Implementación de una puerta de enlace Zigbee

Proyecto de Investigación

Por

Cesar Castillo Alonso

En cumplimiento a los requerimientos para la obtención de la Especialidad de Tecnólogo en Mecatrónica

Revisor académico: Guillermo Ronquillo Lomeli

Santiago de Querétaro, Qro., México, Diciembre 10, 2020

Contenido

Índice de Figuras

Índice de Tablas.

Índice de Listados.

1 Introducción

El paradigma de Internet de las cosas (IoT por sus siglas en inglés) propone la interconexión de dispositivos físicos a través de redes que permiten compartir datos y controlar sus capacidades en tiempo real. Es fácil observar que dicho paradigma tiene una aplicación directa en la domótica y la industria, campo que integra la automatización, la informática y las nuevas tecnologías de la comunicación. Tal campo ha progresado notablemente en la última década, introduciendo nuevas tecnologías y evolucionando desde sistemas compuestos por objetos pasivos que reaccionan según la entrada del usuario, a sistemas basados en fuentes autónomas de información que interactúan con el entorno y anticipan y predicen las acciones de los usuarios.

Bajo el paradigma de la computación en la nube de IoT, las puertas de enlace de IoT generalmente se utilizan para intercambiar mensajes con nodos de IoT y una nube. Wi-Fi y Zigbee se destacan como tecnologías de comunicación preferidas para hogares inteligentes. WiFi se ha vuelto muy popular, pero tiene una aplicación limitada debido a su alto consumo de energía y la falta de capacidades de red en malla estándar para dispositivos de bajo consumo. Por tales razones, Zigbee fue seleccionado por muchos fabricantes para desarrollar dispositivos inalámbricos de automatización. Como consecuencia, estas tecnologías pueden coexistir en la banda de 2,4 GHz, lo que conduce a colisiones, velocidades más bajas y mayores latencias de comunicación.

El estándar de redes inalámbricas Zigbee encaja en un mercado que simplemente no está solucionado con otras tecnologías inalámbricas. Si bien la mayoría de los estándares inalámbricos se esfuerzan por la velocidad, Zigbee apunta a velocidades de datos bajas. Mientras

que otros protocolos inalámbricos agregan más y más funciones, Zigbee apunta a una pequeña librería que se adapta a microcontroladores pequeños. Mientras que otras tecnologías inalámbricas buscan brindar la última milla a Internet o entregar medios de transmisión de alta definición, Zigbee busca controlar una luz o enviar datos de temperatura a un termostato. Mientras que otras tecnologías inalámbricas están diseñadas para funcionar durante horas o quizás días con baterías, Zigbee está diseñado para funcionar durante años. Y mientras que otras tecnologías inalámbricas brindan de 12 a 24 meses de vida útil para un producto, los productos Zigbee generalmente pueden brindar décadas o más de uso.

La categoría de mercado que sirve Zigbee se denomina "control y redes de sensores inalámbricos" o simplemente "control inalámbrico". De hecho, el lema de Zigbee es "Control inalámbrico que simplemente funciona". El mercado de control inalámbrico tiene una serie de necesidades únicas para las que Zigbee es ideal, porque Zigbee es:

- Altamente fiable
- Económico
- Capaz de lograr una potencia muy baja
- Muy seguro
- Un estándar global abierto

Para lograr los criterios de bajo consumo y bajo costo, Zigbee tiene una restricción tecnológica, velocidad de transmisión de datos baja.

2 Planteamiento del problema

Las redes Zigbee no son independientes necesitan interactuar con otras redes o al menos con algún concentrador de datos como una PC. Este trabajo explica cómo funcionan las puertas de enlace en Zigbee y describe algunos retos comunes al implementarlas en el campo. Una puerta de enlace Zigbee, simplemente, es un medio de transferir datos entre una red Zigbee y dispositivos en otra red.

Sin embargo, en la actualidad no hay una especificación Zigbee para la puerta de enlace. Zigbee Alliance continúa mejorando la tecnología, pero las puertas de enlace aún no se han estandarizado. Cada implementación de Zigbee, tanto de proveedores de software como de silicio, incluye alguna forma de interfaz entre la placa Zigbee y una PC de desarrollo a través de un puerto serie, USB o Ethernet, para permitir que el nodo se programe y depure. En este trabajo, se mostrará una pequeña pero eficaz interfaz de puerta de enlace serie Zigbee a través de una aplicación llamada procesador de red Zigbee (ZNP) propuesta por Texas Instruments (TI) que aprovecha un puerto serie.

El objetivo principal detrás de esta interfaz de puerta de enlace es, en primer lugar, mantener simple y fácil de implementar en una MCU básico, que es típica de los nodos Zigbee. La plataforma TI ofrecen un puerto serie (o un puerto USB que actúa como un puerto serie). No hay nada que impida que este protocolo se utilice en sistemas basados en Ethernet alrededor del concepto IP.

3 Objetivos

3.1 Objetivo general.

Evaluar el protocolo Zigbee como solución para la implementación de redes de sensores inalámbricos.

3.2 Objetivos específicos.

- Realizar una investigación completa de aspectos generales del protocolo Zigbee, así como la tecnología actual para su implementación.
- Proponer un sistema de red de sensores inalámbricos que utilice el protocolo Zigbee para su comunicación, indicando sus subsistemas, tipos de dispositivos, así como las acciones que realizan.
- Proponer aplicaciones que realicen las tareas específicas de dispositivos Zigbee e implementarlas en el Kit de desarrollo CC1352- P2 de Texas Instruments
- Proponer una puerta de enlace entre la red creada y redes externas con el propósito que se pueda compartir información.
- Poner en marcha una red inalámbrica y validar que los datos obtenidos por los sensores de forma local son los que visualizamos de forma remota
- Obtener datos del estatus de la red en un formato que puedan ser utilizados su posterior procesamiento

4 Justificación

El CIDESI trabaja sobre tecnología de monitoreo y control de sistemas energéticos utilizando redes de sensores inalámbricas. Estos sistemas están compuestos por múltiples capas de enrutadores en una red en forma de árbol, donde el nivel más bajo es para los nodos de sensores distribuidos dentro de un sistema, y el nivel superior es para el coordinador de red que agrega todos los datos de los sensores. El coordinador de la red se encarga de tomar decisiones dentro del sistema, evalúa las lecturas del sensor que recibe y en consecuencia envía comandos a los diferentes actuadores para ajustar los parámetros ambientales del invernadero. La tecnología inalámbrica utilizada en este modelo es Zigbee.

Por ejemplo, en un sistema de invernaderos, la condensación del rocío en la superficie de las hojas tiene un impacto nocivo en los cultivos; por tanto, es necesario diseñar un sistema de monitoreo y control de invernaderos dedicado a prevenir fenómenos como este. Los nodos de sensores recogen diferentes parámetros que son útiles para calcular los puntos de rocío de las hojas en función de los puntos de rocío. Otro ejemplo, es el monitoreo de consumo de energía eléctrica de un edificio, en donde varios sistemas de medición son instalados estratégicamente y por medio de una red inalámbrica Zigbee se concentran todos los datos en un punto o se publican en la nube para procesos de análisis u optimización del consumo energético.

IoT puede ser parte del sistema de monitoreo y control, y puede tener una contribución más significativa porque el proceso de control puede ser activado de forma remota por el usuario en tiempo real.

Existen varios trabajos relacionados con el diseño de un sistema a partir de la monitorización y/o control de invernaderos o sistemas industriales ya sea de forma local o remoto mediante el

protocolo Zigbee. Por lo que es necesario desarrollar herramientas que permitan la conexión de redes de sensores en invernaderos u otro proceso industrial a través de Internet para aumentar la confiabilidad del sistema y proporcionar tolerancia a fallas.

5 Antecedentes

5.1 Descripción General del Estándar Zigbee

Zigbee es el nombre de la especificación de un conjunto de protocolos de comunicación de alto nivel basado en el estándar IEEE 802.15.4. El cual se utiliza para crear redes inalámbricas de área personal (WPAN, por sus siglas en inglés) de bajo consumo energético, en comparación con otras tecnologías similares (Bluetooth y Wi-fi), como se muestra en la Tabla 1. Tiene amplia utilización en la automatización de casas (domótica), obtención de datos de equipo médico, así como cualquier otro dispositivo de bajo consumo de energía y ancho de banda que requiera conexión inalámbrica (Al-Turjman & Imran, 2020).

Tabla 1. Comparativa de tecnologías inalámbricas (Jiang, Han, & Liu, 2011).

	Banda de frecuencia	**Consumo de energía**	**Distancia de transmisión**	**Nodos de la red**
Wi-Fi	2.4 [GHz]	Alto	1-100[m]	50
Bluetooth	2.4 [GHz]	Medio	1-100[m]	8
Zigbee	2.4[GHz]; 868[MHz]; 915[MHz]	Bajo	1-100[m]	255

En la Tabla 1 se observa las ventajas de una red Zigbee, por una parte, el bajo consumo de energía, algo muy atractivo al emplear sensores (de humedad, temperatura, de presencia, etc.) con comunicación inalámbrica. Así como la cantidad de nodos que soporta la red, 255, una cantidad bastante aceptable tomando en cuenta la cantidad de dispositivos que necesitamos interconectar, teléfonos inteligentes, control de iluminación, calefacción, etc.

5.2 Historia

En enero de 1998 un grupo de trabajo estableció el estándar IEEE 802.15 WPAN, para redes de corto alcance, con las siguientes características: baja tasa de bits, bajo consumo de energía y bajo costo económico. El estándar fue diseñado para utilizarse con dispositivos en redes móviles, como laptop, palmtops, teléfonos móviles, micrófonos y otros instrumentos electrónicos. Los requerimientos funcionales fueron una parte del proyecto BodyLAN por otra parte los requerimientos específicos eran: debían ser baratos, necesidades energéticas mínimas, baja tasa de transmisión de bits, dimensiones pequeñas, ya la capacidad de manejar redes de 16 elementos además de esto que pudieran funcionar al existir otras redes en la misma área. Estas definiciones fueron la base para el diseño de la familia WPAN (Pahlavan & Krishnamurthy, 2002, pág. 581).

En fases tempranas el desarrollo del estándar fue dividido en dos, porque algunas características de los sensores inalámbricos se necesitaban formular, para que pudieran tener compatibilidad con dispositivos de distintos fabricantes. Dos estándares aparecieron IEEE 1451.5 Wireless Smart Transducer Interface y el IEEE 802.15.4 Low Rate Wireless Personal Area Network (Callaway, 2003, pág. 342).

El estándar IEEE 802.15.4 fue publicado en la primavera del 2003, destinado a comunicaciones inalámbricas de corto alcance. Y la gran diferencia con el otro estándar IEEE 802.15, es la baja tasas de bits, esta es la razón por la que se llama Low Rate WPAN (LR-WPAN). El bajo consumo de energía, implementación económica, alta confiabilidad y baja tasa en transferencia de datos son también las principales características del estándar (Zheng & Lee, 2004).

5.3 Características Generales del Estándar IEEE 802.15.4

El estándar IEEE 802.15.5 se basa en modelo OSI (Open System Interconection) de comunicaciones. Todos los componentes de una red en general (no solo inalámbrica) se pueden agrupar dentro de un modelo OSI, este fue creado en 1980 por la International Organization for Standardization (ISO), en la se muestra en Figura 1 el modelo (Krutz & Ronald , 2017)

Figura 1. Modelo OSI

El modelo OSI está compuesto de 7 capas, a continuación, se describen:

Capa física: convierte los paquetes de datos en señales eléctricas u ópticas para enviarlas a los medios de transmisión, como pueden ser cables o alambres. Define las interfaces eléctricas y mecánicas de los medios de transmisión de la red

Capa enlace de datos: incluye la subcapa de Control de Acceso al Medio (MAC) y la subcapa de Control de Enlace Lógico (LLC). Realiza él envió de paquetes sin error

Capa de red: realiza la comprobación de errores, de direccionamiento, enrutamiento y el control de tráfico entre los nodos.

Capa de trasporte: configura la conexión lógica entre transmisor y receptor. La capa de transporte es orientada a la conexión y se asegura de enviar y recibir datos.

Capa de sesión: establece las conexiones de la comunicación, controla la transmisión de datos durante la sesión y finaliza la conexión.

Capa de presentación: realiza traducciones y conversiones para la capa de aplicación incluyendo descompresión, encriptado y desencriptado.

Capa de aplicación: proporciona soporte para aplicaciones y verifica la disponibilidad del receptor

El estándar IEEE 802.15.4 define dos capas inferiores, la capa física y la subcapa MAC la cual es una parte de la capa de Enlace de Datos (Data Link). Actualmente la transferencia de datos con todas sus funcionalidades está en las tareas de la capa física. La subcapa MAC define como se conecta con el canal físico.

El protocolo Zigbee está definido sobre la capa física y la capa MAC del estándar IEEE 802.15.4 y la Alianza Zigbee, se ha concentrado en las capas superiores conformando la llamada pila Zigbee (Zigbee Stack).

Es en la última capa donde se ha dejado abierta para realizar desarrollo en este protocolo y además se ha definido un Framework para permitir la interoperabilidad entre los fabricantes.

5.4 Transferencia de datos y sincronización de transmisiones

El estándar IEEE 802.15.4 utiliza el protocolo basado en el algoritmo CSMA/CA, el cual necesita escuchar el canal antes de empezar a transmitir, esto es para reducir la probabilidad de colisión con otras transmisiones en curso. Esta acción se realiza desde la capa MAC, que es responsable de: generar tramas de reconocimiento, asociación y desasociación, control de seguridad, así como funciones opcionales en la topología estrella (como generación de balizas y garantizar la administración de tiempo de slot) y finalmente dar soporte a las dos posibles topologías descritos en el estándar.

De forma predeterminada IEEE 802.15.4 no admite comunicaciones isócronas (en tiempo real) ni múltiples clases de servicios dentro de una sola red PAN. Sin embargo, el estándar incluye la implementación opcional de una superestructura administrada por el coordinador PAN, para el caso de topología estrella y árbol exclusivamente. Como se observa en la Figura 2 la Supertrama está delimitada por dos mensajes baliza sucesivos y es enviada por el coordinador PAN en intervalos regulares de tiempo. Por lo que cada baliza proporciona sincronización y además contiene información tal como: ID de la red, periodicidad de la baliza, y estructura de la Supertrama. La Supertrama está divida en dos partes. Un periodo de contención de acceso (CAP), durante el cual los dispositivos que deseen unirse a la red se comunican con el coordinador PAN utilizando el CSMA/CA, y un periodo libre de contención (CFP), organizado en 7 slots de tiempo contiguo, llamados slots de tiempo garantizado (GTSs) administrada por el PAN para sincronizar comunicaciones de baja latencia. Los slots de tiempo de garantía (GTSs) hacen posible un servicio de protocolo de alto nivel. Por lo tanto, para aquellas redes que no implementan niveles altos de servicio el empleo de GTSs es opcional. El uso de comunicaciones con balizas habilitadas o no, depende de las aplicaciones: la transmisión de balizas tiene

desventajas cuando los mensajes no son esperados por el coordinador de la red y solo se tiene previsto tráfico ligero de los dispositivos de la red al coordinador (Verdone, Dardari, Mazzini, Conti, & Andrea, 2008).

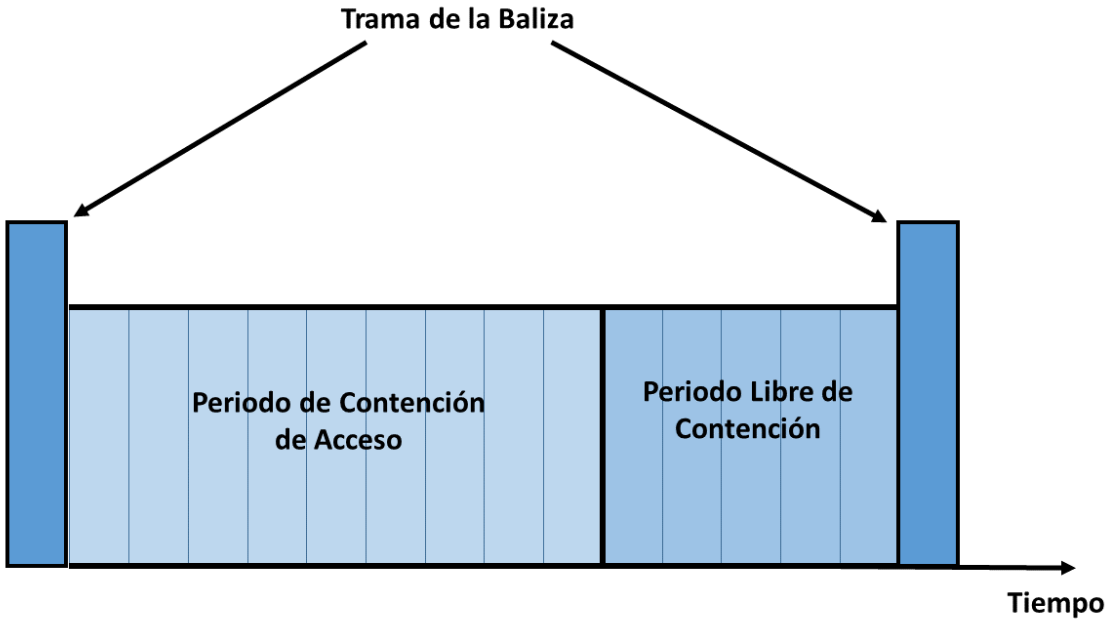


Figura 2. Estructura de la Supertrama.

