[Ilustración 73 Topología B con 5 nodos 96](#_Toc81762091)

[Ilustración 74 Topología con 6 nodos 98](#_Toc81762092)

# ECUACIONES

# TABLAS

[Tabla 1 Abreviaturas y Acrónimos XII](#_Toc81762093)

[Tabla 2 Diferencias M2M – IoT [2] 3](#_Toc81762094)

[Tabla 3 MAC Layer Frame 20](#_Toc81762095)

[Tabla 4 Servicios KBRNT1 48](#_Toc81762096)

[Tabla 5 Comandos Servicio KiBRA 49](#_Toc81762097)

[Tabla 6 Servicio Ajenti 49](#_Toc81762098)

[Tabla 7 Listar dispositivos con dfu-util 54](#_Toc81762099)

[Tabla 8 Actualizar Firmware en dispositivos KTWM102 55](#_Toc81762100)

[Tabla 9 Comando para listar Puertos Serie en Sistemas Linux 59](#_Toc81762101)

[Tabla 10 Comando para listar Puertos Serie en Sistemas en MAC OS 59](#_Toc81762102)

[Tabla 11 Abrir terminal Picocom en Linux / MAC Os 60](#_Toc81762103)

[Tabla 12 Sintaxis comandos KSH 61](#_Toc81762104)

[Tabla 13 Formato del Paquete 65](#_Toc81762105)

[Tabla 14 Bits Byte Type 65](#_Toc81762106)

[Tabla 15 Significado Bits Byte Type 66](#_Toc81762107)

# CÓDIGOS

# INTRODUCCIÓN Y OBJETIVOS DEL TRABAJO

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 Coockie para WSN de la tecnología Thread.

La motivación principal de este proyecto es la implementación hardware de la tecnología THREAD en la plataforma Coockies, 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 Coockies. 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.

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.

# ESTADO DEL ARTE

En los últimos años, se han ido desarrollando cada vez más las tecnologías para Internet of Things, 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 área 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.

## 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. A pesar de esto, el 4G ha mejorado algunas de las capacidades de las redes móviles, dando a los dispositivos IoT acceso a Internet. Dentro de la conectividad 4G hay varios tipos de tecnologías como BLE, WiMaxb, LTE, Zigbee, LoRa, NB-IoT, etc. [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.

También existe otra 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 pueden estar 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 hardware que llevan tanto al desarrollo de una infraestructura IoT como en 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.

### COMPONENTES [2]

Los principales componentes funcionales de las tecnologías IoT son:

* Componente para la interacción y comunicación con otros IoT.
* Componente para operaciones de procesado 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

Cada dispositivo IoT tiene una identidad única que lo hace distinguible respecto al resto. Es un dispositivo que cuenta con sistemas hardware y software y se conecta a una red de con otros dispositivos con los que intercambiar datos. 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.

Los dispositivos restringidos son pequeños, por lo que sus características son limitadas y necesitan de un dispositivo que haga de Gateaway para poder conectarse a servidores externos a la red.

Los dispositivos estándar son similares a pequeños ordenadores y dirigen los datos directamente a la nube de la red sin necesidad de un Gateaway.

#### RED LOCAL

Las redes IoT son usadas para transmitir las señales recogidas en los sensores con el resto de diferentes componentes, como los routers, puentes, LAN, WAN y MAN, por lo que se genera una gran cantidad de datos. Para la conectividad de las redes se puede utilizar cualquiera de las tecnologías disponibles como el Wi-Fi, LTE, etc. Para la elección de esa conectividad se deben seguir las siguientes bases:

* 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.

#### SERVICIOS BACKEND

Los Servicios Backend se definen como un conjunto de servicios basados en la nube que ayudan a construir la aplicación IoT o aplicación Backend. Tienen una alta tase de almacenamiento de datos y una fácil gestión de usuarios, lo que proporciona una solución a empresas dedicadas a IoT. Una vez se recolectan los datos, se envían a la nube, donde se guardarán en servidores que permitirán al usuario la 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.

#### APLICACIONES

Las principales aplicaciones para IoT son:

* Smart Home
* Accesorios
* Coches conectados
* Smart Cities
* IoT en la Agricultura

### 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. Sensores que 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 cuyo 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.

### 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 cuál se ha pensado desde el punto de vista de las funcionalidades.

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

#### 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.

#### 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 requerimientos 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, para desplegar, manejar y organizar (scheduling) comportamientos de los dispositivos 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.

