UNIVERSIDAD NACIONAL DE INGENIERÍA
FACULTAD DE CIENCIAS

TESIS

"OPTIMIZACIÓN DEL CONSUMO DE ENERGÍA EN EL EDGE LEVEL DE ARQUITECTURA FOG COMPUTING"

PARA OBTENER EL TÍTULO PROFESIONAL DE:

LICENCIADO EN CIENCIA DE LA COMPUTACIÓN

ELABORADA POR:

EDGAR JESÚS HUARANGA JUNCO

ASESOR:

DR. JOSÉ MANUEL CASTILLO CARA

LIMA - PERÚ

2021

Agradecimientos

Este trabajo ha sido financiado por FONDECYT / Banco Mundial (ref.026-2019 FONDECYT-BM-INC.INV).

Resumen

En los últimos años las redes de sensores inalámbricas han retomado su popularidad debido a que la complejidad de su implementación y el costo de fabricación se han reducido notablemente, haciéndolo accesible para muchas más personas e instituciones a nivel global. Esta globalización de las redes de sensores también es impulsada por los beneficios de conectividad que nos otorga Internet, teniendo como resultado lo que ahora llamamos Internet de las Cosas.

Sin embargo, a pesar de la diversidad, los avances y las mejoras que se presentan en el mercado y en la academia, las redes de sensores inalámbricas aún presentan un conjunto de problemas y retos en común. El presente trabajo se enfoca principalmente en dos de estos problemas: el primero, es el consumo de energía y tiempo de vida de la red; y el segundo problema es la confiabilidad del envío de información.

El trabajo mostrado en esta tesis consiste en un algoritmo de control de niveles de batería con el objetivo de optimizar el tiempo de vida de la red de sensores con la condición de no perder conectividad, y por lo tanto, información de parte de los nodos componentes.

Índice general

Agradecimientos II

Resumen III

Índice de figuras VII

Índice de cuadros VIII

Índice de acrónimos IX

1. Introducción 1

1.1. Motivación 1

1.2. Objetivos 2

1.3. Estructura del Seminario 3

2. Estado del Arte 5

2.1. Arquitecturas de Internet de las cosas 5

2.1.1. Arquitecturas centralizadas 6

2.1.2. Arquitecturas descentralizadas 7

2.2. Protocolos de transporte de información en redes de sensores 10

2.2.1. Conceptos claves 10

2.2.2. Red de sensores inalámbrica 12

2.2.3. XBee 15

2.2.4. Estándar ZigBee 22

2.2.5. Protocolo MQTT 25

2.3. Métodos de ahorro de energía en una red de sensores 29

2.3.1. Agregación de datos 29

2.3.2. Fuentes de energía alternativas 31

2.3.3. Control de componentes 32
2.4. Conclusiones 33

3. Diseño del estudio experimental **34**
3.1. Topología 34
3.2. Coordinator Node 35
3.2.1. Especificaciones de Software 35
3.3. Router Node y End Node 37
3.3.1. Formato de envío 41
3.4. Configuración XBee 43
3.4.1. XCTU 43
3.4.2. Configuración módulos XBee 44

4. Proceso de optimización de consumo de energía **49**
4.1. Medición del Consumo de energía 49
4.1.1. Caso 1: Batería de ácido de plomo 49
4.1.2. Caso 2: Batería de Litio 51
4.2. Optimización objetiva de Router Node 52
4.3. Algoritmo de balance de energía 54

5. Conclusiones y Trabajo Futuro **56**
5.1. Conclusiones 56
5.2. Trabajo Futuro 57
5.3. Producción científica y tecnológica 58
5.3.1. Contribuciones científicas 58
5.3.2. Patente 59

Referencias bibliográficas **59**

Índice de figuras

2.1. Ejemplo de una Arquitectura Cloud en una WSN Fuente: iiWAS [1] . . . 7
2.2. Arquitectura de Fog Computing. Fuente: Journal of Sensors [2] 8
2.3. Arquitectura Fog Computing Fuente: Sensors [3] 9
2.4. Topologías de red más comunes. Fuente: National Instruments Corp. [4] 15
2.5. Dispositivo XBee. Fuente: Digi International Inc. 15
2.6. Aplicaciones XBee. Fuente: Digi International Inc. 16
2.7. Aplicaciones XBee. Fuente: Digi International Inc. 16
2.8. MAC XBee. Fuente: Digi International Inc. 17
2.9. Conexión serial UART. Fuente: Digi International Inc. 18
2.10. XBee Java Library. Fuente: Digi International Inc. [5] 21
2.11. Capas ZigBee. Fuente: ZigBee Alliance. 23
2.12. Ejemplo de Red ZigBee. Fuente: ZigBee Alliance. 25
2.13. Tipos de QoS. Fuente: TechTarget. 27
2.14. Tipos de mensajes MQTT. Fuente: TechTarget. 28
2.15. Diagrama de flujo del LDD. Fuente: ICCOINS [6] 30
2.16. Modelo de un sistema alimentado por energía solar. Fuente: ICCOINS [7] 31
2.17. Modelo de un nodo en una WSN. Fuente: Sensors [7] 32
2.18. Algoritmo de selección de enrutamiento ECORS. Fuente: INDIN [8] 33