5.5 Componentes de una Low Rate PAN

El estándar IEEE 802.15.4 define dos tipos de nodos en esta red, los cuales tienen características diferentes: dispositivo con funcionalidad total (DFT) y dispositivo con funcionalidad limitada (DFL). Como su nombre lo indica los DFT tienen todas las características y funciones del estándar, esto le permite funcionar como una central de control, también se le llama coordinador PAN, o solo coordinador. Los DFT se pueden comunicar con todo tipo de dispositivos, por el contrario, los DFL no tienen la capacidad de enrutamiento, por lo que actúan como dispositivos finales, normalmente son los encargados de realizar las

mediciones (sensores). Las capacidades de procesamiento de datos de los DFL es limitada, se pueden interconectar un DFT de esta forma y pueden enviar pequeñas cantidades de datos (IEEE Std 802.15.4, 2003) (Zheng & Lee, 2004).

5.6 Redes Ad hoc

Las redes Ad hoc (o de creación espontanea) son de voz y datos, se establecen temporalmente sin tener que crear una infraestructura previa. Puede haber un número limitado de usuarios en tales sistemas. Se establecen para uso personal dentro de un área limitada como pueden ser una oficina o una planta. Se pueden crear en cualquier lugar ya sea de forma temporal o permanente, por lo que las frecuencias portadora utilizadas son industriales, científicas y de banda médica (ISM) (Daal & Upena, 2005).

Las redes Ad hoc son redes inalámbricas o con dispositivos que tienen conexión temporal, donde algunos dispositivos son parte de la red solo durante una sesión de comunicaciones o en el caso de dispositivos móviles o portátiles están muy cercanos a la red. Las redes inalámbricas Ad hoc no necesitan una infraestructura. Esta tecnología ha sido ampliamente implementada en casas y oficinas en donde nuevos dispositivos se tienen unir y desconectar a la red rápidamente, así como tener comunicación inalámbrica entre ellos. Las redes Ad hoc pueden ser de dos tipos (Daal & Upena, 2005):

- PAN para dispositivos personales, si es de comunicación inalámbrica se conoce como WPAN, entre sus características están que tiene una cobertura menor a 100 metros, físicamente seria alrededor de una persona donde ocurre la comunicación Ad hoc.
- El otro tipo es una red local inalámbrica, o WLAN, esta tiene una cobertura de 500 metros. Algunas configuraciones de este tipo necesitan infraestructura para poder

tener acceso a otras redes de trabajo. Algunas tienen una administración centralizada, otras pude ser Ad hoc pura con redes distribuidas

5.7 Topologías de redes inalámbricas

El diseño de la topología de redes una de los principales elementos para crear redes inalámbricas se sensores eficientes. Los criterios para elegir la topología son: tamaño de la red, la aplicación, el medio ambiente y la demanda. El consumo de energía es la característica más crítica, porque en el nodo sensor los recursos de energía son muy limitados. Lo modelos de las topologías LR-WPAN no son muy diferentes a otras topologías de redes inalámbricas. La topología de red puede cambiar al agregarse un nuevo elemento o que uno existente se mueva o se apague.

En la Figura 3 se muestra dos diferentes topologías de redes inalámbricas LR-WPAN. Como se observa en la topología estrella los dispositivos solo se pueden comunicar con el coordinador PAN. Por el contrario, en la topología peer-to-peer, o también conocida red entre iguales, los dispositivos se conectan uno con otro. Dicho de otra manera, se pueden construir redes más complejas utilizando la topología entre pares.

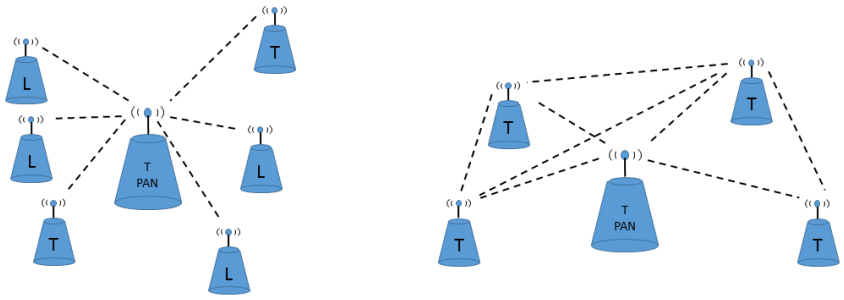


Figura 3. Topologías de red estrella y peer-to-peer

Ambas topologías se pueden configurar de forma automática con todos los dispositivos en una particular banda de frecuencia. Los dispositivos tienen una dirección extendida única de 64 bits, estos se pueden comunicar directamente dentro de la PAN utilizándola, o pueden tener una dirección corta de 16 bits, esta es asignada por el coordinador PAN al ser asociado a la red. El tamaño de la red determinara que tipo de dirección se utilizara (IEEE Std 802.15.4, 2003).

La topología estrella se puede formar con un DFT, que se convertirá en el coordinador PAN. Deberá determinar un número identificador de la red (ID PAN por sus siglas en ingles). Después de esto otros dispositivos, ya sean DFL o incluso DFT en la misma banda de frecuencia podrán unirse a la red (IEEE Std 802.15.4, 2003).

En las redes peer-to-peer los dispositivos se pueden comunicar entre ellos dentro de un rango de comunicación y en la misma banda de frecuencia por lo que son DFT. Básicamente se estará formando una red del tipo ad-hoc (red descentralizada). La formación de la red comienza cuando el primer dispositivo DFT empieza a funcionar dentro de cierta banda de frecuencia. Entonces se

convertirá en el coordinador PAN y formará así el primer clúster, establecido así un clúster principal (CLP). El coordinador PAN establece el identificador de clúster a cero e inicia la trasmisión de las balizas mensajes a los dispositivos vecinos. Los dispositivos DFT y DFL dentro del área enviaran solicitudes de unión a la red, el coordinador PAN se encargará de aceptar o no su ingreso a la red. Los DFT pueden formar su propio clúster, y con esto convertirse en líder de clúster y unirse a la red. El coordinador PAN guardará a estos dispositivos (DFT) como hijos, en su lista de vecinos. Para los dispositivos DFT, el coordinador PAN es su padre, en su lista de vecinos. Después de esto el nuevo clúster principal trasmitirá la baliza de mensaje dentro del rango. Grandes cadenas de red se pueden construir con este tipo de proceso. En la formación de redes también se pueden utilizar topologías maya o token ring (anillo). Comúnmente las redes de sensores tienen más de 10 dispositivos y es una mezcla de las diferentes topologías. Cuando las redes crecen demasiado, a latencia y el consumo de energía también, lo cual es un problema serio. La administración y control de esas redes se dificulta más, a causa de esa razón. Una topología clúster de ilustra en la Figura 4.

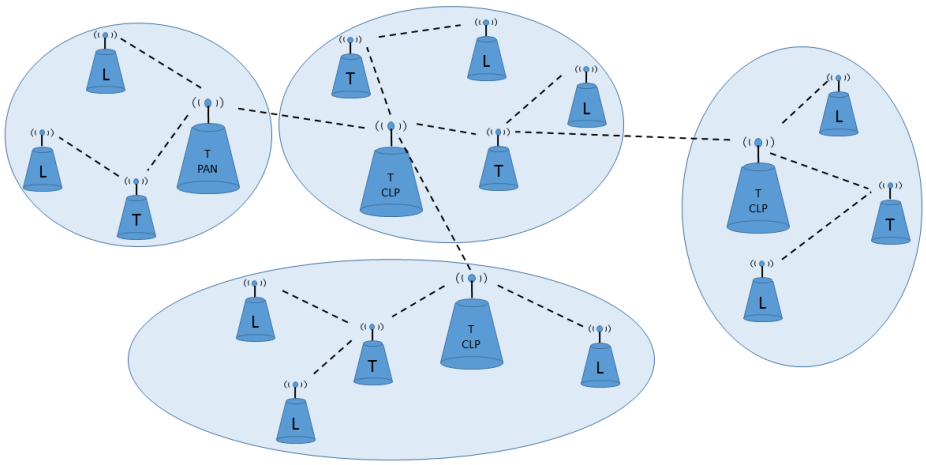


Figura 4. Topología Clúster

5.8 Tipos de dispositivos en una red Zigbee (Texas Instruments, 2005-2018).

Existen tres tipos de dispositivos lógicos en una red Zigbee, coordinador, enrutador y dispositivo final. Una red Zigbee consiste en un dispositivo con capacidad para formar una red (estos pueden ser coordinador o enrutador), múltiples enrutadores y nodos de dispositivos finales.

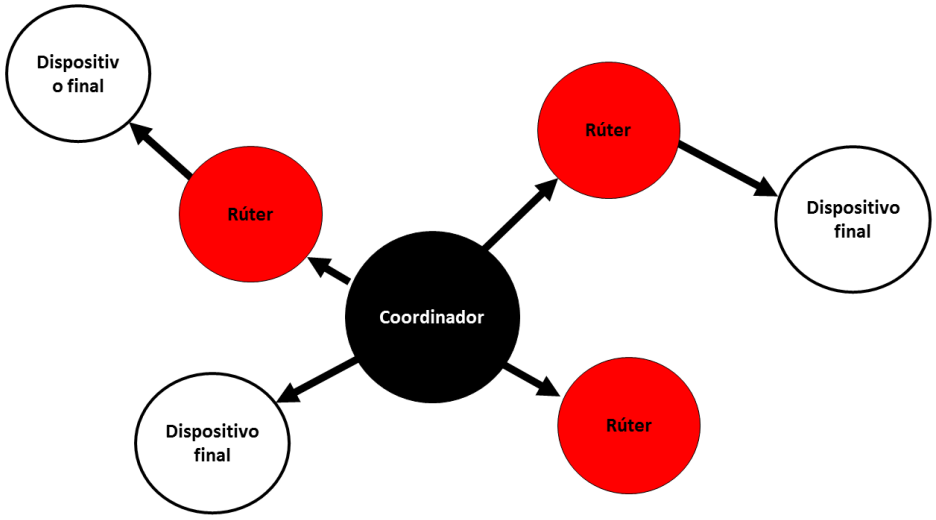


Figura 5 Red Zigbee.

En la Figura 5 Figura 5 observamos un ejemplo típico de una red Zigbee, el círculo negro es el coordinador, los círculos rojos los enrutadores (o rúters) y como dispositivos finales tenemos los círculos blancos.

5.8.1 Coordinador

El coordinador es un dispositivo que tiene la capacidad para formar una red, pero carece de esta para unirse a una red. Esto es, puede crear su propia red, pero no puede unirse a una existente. Para crear una red el coordinador escanea el entorno RF (Radio Frecuencia) de las redes existentes, elige un canal y asigna el identificador de la red (PAN ID).

Las tareas principales del coordinador son iniciar la red, administrar las claves, además de eso se comporta como un dispositivo enrutador. Es importante mencionar que los procedimientos de unión y salida de la red de dispositivos deben ser atendidos por el coordinador, por lo tanto, no debe estar ausente de su propia red.

5.8.2 Rúter (Enrutador)

El rúter realiza las siguientes funciones: permitir que otros dispositivos se unan a la red, enrutamiento de salto múltiple (multi-hop) y asistencia para la comunicación en los dispositivos finales hijos. En general se espera que los rúters estén activos todo el tiempo y por lo tanto deben estar conectados a la red de energía eléctrica.

5.8.3 Dispositivo Final

El dispositivo final no tiene una responsabilidad en el mantenimiento de la infraestructura de la red por lo que puede dormirse o despertarse cuando se desee. Se puede alimentar con una batería. En general los requerimientos de memoria (memoria RAM) son menores para el dispositivo final.

5.9 Puerta de enlace (Gateway)

Es una estación de red, que puede ser tanto lógico o conceptual que sirve de interconexión entre dos redes, que de otra forma serian incompatible. Un Gateway realiza la conversión de protocolos a través de numerosas capas de comunicación. Esto significa que es más complicado que solo un puente de comunicación. La norma ISO lo define como un dispositivo que atraviesa 7 capas. Una definición más actual, pero menos formal, lo describe como cualquier mecanismo que provee acceso a otro sistema (Dimon, 2003, pág. 228).

5.9.1 Arquitectura de una puerta de enlace

Para logar la compatibilidad entre los dos sistemas se utiliza un transceptor dual, es decir se toman las capas de cada red para producir los datos nativos que luego se pueden enviar de retorno al otro sistema.

Las puertas de enlace constan de dos conjuntos completos de siete capas de enlace, excepto en los casos que la arquitectura de red no utilice. Cuando las redes difieren ampliamente como puede ser una red personal conectándose a una red abierta se emplearán estos dispositivos. Técnicamente una puerta de enlace es un transceptor por cada red, con su salida controlada por una computadora, y para que se realice la conversión del protocolo utiliza temporización y diferentes señalizaciones físicas, como se muestra en la Figura 6 (Thompson, Lawrence , & Tim, 2016)

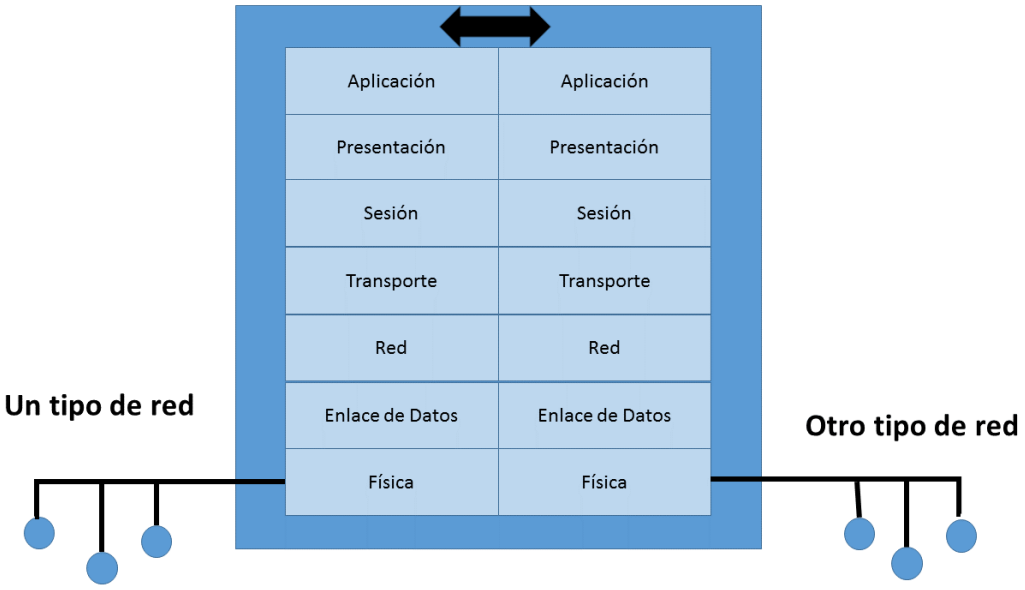


Figura 6. Arquitectura de una puerta de enlace

De la anterior definición concluimos que una puerta de enlace es un dispositivo, que puede ser una computadora personal, la cual nos permitirá vincular dos redes, por ejemplo, puede ser una LAN (red local) conectada hacia el exterior (Internet). Esto significa que se interconectara redes con arquitectura y protocolos distintos. Y el propósito general es traducir la información del protocolo usado en la red LAN al protocolo destino.

5.10 El Internet de las Cosas (IoT)

El termino el Internet de las cosas fue acuñando por Kevin Ashton en 1999, mientras trabajaba como gerente de marca en Procter & Gamble, se interesó en utilizar tecnología RIFD (identificación por radiofrecuencia) para ayudar a administrar la cadena de suministro (supply chain) de P&G (Postolache, Octavian, Mukhopadhyay, & Chandra, 2019, pág. 2)

> Si tuviéramos computadoras que supieran todo lo que hay que saber sobre las cosas, utilizando los datos que recopilaron sin nuestra ayuda, podríamos rastrear y contar todo, de esta forma se reduciría el desperdicio, la pérdida y el costo. Sabríamos cuándo es necesario reemplazar, reparar o retirar las cosas. Necesitamos empoderar a las computadoras con sus propios medios de recopilar información, para que puedan ver, oír y oler el mundo. La tecnología de sensores y RFID permiten computadoras para observar, identificar y comprender el mundo sin las limitaciones de datos ingresados por humanos. (Kevin Aston 1996)

El IoT es un concepto informático que describe un futuro donde los objetos físicos cotidianos estarán conectados a Internet y podrán identificarse con otros dispositivos. IoT es una red de dispositivos que se comunican entre sí, mediante conectividad IP sin interferencia humana. El ecosistema IoT consta de objetos inteligentes, dispositivos inteligentes, teléfonos inteligentes y

tabletas, etc. Utilizará identificación por radiofrecuencia (RFID), códigos de respuesta rápida (QR), sensores o tecnología inalámbrica para permitir la intercomunicación entre dispositivos (Sachchidanand & Nirmala, 2015, pág. 1577).

5.10.1 Desafíos del internet de las cosas

Ha sido muy rápida la adopción del concepto IoT en ámbitos académicos e industriales, por lo que han surgido diversos requerimientos de esta tecnología. Especialmente en la actualidad debido a la alta conectividad a Internet, la cantidad de dispositivos que se interconectan crece de forma espectacular. Por otro lado, la mayoría de dispositivos conectados dentro de IoT tienen recursos limitados, en lo que se refiere específicamente con la batería que suministra energía. Por lo tanto, construir un IoT con eficiencia energética a gran escala es muy importante. En este sentido son tres características esenciales de los requerimientos básicos de IoT:

- Capacidad de percepción: Se realiza a través de varios sensores o identificadores, en concreto proporcionan un enlace de objetos reales (es decir con cantidades físicas medibles) con datos digitales compatibles con Internet. En un panorama general del ecosistema IoT todos estos elementos de percepción (sensores) trabajan en conjunto, desde donde se necesita realizar la medición, hasta donde el usuario lo desee, al usar el internet
- Calidad en Transmisión: El gran avance de las telecomunicaciones y de las redes industriales proporciona no solo una transmisión eficiente y diversa en redes inalámbricas sino también un escenario de comunicación altamente entrelazado. IoT naturalmente utiliza tecnologías de última generación, es su columna vertebral, como por ejemplo la cuarta generación LTE (Long Term Evolution), así como sus diversas

variantes de la tercera generación 3GPP (Generation Partnership Project). Sin embargo, la transmisión de datos en IoT enfrenta nuevos desafíos debido a algunas características especiales de este nuevo paradigma de redes, como lo son alta densidad de nodos, arquitectura de red ad hoc, ancho de banda bajo entre los nodos y limitaciones en la capacidad de las baterías en los nodos regulares.