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

Esta capa se apoya en la tecnología del Middleware que aporta funcionalidades para integrar de manera correcta servicios y aplicaciones en IoT. La tecnología Middleware provee a IoT con una plataforma coste-eficiencia, donde pueden ser reusadas las plataformas Hardware and Software. La principal actividad de esta capa implica las especificaciones de servicio para middleware. Una capa de servicio bien diseñada será capaz de identificar los requerimientos comunes de aplicación y dar APIs y protocolos para soportar los servicios, aplicaciones y necesidades de usuario requeridas. Además, se procesan todos los problemas orientados al servicio, incluyendo el intercambio y almacenamiento de información, gestión de datos, comunicación y motor de búsqueda. Los componentes de esta capa son:

* **Descubrimiento de Servicio**: Encontrar objetos que puedan aportar los servicios requeridos y la información deseada de manera eficiente.
* **Composición de Servicio**: 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 la 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 IoT hay muchos dispositivos involucrados, los cuáles son de diferentes proveedores y no siempre usan los mismos estándares o protocolos, por lo que suele haber problemas en las comunicaciones e intercambios de información entre los dispositivos. Cómo el número de dispositivos sigue creciendo, este problema se incrementa y se hace más complejo. Va surgiendo la necesidad de una capa de Interfaz que simplifique la gestión y la interconexión de los distintos 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.

Tradicionalmente, la capa de servicio daba una API universal a las aplicaciones. Sin embargo, con los nuevos resultados de investigaciones sobre SOA-IoT, se ha visto que el Proceso de provisionamiento de servicios (SPP de sus siglas en inglés) 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. Basados en el contexto de aplicación y la información QoS, todas las instancias de servicios se clasifican y un mecanismo de provisionamiento según demanda se usará para identificar los servicios que encajen con los requerimientos de la aplicación. Finalmente se evalúa el proceso.

## 6LoWPAN [4]

Muchas de las soluciones patentadas, como ZigBee, están vinculadas verticalmente o perpendicularmente 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.

### SIGNIFICADO 6LoWPAN [4]

6LoWPAN es la abreviación de IPv6 over Low power Wireless Personal Area Networks

Vayamos ahora a desglosar 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 **232** direcciones (4,294,967,296 direcciones posibles IP en Internet), **IPv6** ofrece **2128** direcciones posibles (3.4x1038 direcciones).

1. 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.

1. WPAN en 6LoWPAN

**WPAN** corresponde a Wireless Personal Area Networks. Una WPAN es la red personal en área que conecta los dispositivos de una persona. 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.

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

El concepto básico de la pila del 6LoWPAN, se muestra a continuación en Ilustración 1:

![Tabla

Descripción generada automáticamente]()

Ilustración 1 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, ya que esta 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 enmallada.

### PRINCIPALES CARACTERÍSTICAS DE 6LoWPAN [7]

Como se ha indicado tanto en 2.2.1 como en Ilustración 1*,* 6LoWPAN es una capa de adaptación IoT IPv6, la cual está sobre el estándar IEEE 802.15.4.

Este estándar 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. Este estándar 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 necesitará una capa de adaptación, la cual comprimirá la cabecera de 40 a 7 bytes. Cabeceras de extensión IPv6 se comprimirán de igual manera, como las cabeceras UDP.

#### ENRUTAMIENTO

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 “manejadas” por uno o pocos “supernodos” o Border Routers en el caso de 6LoWPAN.

#### SEGURIDAD

Al ser IEEE 802.15.4 un protocolo de capas de enlace y física inalámbricas, será conveniente 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. Para asegurar dicha seguridad end-to-end, como en Internet, se necesita 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, el cuál es adecuado para redes limitadas como es el caso de 6LoWPAN.

#### PROTOCOLOS DE APLICACIÓN

Actualmente hay muchos protocolos de aplicación disponibles para 6LoWPAN, pero los dos más populares son Constrained Applicacion 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 con limitaciones, 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.

### RETOS DE 6LoWPAN

Debido a las limitaciones de recursos que tienen los nodos, tanto de energía, como de memoria y de potencia de procesamiento, hay varios retos a la hora del diseño e implementación de 6LoWPAN, puesto que este implementa IPv6. Algunos de los principales retos son:

* **Sobrecarga de cabecera:** La capa de enlace en 6LoWPAN tiene una Unidad Máxima de Transferencia (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 debido al alto ratio cabecera/payload y se generan frecuentes fragmentaciones y desfragmentaciones.
* **Neighbour Discovery Protocol:** IPv6 usa este protocolo NDP, el cual se encarga 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. Esto formará una dirección de 128 bytes. NDP ayuda bastante 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 todo tipo de redes, la seguridad siempre es un asunto importante. 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.

### 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 hibrido 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 manejo de eventos, se soporta una carga dinámica y reemplazamiento de servicios o programas individualmente. La arquitectura utilizada es la mostrada a continuación en Ilustración 2:

Tabla

Descripción generada automáticamente

Ilustración 2 Arquitectura en Contiki [7]

#### TINYOS

Otro sistema basado en Unix, 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 Neighbor Discovery*, en una versión reducida, configurar 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 se explicará y analizará más en profundidad en el apartado 2.3 THREAD.

dffd

## 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]

### 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 fluidos 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 dar operar segura y fiablemente incluso con fallos o perdidas en dispositivos individualmente.
* **Baja potencia.** Los dispositivos suelen operar durante años con baterías tipo AA gracias al uso de un ciclo duty ajustado.

