# 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.

Dar gracias también a mi pequeña familia de Effetá, por todo lo que he podido aprender de cada persona.

# RESUMEN

El mundo tiende a compartir cada vez más información, a estar cada vez más interconectado. Cada vez existe un mayor número de dispositivos conectados entre ellos y con Internet de múltiples maneras. De todo esto surge lo que se conoce, en inglés, como **Internet of Things** o **IoT**, o en español, Internet de las cosas.

Este concepto consiste en redes de sensores capaces de monitorizar su entorno, reaccionando ante posibles cambios.

# INDICE

[AGRADECIMIENTOS I](#_Toc81499320)

[RESUMEN II](#_Toc81499321)

[INDICE III](#_Toc81499322)

[ABREVIATURAS Y ACRÓNIMOS VII](#_Toc81499323)

[ILUSTRACIONES IX](#_Toc81499324)

[ECUACIONES XI](#_Toc81499325)

[TABLAS XII](#_Toc81499326)

[CÓDIGOS XIII](#_Toc81499327)

[1. INTRODUCCIÓN Y OBJETIVOS DEL TRABAJO 2](#_Toc81499328)

[2. ESTADO DEL ARTE 4](#_Toc81499329)

[2.1. INTERNET OF THINGS (IoT) 4](#_Toc81499330)

[2.1.1. Introducción [2] 4](#_Toc81499331)

[2.1.2. Componentes [2] 5](#_Toc81499332)

[2.1.2.1. Dispositivo 6](#_Toc81499333)

[2.1.2.2. Red Local 6](#_Toc81499334)

[2.1.2.3. Internet 6](#_Toc81499335)

[2.1.2.4. Servicios Backend 6](#_Toc81499336)

[2.1.2.5. Aplicaciones 7](#_Toc81499337)

[2.1.3. Sensores 7](#_Toc81499338)

[2.1.4. Arquitectura [3] 8](#_Toc81499339)

[2.1.4.1. Capa de Sensorizado 8](#_Toc81499340)

[2.1.4.2. Capa de Red 8](#_Toc81499341)

[2.1.4.3. Capa de Servicio 9](#_Toc81499342)

[2.1.4.4. Capa de Interfaz 9](#_Toc81499343)

[2.2. 6LoWPAN [4] 10](#_Toc81499344)

[2.2.1. Significado 6LoWPAN [4] 10](#_Toc81499345)

[2.2.2. Pila de Protocolos de 6LoWPAN [4]–[6] 11](#_Toc81499346)

[2.2.3. Principales características de 6LoWPAN [7] 12](#_Toc81499347)

[2.2.3.1. Compresión de cabecera 12](#_Toc81499348)

[2.2.3.2. Enrutamiento 12](#_Toc81499349)

[2.2.3.3. Seguridad 12](#_Toc81499350)

[2.2.3.4. Protocolos de aplicación 13](#_Toc81499351)

[2.2.4. Retos de 6LoWPAN 13](#_Toc81499352)

[2.2.5. Implementaciones y aplicaciones para 6LoWPAN [5], [7] 14](#_Toc81499353)

[2.2.5.1. Contiki 14](#_Toc81499354)

[2.2.5.2. TinyOS 15](#_Toc81499355)

[2.2.5.3. Thread 15](#_Toc81499356)

[2.3. THREAD 16](#_Toc81499357)

[2.3.1. Introducción a Redes THREAD [8], [10]–[12] 16](#_Toc81499358)

[2.3.2. Tipos de dispositivos 17](#_Toc81499359)

[2.3.3. Protocolo Thread 18](#_Toc81499360)

[2.3.3.1. Redes de área Privada - Private Area Network (PAN) [12] 19](#_Toc81499361)

[2.3.3.2. Capa Física [12], [14] 21](#_Toc81499362)

[2.3.3.3. Capa MAC (o enlace de Datos). [12], [14] 21](#_Toc81499363)

[2.3.3.4. Capa de adaptación 6LoWPAN [12] 23](#_Toc81499364)

[2.3.3.5. Capa de Red [12] 23](#_Toc81499365)

[2.3.3.6. Protocolo de Enrutamiento 24](#_Toc81499366)

[2.3.3.7. Capa de Transporte [12] 24](#_Toc81499367)

[2.3.3.8. Seguridad y Comisión de dispositivos [10], [12] 25](#_Toc81499368)

[3. ANALISIS DE LA TECNOLOGÍA THREAD 27](#_Toc81499399)

[3.1. CONFIGURACIONES INICIALES 27](#_Toc81499400)

[3.1.1. Configuración del Border Router 27](#_Toc81499401)

[3.1.1.1. Requerimientos 27](#_Toc81499402)

[3.1.1.2. Guía de Instalación 27](#_Toc81499403)

[3.1.1.2.1. Descarga del Software Requerido 27](#_Toc81499404)

[3.1.1.2.2. Flashear la imagen en la tarjeta SD 28](#_Toc81499405)

[3.1.1.2.3. Primera Instalación 28](#_Toc81499406)

[3.1.1.3. Panel de Administración Web 31](#_Toc81499407)

[3.1.1.3.1. Cambiar la configuración de red 33](#_Toc81499408)

[3.1.1.3.2. Actualizar KiBRA 34](#_Toc81499409)

[3.1.1.3.3. Configurar Border Router 35](#_Toc81499410)