3.1. Topología de la red. Fuente: Journal of Sensors [2] 35
3.2. Nodo Master de la Red. Fuente: Elaboración propia 36
3.3. Sensor BMP180. Fuente: Adafruit Inc. 37
3.4. Sensor SHARP. Fuente: Adafruit Inc. 38
3.5. Sensor MQ131. Fuente: Adafruit Inc. 38
3.6. Sensor MQ135. Fuente: Adafruit Inc. 39
3.7. Sensor MQ7. Fuente: SparkFun Electronics. 39

3.8. Detector de sonido. Fuente: SparkFun Electronics.....40
3.9. Circuito del nodo. Fuente: Elaboración propia.....40
3.10. Router Node y End Node. Fuente: Elaboración propia.....40
3.11. Layout XCTU. Fuente: Digi International Inc.....44
3.12. Red ZigBee en XCTU. Fuente: Elaboración propia.....48

4.1. Consumo de energía con una batería de ácido de plomo para los dos tipos roles. Es posible observar un consumo constante de la energía en el End Node por un tiempo aproximado de 1 hora y 30 minutos. Mientras que en el Router Node, antes de la hora de uso se muestra un decaimiento significativo en su energía. Fuente: Elaboración propia. 50
4.2. Medición del consumo de energía de un nodo en modo End Node. Se puede apreciar un consumo uniforme de la energía de la batería por un tiempo aproximado de 2 horas. Por el contrario el nivel de energía de un Router Node cae abruptamente en cierto instante entre la hora y media y 2 horas de funcionamiento 51

Índice de cuadros

2.1. Comparación de tecnologías cableadas e inalámbricas..... 14
2.2. Características principales de los diferentes modos de operación del XBee. Fuente: Digi International Inc..... 19
2.3. Ventajas de los diferentes modos de operación XBee. Fuente: Digi International Inc. 20
2.4. Desventajas de los de los diferentes modos de operación XBee. Fuente: Digi International Inc..... 20

3.1. Configuración parámetros XBee. Fuente: Digi International Inc. 45

4.1. Especificaciones técnicas de las baterías utilizadas en el experimento. Fuente: Elaboración propia..... 50

Índice de Acrónimos

WSN	Wireless Sensor Network
SDK	Software Development KIT
JSON	Javascript Object Notation
MQTT	Message Queue Telemetry Transport
RFD	Reduced Function Device
FFD	Full Function Device
QoS	Quality of Service
MAC	Medium Access Control
IoT	Internet of Things
LDD	Local Data Differentiation
RF	Radio Frecuency
PAN	Personal Area Network
SH	Serial Number High
SL	Serial Number Low
NI	Node Identifier
OP	Operating PAN ID
CH	Operating Channel
SC	Scan Channel
UART	Universal Asynchronous Receiver/Transmitter
NWK	Network Layer
APS	Application Sub-Layer
ZDO	ZigBee Device Object
TCP/IP	Transmission Control Protocol/Internet Protocol
SSL/TLS	Secure Socket Layer/Transport Layer Security

Capítulo 1

Introducción

En el presente capítulo se presentan los motivos por el que se decidió realizar este proyecto, se incluye una breve descripción de los capítulos siguientes. Este capítulo tiene como objetivo brindarle al lector un mejor panorama de la forma en la que se realizaron los experimentos, la base tecnológica que la soporta, las propuestas y, finalmente, las discusiones y conclusiones.

1.1. Motivación

La capacidad del ser humano de sentir, de reconocer objetos a través de sus sentidos es inherente. Un punto de quiebre en la historia de la humanidad es cuando se desarrolla la capacidad de medir estas sensaciones, cuantificarlas y tener un registro de ello. Estas mediciones sirvieron como modelos, experiencia y fueron la base de muchas creaciones entre monumentos, equipos y materiales. El desarrollo científico y tecnológico durante cientos de años ahora nos permite recolectar información, transportarla, almacenarla y analizarla casi desde cualquier lugar gracias a Internet.