- Capacidad de procesamiento de datos: Los nodos en el IoT deberían tener la capacidad de tomar decisiones inteligentes, en base a los datos obtenidos. Con de acceso a internet el procesamiento puede ser realizado por computación en la Nube (servicios en la Nube) o con otra estrategia de aceleración remota, pero un dispositivo/nodo IoT debe estar equipado con el menor recurso informático para logar el primer paso de procesamiento de datos.

Los requerimientos del IoT tienen algunos desafíos para su implementación. A diferencia de las redes de sensores tradicionales. IoT logra una escala mayor y se vuelve más complejo. Además, al aumentar de dimensiones necesitan más potencia eléctrica, por lo que la eficiencia energética es un tema que se tiene que tomar enserio (Huang, Jun Hua, & Kun, 2017).

5.11 Sensores inalámbricos

Los sensores inalámbricos están ganando gran popularidad en el monitoreo industrial al ser económicos y fácil de instalar. Los sensores inalámbricos tienen la habilidad de adquirir datos y son una sustitución de los sensores cableados tradicionales, sin embargo, no funcionan exactamente como estos. Estrictamente hablando los sensores inalámbricos no solo son sensores sino nodos autónomos de adquisición de datos, a los que los sensores tradicionales se pueden conectar. Los sensores inalámbricos se consideran una plataforma donde los elementos de computación móvil y comunicación inalámbrica convergen con la detección del transductor.

5.11.1 Componentes de un sensor inalámbrico

Los sensores inalámbricos son generalmente divididos en dos grupos: sensores pasivos y sensores activos. Los sensores pasivos realizan la medición específica de una variable física o química y responden de forma pasiva al estado del sistema que está siendo monitoreado. Estos constan de tres elementos básicos: interfaz del sensor, un procesador de cómputo y un transceptor inalámbrico. En contraste los sensores inalámbricos activos, estos generan señales de control, a continuación, detectan la respuesta del sistema a esas señales y actuaran de acuerdo a la programación establecida. Los sensores activos tienen un elemento más, una interfaz de un actuador, que es la encargada de generar la señal de actuación, como se muestra en la Figura . Al no tener cables, los sensores inalámbricos requieren almacenar internamente energía para su operación. Se pueden utilizar diversas fuentes de energía, tales como las baterías, Identificación por Radio Frecuencia (RFID), fuentes de energía del medioambiente tales como solar, vibratoria o térmica.

Los sensores inalámbricos contienen una interfaz en la cual se pueden conectar transductores de detección. La interfaz de detención es gran parte responsable de convertir la salida analógica de los sensores es una representación digital. Una vez que los datos han sido recopilados por la interfaz de detección el procesador computacional se encarga del procesamiento de datos de forma local así como de las tareas de cómputo, con esto la demanda de recursos computacionales del procesador central de datos se reduce. Para realizar estas tareas, el procesador computacional es proporcionado por un microcontrolador, el cual puede almacenar los datos, ya sea en la RAM o en memoria no volátil. Para poder interactuar con otros sensores inalámbricos y transferir datos a almacenamientos remotos, es necesario utilizar un transceptor inalámbrico, tanto para el trasmisor como para el receptor de datos. Finalmente, una interfaz de un actuador proporciona al

sensor la capacidad de interactuar directamente con el sistema físico. El elemento central de la interfaz del actuador es un convertidos digital analógico (DAC) el cual convierte una señal digital generada por el microcontrolador en una señal de voltaje continua análoga de salida (Chen, Hua-Peng Ni, & Yi-Qing, 2018, págs. 37-38).

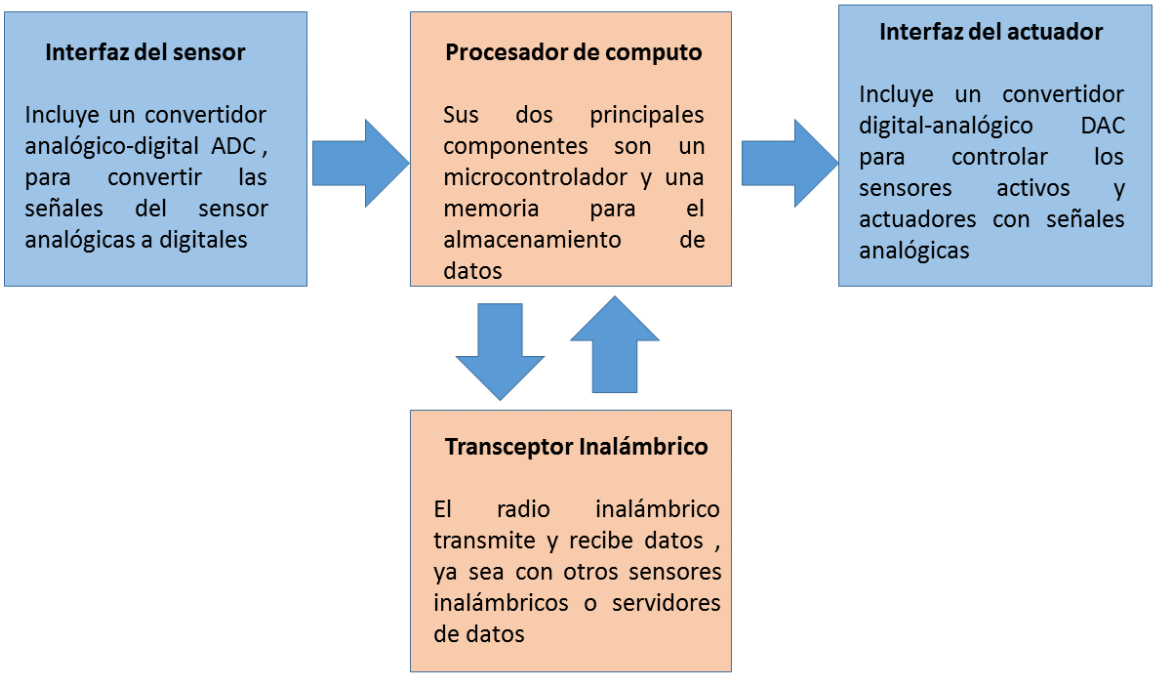


Figura 7. Componentes de un sensor inalámbrico

5.11.2 Modelo de red de sensores inalámbricos

Una red de sensores inalámbricos consiste en cientos o miles de nodos de bajo-costo, los cuales podrían tener una posición fija o en movimiento aleatorio para monitorear el ambiente. Debido a pequeño tamaño tienen ciertas limitantes. Los sensores usualmente se comunican entre ellos utilizando en enfoque multi saltos (multi hop). El flujo de datos termina en nodos llamados estaciones base (a veces también se les denomina sinks, del inglés). La estación base enlaza la red de sensores con otras redes de trabajo (como un Gateway), para trasmitir los datos detectados

para su posterior procesamiento. Las estaciones base tienes mayores capacidades que los nodos simples, ya que deben realizar procesamiento de datos más complejo, esto justifica el hecho que las estaciones base tienen computadoras personales y por supuesto suficiente memoria, energía y recursos computacionales para poder realizar las tareas. Por lo general, la comunicación entre estaciones base se inicia a través de enlaces de gran ancho de banda.

Uno de los mayores problemas de las redes de sensores es el consumo de energía, que se ve muy afectado por la comunicación entre nodos. Para solucionar esto, se introducen puntos agregados a la red (rúters). Esto reduce el número total de mensajes intercambiados entre nodos y ahorra un poco de energía. Usualmente los puntos agregados son nodos regulares que reciben datos de sus nodos vecinos, realizan algún tipo de procesamiento y luego reenvían los datos filtrados al siguiente salto. Similar a agregar puntos es agruparlos. Los nodos sensores están organizados en grupos, cada clúster tiene un "clúster head" o líder del clúster. La comunicación dentro del clúster debe ser a través del líder de clúster, que luego la enviara a otro vecino, líder de clúster también, hasta que la información alcancé su destino, que es la estación base. Otro método para ahorrar energía es configurar los nodos para que estén inactivos cierto tiempo (también conocido como modo dormido), si no son necesarios y activarlos (wake up) cuando sea necesario. El reto es encontrar un patrón en el que es consumo de energía sea uniforme para todos los nodos de la red.

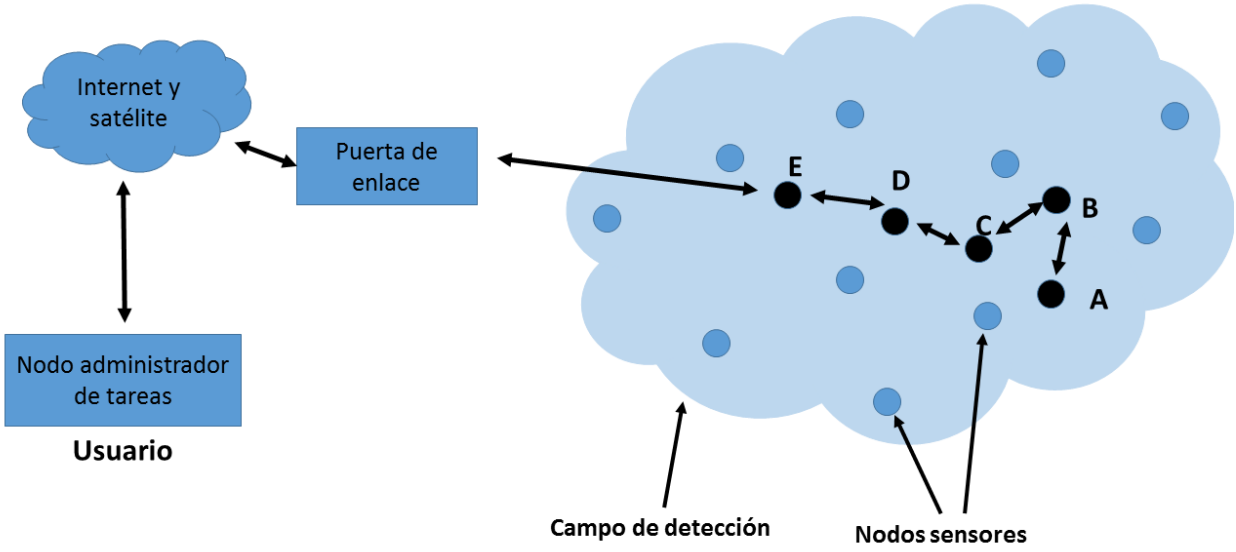


Figura 8. Modelo de una red de sensores inalámbricos.

El diseño de la red descrita en la Figura 8 está determinado por muchos factores, incluyendo, tolerancia a fallas, escalabilidad, costos de producción, operatividad en el medio ambiente, topología de los sensores de la red, restricciones de hardware, medio de transmisión, consumo de energía. Se detallan a continuación

Tolerancia a fallas: algunos sensores pueden fallar o bloquearse debido a la falta de energía, daño físico o interferencia ambiental. Una falla en un nodo sensor no debería afectar la tarea general de los sensores de la red. Este es un problema de confiabilidad o tolerancia a fallas. La tolerancia a fallas es la capacidad de mantener la funcionalidad de la red sin ninguna interrupción debido a las posibles fallas del nodo sensor.

Escalabilidad: el número de sensores desplegados para estudiar un fenómeno puede ser del orden de cientos o miles. Dependiendo de la aplicación, incluso puede alcanzar el número extremo de millones. Los esquemas deben poder trabajar con este número de nodos y también utilizar alta densidad de nodos. La densidad puede variar desde unos pocos sensores hasta cientos en una región con menos de 10 metros de diámetro.

Costo de producción: Dado que las redes constan consisten en una gran cantidad de nodos sensores, el costo de un solo nodo es muy importante para justificar el costo total de la red. El costo de cada nodo sensor debe ser suficientemente bajo. El costo debería ser menor a un dólar para que sea factible.

Restricciones de hardware: un sensor tiene cuatro elementos básicos, unidad de detección, unidad de procesamiento, unidad de transceptor y unidad de energía. Todos estos elementos deben caber dentro de una caja, parecida el tamaño de una caja de cerillos. El tamaño puede ser incluso menor a un centímetro cubico. Aparte de las dimensiones, hay otras restricciones para los nodos sensores, deben consumir muy poca potencia eléctrica, operar en altas densidades volumétricas, es prescindible que opere de forma autónoma y que se adapte al medio ambiente.

Topología de la red de sensores: Cientos o miles de nodos se esparcen en todo el campo de detección, las densidades pueden ser tan altas como 20 nodos/ m^3. La implementación de una gran cantidad de nodos requiere una gran cantidad de mantenimiento topológico.

Operatividad en el medio ambiente: los nodos sensores se despliegan muy densamente cerca o dentro del fenómeno a observar. Usualmente trabajan sin supervisión en áreas remotas geográficamente. Pueden trabajar al interior de una gran maquinaria, en el fondo del océano, o en

un campo contaminado química o biológicamente, en un campo de guerra, más allá de las líneas enemigas o en una casa o edificio grande.

Medios de transmisión: en una red de sensores multi saltos, los nodos de comunicación están conectados por un medio inalámbrico. Estos enlaces están formados por ondas de radio, infrarrojos o medios ópticos. Tanto los medios ópticos como infrarrojos requieren una línea de visión entre emisor y receptor. Para que sea posible la operación global la operación de la red de sensores, el medio de trasmisión elegido debe estar disponible en todo el mundo.

Consumo de energía: Los nodos sensores inalámbricos solo puede equiparse con una fuente de alimentación limitada (0.5 [Ah], 1.2 [V]). La vida útil del sensor depende fuertemente de la batería. En una red de sensores multi saltos cada nodo desempeña el papel de tanto generador como enrutador de datos. La disfuncionalidad de algunos nodos puede causar cambios significativos en la topología de la red y podría requerir el re direccionamiento de paquetes y la reorganización de la red. Por lo tanto, la conservación y administración de energía adquieren gran importancia. El consumo de energía se puede dividir en tres sectores: detección, procesamiento de datos y comunicación. La energía necesitada en la detección depende de la naturaleza de la aplicación, la detección esporádica puede consumir menos energía que un monitoreo constante de eventos. La complejidad en la detección de eventos juega también un papel importante y crucial en la determinación del gasto energético, niveles de ruido altos en el ambiente pueden causar corrupción, y aumentar la complejidad en la detección. La energía utilizada en la comunicación, es tanto para transmisión como recepción de datos. El estado del arte en los transceptores de radio de baja potencia es típicamente tanto potencia de transmisión como recepción de 20 dBm y una potencia de salida de 0 dBm. La energía consumida en el procesamiento de datos es mucho menor comparada con la comunicación. El procesamiento de

datos local es crucial para minimizar el consumo de energía en una red de sensores de multisaltos. Un nodo sensor debe tener habilidades de cómputo y ser capaz de interactuar con su entorno.

6 Metodología

6.1 Descripción general

La Figura 9 muestra el diagrama de bloques del proyecto, del lado derecho tenemos la PC que realiza la función de Host, en la cual se ejecuta una aplicación que obtiene datos del nodo sensor. El Host se comunica por el puerto serie con uno de los kit de desarrollo CC1352-P2, en donde previamente se han cargado la aplicación ZNP. La comunicación inalámbrica entre las dos tarjetas se realiza a través de los servicios del Z-Stack, se utiliza el protocolo Zigbee

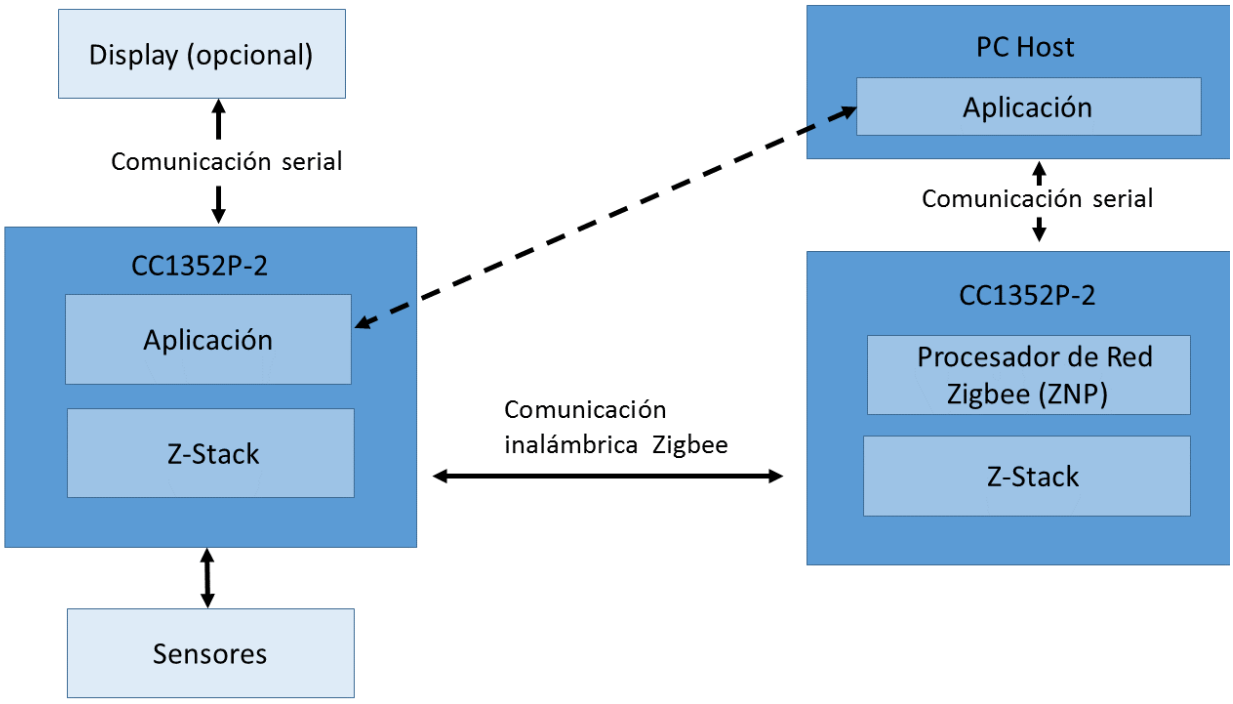


Figura 9. Diagrama de bloques del proyecto

Del lado izquierdo tenemos el otro kit de desarrollo CC1352P-2, se ejecuta una aplicación, que se encarga de los sensores, en este caso particular sensor de temperatura, además se utilizó un monitor para visualizar los datos de forma local, la comunicación entre el dispaly y el MCU es por el puerto serie. Como se puede observar la puerta de enlace la integran la PC, como

procesador Host y el kit de desarrollo CC1352 con la Aplicación ZNP, configurado como coordinador de red. En las secciones posteriores se describen los módulos.