[3.1.1.3.4. Inicio del Border Router (Start-up). 38](#_Toc81499411)

[3.1.1.3.5. Servicios 42](#_Toc81499412)

[3.1.1.3.6. Visual Network 46](#_Toc81499413)

[3.1.1.3.7. Logs 47](#_Toc81499414)

[3.1.1.4. Breve resumen 48](#_Toc81499415)

[3.1.1.4.1. Sistema de ficheros avanzado 48](#_Toc81499416)

[3.1.1.4.2. Servicios críticos 48](#_Toc81499417)

[3.1.1.4.3. Comunicación entre Procesos 49](#_Toc81499418)

[3.1.2. Configuración Inicial Módulo KTWM102 50](#_Toc81499419)

[3.1.2.1. Instalación de Drivers USB y del Bootloader 50](#_Toc81499420)

[3.1.2.1.1. Windows 51](#_Toc81499421)

[3.1.2.1.2. Linux / MAC OS 53](#_Toc81499422)

[3.1.2.2. Install “dfu-util” 54](#_Toc81499423)

[3.1.2.3. Actualización de Firmware 54](#_Toc81499424)

[3.1.2.4. Runtime – Instalación de Drivers USB 56](#_Toc81499425)

[3.1.2.4.1. Windows 56](#_Toc81499426)

[3.1.2.4.2. Linux 58](#_Toc81499427)

[3.1.2.4.3. MAC OS 59](#_Toc81499428)

[3.1.2.5. Configuración de Terminal COM 59](#_Toc81499429)

[3.1.2.5.1. Windows 59](#_Toc81499430)

[3.1.2.5.2. Linux / MAC Os 60](#_Toc81499431)

[3.2. CONFIGURACIÓN DE RED PARA KTWM102 61](#_Toc81499432)

[3.2.1. Kirale Command-Line Shell Reference Guide – Comandos KSH 61](#_Toc81499433)

[3.2.1.1. Sintaxis de los comandos 61](#_Toc81499434)

[3.2.1.2. Sintaxis de los parámetros 62](#_Toc81499435)

[3.2.1.3. Mensajes de Respuesta 63](#_Toc81499436)

[3.2.2. Kirale Binary Interface Reference Guide – Comandos KBI 64](#_Toc81499437)

[3.2.2.1. Operación de Interfaz 64](#_Toc81499438)

[3.2.2.2. Formato del paquete 65](#_Toc81499439)

[3.2.2.3. Representación de datos 66](#_Toc81499440)

[3.2.2.4. Comandos y Respuestas 67](#_Toc81499441)

[3.2.2.5. Notificaciones 67](#_Toc81499442)

[3.2.3. Configuración de Red 68](#_Toc81499443)

[3.2.3.1. Modo Out-of-Band Commissioning Desactivado 68](#_Toc81499444)

[3.2.3.2. Modo Out-of-Band Commissioning Activado 68](#_Toc81499445)

[3.3. PRUEBAS DE RED 69](#_Toc81499446)

[4. DISEÑO E IMPLEMENTACIÓN HARDWARE 70](#_Toc81499447)

[4.1. DISEÑO DE ESQUEMÁTICO PCB 70](#_Toc81499448)

[4.1.1. Jerarquía del circuito 70](#_Toc81499449)

[4.1.2. Circuito de Alimentación 71](#_Toc81499450)

[4.1.3. Integración del módulo KTWM102 72](#_Toc81499451)

[4.1.4. Coockie Connector 73](#_Toc81499452)

[4.1.5. USB 74](#_Toc81499453)

[4.2. LAYOUT 75](#_Toc81499454)

[4.2.1. Layout Capa TOP 76](#_Toc81499455)

[4.2.2. Layout Capa BOTTOM 77](#_Toc81499456)

[4.3. COMPONENTES COMERCIALES UTILIZADOS 78](#_Toc81499457)

[4.3.1. Elementos Hardware Utilizados 78](#_Toc81499458)

[5. PRUEBAS EXPERIMENTALES 80](#_Toc81499459)

[5.1. PRIMERA INTERACCIÓN CON DONGLE USB 80](#_Toc81499460)

[5.2. RED DE DOS NODOS 81](#_Toc81499461)

[5.2.1. Creación de la Red 81](#_Toc81499462)

[5.2.2. Ping entre nodos 82](#_Toc81499463)

[5.2.3. Envío de mensajes UDP a través de Sockets entre ambos nodos 82](#_Toc81499464)

[5.3. PRUEBAS CON EL BORDER ROUTER 83](#_Toc81499465)

[5.3.1. Introducción a la configuración del Router. 83](#_Toc81499466)

[5.3.2. Prueba de conectividad IP entre la red THREAD y la LAN 84](#_Toc81499467)

[5.3.3. Red con el BR y dos nodos KTDG102 84](#_Toc81499468)

[5.3.4. Envío de mensajes UDP por Sockets entre Dongles y entre PC y Dongles 88](#_Toc81499469)

[5.4. PRUEBAS CON PCB COOCKIE THREAD COMO CUARTO NODO 89](#_Toc81499470)

[5.4.1. Pruebas de conectividad con el cuarto nodo 90](#_Toc81499471)

[5.4.2. Envío / Recibo de Sockets 91](#_Toc81499472)

[5.5. PRUEBAS DE ESTABILIDAD CON 5 NODOS 92](#_Toc81499473)

[5.5.1. Pruebas de conectividad 93](#_Toc81499474)