El desarrollo tecnológico también hizo posible que sea cada año más fácil acceder a estas herramientas, entre sensores y componentes de hardware, haciéndolos cada vez más pequeños y baratos. Dependiendo de su naturaleza, se podría medir, por ejemplo, la temperatura, la humedad, radiación UV, radiación solar, etc. Sin embargo, todas estas mediciones también cuentan con limitaciones. Una de estas limitaciones es que es imposible hacer una lectura continua de estos parámetros ya que esa información es infinita y no habría dispositivo con la capacidad suficiente para almacenar esa información.

A pesar de esta limitación, los resultados de recolectar esta información por un conjunto de sensores es muy importante ya que, después de un proceso adecuado, permite tomar decisiones y analizar comportamientos pasados. Un escenario particular como el de la agricultura de precisión utiliza un conjunto de sensores conectados entre sí para recolectar información sobre el campo de cultivo y determinar si las condiciones son óptimas para el crecimiento de un producto específico. Este grupo de sensores interconectados es llamado red de sensores.

Siguiendo en el entorno de agricultura y analizando la realidad en el Perú, que en su rol como país exportador se hace notoria la necesidad de implementar este enfoque. Sin embargo, a pesar de que los volúmenes de exportación se han ido incrementando con los años, estos aún no satisfacen la demanda extranjera. La implementación y difusión de este tipo de tecnologías se puede considerar como una buena alternativa para abordar este problema donde además tenga como resultado un producto de calidad.

En el presente trabajo se propone un control de niveles de energía en una red de sensores inalámbrica con el fin de prolongar su tiempo de vida. De esta manera la red tiene la capacidad de recolectar mayor cantidad de datos de manera automatizada y así enfocarse en el estudio de esta información y su impacto en la actividad a la que está orientada la red.

1.2. Objetivos

El objetivo principal de este trabajo consiste mejorar el control del consumo de energía en el *Edge Level* (nivel donde se encuentra desplegada) de una red de sensores inalámbrica (a partir de ahora WSN, por las siglas en inglés de *Wireless Sensor Network*) basada en el estándar de comunicación ZigBee. Este es el factor principal que permite optimizar los recursos para extender el tiempo de vida útil de la red y obtener una red de sensores más eficiente. El objetivo principal puede ser dividido en objetivos específicos para que se entienda mejor.

- **Definir la arquitectura de red y protocolo de comunicación entre los nodos de la red**.

Se busca definir la arquitectura adecuada para la red de sensores de tal manera que aseguremos el envío de datos y valide el control de nivel de batería en los nodos.

- **Análisis y elección del modo de comunicación en dispositivos XBee.**
  El fin de este objetivo es evaluar las ventajas y desventajas de los modos disponibles para dispositivos XBee y determinar cuál es el más adecuado para el objetivo principal planteado.

- **Análisis de parámetros de comunicación en un estándar ZigBee.**
  Luego de elegir el modo en el que debe actuar el XBee, se analiza los parámetros que caracterizan a este modo para ser modificados durante el tiempo de actividad de la red.

- **Configuración inicial de los dispositivos XBee.**
  Una vez definida la arquitectura, este punto tiene como objetivo configurar los parámetros de cada uno de los dispositivos XBee para dar inicio a la red.

- **Propuesta del algoritmo controlador de roles.**
  Con la red configurada, el objetivo siguiente es administrar los parámetros de los dispositivos XBee en cada nodo de manera remota desde el nodo máster bajo determinadas condiciones.

1.3. Estructura del Seminario

Para brindar al lector una idea global del contenido de este trabajo, se hace una breve descripción del propósito de cada capítulo y de su importancia en el desarrollo y construcción del objetivo final.

- **Introducción:**
  En este capítulo se presentan los motivos por el que se decidió realizar este proyecto, se incluye una breve descripción de los capítulos siguientes. Este capítulo tiene como objetivo brindarle al lector un mejor panorama de la forma

en la que se realizaron los experimentos, la base tecnológica que la soporta, las propuestas y, finalmente, las discusiones y conclusiones.

- **Estado del Arte:**

  El segundo capítulo de este trabajo consiste en la descripción de métodos e investigaciones que comparten el objetivo de reducir el consumo de energía en las WSN. En este capítulo también se explican las bases tecnológicas y conceptos utilizados para el diseño de estudio experimental, el proceso de optimización y las conclusiones finales.