6.2 Z-Stack

6.2.1 Perfil del Stack

El conjunto de parámetros con valores específicos que definen al dispositivo, se le llama perfil de la pila (stack profile). Dichos parámetros son definidos por la alianza Zigbee (The Zigbee Alliance). Todos los dispositivos en una red particular deben tener el mismo perfil de pila (todos los dispositivos de la red deben tener la configuración de los parámetros con los mismos valores).

Si el desarrollador de aplicaciones cambia alguno de esos valores, deberá saber que dichos dispositivos no podrán interactuar con dispositivos de otros vendedores que siguen las especificaciones del stack profile Zigbee. Por lo tanto los desarrolladores de redes-cerradas (closed -network) deben cambiar estos valores. Este tipo de stack profile son llamados “network- specific” stack profile (perfil del stack “red-especifica”).

El identificador del stack profile del dispositivo se encuentra en la baliza transmitida por el dispositivo. Esto posibilita al dispositivo determinar el stack profile de la red antes de unirse a ella.

6.2.2 Direccionamiento

6.2.2.1 Tipos de dirección

Los dispositivos Zigbee tienes dos tipos de dirección. La dirección IEEE de 64 bits (también llamada MAC Address o Dirección Extendida) y la dirección de red de 16 bits (también llamada dirección lógica o dirección corta).

La dirección de 64 bits es única en todo el mundo y es asignada para todo el tiempo de vida del dispositivo, es usualmente asignada durante su fabricación o en su instalación. Está determinada y configurada por el IEEE (los últimos 24 bits) y por el fabricante (primeros 24 bits) utilizando el Organizationally Unique Identifier.

La dirección de 16 bits es asignada al dispositivo cuando este se une a la red y está diseñada para usarse mientras permanezca dentro de esta, debe ser única, ya que se utiliza para identificar los dispositivos, así como para enviar y recibir mensajes dentro de la red.

6.2.2.2 Asignación de la dirección de red

6.2.2.2.1 Direccionamiento Estocástico (Aleatorio)

Zigbee Pro usa el esquema de direccionamiento aleatorio para asignar direcciones en la red. Este esquema de direccionamiento aleatorio asigna un short address (dirección corta) a los nuevos dispositivos y después revisa en el resto de los dispositivos para asegurarse que no está duplicada dicha dirección.

Cuando un dispositivo se une una red este recibe una dirección aleatoria la cual ha sido generada por su padre. El nuevo nodo de la red entonces genera un “Device Announce” (anuncio del dispositivo, el cual contiene tanto la MAC Address como el short address) en toda la red. Si existe otro dispositivo con el mismo short address, un nodo (rúter) emitirá un mensaje diciendo

que el estado de la red está en conflicto por una dirección “Network Status – Address Conflict” y entonces los dispositivos en conflicto cambiaran su short address. Cuando los dispositivos conflictivos han cambiado su short address emitirán de nuevo su “Device Announce” para verificar si aún existen conflictos en la red.

Los dispositivos finales no participan en el conflicto de direcciones (Addrees Conflict), sus padres lo hacen por ellos. En caso de que ocurra un conflicto en un dispositivo final los padres emitirán el mensaje “Rejoin Response” para cambiar el short address de los dispositivos finales y entonces los dispositivos finales emitirán un “Device Announce” y verificaran si existe conflicto en la red con su nuevo short address.

Cuando un “Device Announce” es recibido las tablas de asociación y enlace son actualizadas con el nuevo short address, la tabla de enrutamiento no es actualizada (se deben establecer las nuevas rutas). Si un padre determina que un “Device Announce” pertenece a alguno de sus dispositivos finales hijos, pero este no proviene directamente de su hijo, el padre asume que el dispositivo hijo se ha movido con otro padre.

6.2.2.2.2 Direccionamiento en la Pila (Stack)

Para enviar datos a un dispositivo en una red Zigbee la aplicación generalmente usa la función Zstackapi_AfDataReq(). Al dispositivo que se le enviaran los datos es del tipo zstack_AFAddr_t como se muestra en la Lista 1, este código es extraído del archivo zstack.h

Lista 1. Estructura para la definir una dirección

```
typedef struct _zstack_afaddr_t
{
    /** Modo de direccionamiento */
    zstack_AFAddrMode addrMode;
    /** Dirección, unión de 16 bits (dirección corta) and 64 bits (dirección IEEE) */
    union
    {
        /** dirección de red de 16 bits*/
```

```
        uint16_t shortAddr;
        /** dirección IEEE de 64 bits */
        zstack_LongAddr_t extAddr;
    } addr;
    /** Elemento de dirección punto final, opcional si se direcciona a un punto
final,
     * puede ser 0xFF para direccionar a todos los puntos finales del dispositivo.
     */
    uint8_t endpoint;
    /** PAN ID – para uso con Inter-PAN */
    uint16_t panID;
  } zstack_AFAddr_t;
```

Note que además de la dirección de red el parámetro address mode (modo de dirección) también necesita ser especificado. El address mode destino puede tomar uno de los siguientes valores que se muestran Lista 2 (los modos de direccionamiento AF son definidos en el archivo AF.h).

Lista 2 Configuración modo de direccionamiento

```
/** Modo de direccionamiento */
typedef enum
{
   // Dirección no presente!
    zstack_AFAddrMode_NONE = 0,
    //Direccionamiento Grupal (uint16_t)
    zstack_AFAddrMode_GROUP = 1,
    //Direccionamiento con la dirección corta (uint16_t)
    zstack_AFAddrMode_SHORT = 2,
    //Con la dirección extendida (8 bytes/64 bits)
    zstack_AFAddrMode_EXT = 3,
    //Direccionamiento Broadcast enviado a toda la red (uint16_t)
    zstack_AFAddrMode_BROADCAST = 15,
} zstack_AFAddrMode;
```

El parámetro address mode es necesario porque en Zigbee, los paquetes pueden ser unicast, multicast o broadcast. El paquete unicast es enviado a un solo dispositivo, el multicast es destinado a un grupo de dispositivos y el broadcast es generalmente enviado a toda la red

6.2.2.2.3 Unicast (unidifusión)

Es el modo de direccionamiento normal y es usado para enviar paquetes a un dispositivo el cual es conocida su dirección de red. El addrMode se establece en Addr16Bit y la dirección de la red destino es enviada en el paquete.

6.2.2.2.4 Indirecta

Esto es cuando la aplicación no tiene conocimiento del destino final del paquete el modo para establecerlo es con AddrNotPresent y la dirección destino no es especificada. En lugar de eso el destino es buscado en una tabla de enlace ("binding table") la cual se encuentra en el stack del dispositivo emisor. Esta característica se llama Source binding.

Cuando el paquete es enviado al stack la dirección destino (destination address) y el punto final (end point) es buscado en la tabla de enlace para su utilización. Y entonces es paquete es tratado como un paquete unicast normal. Si más de un dispositivo destino es encontrado en la tabla de enlace una copia es enviado a cada uno. Sino se encuentra en la tabla de enlace, el paquete no será enviado.

6.2.2.2.5 Broadcast

Este modo de direccionamiento es utilizado cuando la aplicación desea enviar un paquete a todos los dispositivos de la red. El modo de direccionamiento se establece con AddrBroadcast y la dirección destino se puede establecer con una de las siguientes direcciones broadcast.

NWK_BROADCAST_SHORTADDR_DEVALL (0xFFFF): el mensaje es enviado a todos los dispositivos de la red (incluyendo los dispositivos dormidos). Para los dispositivos dormidos el mensaje es tomado por su padre hasta que esté listo para tomar el mensaje o que envié mensaje

de tiempo de respuesta agotado (timed out) NWK_INDIRECT_MSG_TIMEOUT en f8wConfig.h.

NWK_BROADCAST_SHORTADDR_DEVRXON (0xFFFD): el mensaje será enviado a todos los dispositivos que estén activos (RXONWHENIDLE). Es decir, a todos, excepto los dormidos.

NWK_BROADCAST_SHORTADDR_DEVZCZR (0xFFFC): el mensaje será enviado a todos los rúters (incluyendo al coordinador).

6.2.2.2.6 Direccionamiento grupal (Group Addressing)

Este modo de direccionamiento es usado cuando las aplicaciones quieren enviar paquetes a un grupo de dispositivos. El modo de direccionamiento se establece con zstack_AFAddrMode_GROUP y el addr.shortAddr se fija como identificador del grupo. Antes de usar esta función, los grupos deben ser definidos en la red. Observe, que los grupos también se pude utilizar el direccionamiento indirecto. La dirección destino debe encontrar en la tabla de enlace (table binding) y puede ser tanto unicast o dirección grupal (group address).

Tenga en cuenta también que el direccionamiento broadcast es simplemente un caso especial de un direccionamiento grupal donde los grupos están configurados antes de tiempo. El código muestra un dispositivo que se ha agregado a un grupo con identificador, ver Lista 3

Lista 3 Asignación ID a un grupo.

```
#define GROUP_NAME "Group1"
zstack_apsAddGroup_t group;
group.endpoint = SAMPLEAPP_ENDPOINT;
/* Asignar el número de identificación 1 al grupo */
group.groupID = 0x0001;
/* El primer bit es la longitud de la cadena*/
```

```
group.n_name[0] = 6;
osal_memcpy( &(group.n_name[1]), GROUP_NAME, 6);
Zstackapi_ApsAddGroupReq(appEntity, &group);
```

6.3 Aplicación Sensor de Temperatura

6.3.1 Arquitectura de la aplicación

La Figura 10 muestra el diagrama de bloques de la aplicación sensor de temperatura.

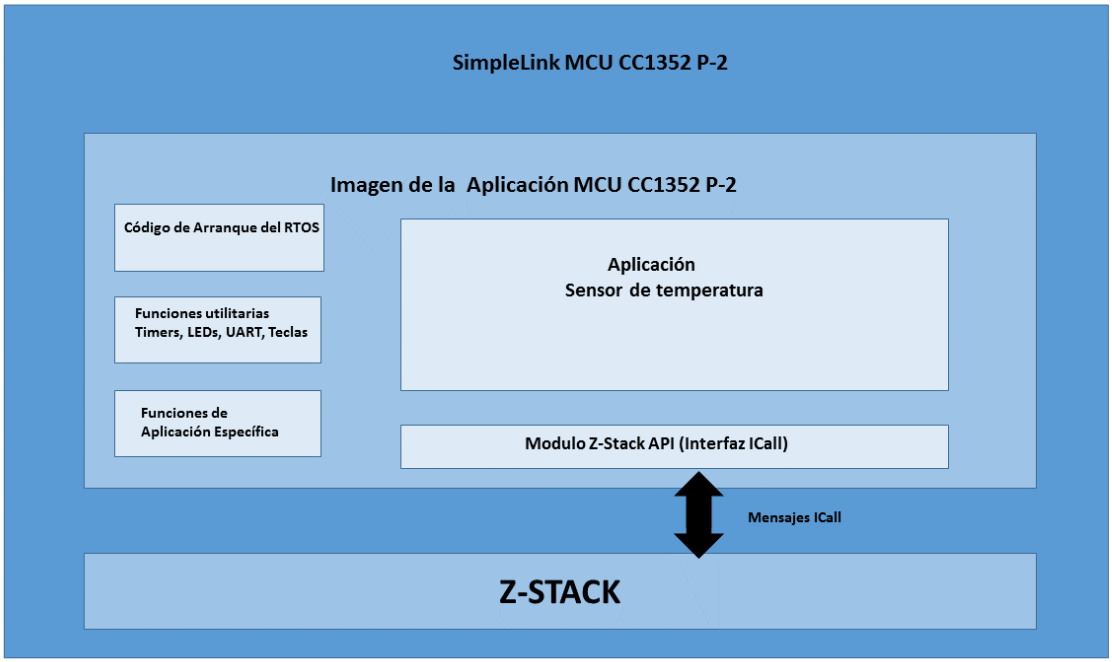


Figura 10. Diagrama de bloques de una aplicación

Las características generales de cada bloque se mencionan a continuación:

Código TI-RTOS Start-up: Inicializa la aplicación.

Aplicación: Se implementa en una plataforma independiente del ejemplo utilizado de acuerdo al caso. El desarrollador puede modificarlo y adecuarlo a sus necesidades y requerimientos.

Funciones utilitarias: Provee varias plataformas de utilidades las cuales se pueden utilizar en la aplicación. Incluye ejemplos como UART, LEDs, timers, botones, etc.

Funciones de Aplicación Específica: Implementa funciones de plataforma- especifica tales como almacenamiento de datos durante ciclo de encendido (no volátiles NV) y proporciona funciones de interfaz de usuario como manejo de accionamiento de botones o mostrar información esencial en UART.

Modulo Z-Stack API (API Z-Stack Module): Este módulo proporciona una interfaz para el almacenamiento y servicio de datos del Stack, lo realiza a través del módulo Indirect Call.

6.3.1.1 Framework (ICALL)

En la Figura 11 notamos que el proyecto lo podemos dividir en 2 bloques generales el primero es la imagen de la Aplicación y el segundo el Z-Stack, también observamos que la comunicación entre la aplicación ejemplo y el Z-Stack se realiza mediante el módulo ICALL.

Aquí notamos la razón de dividir el proyecto en carpetas, dependiendo de las necesidades de la aplicación particular podemos modificar estos archivos. Cabe aclarar una situación que no es evidente y es que los procesos realizados por el Z-Stack tienen prioridad sobre la Aplicación, esto se debe tener en cuenta al realizar el diseño, pues si no se sigue la estructura definida se pueden tener problemas en la ejecución del programa.

6.3.1.2 Módulo ICALL

ICALL es un módulo que proporciona un mecanismo interfaz entre la Aplicación y los Servicios del Stack (como las APIs del Stack), así como ciertos servicios primitivos (por ejemplo, la sincronización de procesos) proporcionados por el sistema operativo en tiempo real

RTOS. ICALL permite que la Aplicación y las tareas del protocolo del Stack, operen eficientemente, se comuniquen y compartan recursos en un ambiente unificado de RTOS.

El componente central de la arquitectura ICALL es el dispatcher (despachador) el cual proporciona un programa interfaz entre la Aplicación y las tareas del Stack. A pesar de que la mayoría de las interacciones del módulo ICALL dentro de la APIs del Stack son abstractas, es importante que el desarrollador entienda la arquitectura fundamental, con el fin de que tenga una operación adecuada el protocolo del Stack, en un entorno multiproceso RTOS. El código fuente del módulo ICALL se encuentra en el folder ICALL IDE, en la carpeta Aplicación.

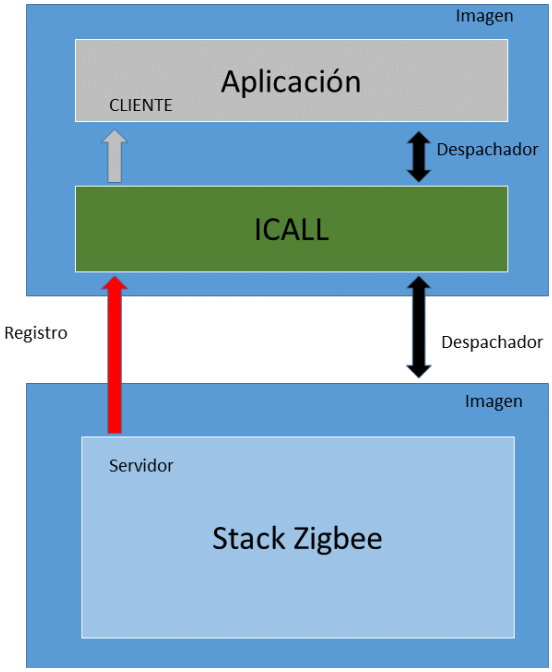


Figura 11 Aplicación ICALL.

6.3.1.3 Servicio del Stack Protocolo ICALL

Como se muestra en la Figura 11 ICALL se encarga de los mensajes involucrados entre una entidad servidor (las tareas del Z-Stack) y una entidad cliente (las tareas de la Aplicación). La razón de esta arquitectura es por dos motivos: para permitir la actualización independiente de la

Aplicación y Z-Stack, y para mantener una coherencia de la API a medida que el software se trasfiere a plataformas posteriores (por ejemplo, OSAL a CC253x) a el CC13x2 TI-RTOS. El Servicio ICALL del Z-Stack funciona como una interfaz de Aplicación para todas las APIs del Z-Stack. Internamente cuando una API del protocolo Z-Stack es llamada, el módulo ICALL enruta (despacha, envía) el comando al Z-Stack, y cuando es apropiado, enruta mensajes desde Z-Stack a la aplicación.