[5.5.2. Envío / Recibo de Sockets 93](#_Toc81499475)

[5.6. PRUEBAS CON 6 NODOS Y CON ENVÍO DE MENSAJES CON MULTISALTO 95](#_Toc81499476)

[6. CONCLUSIONES 97](#_Toc81499477)

[7. LINEAS FUTURAS 98](#_Toc81499478)

[8. PLANIFICACIÓN TEMPORAL Y PRESUPUESTO 99](#_Toc81499479)

[9. BIBLIOGRAFÍA 100](#_Toc81499480)

# 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) |

Tabla 1 Abreviaturas y Acrónimos

# ILUSTRACIONES

[Ilustración 1 Estructura Básica 6LoWPAN [6] 11](#_Toc81499811)

[Ilustración 2 Arquitectura en Contiki [7] 14](#_Toc81499812)

[Ilustración 3 Ejemplo de una topología de Red Thread en malla [9] 18](#_Toc81499813)

[Ilustración 4 Pila de Protocolo Thread en un sistema IoT [12], [13] 19](#_Toc81499814)

[Ilustración 5 Ejemplo PAN Thread [12], [13] 19](#_Toc81499815)

[Ilustración 7 Balena Etcher 28](#_Toc81499817)

[Ilustración 8 Border Router en Administrador de Dispositivos Previo a la instalación de Drivers 29](#_Toc81499818)

[Ilustración 21 Instalación Drivers de Border Router con Zadig 30](#_Toc81499819)

[Ilustración 22 MobaXterm 30](F:\\Jorge\\UPM\\master\\TFM\\Documentos\\TFM_Contreras_v1_correcciones.docx" \l "_Toc81499820)

[Ilustración 23 Login Panel Administración Web 32](#_Toc81499821)

[Ilustración 24 Pestaña Network 33](#_Toc81499822)

[Ilustración 25 Actualización KiBRA 34](#_Toc81499823)

[Ilustración 26 Pestaña Settings 35](#_Toc81499824)

[Ilustración 27 Parámetros a configurar con Out-of-band Commissioning Activado 36](F:\\Jorge\\UPM\\master\\TFM\\Documentos\\TFM_Contreras_v1_correcciones.docx" \l "_Toc81499825)

[Ilustración 28 Parámetros a configurar con Out-of-band Commissioning Desactivado 36](F:\\Jorge\\UPM\\master\\TFM\\Documentos\\TFM_Contreras_v1_correcciones.docx" \l "_Toc81499826)

[Ilustración 29 Configuración Prefijo de Red 37](#_Toc81499827)

[Ilustración 30 Menú KiBRA - Inicio de Border Router 38](#_Toc81499828)

[Ilustración 31 Menú de KiBRA - Border Router Iniciado 39](#_Toc81499829)

[Ilustración 32 Pestaña Settings - Export Settings 40](#_Toc81499830)

[Ilustración 33 Ventana Export Commissioning Information 41](#_Toc81499831)

[Ilustración 34 Servidor Backbone Router 42](#_Toc81499832)

[Ilustración 35 Servicio DHCP 43](#_Toc81499833)

[Ilustración 36 Servicio NAT64 44](#_Toc81499834)

[Ilustración 37 Servicio Commissioner 45](#_Toc81499835)

[Ilustración 38 Pestaña Visual Network 46](#_Toc81499836)

[Ilustración 39 Pestaña Logs 47](#_Toc81499837)

[Ilustración 40 Administrador de dispositivo antes de instalar los Drivers de KTWM102 50](#_Toc81499838)

[Ilustración 41 Instalación Drivers con Zadig Paso 1 51](#_Toc81499839)

[Ilustración 42 Instalación Drivers con Zadig Paso 2 52](#_Toc81499840)

[Ilustración 43 Instalación Drivers con Zadig Finalizada 52](#_Toc81499841)

[Ilustración 44 Administrador de Dispositivos Después de Instalar Drivers 53](#_Toc81499842)

[Ilustración 45 Instalar libusbk con Zadig 56](#_Toc81499843)

[Ilustración 46 Instalación Driver Libusbk xon Zadig Finalizada 57](#_Toc81499844)

[Ilustración 47 Instalar USB SERIAL (CDC) con Zadig 57](#_Toc81499845)

[Ilustración 48 Instalación Driver USB Serial (CDC) con Zadig Finalizada 57](#_Toc81499846)

[Ilustración 49 Administrador de Dispositivos después de la instalación 58](#_Toc81499847)

[Ilustración 50 Terminal Termite 60](#_Toc81499848)

[Ilustración 51 Herramienta KiTools 62](#_Toc81499849)

[Ilustración 52 IP como parámetros 63](#_Toc81499850)

[Ilustración 53 Esquema de comunicación entre Host Externo y el dispositivo KTWM102 64](F:\\Jorge\\UPM\\master\\TFM\\Documentos\\TFM_Contreras_v1_correcciones.docx" \l "_Toc81499851)

[Ilustración 54 Ejemplo Rutina de Comando - Esperar Respuesta 67](#_Toc81499852)

[Ilustración 55 Esquema montaje Red de Dos Nodos 69](F:\\Jorge\\UPM\\master\\TFM\\Documentos\\TFM_Contreras_v1_correcciones.docx" \l "_Toc81499853)