- **Diseño del estudio experimental:**

  A partir de los conceptos desarrollados en el capítulo anterior, en este se describe la configuración de los equipos de radiofrecuencia, la topología elegida, formato de envío de información entre nodos, transmisión al Core Level y los procedimientos para realizar los experimentos necesarios para justificar algunas hipótesis planteadas en el capítulo anterior.

- **Proceso de optimización del consumo de energía:**

  En este capítulo se muestran los resultados del procedimientos que justifican el desarrollo del enfoque propuesto para maximizar el tiempo de vida de una WSN. Aquí también se propone un juego de cambio de roles correspondientes al estándar ZigBee a partir de la topología propuesta en el capítulo anterior y el análisis del consumo de cada tipo de nodo en la WSN.

- **Conclusiones y Trabajo a Futuro:**

  En este capítulo se exponen las conclusiones obtenidas de este trabajo a partir de los experimentos realizados. Se discuten los resultados a partir de los conceptos y base tecnológica del segundo capítulo para finalmente, proponer trabajos a futuro para cubrir áreas no consideradas o estudiar nuevas aplicaciones a este control de los parámetros en un dispositivo XBee.

Capítulo 2

Estado del Arte

Este capítulo describe los las características, tipos y limitaciones de las arquitecturas existentes para las WSN, se incluye también una descripción del protocolo de comunicación utilizado para el transporte de información entre nodos. También se hace un estudio de los métodos e investigaciones que comparten el objetivo de reducir el consumo de energía en las WSN.

De acuerdo a lo mencionado en el primer capítulo, la complejidad para implementar redes de sensores cada vez es más fácil gracias al desarrollo de componentes más pequeños y baratos. No existe un estándar por defecto para WSN, sino que los componentes y tecnologías utilizadas son elegidas según la necesidad que se quiera abordar.

2.1. Arquitecturas de Internet de las cosas

Una de las características fundamentales de Internet de las cosas (a partir de ahora IoT, por las siglas en inglés de *Internet of Things*) descrita en [9] es que el conjunto de sensores en una WSN recolecten información con la menor interacción humana posible a fin de brindar un nuevo tipo de aplicación. En el contexto del objetivo fundamental del proyecto, donde se debe extender el tiempo de vida de la red, se alinea con la reducción de interacción humana al mismo tiempo que la cantidad de información recolectada es mayor.

Una de las principales decisiones que se deben tomar antes del despliegue de la red es sobre la topología de la red si va a ser centralizada o descentralizada. Por un lado, elegir una topología centralizada podría simplificar el control y mantenimiento

de la red. Mientras que una topología descentralizada muchos servicios reducen la sobrecarga del nodo central de la red. A continuación de describe en mayor detalle cada una de las arquitecturas.

2.1.1. Arquitecturas centralizadas

Nos referimos a arquitecturas centralizadas cuando tenemos un conjunto de servicios desarrollados en un mismo punto de nuestra arquitectura, esto es, Cloud Computing como arquitectura más ampliamente desplegada. El desarrollo de Cloud Computing también ocupa un espacio en la implementación de las redes de sensores. Apostar por un enfoque Cloud significa centralizar muchas actividades que usualmente se hacían a nivel local y cerca a donde la red de sensores esté localizada. Además, estando en la nube, se puede aprovechar muchos recursos como por ejemplo el poder de procesamiento computacional, la capacidad de almacenamiento y la alta disponibilidad para llegar al usuario final de la aplicación de la WSN. Un ejemplo de este tipo de arquitectura puede verse en la Figura 2.1 propuesta por W. Kurschl et al. [1].

Las ventajas de utilizar una arquitectura como esta en una WSN pueden ser las siguientes:

- **Escalabilidad** de los recursos computacionales. A diferencia de los nodos de la red, en un entorno Cloud se puede incrementar tanto el poder de procesamiento computacional como el almacenamiento para procesar la información proveniente de los nodos.
- **Alcance** a los usuarios finales. En un entorno Cloud las personas a cargo de analizar, procesar o tomar decisiones a partir de la información recogida por la WSN pueden realizar su tarea desde cualquier lugar que tenga conexión a Internet, acelerando muchos procesos. Un ejemplo es el envío temprano de alertas.
- **Interoperabilidad** entre los diferentes tipos de dispositivos, servicios. La abstracción para manejar los diferentes tipos de dispositivos e información es manejada de manera central en la nube.