### 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]

* **Border Routers.** Es un router específico que da conectividad desde 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.
* **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).
* **Router-Eligible End Devices.** Estos dispositivos pueden convertirse en Routers acorde a la topología o condiciones de la red, pero no actúan como 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 cambió a Routers es manejado por la propia red automáticamente sin necesidad de ser manejado por ningún usuario.
* **Sleepy End Devices.** Son dispositivos más sencillos. 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 ya 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.** Funcionalidad descrita en **Border Routers**.
  + **Active Router.** Funcionalidad descrita en **Routers**.
  + **REEDs.** Funcionalidad descrita en **Router-Eligible End Devices**. 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**.

Imagen que contiene Gráfico

Descripción generada automáticamente

Ilustración 3 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.

### PROTOCOLO THREAD

En este apartado se describirá la pila del protocolo Thread con las diferentes capas que la integran, las cuales podemos ver en Ilustración 4:

Diagrama, Esquemático

Descripción generada automáticamente

Ilustración 4 Pila de Protocolo Thread en un sistema IoT

#### 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 o puertas de enlace, también conocidos como Border Routers.

Antes de que un dispositivo pueda unirse a una PAN, este deberá ser introducido 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 5:

Gráfico

Descripción generada automáticamente con confianza media

Ilustración 5 Ejemplo PAN Thread [12], [13]

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

* **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. Estas radios inalámbricas 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/desencriptación de datos, realizar la transmisión, etc.
* **Servicio ubicuo.** Se ofrece a sus usuarios un servicios 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 estas cinco restricciones:

* **Bajo consumo.** Debido a que mucho de los dispositivos están conectados a baterías, y el alargar la vida de dicha batería es un indispensable. 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.** Estas redes PAN, trabajan en interiores, donde suele haber bastantes obstáculos para la comunicación. Aparte, esos obstáculos, pueden variar su posición, al igual que pueden hacerlo los dispositivos, aparecen nuevos dispositivos y los viejos desaparecen. Varios dispositivos móviles pueden salir y reunirse a la red sin ser notificados. Thread usa DTLS para poder solventar este tipo de comunicaciones poco fiables. DTLS asume una capa de Transporte poco fiable. Debido al bajo consumo, la señal de comunicaciones es débil, por lo que lo complica más, 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 y conectando hasta más de 250 dispositivos en una PAN.

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

Otra parte por implementar de la especificación IEEE 802.15.4 en los dispositivos Thread es un conjunto de Control de Acceso a los Medios (Media Access Control – MAC). Dependiendo de si el dispositivo tiene o no capacidad de enrutamiento, los dispositivos tendrán que funcionar como un dispositivo MAC conmpleto (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 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 siguiente:

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

Tabla 3 MAC Layer Frame

* **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).

#### CAPA DE ADAPTACIÓN 6LoWPAN [12]

Thread usa IPv6 over Low-power Wireless Personal Area Network o 6LoWPAN, según se especifica en RFC 4944 [16] y RFC 6282 [17]. 6LoWPAN permite el uso de IPv6 dentro de las PAN, a pesar de que Thread las implementa con unas restricciones menores. 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.

* 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 (Líder, DHCPv6, Servicio, Commissioning, Descubrimiento de Vecinos) o incluso Identifcador de Punto Final (EID – Endpoint Identifier).

Las redes Thread no depende del Protocolo de Configuración de Host Dinámico v6 (DHCPv6), por lo que los dispositivos no tendrán que implementarlo. Lo que sí que deberán implementar es el Protocolo para Control de Mensajes Internet (ICMPv6), RFC 4443 [20].

#### PROTOCOLO DE ENRUTAMIENTO

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.
* Enlace
* Ruta

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 Lideres 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, esta 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.

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

Para dicho Commissioning Thread usa MeshCoP (Mesh Commissioning Protocol), el cual permite una autenticación, una comisión 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 elegible 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 para la comisión que tiene Thread son:

* Usando el método Out-of-band, configurar directamente sobre el dispositivo toda la información necesaria para la comisión. Esta información permitirá a los dispositivos que quieran unirse, que se unan a la red Thread correcta.
* 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 Comisario . 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 chequea 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 Comisario le proporciona un secreto compartido entre JD y JR, completando así la unión del JD.

# ANALISIS DE LA TECNOLOGÍA THREAD

## 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 Coockie 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.

### CARACTERÍSTICAS TÉCNOLOGICAS 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 hot 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

En primera instancia, el módulo KTDG102 integra el módulo RF KTWM102 comentado en 3.1.1.1CARACTERÍSTICAS 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 control 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 cuál 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, KTBN1 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.