[Ilustración 45 Montaje Border Router 69](F:\\Jorge\\UPM\\master\\TFM\\Documentos\\TFM_Contreras_v1_correcciones.docx" \l "_Toc81499854)

[Ilustración 6 Jerarquía Circuito 70](F:\\Jorge\\UPM\\master\\TFM\\Documentos\\TFM_Contreras_v1_correcciones.docx" \l "_Toc81499855)

[Ilustración 7 Circuito Integración del Módulo 72](#_Toc81499856)

[Ilustración 8 Dispositivo KTWM102 72](#_Toc81499857)

[Ilustración 9 Conectores Verticales 73](#_Toc81499858)

[Ilustración 10 Conector USB 74](#_Toc81499859)

[Ilustración 11 Layout Completo 75](#_Toc81499860)

[Ilustración 12 Layout Capa TOP 76](#_Toc81499861)

[Ilustración 13 Layout Capa BOTTOM 77](#_Toc81499862)

[Ilustración 14 Kit de Desarrollo STM32F407G-DISC1 78](#_Toc81499863)

[Ilustración 15 KTDG102 Evaluation Dongle 78](#_Toc81499864)

[Ilustración 16 Módulo KTWM102 79](#_Toc81499865)

[Ilustración 17 Border Router 79](#_Toc81499866)

[Ilustración 57 Módulos de Procesamiento y de Alimentación de la Coockie 79](#_Toc81499867)

[Ilustración 49 Diagrama de conexión PC - Dongle - uC 80](F:\\Jorge\\UPM\\master\\TFM\\Documentos\\TFM_Contreras_v1_correcciones.docx" \l "_Toc81499868)

[Ilustración 42 Esquema montaje Red de Dos Nodos 81](F:\\Jorge\\UPM\\master\\TFM\\Documentos\\TFM_Contreras_v1_correcciones.docx" \l "_Toc81499869)

[Ilustración 43 Logs KiTools al REALIZAR un Ping 82](#_Toc81499870)

[Ilustración 44 Logs KiTools al RECIBIRun Ping 82](#_Toc81499871)

[Ilustración 45 Montaje Border Router 83](F:\\Jorge\\UPM\\master\\TFM\\Documentos\\TFM_Contreras_v1_correcciones.docx" \l "_Toc81499872)

[Ilustración 46 Topología de Red 1 nodo con BR 84](#_Toc81499873)

[Ilustración 47 Topología 1 de BR con 2 nodos Dongle 85](#_Toc81499874)

[Ilustración 48 Topología 2 de BR con dos nodos Dongle 85](#_Toc81499875)

[Ilustración 49 Ping desde PC a BR 86](#_Toc81499876)

[Ilustración 50 Ping desde PC a nodo LEADER 86](#_Toc81499877)

[Ilustración 51 Ping desde PC a nodo MED 86](#_Toc81499878)

[Ilustración 52 Topología para envío de mensajes UDP vía Sockets 88](#_Toc81499879)

[Ilustración 53 Topología 4 nodos con un Dongle como LEADER 89](#_Toc81499880)

[Ilustración 54 Topología 4 nodos con BR como Leader 90](#_Toc81499881)

[Ilustración 55 Ping desde PC a nuevo nodo MED. 90](#_Toc81499882)

[Ilustración 56 Topología A de 5 nodos 92](#_Toc81499883)

[Ilustración 57 Topología B con 5 nodos 93](#_Toc81499884)

[Ilustración 58 Topología con 6 nodos 95](#_Toc81499885)

[Ilustración 59 Jerarquía Circuito 96](F:\\Jorge\\UPM\\master\\TFM\\Documentos\\TFM_Contreras_v1_correcciones.docx" \l "_Toc81499886)

# ECUACIONES

# TABLAS

[Tabla 1 Abreviaturas y Acrónimos VIII](#_Toc81499561)

[Tabla 2 Diferencias M2M – IoT [2] 5](#_Toc81499562)

[Tabla 3 MAC Layer Frame 22](#_Toc81499563)

[Tabla 4 Servicios KBRNT1 48](#_Toc81499564)

[Tabla 5 Comandos Servicio KiBRA 49](#_Toc81499565)

[Tabla 6 Servicio Ajenti 49](#_Toc81499566)

[Tabla 7 Listar dispositivos con dfu-util 54](#_Toc81499567)

[Tabla 8 Actualizar Firmware en dispositivos KTWM102 55](#_Toc81499568)

[Tabla 9 Comando para listar Puertos Serie en Sistemas Linux 59](#_Toc81499569)

[Tabla 10 Comando para listar Puertos Serie en Sistemas en MAC OS 59](#_Toc81499570)

[Tabla 11 Abrir terminal Picocom en Linux / MAC Os 60](#_Toc81499571)

[Tabla 12 Sintaxis comandos KSH 61](#_Toc81499572)

[Tabla 13 Formato del Paquete 65](#_Toc81499573)

[Tabla 14 Bits Byte Type 65](#_Toc81499574)

[Tabla 15 Significado Bits Byte Type 66](#_Toc81499575)

# CÓDIGOS

# INTRODUCCIÓN Y OBJETIVOS DEL TRABAJO

La estructura del documento debe ser:

* Introducción y objetivos del trabajo
* Estado del arte
* Análisis de la tecnología Thread y configuración de dispositivos
* Diseño e implementación hardware
* Pruebas experimentales
* Conclusiones y líneas futuras

Los objetivos 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.