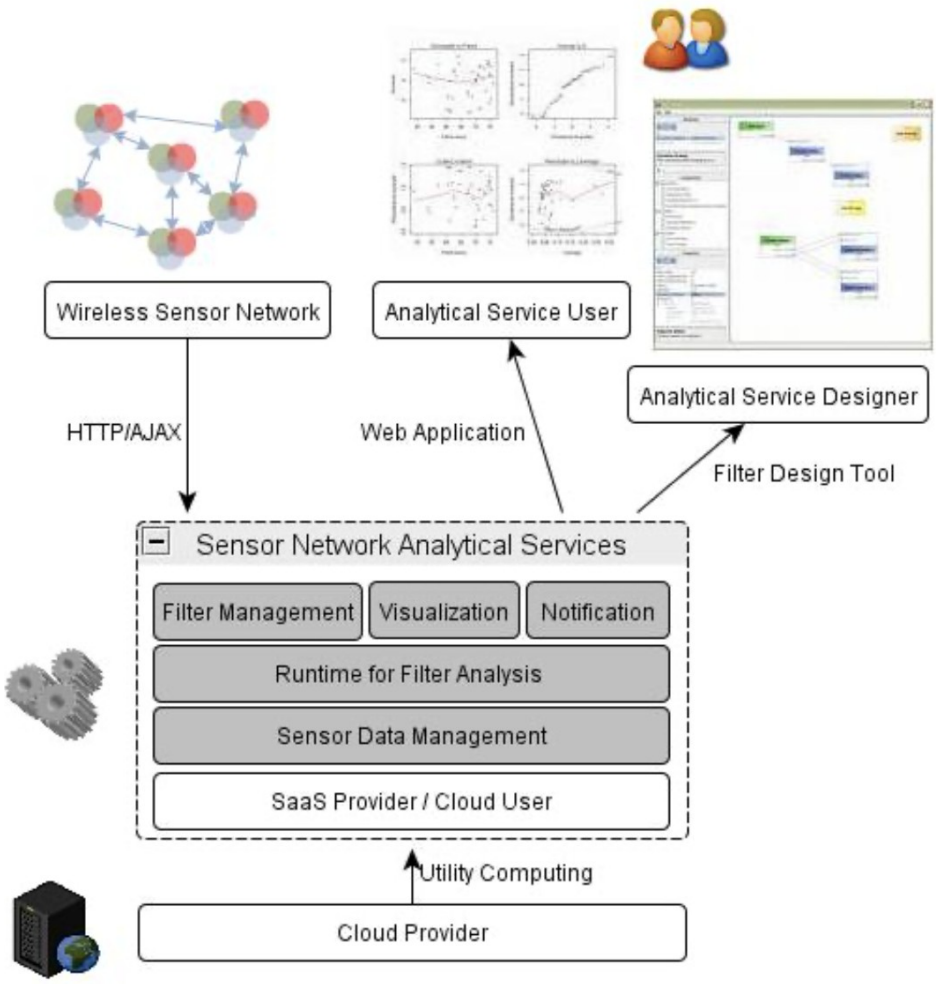


FIGURA 2.1: Ejemplo de una Arquitectura Cloud en una WSN Fuente: iiWAS [1]

Este tipo de enfoques, ha sido potenciado por el mismo crecimiento del negocio de Cloud Computing en el mundo. Resulta evidente la escalabilidad de implementar una WSN haciendo uso de esta tecnología, simplifica el monitoreo desde cualquier lugar del mundo que tenga conexión a Internet.

2.1.2. Arquitecturas descentralizadas

El impresionante desarrollo del lado de hardware y software de los componentes de los nodos en una WSN trajo consigo un nuevo paradigma para descentralizar recursos. Fog Computing, que puede ser entendido como una extensión natural de Cloud Computing donde su principal característica es que el procesamiento de datos

y algunos servicios son realizados por dispositivos inteligentes intermedios llamados Fog Nodes. Este tipo de dispositivos inteligentes se encuentran físicamente más cerca a los usuarios finales que del procesamiento central.

En este contexto, una arquitectura de Fog Computing es dividida en dos niveles: (i) el Core level; y (ii) el Edge Level. Los componentes y tecnologías para cada nivel son adaptados para los servicios que van a proveer cada uno de ellos como se muestra en la Figura 2.2. El Core Level comprende los principales componentes del sistema, por ejemplo, la base de datos principal y el intermediario, entre otros [10]; mientras que el Edge Level está compuesto por otros componentes: redes de sensores y actuadores inalámbricos y los nodos Fog. El último comprende un broker local, bases de datos locales y un servidor de procesamiento y visualización [10, 11, 12]. Los Fog Nodes están a cargo de procesar los datos sensibles al tiempo mientras hacen un mejor uso de la infraestructura de red subyacente.