#### 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.

### EJEMPLOS DE OTROS DISPOSITIVOS

Como ya se ha mencionado en el análisis realizado en [3.1. ANÁLISIS INICIAL DE LA TECNOLOGÍA THREAD](#_ANÁLISIS_INICIAL_) 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](https://www.nxp.com/support/developer-resources/software-development-tools/mcuxpresso-software-and-tools:MCUXPRESSO" \t "new)), Nordic Semiconductor ([nRF Connect SDK](https://www.nordicsemi.com/Software-and-tools/Software/nRF-Connect-SDK" \t "_blank)), Silicon Labs ([Simplicity Studio](https://www.silabs.com/products/development-tools/software/simplicity-studio" \t "_blank)) y MMB Networks ([RapidConnect Desktop](http://mmbnetworks.com/rapidconnect" \t "_blank)), 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](https://www.nordicsemi.com/Software-and-tools/Development-Kits/nRF52840-Dongle" \t "_blank)) y NXP ([OM15080-K32W](https://www.nxp.com/products/wireless/bluetooth-low-energy/om15080-k32w-k32w-usb-dongle-for-bluetooth-le-zigbee-and-thread-networks:OM15080-K32W" \t "_blank)) cuentan con mayor variedad de dispositivos hardware, pero también existen otros dispositivos desarrollados por empresas tecnológicas como Qualcomm ([QCA 4020/ 4024/ 4025](https://developer.qualcomm.com/hardware/qca4020-qca4024" \t "_blank)) o Texas Instuments ([SimpleLink LaunchPad™ Development Kits](https://www.ti.com/wireless-connectivity/simplelink-solutions/thread/design-development.html" \t "_blank)).

En módulos RF con certificado Thread, el desarrollador con mayor variedad es Silicon Labs (teniendo un equivalente al módulo de Kirale: [MGM210L / MGM210P](https://www.silabs.com/wireless/zigbee/efr32mg21-series-2-modules" \t "_blank)), pero también se cuenta con desarrolladores como MMB Networks ([RapidConnect Module](http://mmbnetworks.com/rapidconnect-module" \t "_blank)) y NXP ([Azurewave NXP Platform Wireless Modules](https://www.azurewave.com/wireless-modules-nxp.html" \t "_blank)).

Para sistemas operativos, tenemos desarrolladores como Silicon Labs ([Micrium OS](https://www.silabs.com/products/development-tools/software/micrium-os" \t "_blank)) o Zephyr ([Zephyr RTOS](https://docs.zephyrproject.org/latest/reference/networking/thread.html?highlight=thread" \t "_blank)).

Finalmente, aparte de Kirale, hay también más desarrolladores de Stack Software, como son OpenThread ([OpenThread](https://openthread.io/" \t "_blank)) y Silicon LAbs ([Silicon Labs Thread Stack](https://www.silabs.com/products/wireless/mesh-networking/thread" \t "_blank)).

## CONFIGURACIONES INICIALES

### CONFIGURACIÓN DEL BORDER ROUTER

#### REQUERIMIENTOS

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

* Un dispositivo Border Router KTBRN1.
* 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

#### GUÍA DE INSTALACIÓN

Para la instalación del Software del Border Router se deberán seguir los siguientes pasos.

##### DESCARGA DEL SOFTWARE REQUERIDO

Se deberá descargar la imagen, basada en Debian, en su última versión para el KTBRN1. Este software incluye el software de KiBRA. Esta imagen la podemos encontrar en el siguiente enlace:

* [KTBRN1 + KiBRA image file](https://www.kirale.com/products/ktbrn1/" \l "resources)

Una vez descargada esta imagen, se descargará el fichero [KiBRA-v2.x.x.zip](https://www.kirale.com/products/ktbrn1/#resources) para uso posterior.

Una vez descargada la imagen y el fichero .ZIP, se necesitará un software para actualizar la imagen guardada en la SD o flashear una nueva. Un ejemplo de este software sería:

* [Balena Etcher](https://www.balena.io/etcher/)

Por otro lado se necesitará una terminal Serie y/o un cliente SSH para la conexión con el dispositivo KTBRN1.

* [MobaXterm free](https://mobaxterm.mobatek.net/): Para el cliente SSH y la terminal Serie. Admite ambos tipos de sesiones a la vez, pero puede usarse cualquier otro.

[Zadig:](https://zadig.akeo.ie/) Se usará en caso de necesitar instalar los drivers de USB Serie.

##### FLASHEAR LA IMAGEN EN LA TARJETA SD

En caso de necesitar actualizar o flashear una imagen en una tarjeta SD, se deberán seguir las siguientes instrucciones:

* Instalar y abrir Balena Etcher.
* Seleccionar el fichero con extensión *.gz* (la imagen) en Etcher.
* Introducir la SD en el lector de tarjetas del ordenador y seleccionarla en Etcher. Se recomienda el uso de una micro SD clase 10 de al menos 2 GB de capacidad.
* Se clicará 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.

Captura de pantalla de un celular

Descripción generada automáticamente

Ilustración 6 Balena Etcher

##### PRIMERA INSTALACIÓN

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.

###### 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.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Ilustración 7 Border Router en Administrador de Dispositivos Previo a la instalación de Drivers

Si sale como en la Ilustración 7, 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 8*.* Una vez se hayan seleccionado las opciones comentadas se clicará en *Upgrade Driver*.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Ilustración 8 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.