En este estudio nos centraremos 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]

Mientras que IoT ofrece muy buenas oportunidades, sigue siendo un reto pendiente el manejo de todo lo que lleva integración continua del mundo físico. Actualmente hay dos APIs para las comunicaciones IoT:

* REST basado en API.
* WebSocket basado en API.

IoT es un término paraguas, que acoge a muchas tecnologías dentro de sí. Las siguientes tecnologías son ejemplos de las que dan paso al desarrollo de IoT:

* Wireless Sensor Networks (WSN).
* Cloud Computing.
* Big Data Analytics.
* Sistemas Embebidos
* Arquitectura y Protocolos de Seguridad
* Protocolos de Comunicación
* Servicios Web.
* Internet Móvil
* Motor de búsqueda semántico.

Dentro de todas estas tecnologías, las WSN son el corazón de IoT.

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 en esta tabla.

|  |  |
| --- | --- |
| **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 esta 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

Los dispositivos IoT tienen identidades únicas. Es un dispositivo físico que está embebido con sensores, actuadores, electrónica, software, una red de conectividad con otros objetos con los que intercambiar datos. Se realizan tareas como sensorizado, actuación y monitorización de manera remota. Hay mucha variedad de dispositivos, pero todos se pueden reducir a dos tipos: dispositivos estándar o dispositivos restringido. 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, al igual que sus capacidades de cómputo, memoria y otras características. Estos pequeños dispositivos restringidos necesitan de un dispositivo que haga de Gateaway para poder conectarse a la plataforma de la nube.

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

Como se ha comentado anteriormente, IoT es una red de dispositivos interconectados y se generarán una gran cantidad de datos. Estas redes son usadas para transmitir las señales que se recogerán posteriormente en los sensores con el resto de diferentes componentes, como los routers, puentes, LAN, WAN y MAN. Esta conectividad de las redes para los sensores se puede utilizar cualquiera de las tecnologías disponibles como el Wi-Fi, LTE, etc. Para la generación de esta red 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. Internet da la posibilidad de nuevos servicios, acelerando y permitiendo nuevas zonas de interacción con mensajes instantáneos y social networking, entre otros.

#### Servicios Backend

Los servicios principales que se ofrecen son:

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

Estos Servicios Backend se definen como un conjunto de servicios basados en la nube que ayudan a construir la aplicación IoT o aplicación Backend. Tienen un rápido almacenamiento de datos y una fácil gestión de usuarios. Para toda estrategia de una empresa IoT es una solución bonita. La parte delantera de una solución IoT asegura al usuario final un aumento de seguridad o añadirá valor en sus vidas.

Una vez se recolectan los datos con el producto, se envían a la nube, donde se guardarán en servidores que permitirán al usuario computarlos.

#### 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 una señal continua de salida. Sensores de este tipo son como los acelerómetros, sensor de presión, de sonido, de temperatura, etc. Sensores que captan cantidades analógicas y que son continuas en la naturaleza.
2. **Sensores Digitales:** Son usados para medidas analíticas generalmente. 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 otro lado, según el **tipo de dato** tenemos las siguientes clases de sensores:

1. **Sensores escalares:** La señal de salida generada por el sensor, o el voltaje, es proporcional a la magnitud que se está midiendo. Mediciones físicas como temperatura, presión, etc., son magnitudes escalares cuyo valor es suficiente información. Dichas mediciones también variarán respondiendo proporcionalmente a los cambios en la medida realizada.
2. **Sensores vectoriales:** Se produce una señal de salida, o voltaje, proporcionales 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

IoT puede considerarse una red física globalmente interconectada, en la cual, las cosas pueden ser conectadas o ser controladas remotamente. 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 conectar todo entre sí y permitir el envío de información entre 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 redes heterogéneas y los dispositivos relacionados se introducen en el servicio de Internet.

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, es importante que se automatice el encontrar y mapear los dispositivos o “cosas” para una dinámica red cambiante. 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 necesite. Esto habilita a los dispositivos para 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 dificultoso. Va apareciendo 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.

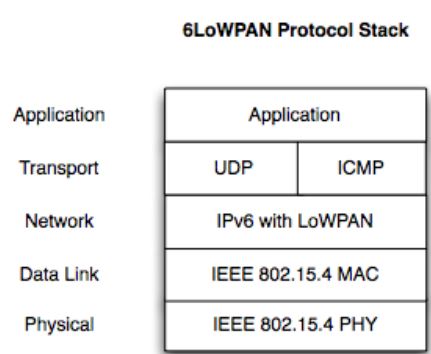
1. **WPAN en 6LoWPAN**

**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, lo ilustramos en la siguiente imagen:



*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 los esfuerzos de los diferentes equipos de trabajo en la investigación 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 apoyan 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 el apartado anterior como en *Ilustración 1,* 6LoWPAN es una capa de adaptación IoT IPv6, la cuál 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 perdidas (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, como de energía, memoria y 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.
* **Neighbor Discovery Protocol:** IPv6 usa este protocolo NDP, el cual se encarga de reconocer a los diferentes equipos 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 adaptarse 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. Contiki es de código abierto, muy portable y con capacidad de multitarea y control de eventos, el cuál fue pensado 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. Contiki 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 en la siguiente imagen:



*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

Ver [apartado 2.3.](#_THREAD)

## [TH](#_THREAD)READ

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 a la vez que ocurren.
* **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 típicos dispositivos junto con 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

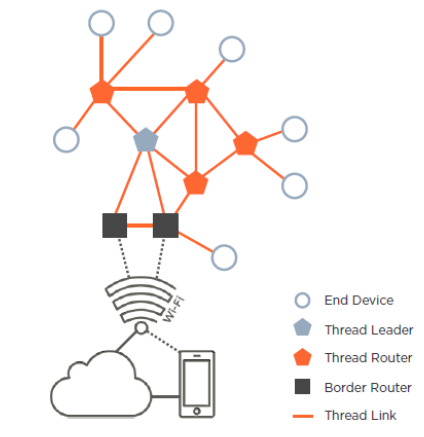
En estas redes, hay 2 clasificaciones para los posibles tipos de nodos que puede haber en la propia red.