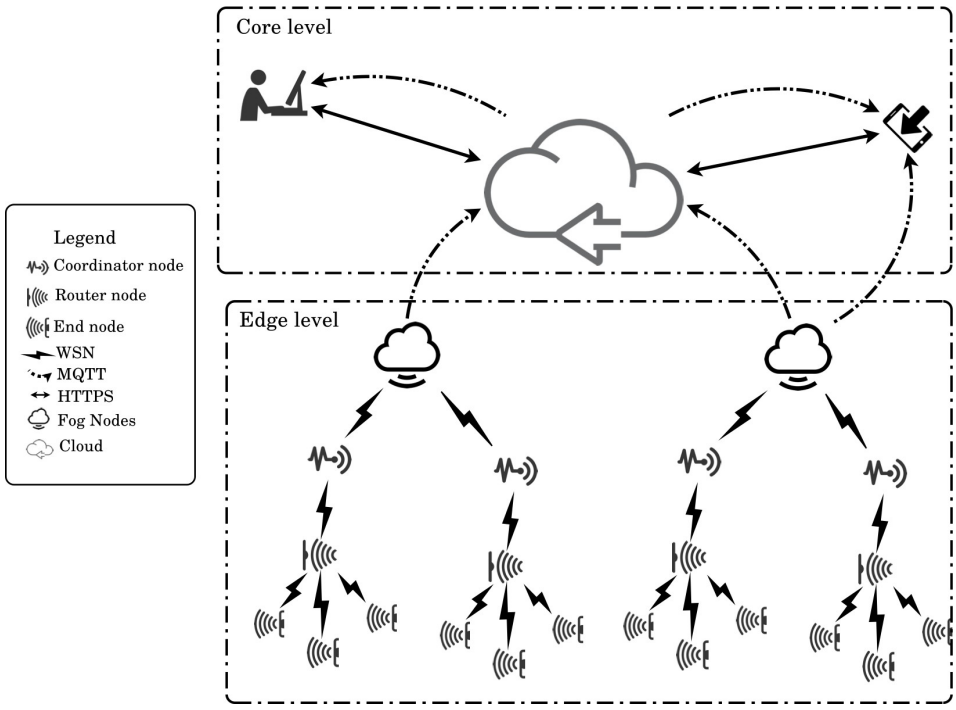


FIGURA 2.2: Arquitectura de Fog Computing. Fuente: Journal of Sensors [2]

Las mejoras en software y en hardware de los componentes de los nodos en una red de sensores, han hecho posible posible tener el mismo poder de cómputo y capacidad de almacenamiento que un entorno Cloud de hace 10 años en una micro computadora

del tamaño de la mitad de una tarjeta de crédito. Este tipo de avances en la tecnología permite proponer nuevas arquitecturas para las redes de sensores atentiendo algunos problemas surgidos frente a una arquitectura centralizada.

Uno de estos retos es la manera en que la información llega a los usuarios finales físicamente cercanos a la red de sensores. El viaje ida y vuelta de los paquetes de información hasta la nube generaba un tiempo perdido en la adquisición de información. Dada las capacidades de los nuevos componentes de las redes de sensores la propuesta de procesar ciertas operaciones en estos nodos intermedios con la capacidad suficiente es una buena opción que puede reducir ese tiempo de respuesta ante operaciones específicas. Esta capa intermedia entre el Cloud y los dispositivos finales, como se describe en la Figura 2.3 recibe el nombre de Fog Computing.

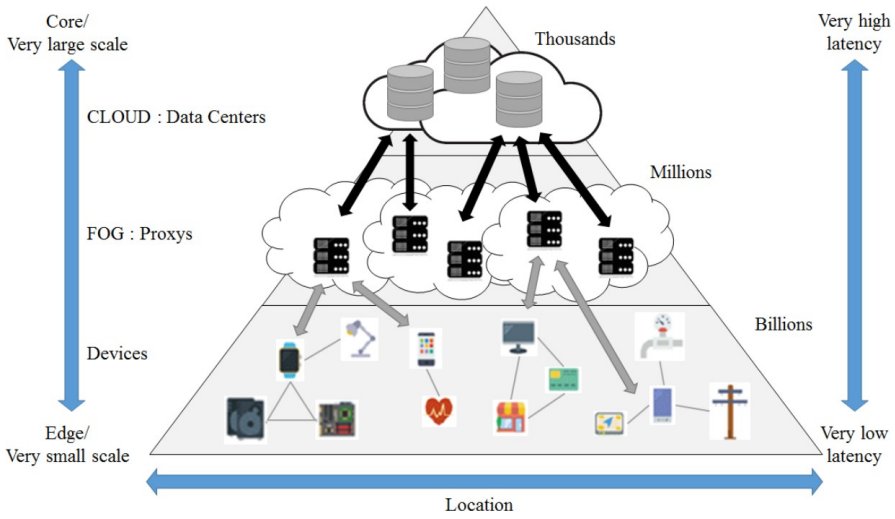


FIGURA 2.3: Arquitectura Fog Computing Fuente: Sensors [3]

Otras ventajas de apostar por un enfoque descentralizado son:

- **Menor latencia** ya que los datos no necesitan ser enviados hasta la nube para ser procesados.
- **Menor consumo de ancho de banda** ya que, según las necesidades, únicamente información relevante puede ser enviada al servidor central en vez de un gran conjunto de datos no relevantes.