*Texto

Descripción generada automáticamente*

Ilustración 9 MobaXterm

Tras realizar esto aparecerá una consola de inició de sesión, en la cual se deberá iniciar sesión con el usuario root y la contraseña kirale123.

La pantalla de bienvenida mostrará información sobre las direcciones IP configuradas en el KTBRN1 y que versión de KiBRA está utilizando, tal como se ve en Ilustración 9. Por defecto, la imagen instalada viene configurada con una dirección IPv4 estática para la interfaz Ethernet.

Dirección IPv4 por defecto: 192.168.75.84/24

Está dirección se deberá proceder a cambiarla a una dirección IPv4 que esté dentro de la red local en la que se vaya a trabajar, para poder ser visible al resto de los equipos.

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

#### PANEL DE ADMINISTRACIÓN WEB

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, aparecerá la página de acceso / login.

***Nota:*** *El ordenador deberá estar en la misma red que el dispositivo KTBRN1.*

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Ilustración 10 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.*

##### CAMBIAR LA CONFIGURACIÓN DE RED

El administrador puede querer cambiar y permitir una configuración automática de DHCPv4 o cambiar la dirección IPv4 por otra. Esto podrá realizarse accediendo al menú “Network” en la administración Web.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Ilustración 11 Pestaña Network

Como se muestra en la imagen de arriba, la nueva dirección IPv4 del DHCP es 192.168.0.102. Ahora será posible acceder por cliente SSH a esta dirección e incluso por enlace local de dirección IPv6 si el ordenador está conectado a la misma red. KiBRA también usará esta dirección IPv4 externa para su funcionalidad NAT64.

***Nota:*** *Se necesitará un reinicio del dispositivo KTBRN1 para asegurar que se aplican correctamente las nuevas configuraciones.*

##### ACTUALIZAR KiBRA

Para actualizar la versión de KiBRA, se deberá entraral menú de KiBRA y se clicará en el icono de “Upgrade”, situado al lado de la imagen del KTBRN1 en la sección de “System”.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Ilustración 12 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.

##### 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.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Ilustración 13 Pestaña Settings

###### 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 Commisioning:**  Permite seleccionar o desactivar este tipo de unión a la red cuando el sistema arranca.
   1. Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

      Descripción generada automáticamente**Activando**  este modo los parámetros a configurar de la red serán los mostrados en la siguiente imagen:

Ilustración 14 Parámetros a configurar con Out-of-band Commissioning Activado

* 1. **Desactivando** esta opción, los parámetros a configurar serán:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Ilustración 15 Parámetros a configurar con Out-of-band Commissioning Desactivado

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

Da la posibilidad de habilitar o deshabilitar la función BBR. De igual manera, el administrador de la red podrá configurar los parámetros específicos que usará el Servidor del Border Router

.

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

Esta opción activa la configuración manual del prefijo de la red, el cuál será usado en la red Thread. El usuario puede decidir entre diferentes opciones como será el direccionamiento IPv6 de los nodos dentro de la red, como DHCP o SLAAC. Si el BBR está activado, la opción de prefijo DUA será activada.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Ilustración 16 Configuración Prefijo de Red

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

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

Ir a la pestaña “KiBRA” en el menú para encender el “Border Router Engine”.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Ilustración 17 Menú KiBRA - Inicio de Border Router

Después se clicará en el botón de Start, y el Border Router se unirá a la red seleccionada o formará una nueva, según los ajustes configurados por el administrador.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Ilustración 18 Menú de KiBRA - Border Router Iniciado

Volviendo al submenú de “Settings” (Ajustes), es 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.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Ilustración 19 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 gracias al botón situado en la esquina inferior derecha llamado “Export Settings”, el cual permite copiar la información de “commissioning”, requerida para la configuración del nuevo dispositivo e introducirlo en la red, utilizando comandos KiNOS.a través de la interfaz de KiTools proporcionada por Kirale.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Ilustración 20 Ventana Export Commissioning Information

##### 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.

###### SERVIDOR BACKBONE ROUTER

Cuando la opción “Backbone Router Server” esté habilitada en el menú de “Settings”, los datos relacionados aparecerán en esta página.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Ilustración 21 Servidor Backbone Router

###### DHCP

Si la opción DHCP se activa al configurar el prefijo de red, la siguiente página mostrará una lista de los nodos que han adquirido una dirección IPv6 vía DHCP y cuál es.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Ilustración 22 Servicio DHCP

###### NAT64

Siempre que haya una dirección IPv4 configurada en la interfaz externa, esta será usada para realizar una función de NAT64 en el Border Router. La tabla de la sesión NAT se mostrará en esta pestaña.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Ilustración 23 Servicio NAT64

###### 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.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Ilustración 24 Servicio Commissioner

##### VISUAL NETWORK

El mapa de topología de red es un mapa que permite al administrador de la red ver el Layout físico 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 los problemas.

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza baja

Ilustración 25 Pestaña Visual Network

##### 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.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Ilustración 26 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.