Al ser el módulo ICALL parte del proyecto de Aplicación. Las tareas de la Aplicación pueden acceder a ICALL con llamadas a función directas. No es recomendable modificar el código fuente de ICALL. Además, las tareas del Z-Stack se ejecutan con la máxima prioridad, por lo la tarea de Aplicación se bloquea hasta que se recibe la respuesta. Algunas APIs del protocolo del Stack pueden responder inmediatamente, sin embargo, los procesos de la Aplicación se bloquean por que la API está siendo despachada por el Z-Stack a través de ICALL. Otras APIs del Z-Stack (por ejemplo, actualizaciones de eventos) pueden responder de forma asíncrona a la Aplicación a través de ICALL, con la respuesta enviada a la tarea controlador de eventos de la Aplicación.

6.3.1.4 Servicio Primitivo ICALL

ICALL incluye un servicio primitivo el cual es una abstracción varias funciones relacionadas con el sistema operativo. Debido a que comparte recursos y mantiene comunicación entre procesos, la Aplicación debe utilizar las siguientes funciones de servicio primitivas ICALL.

- Mensajería y Sincronización de procesos.
- Asignación y Administración Heap.
- Mensajería y Sincronización procesos.

Las funciones de mensajería y sincronización de procesos proporcionadas por ICALL, permiten a los usuarios diseñar una aplicación interfaz del protocolo del Stack en un ambiente multiprocesos RTOS. Dentro de ICALL, los mensajes entre dos tareas se logran enviando un bloque de mensajes de un proceso a otro, utilizando una cola de mensajes (message queue). El emisor asigna memoria, escribe el contenido del mensaje en el bloque de memoria y después envía (pone en cola) el bloque de memoria al destinatario. La notificación de envió de mensaje es ejecutada utilizando una señal de semáforo. El receptor se levanta con el semáforo copia el bloque de memoria del mensaje (o bloques), procesa el mensaje y devuelve (libera) el bloque de memoria al heap.

El Stack utiliza a ICALL para notificar y enviar mensajes a la Aplicación. Este servicio de mensajes (por ejemplo, notificaciones de cambio de estado) es recibido por la tarea de la Aplicación sin enviados por ICALL y procesados en el contexto de la Aplicación.

Las funciones de mensajería y sincronización de subprocesos proporcionados por ICALL permiten a los usuarios diseñar una aplicación con la interfaz del protocolo del Stack en un entorno RTOS multiproceso. Dentro de ICALL la mensajería entre las dos tares se logra enviando un bloque de mensajes de un subproceso a otro utilizando una cola de mensajes (message queue). El emisor asigna memoria escribe el contenido del mensaje en un bloque de memoria y luego lo envía (poner en cola) al bloque de memoria del receptor. La notificación de mensaje enviado es acompañada con una señalización de semáforo. El receptor se activa por el semáforo, copia el mensaje en un bloque de memoria (o bloques), procesa el mensaje y lo devuelve (libera) al bloque de memoria para el heap.

El Stack utiliza el ICALL para notificar y enviar mensajes a la Aplicación. Este servicio de mensajes (como notificaciones de cambio de estado) es recibido por las tareas de la Aplicación y enviados por ICALL y procesados en el contexto de las tareas de la Aplicación.

6.3.1.5 Asignación y Administración Heap

ICALL proporciona la Aplicación con APIs heap globales para una asignación de memoria dinámica. El tamaño del heap ICALL es configurado con el preprocesador HEAPMGR_SIZE definido en la Aplicación del proyecto. ICALL utiliza este heap para todos los mensajes del protocolo del Stack, así como para obtener memoria para otros servicios ICALL. TI recomienda que la Aplicación utilice estas APIs ICALL para la asignación de memoria dinámica dentro de la Aplicación.

6.3.1.6 Sincronización de Procesos ICALL

EL módulo ICALL conmuta entre la Aplicación y los procesos del Stack a través de servicios de sincronización, utilizando el semáforo y la prioridad, los cuales son proporcionados por el RTOS. Las dos funciones de ICALL para recuperar y poner en cola mensajes no son funciones de bloqueo. Verifican si hay un mensaje recibido en la cola, y si no hay ningún mensaje, las funciones regresan inmediatamente con el valor ICALL_ERRNO_NOMSG.

Para permitir que un proceso de un cliente o un servidor se bloquee hasta recibir un mensaje, ICALL proporciona la siguiente función, Lista 4 , que se bloquea hasta que el semáforo asociado con el proceso que llama a RTOS es posteado.

Lista 4. Función de bloqueo para sincronizar cliente y servidor

```
    // Línea estática ICall_Errno ICall_wait(milisegundos)
ICall_Errno errno = ICall_wait(ICALL_TIMEOUT_FOREVER);
```

En la función anterior, milisegundos es el periodo de timeout, si después de este tiempo la función aún no se ha completado es retornado ICALL_ERRNO_TIMEOUT. Si se colca ICALL_TIMEOUT_FOREVER en lugar de milisegundos en ICall_wait() se bloqueara por siempre hasta que sea posteado el semáforo como se muestra en **¡Error! No se encuentra el origen de la referencia.**. Al permitir a una Aplicación o proceso del servidor se bloquee, se cede los recursos del procesador a otra actividad de menor prioridad o para conservar energía se apaga y se domina el reloj, tanto como sea posible.

El semáforo asociado con el proceso RTOS es activado también por las siguientes condiciones.

- Un nuevo mensaje en la cola de procesos RTOS de la aplicación.
- ICall_signal() llamado por el semáforo.
- Icall_signal() proporcionado para que una Aplicación o servidor puedan agregar sus propios eventos de desbloqueo ICall_wait() y sincronización de procesos.

Icall_signal() acepta el controlador del semáforo como su único argumento de la siguiente manera, Lista 5.

Lista 5. Activación con ICall_signal

```
//Estático en línea ICall_Errno ICall_signal(mensaje Icall_Semaphore);
ICall_signal(sem);
```

El control del semáforo asociado con el proceso se obtiene mediante la llamada ICall_enrollService() o con la llamada ICall_registerApp().

6.3.2 Descripción general del código de la aplicación

6.3.2.1 Inicio en main()

La función main() dentro del archivo main.c es el punto de inicio de la aplicación. Aquí es donde se activa la tarjeta con las interrupciones inhabilitadas y se inicializan los componentes. Las tareas en esta función se inicializan al establecer su prioridad, así como el tamaño del Stack de la aplicación. Como paso final, las interrupciones son habilitadas y *BIOS* es iniciado con la llamada a BIOS_start (), parte del código se muestra en la Lista 6

Lista 6. Función main()

```
void main()
{
    Task_Params taskParams;

#ifndef USE_DEFAULT_USER_CFG
    user0Cfg[0].pAssertFP = macHalAssertHandler;
#endif


    /* habilitar iCache predefinido */
    VIMSConfigure(VIMS_BASE, TRUE, TRUE);

#if defined(USE_CACHE_RAM)
    /* Disable cache */
      /* Deshabilitar cache */
    VIMSModeSet( VIMS_BASE, VIMS_MODE_DISABLED);
#else
    /* Enable cache */
    VIMSModeSet( VIMS_BASE, VIMS_MODE_ENABLED);
#endif


/*  Inicializar los dispositivos de la tarjeta como LEDS y botones
siguiendo la convención TI RTOS  */
#if defined(CC13X2R1_LAUNCHXL) && !defined(NO_CC1312R1_SUPPORT) &&
!defined(NO_CC1352P1_SUPPORT)
  CC13X2R1_LAUNCHXL_Pin_init();
#else
  PIN_init(BoardGpioInitTable);
#endif

    /* Configuración de tareas. */

    Task_Params_init(&taskParams); // Inicialización parámetros de tareas
    taskParams.stack = myTaskStack;
```

```
    taskParams.stackSize = APP_TASK_STACK_SIZE;          // Tamaño del stack
    taskParams.priority = 1;                                //Prioridad
    Task_construct(&myTask, taskFxn, &taskParams, NULL); // Constructor
                                                            //de tareas

#ifdef DEBUG_SW_TRACE
    IOCPortConfigureSet(IOID_8, IOC_PORT_RFC_TRC, IOC_STD_OUTPUT
                      | IOC_CURRENT_4MA | IOC_SLEW_ENABLE);
#endif /* DEBUG_SW_TRACE */

    BIOS_start(); /* Arranque del Sistema operativo SYS/BIOS */
}
```

6.3.2.2 Inicialización y Registro de ICALL

Para inicializar el servicio ICALL, la aplicación debe llamar a las siguientes funciones en main.c, antes de iniciar SYS/BIOS (Sistema Operativo en Tiempo Real). Llamar a ICall_init(); que inicializa el servicio primitivo de ICALL (por ejemplo, administración del heap) y el framework. Y llamar a ICall_createRemoteTasks(), que crea pero no inicializa las tareas del protocolo Z-Stack, esto lo realiza en la función llamada taskFxn () como se muestra el código en Lista 7. La dirección primaria IEEE (programada por TI) es obtenida del área CCFG de la memoria Flash y los drivers NV son inicializados, las tareas de la aplicación son inicializas y arrancan.

Lista 7. Función taskFxn

```
Void taskFxn(UArg a0, UArg a1)
{

    /*
    No permite apagar JTAG, VIMS, SYSBUS durante el estado inactivo
     ya que TIMAC requiere SYSBUS durante la inactividad
     */
    Power_setConstraint(PowerCC26XX_IDLE_PD_DISALLOW);

#if defined(USE_CACHE_RAM)
    /* Retiene la memoria RAM Cache */
    Power_setConstraint(PowerCC26XX_SB_VIMS_CACHE_RETAIN);
#endif

    /* Inicialización de modulo Icall */
    ICall_init();
```

```
    /*
     * Copia la dirección extendida del area de memoria CCFG
     * Asumiendo que: la memoria en CCFG_IEEE_MAC_0 y CCFG_IEEE_MAC_1
     Contiguas contiguo y primero el bit LSB.
     */

    memcpy(zstack_user0Cfg.extendedAddress, (uint8_t *)&(__ccfg.CCFG_IEEE_MAC_0),
           (APIMAC_SADDR_EXT_LEN));

    /* Verifica si la CCFG IEEE es valida */
    if(memcmp(zstack_user0Cfg.extendedAddress, dummyExtAddr, APIMAC_SADDR_EXT_LEN) ==
0)
    {
        /* Si no es valida. Se obtiene la dirección primaria IEEE */
        memcpy(zstack_user0Cfg.extendedAddress, (uint8_t *)(FCFG1_BASE +
EXTADDR_OFFSET),
                      (APIMAC_SADDR_EXT_LEN));
    }


#ifdef NV_RESTORE
    /* Setup the NV driver */
#ifdef ONE_PAGE_NV
    NVOCOP_loadApiPtrs(&zstack_user0Cfg.nvFps);
#else
    NVOCTP_loadApiPtrs(&zstack_user0Cfg.nvFps);
#endif

    if(zstack_user0Cfg.nvFps.initNV)
    {
        zstack_user0Cfg.nvFps.initNV( NULL);
    }
#endif


    /* Inicia las tareas de Imágenes externas */
    ICall_createRemoteTasks();

    /* Arranca la Aplicación */
    zclSampleTemperatureSensor_task(&zstack_user0Cfg.nvFps); // Inicia las tareas de
la aplicación
}
```

Antes de utilizar los servicios del protocolo ICALL, tanto el servidor como el cliente deben registrarse con ICALL. El servidor inscribe un servicio cuando asigna un ID al momento de construirlo. La función control de registro utiliza un identificador único para cada servicio.

El mecanismo de registro es utilizado por el cliente para enviar y recibir mensajes a través del despachador ICALL. Para que un cliente (por ejemplo, alguna tarea de la Aplicación) utilice las

APIs del Z-Stack, el cliente debe primero registrar la tarea con ICALL. Este registro es hecho por la Aplicación en la función zclSampleTemperatureSensor_initialization() el cual es llamado por las funciones de inicialización de la aplicación.

En la Lista 8 se muestra una llamada de ICALL en la función zclSampleTemperatureSensor_initialization() en el archivo zcl_sampletemperaturesensor.c

Lista 8. Función de inicialización

```
static void zclSampleTemperatureSensor_initialization(void)
{
    uint8 key;

    /* Inicializa los relojes utilizados */
    zclSampleTemperatureSensor_initializeClocks();

    /* Inicializa los botones */
    key = Board_Key_initialize(zclSampleTemperatureSensor_changeKeyCallback);


    /* Inicializa los LEDS */
    Board_Led_initialize();

    // Registra el proceso actual como una aplicación del despachador ICall
    // entonces la aplicación puede enviar y recibir mensajes
    ICall_registerApp(&zclSampleTemperatureSensor_Entity, &sem);

    if(key == KEY_RIGHT)
    {
        Zstackapi_bdbResetLocalActionReq(zclSampleTemperatureSensor_Entity);
    }


    //Initialize stack
    zclSampleTemperatureSensor_Init();
}
```

El archivo zcl_sampleTemperatureSensor.c proporciona las entradas zclSampleTemperatureSensor_Entity y sem, donde hasta que regrese ICall_registerApp(), son inicializadas por las tareas del cliente (por ejemplo la Aplicación). Estos objetos son después utilizados por ICALL que facilita los mensajes entre la Aplicación y las tareas del servidor. El argumento sem representa el semáforo utilizado para señalización, mientras que el

zclSampleTemperatureSensor_Entity representa la cola de mensajes de la tarea destino. Cada registro de tarea con ICALL tiene un único sem (semaforo) y un identificador zclSampleTemperatureSensor_Entity.

6.3.2.3 Procesamiento de Eventos en la Función Task

Después de inicializar los periféricos, configurar la aplicación e implementar la inicialización de la función, del código anterior tenemos la función zclSampleTemperatureSensor_task(), la cual entra a un lazo infinito, con el propósito de procesar continuamente una tarea independiente y no parar hasta que finalice, Lista 9.

Lista 9. Función Task

```
void zclSampleTemperatureSensor_task(NVINTF_nvFuncts_t *pfnNV)

{

  // Registra y guarda los punteros de las funciones del controlador NV (memoria no
volatil)
  pfnZdlNV = pfnNV;
  zclport_registerNV(pfnZdlNV, ZCL_PORT_SCENE_TABLE_NV_ID);


// Inicializa la aplicación
  zclSampleTemperatureSensor_initialization();

  // No regresa del proceso de las tareas
  zclSampleTemperatureSensor_process_loop(); // Entra a un lazo infinito

}
```

Para tener una mejor visualización del proceso tenemos el diagrama de flujo de la Figura 12 observamos que inicia en la función task (tareas), enseguida siguen las funciones de inicialización y la del proceso loop (ciclo infinito). Una vez entrando a la función loop se verifica si hay mensajes del Stack, si es cierto, los atiende, de no serlo continúa revisando los mensajes de la aplicación, una vez atendidos regresa a revisar los mensajes del Stack, este ciclo se mantendrá de forma infinita en la Aplicación.

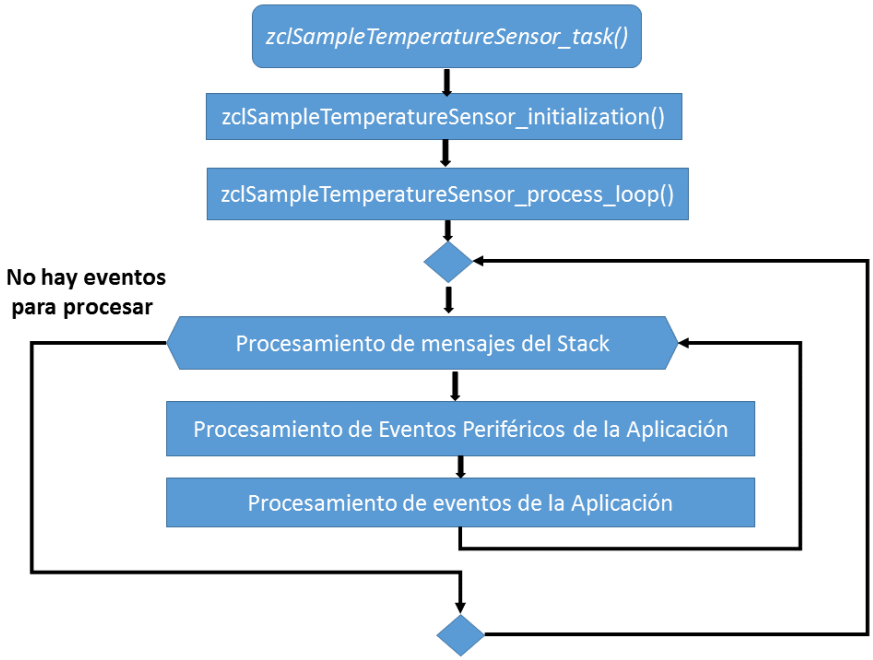


Figura 12 Diagrama de flujo de la Aplicación

6.3.2.4 Señalización de Eventos a Través de la Variable de Evento Interna

La tarea Aplicación utiliza mascara de bits como variable de evento para identificar la acción ha causado que el proceso se active, y realizar así una acción apropiada, en el archivo zcl_sapletemperaturesensor.h se definen los eventos. Cada bit de la variable evento se define como se muestra en la Lista 10

Lista 10 Señalización de eventos

```
// Eventos de la Aplicación
// Evento envía la temperatura del sensor
#define SAMPLETEMPERATURESENSOR_TEMP_SEND_EVT      0x0001
// Eventos re-unión de dispositivos finales
#define SAMPLEAPP_END_DEVICE_REJOIN_EVT            0x0002
#define SAMPLEAPP_KEY_EVT                          0x4000

// Eventos UI (Interfaz de usuario
#define SAMPLEAPP_UI_AUTO_REFRESH_EVT              0x0010
#define SAMPLEAPP_UI_INPUT_EVT                     0x0040

// Green Power Events
#define SAMPLEAPP_PROCESS_GP_DATA_SEND_EVT          0x0100
#define SAMPLEAPP_PROCESS_GP_EXPIRE_DUPLICATE_EVT      0x0200
```

```
#define SAMPLEAPP_PROCESS_GP_TEMP_MASTER_EVT           0x0400

#define SAMPLEAPP_END_DEVICE_REJOIN_DELAY 10000
```

Cualquier función que utilicé esa variable evento, también debe asegurarse de postearlo al semáforo, para activar la aplicación para su procesamiento. Un ejemplo de esto es el control de reloj para timeouts, como se muestra en Lista 11.