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. 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. También se encargan de servicios de seguridad y de unión a los diferentes dispositivos que quieran unirse a la red. Estos routers, no están pensados para un modo “dormido”. Pueden degradar su funcionalidad y convertirse en REEDs (Router-Eligible End Devices).
* **Router-Eligible End Devices.** Estos REEDs pueden convertirse en Routers, pero acorde a la topología o condiciones de la red, 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. Su cambió a Routers es manejado por la propia red automáticamente sin necesidad de un usuario.
* **Sleepy End Devices.** Son dispositivos anfitriones o Host de la red. Solo se pueden comunicar a través de su 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. (**Misma funcionalidad comentada en la primera clasificación.)
  + **Active Router.** (Misma funcionalidad comentada en la primera clasificación.)
  + **REEDs.** (Misma funcionalidad comentada en la primera clasificación.) Pueden tener varios nodos hijos y mantener enlaces con los Routers vecinos.
  + **Full End Device (FED).** Es un FTD normal, que siempre actuará como un Dispositivo Final (End Device – ED). Un dispositivo FED nunca pedirá convertirse en Router, como haría un REED.
* **Minimal Thread Device (MTD).** Estos dispositivos solo pueden comunicar con el padre FTD al que están unidos. Los roles que pueden adquirir son:
  + **Minimal End Device (MED).** Es un dispositivo MTD cuyo receptor está habilitado todo el tiempo y puede comunicarse con su nodo padre en cualquier momento.
  + **Sleepy End Device.** **(**Misma funcionalidad comentada en la primera clasificación.)

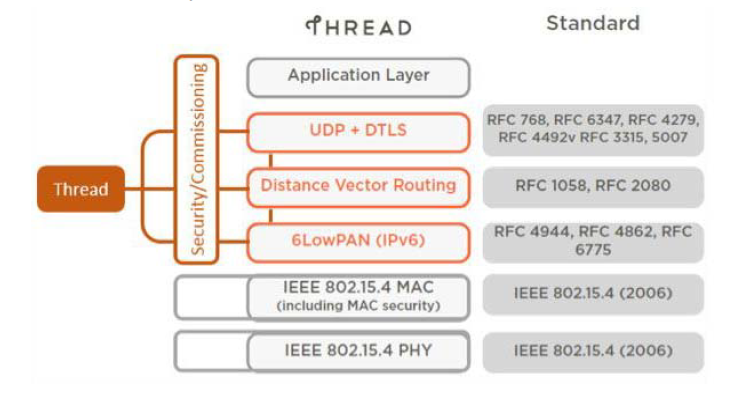


*Ilustración 3 Ejemplo de una topología de Red Thread en malla [9]*

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

### Protocolo Thread

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



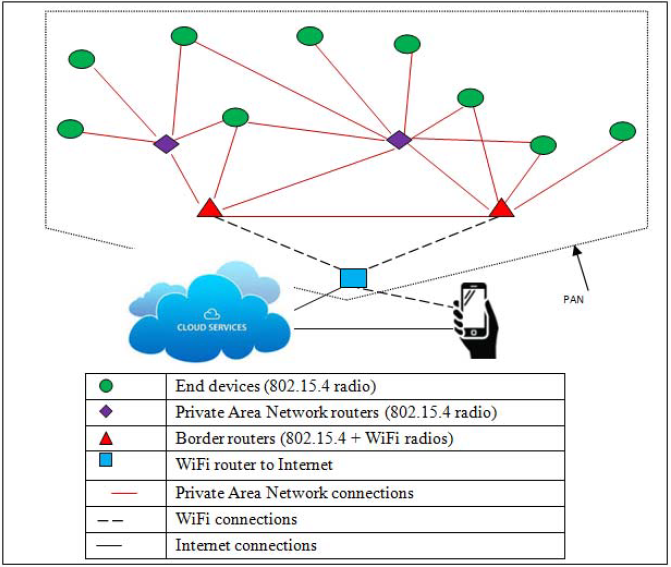
*Ilustración 4 Pila de Protocolo Thread en un sistema IoT [12], [13]*

#### Redes de área Privada - Private Area Network (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.

Antes de que un dispositivo pueda unirse a una PAN, este debe ser puesto 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, 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 en la Ilustración 5:



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

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

* **Nodos.** Las redes Thread se componen de nodos. Cada dispositivo o nodo tendrá unas determinadas características, unos darán conectividad a Internet a la PAN, otros solo podrán trasmitir a su vecino más cercano, otros podrán mandar datos de un nodo a otro. Los distintos tipos de dispositivos o nodos posibles son los vistos la sección [2.3.2. Tipos de dispositivos](#_Tipos_de_dispositivos).
* **Topología.** Uno de los requerimientos para las PAN Thread es no tener un punto singular de fallo. Para esto, Thread ha escogido una topología de red enmallada o de malla para sus redes PAN. El motivo de esta topología es su alta resiliencia, escalabilidad, robustez y una habilidad de auto reparación en caso de fallo de algún componente.
* **Enlaces de comunicación.** Para estos enlaces de la red, las redes PAN Thread usan radios inalámbricas. 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 deber. 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 solventa 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 el envío 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 malla.