##### 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 perdida 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.

##### 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.

|  |
| --- |
| 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

##### COMUNICACIÓN ENTRE PROCESOS

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

### 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. Esto último se explicará tanto para configurarlo a través del PC por conexión USB, como a través de un microcontrolador vía UART.

***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

Conectar 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*”.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Ilustración 27 Administrador de dispositivo antes de instalar los Drivers de KTWM102

##### 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 una herramienta gratuita llamada **Zadig**. Esta herramienta puede descargarse de la página de Zadig <https://zadig.akeo.ie/>**[¡Error! Referencia de hipervínculo no válida.](https://zadig.akeo.ie/)**. 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”.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Ilustración 28 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 despegable, y seleccionar el driver “liusbK” y clicar en “***Install / Replace Driver***”.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Ilustración 29 Instalación Drivers con Zadig Paso 2

El proceso tomará alrededor de un segundo y el resultado será un mensaje de éxito:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Ilustración 30 Instalación Drivers con Zadig Finalizada

Ahora, en el *Administrador de Dispositivos*, el KiNOS Boot DFU deberá salir como se muestra en la Ilustración 31:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Ilustración 31 Administrador de Dispositivos Después de Instalar Drivers

##### LINUX / 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.

1. En Windos: descargar de <http://dfu-util.sourceforge.net/> y extraerlo en la carpeta deseada.
2. En MAC OS:

$ brew install dfu-util

([Get Brew](https://brew.sh/))

1. En Linux:

$ sudo apt install dfu-util

#### ACTUALIZACIÓN DE FIRMWARE

Abrir dfu-util desde la ventana de comandos y listar los dispositivos conectados para encontrar el dispositivo deseado. El USB Product ID par 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

Flashear el fichero del firmware al dispositivo deseado (especificando el número de serie). Esta transferencia del archivo puede tardar varios segundos.

|  |
| --- |
| $ dfu-util—download KiNOS-GEN-KTWM102-1.1.6533.62822.dfu—serial 8404D2000000045B  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](https://www.kirale.com/support/" \l "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).
* 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:

##### 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 despegable, seleccionar el driver “*libusbK”* y seleccionar “*Install/Replace Driver”*.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Ilustración 32 Instalar libusbk con Zadig

El proceso tardará alrededor de un segundo y saldrá un mensaje de éxito

.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Ilustración 33 Instalación Driver Libusbk xon Zadig Finalizada

Después, se selecciona *KiNOS Virtual COM (Interface 1)* en la lista despegable, seleccionar el dirver “***USB Serial (CDC)”*** y cliclar en “*Install/Replace Driver”*.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Ilustración 34 Instalar USB SERIAL (CDC) con Zadig

El proceso tardará alrededor de un segundo y saldrá un mensaje de éxito.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Ilustración 35 Instalación Driver USB Serial (CDC) con Zadig Finalizada

Ahora, en *Administrador de Dispositivos*, la interfaz KiNOS debería aparecer como se muestra en la Ilustración 36.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Ilustración 36 Administrador de Dispositivos después de la instalación

##### 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

##### 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 |

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

#### CONFIGURACIÓN DE TERMINAL COM

##### WINDOWS

Hay gran variedad de terminales serie COM disponibles para sistemas Windows. En esta guía se usa una genérica llamada “Termite”, pero valdría cualquier otra terminal.

Abrir la terminal serie y configurar con las siguientes configuraciones para comunicaciones serie con dispositivos de Kirale:

Configuración Estándar para puerto serie USB: 9600 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.

Interfaz de usuario gráfica, Aplicación, Word

Descripción generada automáticamente

Ilustración 37 Terminal Termite

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

##### 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***

## 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.

### 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 38 Herramienta KiTools.

Texto

Descripción generada automáticamente

Ilustración 38 Herramienta KiTools

Para ver más en detalle la sintaxis de cada comando, descargar y ver la [KSH Reference Guide](https://www.kirale.com/products/ktwm102/#083596d3193c172fa).

#### 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).

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Ilustración 39 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”.

#### 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.

### 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 40, se muestra un esquema de un Host externo, como puede ser un microcontrolador usando KBI para comunicarse con el módulo KTWM102 y el sistema KiNOS a través del puerto UART:

Diagrama

Descripción generada automáticamente

Ilustración 40 Esquema de comunicación entre Host Externo y el dispositivo KTWM102

Una vez conectado el sistema como se indicada en el esquema mostrado en Ilustración 40, se deberá configurar el puerto serie de nuestro Host de la siguiente manera:

**Configuración Serie:** 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 40, 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 recibidor 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 ***Consisten Overhead Byte Stuffing (COBS)*** implementado en KiNOS. Este sistema se explica con más detalle en la [KBI Reference Guide](https://www.kirale.com/products/ktwm102/#083596d3193c172fa) y en [draft-ietf-pppext-cobs-00](https://datatracker.ietf.org/doc/html/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:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Header | | | | | Payload |
| L0 | L1 | TYPE | CMD | CKS | (Optional ≤ 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 Misssing | - | - |
| 7 | - | - | Firmware Update Error | - | - |
| 8 | - | - | Busy | - | - |
| 9 to 15 | - | - | - | - | - |

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:

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](#_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 41:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Ilustración 41 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.