Lista 11. Posteo de un evento

```
static void zclSampleTemperatureSensor_processEndDeviceRejoinTimeoutCallback(UArg a0)
{
    (void)a0; // Parámetro no utilizado

    events |= SAMPLEAPP_END_DEVICE_REJOIN_EVT;

    // Activa un proceso de la aplicación cuando está esperando por el evento del
reloj              Semaphore_post(sem);
  }
```

Que después de tomado el evento, se limpia el bit que se ha activado a medida que se procesa el evento, se muestra en la Lista 12.

Lista 12. Limpiar evento

```
  if ( events & SAMPLEAPP_END_DEVICE_REJOIN_EVT )    // Se  activa la bandera
            {
              zstack_bdbZedAttemptRecoverNwkRsp_t zstack_bdbZedAttemptRecoverNwkRsp;


Zstackapi_bdbZedAttemptRecoverNwkReq(zclSampleTemperatureSensor_Entity,&zstack_bdbZed
AttemptRecoverNwkRsp);

              events &= ~SAMPLEAPP_END_DEVICE_REJOIN_EVT;    // Se limpia el evento
               }
```

Cuando se agrega un evento, este debe ser único para una tarea y tener una potencia de 2 (de modo que solo se establezca un bit). Debido a que la variable evento se ha inicializado como uint16_t, esta configuración permite un máximo de 16 eventos internos.

6.4 Procesador de Red Zigbee (ZNP)

6.4.1 Interfaz de Comandos ZNP

Los comandos de la interfaz están divididos en las siguientes categorías:

- SYS (MT_SYS): proporciona al procesador de aplicación una interfaz de bajo nivel para el hardware y software del ZNP.
- AF (MT_AF) y ZDO (MT_ZDO): interfaces con características completas de Zigbee, que se utilizan para crear una gran cantidad de aplicaciones compatibles con Zigbee. La interfaz AF (Application Framework) permite al procesador de aplicación registrar la aplicación con el ZNP para poder enviar y recibir datos. La interfaz ZDO nos proporciona varias funciones de administración de dispositivos y servicios de descubrimiento.
- UTIL (MT_UTIL): nos provee varias funcionalidades de soporte configuración de ID PAN, obtener información de un dispositivo, obtener información de NV, subscribir callbacks etc.
- APP CONF (MT_APP_CNF): esta interfaz nos proporciona soporte para la funcionalidad BDB, como por ejemplo configurar, códigos de instalación, canal primario y secundario, activación de diferentes métodos Commissioning y otras configuraciones de Trust Center.

Para enviar los comandos utilizaremos la herramienta de desarrollo Z-Tool de Texas Instruments, Figura 13, obsérvese que en la esquina superior izquierda tenemos la interfaz de comandos mencionados anteriormente, este software nos facilitará configurar al ZNP para que funcione como un coordinador de red y realice las tareas correspondientes.

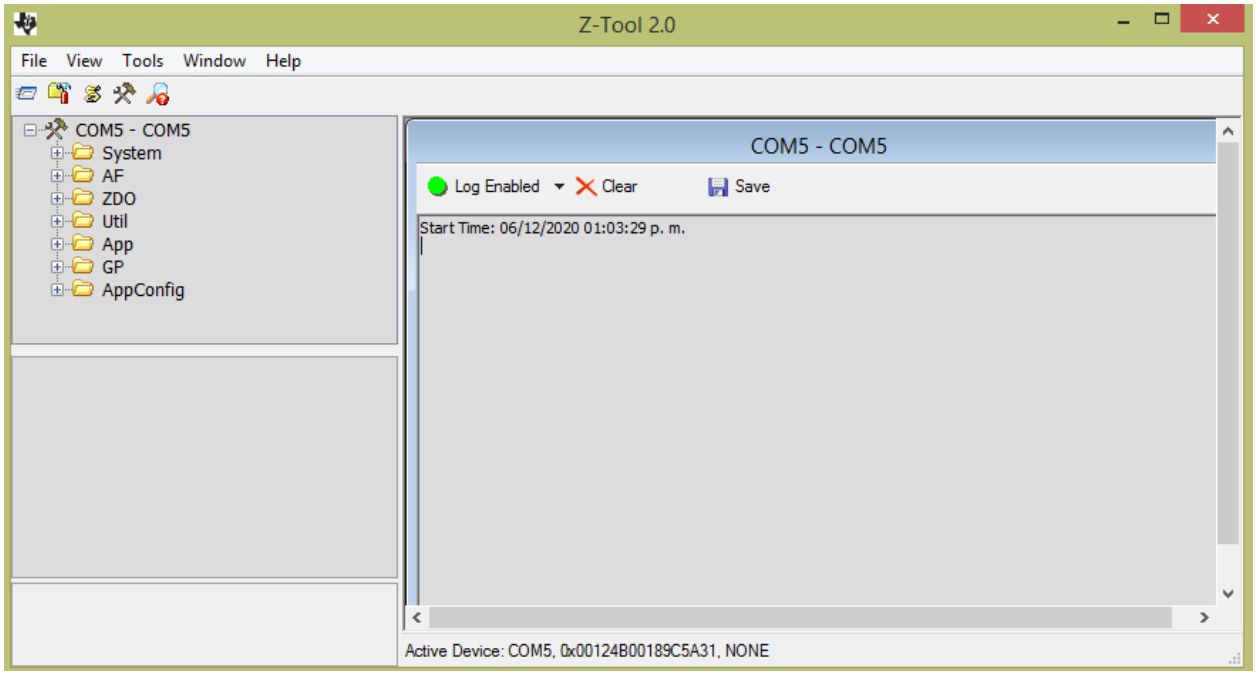


Figura 13. Herramienta Z- Tool 2.0

6.4.2 Comunicación Serial

Como se mencionó anteriormente la comunicación entre la aplicación ZNP con la computadora personal se realiza mediante tramas. La interfaz de comunicación se utiliza en puerto serial UART, con las siguientes características

- baud rate: 115200.
- Formato de datos 8bits.
- Sin control de flujo (predeterminado), pero se puede activar.

6.4.2.1 Formato de las tramas

Él envió de datos a través del puerto serial se realiza a través de tramas por lo que a continuación se detalla la estructura que tienen. En la Tabla 2 se muestra en el formato de

tramas del transporte UART, primero se trasmite el elemento que está a la izquierda (SOF) y así sucesivamente con los demás.

Tabla 2 Formato de la trama del transporte UART.

	SOF	**Formato General de Trama**	**FCS**
bytes	1	3 - 253	1

de la **¡Error! No se encuentra el origen de la referencia.** tenemos que:

- **SOF (Start of Frame):** Indica el inicio de la trama, se activa con el valor 0xFE
- **FCS (Frame Check Sequence)** Secuencia de Verificación de la Trama, se calcula con una función XOR a todos los bytes del Formato General de Trama
- **FORMATO GENERAL DE TRAMA** Es la estructura que tiene la trama de datos a enviar, se explica en el siguiente punto.

6.4.2.1.1 Formato General de Trama

El Formato General de Trama se muestra en la Tabla 3, el primer elemento en transmitir es el que está a la izquierda, en este caso particular es la longitud del dato.

Tabla 3. Formato General de Trama

	Longitud	**Comando**	**Dato**
Bytes	1	2	0 - 250

Los campos que tenemos son:

- **Longitud:** es el tamaño de dato a transmitir, está en un rango de 0 a 250

- **Comando:** Indica el ID del comando que lleva el mensaje (se explica en la siguiente sección).
- **Dato:** Es la trama que contiene el dato enviado. El tamaño puede ser de 0 a 250 bytes

6.4.2.1.2 Estructura del Campo Comando

Se compone de 2 bytes, tiene el formato que se muestra en Tabla 4. De forma similar que las tablas anteriores, primero se transmite el elemento a la izquierda, Cmd0 y a continuación Cmd1

Tabla 4. Campo de comando.

	Cmd0		**Cmd1**
Bit	7-5	4 - 0	7-0
	Tipo	Subsistema	ID

En cuanto al Tipo, se encuentra definido con los bits del 7 al 5 y estos valores son de acuerdo a la Tabla 5.

Tabla 5. Tipo de comando

Tipo	**Valor de Cmd0**
POLL	0x00
SREQ	0x20
AREQ	0x40
SRSP	0x60

donde:

- POLL: No se utiliza en el Z-Stack
- SREQ: Es una solicitud síncrona que requiere una respuesta inmediata.
- AREQ: Solicitud asíncrona

- SRSP: Respuesta síncrona
- 4 -7 Reservado

El subsistema por su parte tiene los siguientes valores, Tabla 6.

Tabla 6. Valor del subsistema.

Valor del subsistema	Nombre del subsistema
0x00	RPC Error de la interfaz
0x01	Interfaz SYS
0x02	Interfaz MAC
0x03	Interfaz NWK
0x04	Interfaz AF
0x05	Interfaz ZDO
0x06	Interfaz Simple API
0x07	Interfaz UTIL
0x08	Interfaz Debug
0x09	Interfaz APP
0x0F	Config APP
0x15	Green Power

En cuanto a ID del comando es un numero identificador de 8 bits, el cual se asigna a la interfaz especificada en el campo subsistema (en el campo Cmd0). De aquí se concluye que, cada subsistema de MT puede tener hasta 256 funciones de control de mensajes.

7 Resultados

7.1 Implementación del sistema experimental

En la Figura 14 se muestra la interconexión del sistema, en la derecha tenemos la PC que trabaja como Host, en ella se ejecutará la herramienta de ayuda Z-Tool, este software tiene funcionalidades que permiten realizar tareas de un coordinador de red, está conectada a una tarjeta CC1352 P-2 a través del serial, esta cumplirá la función de coordinador ZNP.

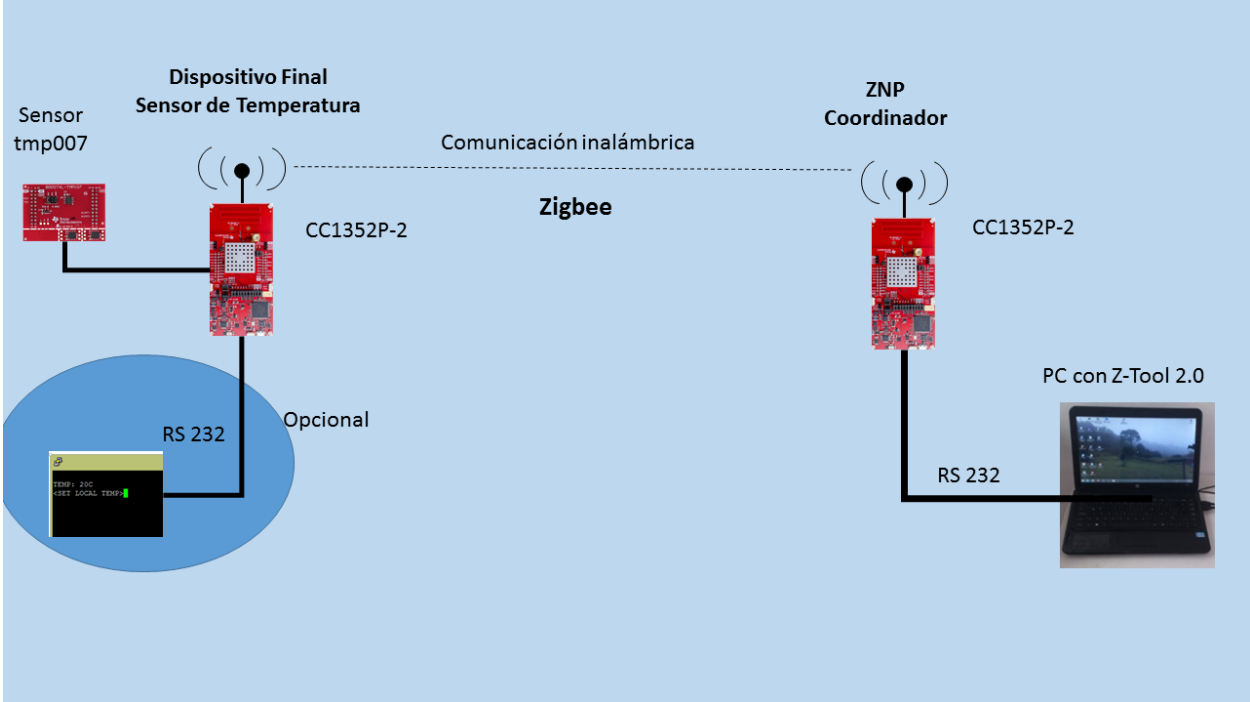


Figura 14. Esquema de red Zigbee simple para prueba.

Del lado izquierdo tenemos otra tarjeta CC1352 P-2, este es el dispositivo final, además de la tarjeta cuanta con un módulo tmp 007 que es el sensor de temperatura. Para fines de visualización se ha utilizado el software Putty, que corre en una PC externa y se comunica por el puerto serial, se puede omitir el display, sin que altere significativamente el sistema. La comunicación entre ambos kits de desarrollo es inalámbrica, utilizando el protocolo Zigbee.

7.1.1 Plataforma de Desarrollo Kit CC1352-P2.

La tarjeta de desarrollo CC1352-P2 MCU multi-banda inalámbrico CC1352 P-2, Figura 23 es parte de la plataforma de microcontroladores SimpleLink™ la cual consiste en tecnologías Wi-Fi®, Bluetooth® Low Energy, Sub-1 GHz, Thread, Zigbee®, 802.15.4 y microcontroladores host, que tienen características en comunes como: fácil ambiente de desarrollo, el mismo kit software de desarrollo SDK así como un amplio conjunto de herramientas. Con una plataforma única de integración SimpleLink le permite agregar cualquier combinación de los dispositivos al diseño, este permite poder reutilizar y código cuando se requieran hacer cambios en el diseño. Este kit de desarrollo soporta programación y depuración (debug) de Code Composer Studio™ y de IAR Embedded Workbench® como Entono Integrados de Desarrollo (IDEs).

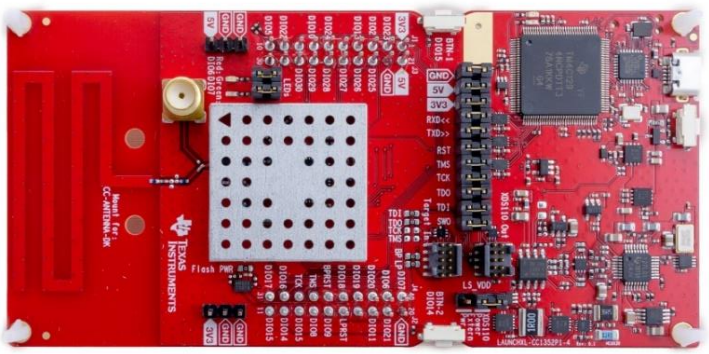


Figura 23. Kit de Desarrollo CC1352 P-2

7.1.1.1 Arquitectura.

La plataforma de desarrollo CC1352 –P2 es un MCU inalámbrico de bajo consumo energético que se emplea para una amplia gama de aplicaciones, en un sistema embebido (system-on-chips SoCs). La combinación de un procesador Arm® Cortex®-M4F a 48 MHz, memoria flash y una amplia selección de periféricos hace que la plataforma sea especialmente implementada en redes de comunicación RF de baja potencia (TI Technical Reference Manual CC13x2, 2019)

7.1.1.2 Componentes.

La Figura 24 Figura 24muestra el diagrama de bloques del dispositivo, que tiene las siguientes características

- Procesador Arm® Cortex®-M4F.
  - Oscilador RC de 48 MHz y oscilador de cristal de 48 MHz.
  - Oscilador de cristal de 32 kHz, oscilador RC de 32 kHz para modos de bajo consumo energético.
  - Arm® Cortex® SysTick timer.
- Memoria.
  - Memoria Flash de 8 KB con 4 formas de configurar-asociar con el cache de la RAM para variar la velocidad y consumo energético.
  - Sistema RAM con retención configurable en bloques se 16 KB.
- Administrador de energía.
  - Amplio rango en voltaje de alimentación.
  - Convertidor eficiente DC-DC para reducir el consumo de energía
  - Flexibilidad en la frecuencia de operación, lo que permite un bajo consumo de energía.
- Interfaz de sensor.
  - Interfaz de sensor inteligente y autónomo que puede activarse independientemente de la CPU del sistema.
  - Convertidor analógico-digital de 12 bits con 8 canales digitales de entrada.
  - Comparador análogo de bajo consumo.
  - Interface de comunicación serial.

- Integración serial avanzada
    - Trasmisor/ Receptor Asíncrono Universal (UART)
    - Módulo I2C
    - Módulos de interfaz serie síncronos (SSIs)
    - Módulo Interfaz de Audio I2 S
- Sistemas integrados.
    - Controlador de acceso directo a memoria (DMA).
    - Cuatro temporizadores de 32 bits (hasta ocho de 16 bits) con capacidad de modulación de ancho de pulso (PWM) y sincronización.
    - Reloj de tiempo real de 32 kHz (RTC).
    - Watchdog timer.
    - Sensor de temperatura y voltaje.