Las direcciones Unicast IPv6 son usadas para el uso interno de la malla. Las Unicast pueden ser localizadores de enrutamientos, localizador de cualquier tipo (Líder, DHCPv6, Servicio, Commision, 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 si 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 rúters de la PAN envían periódicamente sus “costes” de enrutamiento a todos los demás rúters junto con la calidad de enlace de un salto. Los costes de enrutamiento para un mensaje se sacarán de este último dato.

Cada rúter 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 rúters de la PAN mantienen un seguimiento de la versión Thread de los enlaces de sus hijos y, de manera proactiva, manda actualizaciones de versiones más nuevas. Además, si el rúter es un nodo Leader, este tiene una base de datos con las asignaciones de id. Los rúters 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 implementación obligatoria cifrados. 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, lo cual reduce el ratio de compresión, dejando paso a su vez a más transmisiones de paquetes y potencias consumidas mayores-

#### Seguridad y Comisión 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 la comisión 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 comisión entre un aplicación de comisión, 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 Comisario deberá autenticarse también. Para ello se establece una conexión socket cliente/servidor, la cuál se le llamará Sesión de Comisión y sucede entre el Comisario 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 Comisario se registra con el BR, y este último comparte la información del Comisario al Nodo Leader, el cuál procederá a aceptar o rechazar al Comisario. En caso de ser aceptado el Comisario, el Líder 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 cuál los transmitirá al Comisario. 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 Comisario.

El Comisario 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 Comisario 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

## 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 debemos seguir los siguientes pasos.

##### Descarga del Software Requerido

Deberemos 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, descargaremos el fichero [KiBRA-v2.x.x.zip](https://www.kirale.com/products/ktbrn1/#resources) para un uso posterior.

Una vez descargada la imagen y el fichero .ZIP, necesitaremos 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 necesitaremos 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, deberemos 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. Recordemos que es recomendado el uso de una micro SD clase 10 de al menos 2 GB de capacidad.
* Seleccionaremos Flash y esperaremos a que termine.
* Expulsaremos la tarjeta SD y la introduciremos en la ranura para micro SD del módulo KTBRN1.

Captura de pantalla de un celular

Descripción generada automáticamente

Ilustración 7 Balena Etcher

##### Primera Instalación

Conectaremos con un cable USB el dispositivo KTBRN1 al PC. La primera vez que encendamos 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, seguiremos los pasos descritos debajo 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 instalemos el driver para el puerto USB a Serie, para ello comprobaremos si nuestro ordenador lo reconoce.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Ilustración 8 Border Router en Administrador de Dispositivos Previo a la instalación de Drivers

Si sale como en la imagen anterior, instalaremos los driver usando la herramienta Zadig.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

*Ilustración 21 Instalación Drivers de Border Router con Zadig*

Texto

Descripción generada automáticamenteUna vez tengamos acceso al KTBRN1 vía USB Serie, abriremos MobaXterm con una nueva sesión Serie. Seleccionaremos el puerto asignado al KTBRN1 con un BaudRate de 115200. Tras realizar esto aparecerá una consola de inició de sesión, en la cual deberemos iniciar sesión con el usuario *root* y la contraseña *kirale123*.

*Ilustración 22 MobaXterm*

La pantalla de bienvenida mostrará información sobre las direcciones IP configuradas en el KTBRN1 y que versión de KiBRA está utilizando. 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 deberemos cambiarla a una dirección IPv4 que esté dentro de la red local nuestra, 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. Accederemos 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 23 Login Panel Administración Web*

Las credenciales son las mismas a las mencionadas anteriormente. 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 24 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, iremos al menú de KiBRA y pincharemos en el icono de “Upgrade”, que es el que se sitúa al lado de la imagen del KTBRN1 en la sección de “System”.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

*Ilustración 25 Actualización KiBRA*

Después seleccionaremos el fichero KiBRA-v2.x.x.zip descargado anteriormente en nuestro ordenador, pincharemos en el botón “Install” y seguiremos las instrucciones que nos aparecerán en pantalla.

##### Configurar Border Router

Pincharemos 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 26 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 27 Parámetros a configurar con Out-of-band Commissioning Activado*

* 1. **Desactivando** esta opción, los parámetros a configurar seránInterfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

     Descripción generada automáticamente:

*Ilustración 28 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 29 Configuración Prefijo de Red*

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

##### Inicio del Border Router (Start-up).

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

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

*Ilustración 30 Menú KiBRA - Inicio de Border Router*

Después de clickear en el botón de Start, 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 31 Menú de KiBRA - Border Router Iniciado*

Si volvemos 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 32 Pestaña Settings - Export Settings*

En esta pestaña, podemos coger la información necesaria para que otros dispositivos puedan unirse 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.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

*Ilustración 33 Ventana Export Commissioning Information*

##### Servicios

El administrador de la red podrá ver que 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 34 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 35 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 36 Servicio NAT64*