Para más detalle de los comandos ver en la [KBI Reference Guide](https://www.kirale.com/products/ktwm102/#083596d3193c172fa)

### CONFIGURACIÓN DE RED

Una vez explicadas las diferentes maneras 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 Commisioning “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 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.*

#### 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 Commisioning”.

Una vez se aseguré 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 Commisioning,* 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”***.**

# 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.

## COMPONENTES COMERCIALES UTILIZADOS

### ELEMENTOS HARDWARE UTILIZADOS

Los elementos Hardware comerciales utilizados han sido:

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

Un circuito electrónico

Descripción generada automáticamente

Ilustración 42 Kit de Desarrollo STM32F407G-DISC1

1. **KTDG102 Evaluation Dongle:** Módulo Thread de Evaluación de Kirale. Se han utilizado 2 unidades.

Un circuito electrónico

Descripción generada automáticamente con confianza media

Ilustración 43 KTDG102 Evaluation Dongle

1. **Módulo KTWM102:** Módulo Thread de Kirale, el cual se integra en la Coockie PCB

Imagen de la pantalla de un celular con letras

Descripción generada automáticamente con confianza baja

Ilustración 44 Módulo KTWM102

1. **Border Router:** Border Router propio de Kirale.

Imagen de la pantalla de un celular con letras

Descripción generada automáticamente con confianza baja

Ilustración 45 Border Router

1. **Coockie:** Módulo sobre el que se implementa el diseño final de la PCB realizada.

Imagen que contiene electrónica, circuito

Descripción generada automáticamente

Ilustración 46 Módulos de Procesamiento y de Alimentación de la Coockie

### 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 Coockie.
* **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.

## DISEÑO DE ESQUEMÁTICO PCB

### 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.
* **Coockie Connector:** Zona donde se definen los conectores a utilizar para integrar la PCB a la Coockie.
* **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 47:

Diagrama

Descripción generada automáticamente

Ilustración 47 Jerarquía Circuito

### CIRCUITO DE ALIMENTACIÓN

En la parte de alimentación, el circuito realizado ha sido el mostrado a continuación en Ilustración 48:

Diagrama, Esquemático

Descripción generada automáticamente

Ilustración 48 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 Coockie (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 Coockie 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 Coockie. 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.

### INTEGRACIÓN DEL MÓDULO KTWM102

El circuito realizado para integrar el módulo KTWM102 es el mostrado a continuación en Ilustración 49:

Diagrama, Esquemático

Descripción generada automáticamente

Ilustración 49 Circuito Integración del Módulo

El módulo KTWM102 es el mostrado en la Ilustración 44, 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…).

### COOCKIE CONNECTOR

Esta parte del circuito es la interfaz con el resto de PCB integrantes de la Coockie a través de conectores verticales, el esquemático es el mostrado a continuación en Ilustración 50:

Tabla

Descripción generada automáticamente con confianza baja

Ilustración 50 Conectores Verticales

### 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 51:

Gráfico

Descripción generada automáticamente

Ilustración 51 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.

## LAYOUT

El Layout completo realizado es el mostrado a continuación en Ilustración 52. Posteriormente se mostrará el Layout elegido separando las diferentes capas.

Imagen de la pantalla de un computador

Descripción generada automáticamente con confianza baja

Ilustración 52 Layout Completo

### LAYOUT CAPA TOP

Como se muestra a continuación, en Ilustración 53, 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.

Diagrama, Esquemático

Descripción generada automáticamente

Ilustración 53 Layout Capa TOP

### LAYOUT CAPA BOTTOM

Como se muestra a continuación, en Ilustración 54 , 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.

Imagen de la pantalla de un computador

Descripción generada automáticamente con confianza baja

Ilustración 54 Layout Capa BOTTOM

### RESULTADO FINAL DEL DISEÑO

Una vez finalizado el diseño se mandó fabricar la PCB quedando como se ve en Ilustración 55 y en Ilustración 56.

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 55 PCB Coockie Thread TOP

Imagen que contiene circuito

Descripción generada automáticamente

Ilustración 56 PCB Coockie Thread Capa BOTTOM

# PRUEBAS EXPERIMENTALES

## PRIMERA INTERACCIÓN CON DONGLE USB

Imagen que contiene circuito

Descripción generada automáticamenteEsta 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 57:

Ilustración 57 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:
   1. 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.
   2. 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.

## 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.

Imagen de la pantalla de un celular con texto e imágenes

Descripción generada automáticamente con confianza mediaSe han realizado varias pruebas para las cuales el esquema de montaje ha sido el mostrado en Ilustración 58:

Ilustración 58 Esquema montaje Red de Dos Nodos

### 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 Commisioning 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.