- **Mejor seguridad** ya que la información está distribuida en diferentes nodos en vez de una estación central.

Un entorno Fog puede ser más complejo de implementar y planear que una arquitectura Cloud. Sin embargo, esto no significa que sean tecnologías excluyentes; al contrario, el Fog Computing debe ser visto como una extensión natural del Cloud Computing a fin de resolver algunos problemas surgidos por la cantidad inmensa de dispositivos con capacidad de conectarse a la nube del proyecto.

2.2. Protocolos de transporte de información en redes de sensores

Dentro de una red de sensores, es importante describir específicamente los tipos de nodo que se van a utilizar, así como la función que realiza cada uno de ellos. En este trabajo se utilizó la tecnología ZigBee como protocolo de comunicación, por lo tanto, se manejaron 3 tipos de nodos: (i) el Coordinator Node conectado a un microcomputador con conexión a Internet; (ii) un Router Node como un enrutador para que; (iii) los End Nodes puedan enviar información al Coordinator Node a pesar de estar fuera de su alcance directo.

2.2.1. Conceptos claves

Los protocolos y dispositivos de radiofrecuencia comunmente utilizan conceptos fundamentales que son descritos a continuación [13]:

- Una **Señal** de radiofrecuencia es una variación de energía eléctrica, con oscilación entre un valor máximo y mínimo.
- **Amplitud** es la diferencia entre el máximo y mínimo valor durante el ciclo de una señal.
- **Frecuencia** es el número de veces que una señal completa un ciclo de oscilación en un segundo y se mide en Hertz. Los dispositivos de radiofrecuencia se caracterizan por el rango de frecuencias que pueden generar.

- **Banda de frecuencia** es una manera estándar de nombrar a un rango específico de frecuencias. Las bandas más usadas son las siguientes:

  - HF High Frecuency 3 MHz - 30MHz
  - VHF Very High Frecuency 30MHz - 300MHz
  - UHF Ultra High Frecuency 300MHz - 3GHz
  - SHF Super High Frecuency 3 GHz - 30GHz
  - EHF Extra High Frecuency 30 GHz - 300GHz

  Existen bandas denominadas ISM (Industrial Scientific and Medical) están internacionalmente reservadas para uso no comercial para el campo científico, industrial o médico. Algunas de estas bandas son las siguientes:

  - 900 MHz ISM Band 902 MHz - 928 MHz (USA).
  - 2.4 GHz ISM Band 2400 MHz - 2483.5 MHz
  - 5 GHz ISM Band 5725 MHz - 5875 MHz

- **Ancho de banda** es el término utilizado para expresar cuánta información se transmite en un periodo de tiempo.

- **Longitud de onda** es la distancia que viaja una onda durante un ciclo. Esta distancia puede obtenerse a partir de la ecuación $\lambda = \frac{c}{f}$ donde c es la velocidad de la luz y f es la frecuencia de la onda.

- **Potencia** se mide en Watts y es el resultado de multiplicar el voltaje, o amplitud de onda y la corriente, en amperios.

- **Decibel** es una expresión logarítmica entre la potencia, voltaje o corriente entre dos señales. Cuando una medición de decibel es positiva significa que la potencia de transmisión de la primera onda es mayor que la segunda, si es negativa entonces la potencia de la segunda es mayor. El estándar de referencia es de $1mW$ $(0.001 Watts)$.