###### Commissioner

El Border Router puede hacer también de comisario 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 37 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 38 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 39 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 cuál 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 siguientes comandos:

|  |
| --- |
| 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:

|  |
| --- |
| 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 40 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, necesitaremos una herramienta gratuita llamada **Zadig**. Esta herramienta puede descargarse de la página de Zadig <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 41 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 pinchar en “***Install / Replace Driver***”.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

*Ilustración 42 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 43 Instalación Drivers con Zadig Finalizada*

Ahora, en el *Administrador de Dispositivos*, el KiNOS Boot DFU deberá salir de la siguiente manera:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

*Ilustración 44 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>

#### 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 45 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 46 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 pinchar “*Install/Replace Driver”*.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

*Ilustración 47 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 48 Instalación Driver USB Serial (CDC) con Zadig Finalizada*

Ahora, en *Administrador de Dispositivos*, la interfaz KiNOS debería aparecer de la siguiente manera:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

*Ilustración 49 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:

|  |
| --- |
| ~$ 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:

|  |
| --- |
| ~$ 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 50 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.

|  |
| --- |
| ~$ 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 tenemos 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 Reference Guide – 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 de la siguiente manera:

|  |
| --- |
| 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.

Texto

Descripción generada automáticamente

Ilustración 51 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 52 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 Reference Guide – Comandos KBI

#### Operación de Interfaz

Diagrama

Descripción generada automáticamenteAntes de entrar en el funcionamiento de como usar los comandos KBI por puerto UART, en la siguiente imagen se muestra un esquema de un Host externo usando KBI para comunicarse con el módulo KTWM102 y el sistema KiNOS a través del puerto UART:

Ilustración 53 Esquema de comunicación entre Host Externo y el dispositivo KTWM102

Una vez conectado el sistema como en el esquema, deberemos 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 imagen, 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 siguiente:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 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:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 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:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Ilustración 54 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 trafico 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.

Antes de introducir los parámetros de la red, deberemos tener en cuenta una cosa: si deseamos o no realizar el proceso de “In Band” o no. Si queremos unirnos a una red en específico, deberemos activar el modo de Commisioning “Out of Band”, y posteriormente introducir todos los parámetros necesarios.

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 aún 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, y se generarán automáticamente en caso de no especificarles.

Una vez enviado el comando IfUp, el módulo tardará unos pocos segundos el realizar el proceso de unirse/crear la red. Una vez unido a la red, se guardará su configuración como configuración válida y ya no podrá modificarse salvo realizar 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”***.**

## PRUEBAS DE RED

En este capítulo describiremos las diferentes pruebas realizadas. En estas pruebas se han utilizado los siguientes elementos Hardware:

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  | |
|  | |

En cuanto a elementos Software se han usado:

* **Keil uVision 5:** Programación y debug del microcontrolador ARM de STM.
* **Keil uVision 3**: Programación para el ADuC841 usado en la Coockie.
* **KiTools:** Debug de 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.
* **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.

### 

#### 

#### 

#### 

#### 

# 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.

## 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.

Diagrama

Descripción generada automáticamenteEstas cuatro partes se integran y se conectan entre sí de la siguiente manera:

Ilustración 6 Jerarquía Circuito

### Circuito de Alimentación

En la parte de alimentación, el circuito realizado ha sido el siguiente:

Diagrama, Esquemático

Descripción generada automáticamente

Ilustración Circuito de Alimentación

Para entender el circuito deberemos 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, asegurándonos 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 siguiente:

Diagrama, Esquemático

Descripción generada automáticamente

Ilustración 7 Circuito Integración del Módulo

El dispositivo KTWM102, tal como hemos mostrado anteriormente, es el siguiente:

Imagen de la pantalla de un celular con letras

Descripción generada automáticamente con confianza baja

Ilustración 8 Dispositivo KTWM102

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:

Tabla

Descripción generada automáticamente con confianza baja

Ilustración 9 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.

Gráfico

Descripción generada automáticamente

Ilustración 10 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 en la Ilustración 11. Posteriormente se mostrará el Layout elegido separando las diferentes capas.

Imagen de la pantalla de un computador

Descripción generada automáticamente con confianza baja

Ilustración 11 Layout Completo

### Layout Capa TOP

Como se ve el la Ilustración 12, en la capa TOP se han posicionado la mayoría de las pistas y se ha realizado el posicionado de los componentes exceptuando 2 conectores.

Diagrama, Esquemático

Descripción generada automáticamente

Ilustración 12 Layout Capa TOP

### Layout Capa BOTTOM

Como se muestra en la Ilustración 13, en la capa BOTTOM se han posicionado 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 13 Layout Capa BOTTOM

## 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 14 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 15 KTDG102 Evaluation Dongle

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

Imagen de la pantalla de un celular con letras

Descripción generada automáticamente con confianza baja

Ilustración 16 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 17 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 57 Módulos de Procesamiento y de Alimentación de la Coockie

# PRUEBAS EXPERIMENTALES

## 

Ilustración 49 Diagrama de conexión PC - Dongle - uC

## 

### 

### 

### 

## 

### 

### 

### 

### 

## 

### 

### 

## 

### 

### 

## 

# CONCLUSIONES

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.

# LINEAS FUTURAS

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 estados 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.

# PLANIFICACIÓN TEMPORAL Y PRESUPUESTO

# 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.