### 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 59:



Ilustración 59 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 60:

Texto

Descripción generada automáticamente con confianza media

Ilustración 60 Logs KiTools al RECIBIR un Ping

### 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.

## PRUEBAS CON EL BORDER ROUTER

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

### 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 61:

Imagen de la pantalla de un celular con letras

Descripción generada automáticamente con confianza media

Ilustración 61 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 loggeados 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.

### 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 62:

Gráfico, Gráfico de líneas

Descripción generada automáticamente

Ilustración 62 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](https://www.kirale.com/support/kb/ip-connectivity-between-the-thread-network-and-the-lan-part-i/)
2. [Conectividad IPv6](https://www.kirale.com/support/kb/ip-connectivity-between-the-thread-network-and-the-lan-part-ii/)
3. [Conectividad IPv4 a IPv6](https://www.kirale.com/support/kb/ip-connectivity-between-the-thread-network-and-the-lan-part-iii/)
4. [Conectividad IPv6 a IPv4](https://www.kirale.com/support/kb/ip-connectivity-between-the-thread-network-and-the-lan-part-iv/)

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

### 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 63:

![Diagrama

Descripción generada automáticamente]()

Ilustración 63 Topología 1 de BR con 2 nodos Dongle

Siendo el Dongle Leader el 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 64:

![Diagrama

Descripción generada automáticamente]()

Ilustración 64 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 65, Ilustración 66 e Ilustración 67:

![Texto

Descripción generada automáticamente]()

Ilustración 65 Ping desde PC a BR

![Leader
]()

Ilustración 66 Ping desde PC a nodo LEADER

![MED
]()

Ilustración 67 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.

### 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 68:

![Diagrama

Descripción generada automáticamente]()

Ilustración 68 Topología para envío de mensajes UDP vía Sockets

Para esto, como se ha mencionado en 5.2.3 ENVÍO DE MENSAJES UDP A TRAVÉS DE SOCKETS 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 33ºC durante toda la prueba.

## PRUEBAS CON PCB COOCKIE 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 Coockie 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 69

Diagrama

Descripción generada automáticamente

Ilustración 69 Topología 4 nodos con un Dongle como LEADER

1. 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 70:

Imagen que contiene objeto, reloj

Descripción generada automáticamente

Ilustración 70 Topología 4 nodos con BR como Leader

### 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 ping desde el PC a este nuevo nodo y comprobamos que el resultado medio es de aproximadamente 46 ms:

Texto

Descripción generada automáticamente

Ilustración 71 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.

### ENVÍO / RECIBO DE SOCKETS

Siguiendo con el uso de las dos topologías mencionadas en 5.4 PRUEBAS CON PCB COOCKIE 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 COOCKIE 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 Coockie 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 ENTRE DONGLES Y ENTRE PC Y DONGLES, 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.

## PRUEBAS DE ESTABILIDAD CON 5 NODOS

Tras validar el funcionamiento de la primera PCB Coockie Thread, se crea el código para el manejo de los módulos KTWM102 desde el kit de la Coockie y se añade como 5 nodo a la red otra PCB de Coockie 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 72.

Diagrama

Descripción generada automáticamente

Ilustración 72 Topología A de 5 nodos

1. El nodo REED se conecta al BR como nodo hijo, y el los MED se conectan al REED. Ver Ilustración 73.

Diagrama

Descripción generada automáticamente

Ilustración 73 Topología B con 5 nodos

### 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.

### 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 Coockie Completa (PCB Thread + Coockie) 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 Coockie 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 que vayan 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 perdida de bytes en los diferentes altos.

## 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 Coockie Thread controlada desde el PC Se ha configurado la red con el BR como nodo Líder, un Dongle KTDG102 y una PCB Coockie Thread como REEDs o Routers, y las otras dos PCB (una de ellas con la Coockie 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 74:

Gráfico, Gráfico radial

Descripción generada automáticamente

Ilustración 74 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 Coockie Completa (0x2402)**: Envía al nodo REED formado por el Dongle.
* **Nodos MED (0x2401) y REED (0x2400)** de PCB Coockie 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.

# CONCLUSIONES Y LÍNEAS FUTURAS

En este proyecto se ha buscado la viabilidad de la implementación de las comunicaciones THREAD en el sistema de la Coockie. 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:

* 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 el sol impactando directamente sobre dichos nodos. Por lo que, en ambientes normales sin mucha carga de trabajo sobre el nodo, no habrá problema con los nodos.

Un posible desarrollo futuro para este proyecto es la integración de la capa de sensores a los nodos desarrollados. A su vez se podría desarrollar una plataforma Web donde se puedan enviar por mensajes UDP mediante Sockets las medidas tomadas por los diferentes nodos y se pueda ir gestionando la red mediante esta Web, ya sea desde la configuración de los tiempos de envío 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.

# 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. ICCCSP 2018*, no. Icccsp, pp. 1–5, 2018, doi: 10.1109/ICCCSP.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.

[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.