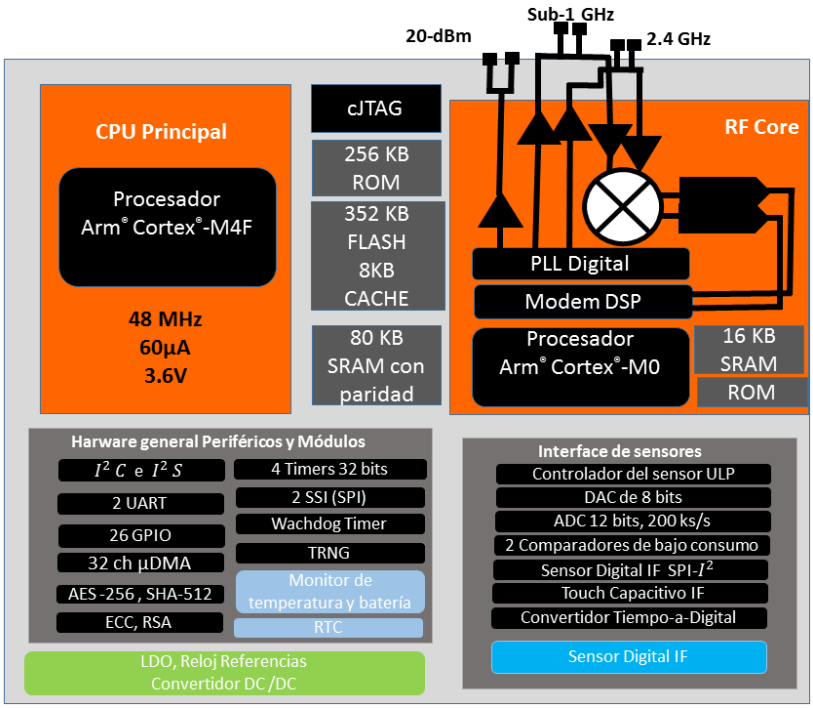


Figura 24. Diagrama de bloques del CC1352P-2.

7.1.2 Sensor de Temperatura

Para verificar el funcionamiento del sensor de temperatura de forma local podemos usar un software auxiliar de puerto serie, Putty, como se mencionó anteriormente es solo con fines de visualización. Para esto se ejecuta el programa Putty, nos aparece una ventana como se muestra en la Figura 15 a). Nos pide el puerto donde se encuentra conectado el dispositivo, revisamos en que puerto se encuentra conectado el sensor de temperatura (se recomienda ver en el administrador de dispositivos de Windows), una vez identificado se realiza la configuración, tasa de transmisión 1152000 baudios, 8 bits, sin control de flujo, se muestra en la Figura 15 a)

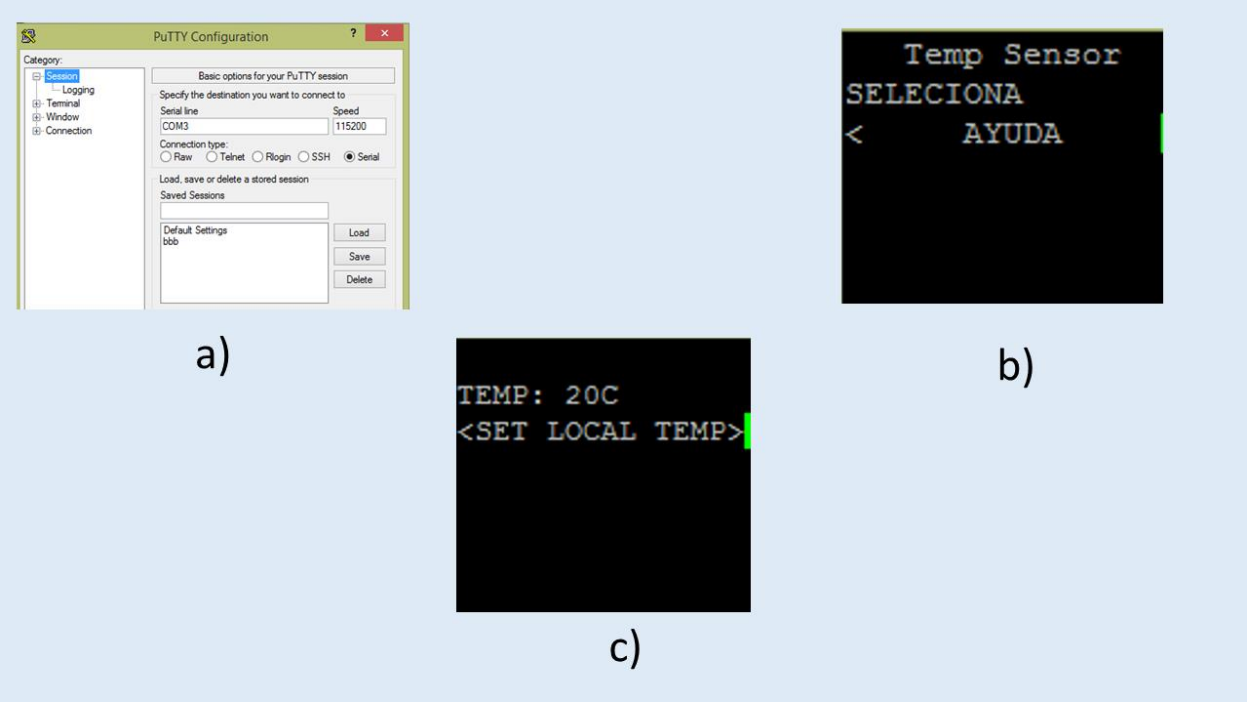


Figura 15. Display sensor de temperatura a) Configuración, b) Menú principal c) Medición temperatura local

Presionamos la tecla enter, a continuación, entramos al menú de la aplicación, Figura 15 b), del sensor de temperatura, se observa un menú cíclico, que nos permite interactuar directamente

con el nodo sensor, buscamos temperatura y encontramos la medición de la temperatura local, Figura 15 c), se observa una medición de temperatura de 20 °C. La utilidad del menú es verificar que la información local obtenida es igual a la que se tiene en el coordinador.

7.1.3 Coordinador ZNP

Para trabajar con el coordinador se tiene que realizar una configuración previa al ZNP. Recordemos que el procesador de red Zigbee (ZNP) es un nodo con características generales de un dispositivo Zigbee, es utilizado para realizar pruebas a una red, por lo que podría ser utilizado como rúter o un dispositivo final.

Se inicia el proceso con un reinicio del ZNP, como se observa en la Figura 16. El procesador Host envía una requisición síncrona con el comando 0x4100, que es una solicitud para reiniciar al ZNP. El rectángulo rojo en posición vertical que se ve al final de la flecha indica que el procesador Host está esperando una respuesta a la solicitud enviada. Por lo que el ZNP debe enviar un estatus a la requisición solicitada.

Después de e reinicio la NV debe tener valores de configuración, como definir el tipo lógico de dispositivo que es, en este caso particular el valor es 0x00 por default, lo que significa que es un coordinador. Si se desea cambiar los valores para el enrutador y el dispositivo final son 0x01 y 0x02 respectivamente. Se realiza una petición de reinicio nuevamente como se muestra en la Figura 16.

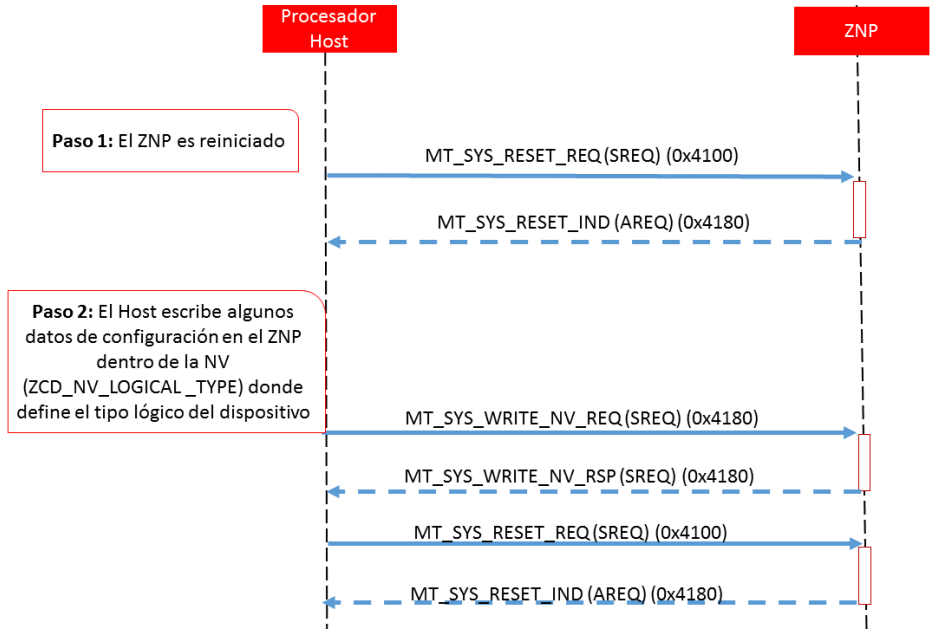


Figura 16. Configuración ZNP como coordinador, paso 1 y 2

El tercer paso de nuestro proceso es el registro del punto final (también conocido como end point), para esto se toman los valores de la Lista 13 y se utiliza el comando AF_REGISTER.

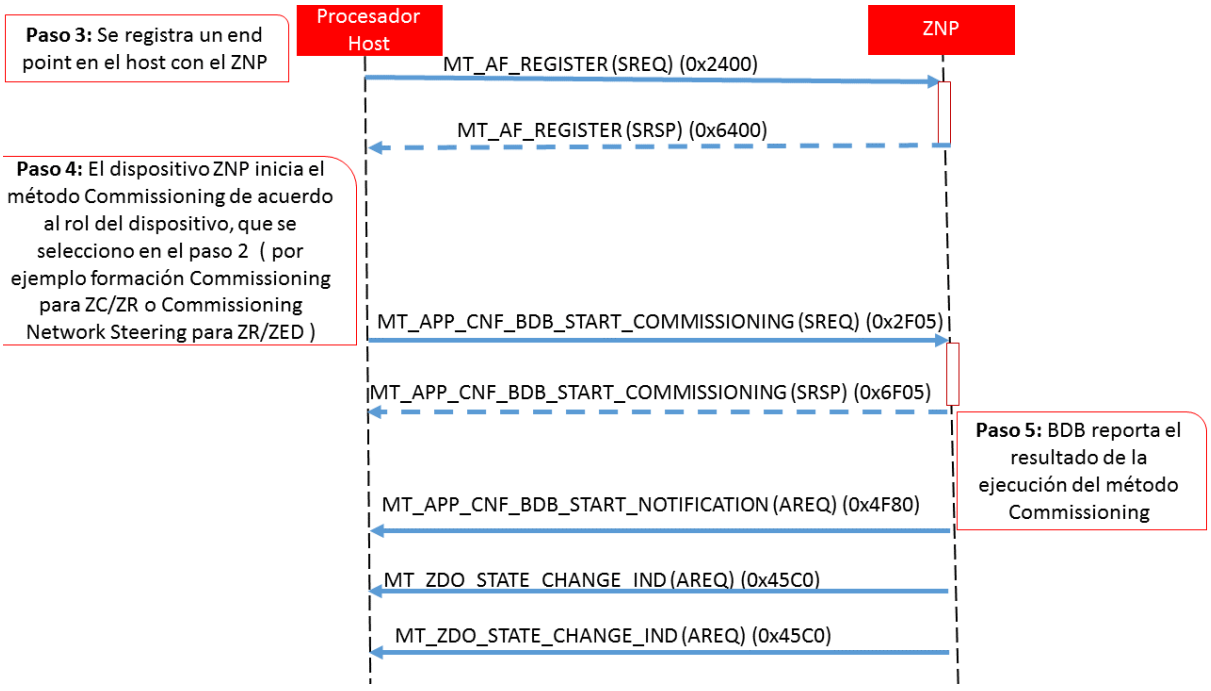


Figura 17. Configuración ZNP como coordinador, paso 3,4 y5

A continuación de esto se inicia el proceso Commissioning, seleccionando formación de red, el Host deberá responder a la solicitud. Así como notificara el estatus del resultado del método Commissioning con el comando 0x4F8. El ZNP indicara al Host los cambios de estado del dispositivo al ejecutarse el proceso Commissioning, como se muestra en la Figura 17.

Los valores con los que se ha realizado la configuración, que se muestran a continuación, Lista 13.

Lista 13. Valores AF_REGISTER

```
EndPoint:0x08
AppProfID:0x104
AppDeviceId:0x301
AppDevVer:0x00
LatencyReq:NO_LATENCY_REQ(0x0)
AppNumInClusters:0x03
AppClusterList:0x0000, 0x0003, 0x0201
AppNumOutClusters:0x02
AppOutClusterList:0x0003, 0x0402
```

Una vez creada la red los dispositivos disponibles comenzaran a enviar solicitud de unión a el coordinador, por lo que el ZNP le notificara al Host, eso lo vemos en el paso 6, Figura 18. Con una red creada y con dispositivos enlazados se realizará un intercambio de mensajes entre el Host y el ZNP, como se muestra en el paso 7.

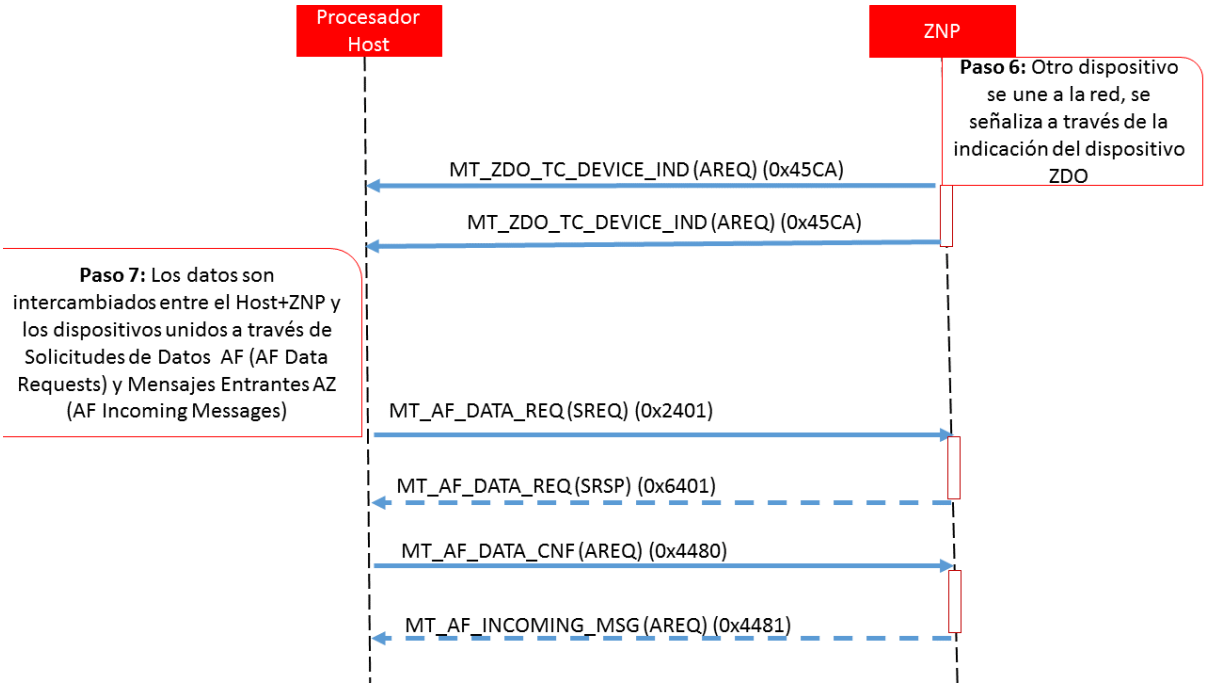


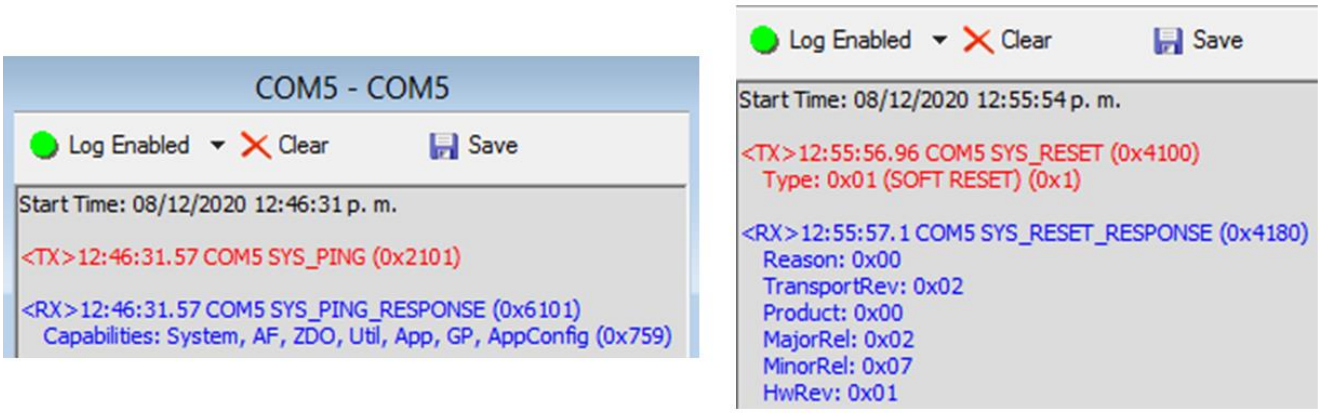
Figura 18. Configuración ZNP como coordinador, paso 6 y 7

Con el comando AF_DATA_REQUEST podemos solicitar información específica de un dispositivo con el comando 0x2401, al ser el ZNP coordinador de red podrá solicitar información a elementos unidos. A través de los mensajes entrantes realizados por el comando 0x4481 el ZNP le enviara al host la información solicitada.

Para el proceso de convertir al ZNP como coordinador de red, nos apoyaremos con la herramienta Z-Tool, Figura 13, que nos provee Texas Instruments, como se explicó en sección Interfaz de Comandos ZNP. La interacción entre el procesador Host (PC) y el ZNP se realiza a través de requisiciones (solicitudes) y respuestas, a continuación, muestra el proceso para inicializar el coordinador y que pueda administrar la red.

Se ejecuta el programa Z-Tool, nos aparecerá una ventana de menú buscamos la pestaña Tools, enseguida nos dirigimos a Settings y habilitamos el puerto COM donde se encuentra el

ZNP (revisar en que puerto se encuentra conectado el ZNP). A continuación, ejecutamos Scan for Devices, y si es exitoso el proceso, nos enviara la siguiente ventana Figura 19 Figura 19a)



a) b)

Figura 19. Herramienta Z-Tool a) búsqueda de dispositivo ZNP b) reinicio.

Observe que TX (Transmisión) en color rojo corresponde a comandos enviados por el Host y en color azul es la respuesta del ZNP, al realizar la búsqueda de dispositivos se ha enviado un PING al ZNP, este responde y muestra las capacidades de comandos que tiene.

En la Figura 19 Figura b), se muestra la solicitud de reinicio al ZNP con el ID de comando 0x4100. El ZNP se reinicia y responde con el comando 0x4180.

A continuación, se realiza el paso 3, que es el registro del end point. Para esto se utilizará el comando AF REGISTER, que tiene el ID 0x2400, recordemos que se configura con los valores de la Lista 13, en la Figura 20 a) se muestra el proceso y si se realiza de forma exitosa nos regresa un estatus 0x00.

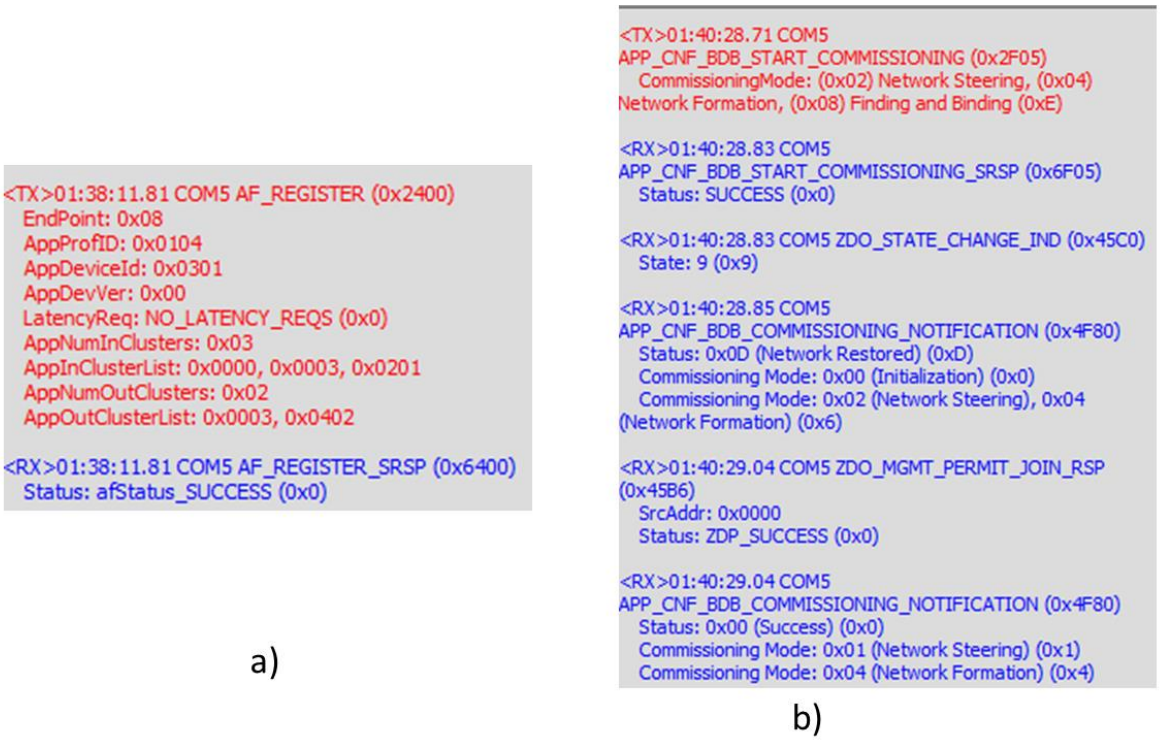


Figura 20. Herramienta Z-Tool, a) registro de aplicación b) enlace de red.

Ya hemos configurado los valores del coordinador, ahora es necesario unir los dispositivos, para esto utilizaremos el método Commissioning. En la Figura 20 b) se muestra el proceso, enviamos el comando 0x2F05 al ZNP, este nos responde que se ha realizado de forma exitosa y nos indica el cambio de estado del ZDO.

A continuación, se presiona el botón 1 de la tarjeta sensor de temperatura y así como observamos que se ha permitido en enlace de un dispositivo (comando 0x4586), que en este caso particular es el nodo sensor de temperatura.

7.2 Resultados, operación de la res de sensores.

Una vez logrado la creación de la red, enlace de los dispositivos en la red y la unión del nodo sensor a la red; la aplicación del sensor (dispositivo final en la red) reportará periódicamente el

dato de temperatura actual de la temperatura al ZNP (coordinador en la red), el cual estará los recibe y retrasmite vía serial hacia al Host, como se muestra en la Figura 21.Figura 21Figura 21Figura 21Figura 21

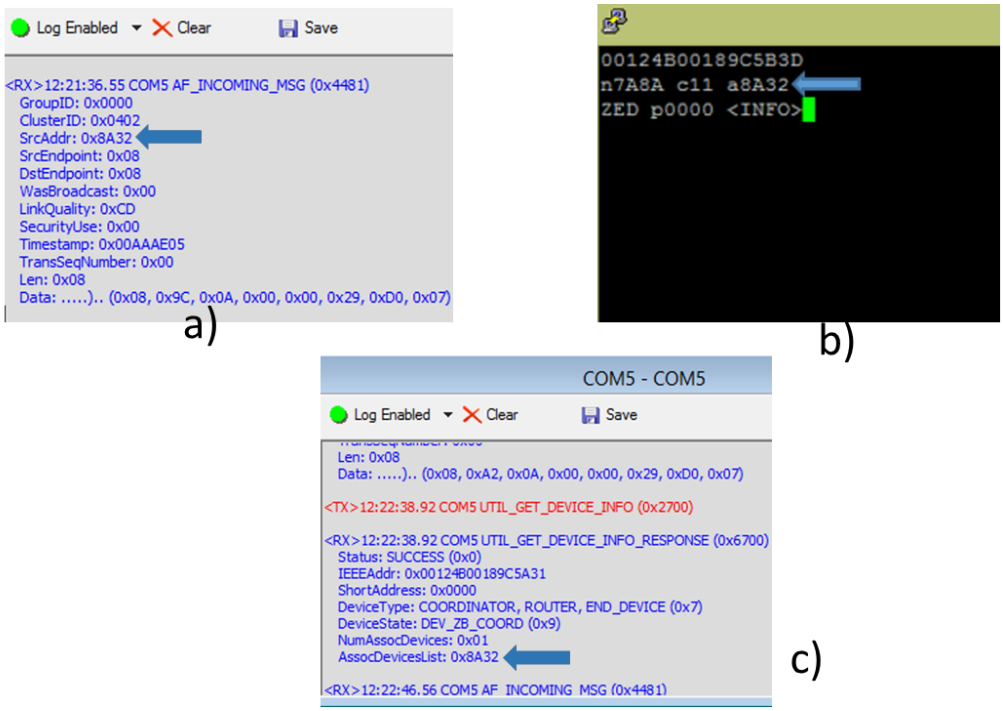


Figura 21. Datos recibidos del estatus de la red

En Figura 21 Figura 21a) muestra los datos recibidos: el identificador del grupo, del clúster, etc. Dos datos importantes para nuestro proyecto son la dirección del dispositivo que se están obteniendo los datos (dispositivo fuente) y el valor de la temperatura, pues es un sensor. Como se puede observar la dirección del dispositivo fuente es 0x8A32. Para verificar este valor, podemos entrar a la aplicación del sensor de temperatura y validar si es correcta la información. En la Figura 21 b)**¡Error! No se encuentra el origen de la referencia.**, vemos que efectivamente es el valor correcto.

Para verificar que efectivamente hemos creado la red, enviamos el comando GET_DEVICE_INFO, Figura 21 c), y vemos que efectivamente esta enlazado con un dispositivo y verificamos que dirección corta del dispositivo que es exactamente 0x8A32.

Como se mencionó anteriormente la aplicación nos estará enviando de forma periódica la información del sensor de temperatura, en el parámetro Data tenemos este valor, como se puede observar los dos últimos valores son 0xD0 y 0x07, Figura 21 a). Estos dos últimos valores son los de la temperatura remota, recordemos que primero se recibe el bit menos significativo, por lo que el orden del valor numérico es el siguiente 0x07 0xD0, son dos dígitos en numeración decimal, que al convertirlos a decimal nos da el valor de 20.00, tomamos las dos primeras cifras vemos que el valor de la temperatura es de 20°C. Para verificar esa afirmación, aumentamos la temperatura en el sensor de temperatura hasta 25°C, como se muestra en la Figura 22 a).

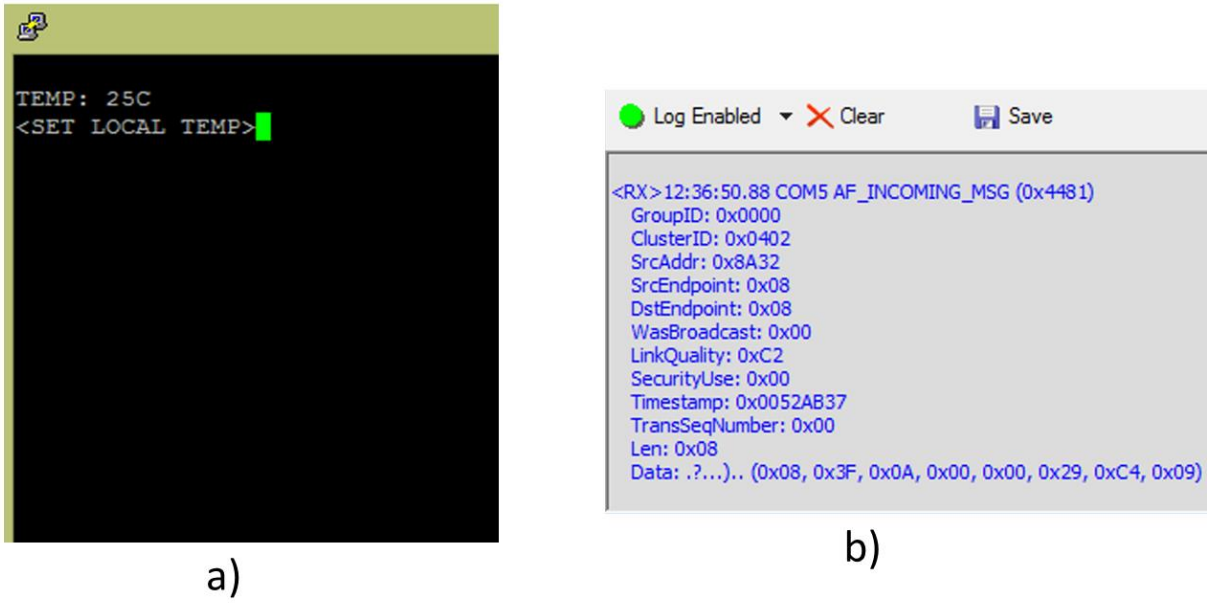


Figura 22. Medición de temperatura, a) local y b) remota.

Se observa la pantalla del Z-Tool, Figura 22 b) se nota que el valor cambia a 0x09, se realiza la conversión a decimal (0x09 0xC4) y obtenemos el valor 25 00, por lo que la temperatura es 25°. El Host estará recibiendo tramas de datos con la información del estado del nodo sensor, en este caso particular solo tenemos un sensor, pero podríamos agregar hasta 255 nodos y crear así una red PAN.

La puerta de enlace se construyó con el ZNP configurado como coordinador de red y utilizando la PC como procesador Host. Un siguiente proyecto podría ser omitir la utilización de la herramienta Z-Tool y realizar una aplicación para leer datos del puerto serial de la PC. Las tramas de datos que obtienen directamente del puerto serie son con el formato de trama visto en la sección 6.6. Un proyecto fututo puede ser desarrollar una aplicación para procesar la información obtenida de las tramas de datos, y posteriormente ya sea almacenar la información o transmitirla a través de internet.

8 Conclusiones

El panorama mundial apunta a un futuro donde todo nuestro entorno esta interconectado, es muy conocido el gran empuje que tienen las casas inteligentes, sin embargo, el internet de las cosas se expandirá a todos los ámbitos del ser humano, por ejemplo, el trabajo, en espacios públicos, recreación e incluso conflictos bélicos. El desarrollo de nodos sensores de bajo costo y muy eficientes ha permitido esta rápida expansión.

Como se muestra en este trabajo son tres dispositivos elementales en una red Zigbee: dispositivo final, enrutador y coordinador. El coordinador de red se encarga de la administración de red, sin embargo, también es el vínculo a redes de comunicación más grandes como lo es internet. El coordinador es una puerta de enlace entre la red local con la red global. De aquí viene la importancia de la puerta de enlace, para que sea una realidad el internet de las cosas es necesario desarrollar esta tecnología para acceder a redes grandes (Zigbee) a través de una sola IP.

La puerta de enlace desarrollada en este proyecto está conformando por un nodo ZNP (con atributos de un coordinador) y un Host (PC). A través de envío y recepción de comandos entre el Host y el ZNP se administra la red, de esta forma se puede acceder al status de los sensores, así como datos obtenidos de sus mediciones. La información es obtenida a través del puerto serie, esta información se podría almacenar y posteriormente compartirse en Internet completando de esta manera el concepto de IoT. Finalmente, dado que la red cumple con la normativa Zigbee se pude agregar más dispositivos al sistema, siempre y cuando que cumplan con el estándar.

9 Referencias

Al-Turjman, F., & Imran, M. (2020). *IoT Technologies in Smart Cities - From Sensors to Big Data, Security and Trust - 5.5.7 Zigbee.* Nicosia, Cyprus: The Institution of Engineering and Technology.

Callaway, E. (2003). *Wireless Sensor Network: Architectures and Protocols* . Florida, USA: CRC Press LLC.

Chen, Hua-Peng Ni, & Yi-Qing. (2018). Components of Wireless Sensors. En *Structural Health Monitoring of Large Civil Engineering Structures* (págs. 37-38). Hon Kong: John Wiley & Sons. Obtenido de https://app.knovel.com/hotlink/pdf/id:kt011UKTC2/structural-health-monitoring/front-matter

Daal, & Upena. (2005). *Wireless Communication and Networks.* USA: Oxford University Press. Obtenido de https://app.knovel.com/hotlink/pdf/id:kt00UPPEND/wireless-communication/ad-hoc-net-introduction

Dimon, T. G. (2003). *Automation, Systems, and Instrumentation Dictionary.* Estados Unidos: ISA. Obtenido de https://app.knovel.com/hotlink/pdf/id:kt004RBMT1/automation-systems-instrumentation/g-to-gyro-wheel

Huang, Jun Hua, & Kun. (2017). *Managing the Internet of Things - Architectures, Theories and Applications.* China: Institution of Engineering and Technology.

IEEE Std 802.15.4. (September de 2003). Standar for Part 15.4 Wireless Medium Acces Control (MAC) and Physical Layer Specifications for Low-Rate Wireless Personal Area

Networks (LR. WPANs). 679. New York, USA. Obtenido de http://standards.ieee.org/getieee802/download/802.15.4-2003.pdf

Jiang, Z., Han, J., & Liu, X. (2011). *Manufacturing Processes and Systems.* Shenzhen, China: Trans Tech Pubn.

Krutz, & Ronald , L. (2017). *Industrial Automation and Control System Security Principles-Protecting the Critical Infrastructure.* ISA. Obtenido de https://app.knovel.com/web/view/khtml/show.v/rcid:kpIACSSPPL/cid:kt0113TG31/viewerType:khtml//root_slug:industrial-automation/url_slug:the-tcp-ip-model?page=1&view=collapsed&zoom=1

Pahlavan, K., & Krishnamurthy, P. (2002). *Principles of Wireless Network.* Upper Saddle River: Pretince Hall PTR.

Postolache, Octavian, A. S., Mukhopadhyay, E., & Chandra, S. (2019). *Sensors in the Age of the Internet of Things - Technologies and Applications* -. Obtenido de https://app.knovel.com/hotlink/pdf/id:kt0124LBC1/sensors-in-age-internet/introduction

Sachchidanand, S., & Nirmala, S. (2015). Internet of Things(IoT): Security Challenges,Business Opportunities & Reference Architecture for E-commerce. *International Conference on Green Computing and Internet of Things (ICGCIoT)*, (pág. 1577).

Texas Instruments. (2005-2018). *Developing Zigbee Applications*. Obtenido de http://software-dl.ti.com/simplelink/esd/simplelink_cc13x2_sdk/2.30.00.45/exports/docs/zstack/html/zigbee/application_overview.html

Thompson, Lawrence , M. S., & Tim. (2016). *Industrial Data Communications.* Marlin Texas: ISA. Obtenido de https://app.knovel.com/hotlink/pdf/id:kt0112OPXC/industrial-data-communications/gateways

TI Technical Reference Manual CC13x2, C. S. (Octubre de 2019). Technical Reference Manual CC13x2, CC26x2 SimpleLink™ Wireless MCU. USA. Obtenido de https://www.ti.com/tool/LAUNCHXL-CC1352P

Verdone, Dardari, R., Mazzini, D., Conti, G., & Andrea. (2008). ZigBee MAC Sublayer. En *Wireless Sensor and Actuator Networks - Technologies, Analysis and Design* (págs. 131-132). Great Britain: Elsevier. Obtenido de https://app.knovel.com/hotlink/pdf/id:kt0091MFQ1/wireless-sensor-actuator/zigbee-mac-sublayer

Zheng, J., & Lee, M. (2004). Will IEEE 802.15.4 make ubiquitions networking a reality? *IEEE Comunications Magazine*, 140-146.