2.2.2. Red de sensores inalámbrica

Una WSN consiste en dispositivos distribuidos en un determinado espacio que tienen la capacidad de monitorear parámetros físicos o ambientales [4]. La comunicación entre los dispositivos de la WSN está definido por un protocolo de transporte de información inalámbrico, los más utilizados son los que pertenecen a las bandas ISM basados en el estándar IEEE 802.15.4 ZigBee o IEEE 802.11 Wi-Fi/Bluetooth. Para entender las necesidades de una WSN podemos describirla a partir de los siguiente puntos:

- **Protocolos**. Existen diversos protocolos de comunicación, cada uno está orientado a determinados requerimientos y limitaciones de la aplicación de la WSN. Usualmente, los factores a considerar son distancia entre dispositivos, cantidad de información que se recoje y envía y costo de implementación. Es necesario determinar también si realmente la aplicación necesita comunicarse de manera inalámbrica o no, para ese propósito se toma en cuenta la Tabla 2.1. Si se compara los protocolos inalámbricos basados en el estándar IEEE 802.11 e IEEE 802.15.4, se puede apreciar que existe un mayor ancho de banda para Wi-Fi. Mientras que para ZigBee se tiene un mayor rango de alcance. Lo mismo pasa con la batería del dispositivo conectado al WSN, si el ancho de banda es mayor, entonces el consumo de batería es también mayor. En otras palabras, energía y distancia son el precio a pagar por un mayor ancho de banda [4].

- **Topologías de red en una WSN**. Cada dispositivo que forma parte de una WSN es llamado nodo. La manera en que los nodos de una WSN están conectados define una topología. Una topología de tipo estrella es aquella donde cada nodo se conecta directamente al nodo que enviará la información a una central de datos o también llamado nodo *gateway*. Una topología de árbol es aquella donde existen jerarquías hasta llegar al nodo gateway. Y las topologías de tipo mesh son aquellas donde cada nodo puede conectarse a más de un nodo para ofrecer mayor confiabilidad. La Figura 2.4 muestra de manera gráfica estas topologías.

- **Componentes del nodo de una WSN**. Cada nodo en una WSN tienen distintos componentes electrónicos, estos dependen del tipo de información que se va a medir. Sin embargo, de manera general todo nodo incluye lo siguiente:

  - **Dispositivo de comunicación**: Son los encargados de la transmisión y/o recepción de señales.
  - **Fuente de energía**: Dispositivo que puede ser constante o una batería que se añade al nodo en caso se quiera independencia. El consumo de la batería se determina por los otros componentes del nodo.
  - **Microcontrolador**: Es el encargado de realizar las lecturas de los sensores y convertirlas a formatos adecuados dependiendo del sistema de medición acorde al parámetro. También enviar los datos al dispositivo de comunicación o manejar los datos que llegan a través de él.
  - **Circuito analógico**: Dispositivo necesario para que el microcontrolador reciba las señales analógicas o digitales de los distintos sensores del nodo.
  - **Sensores**: Acorde a los parámetros que se quieren medir según la aplicación que se propone.

- **Restricciones en una WSN**. Debido a los recursos de cómputo y energía que tienen los nodos, existen ciertos requisitos que se deben cumplir para una red de sensores [14]:

  - **Autonomía**: El problema de tener un único nodo encargado del manejo de envío de información y enrutamiento lo hace vulnerable a un ataque. Si cada nodo fuera autónomo, un nodo gateway no se hace necesario y existiría una forma más segura de enviar los datos.
  - **Eficiencia de la energía**: Los protocolos de comunicación deberían prolongar el tiempo de vida al mismo tiempo que mantienen interconectado cada nodo. Se debe tomar en cuenta también que reemplazar baterías no es una opción para prolongar el tiempo de vida ya que podría haber nodos que son inaccesibles.

	Cable Ethernet	**Fibra óptica**	**Wireless 802.11**	**Wireless 802.15.4**
Frecuencia RF/Material	Cobre	Fibra óptica	2.4 GHz	2.4 GHz
Ancho de banda (máximo)	100 Mbit/s	100 Mbit/s	54 Mbit/s	250 kbit/s
Rango	100 m	400 m	~100 m	~800 m
Energía requerida	Alto	Alto	Medio	Poco
Tiempo de vida de una batería	-	-	1 - 2 días	3 - 5 años

TABLA 2.1: Comparación de tecnologías cableadas e inalámbricas

- **Escalabilidad:** Una WSN puede estar compuesta desde unos cuantos nodos hasta cientos de nodos. Entonces los protocolos de enrutamiento deben seguir funcionando a pesar del número de nodos que este WSN tiene.
- **Capacidad de recuperación:** En la práctica, unos nodos podrían dejar de funcionar debido a condiciones ambientales o la descarga de la batería. Los protocolos de enrutamiento deben seguir funcionando a pesar de estas caídas.
- **Heterogeneidad de dispositivos:** A pesar que la mayoría de aplicaciones tienen nodos simulares midiendo los mismos parámetros, tener nodos heterogéneos significaría una mejora en las características en la red, por ejemplo en el ancho de banda, el consumo de energía o la escalabilidad de la red.

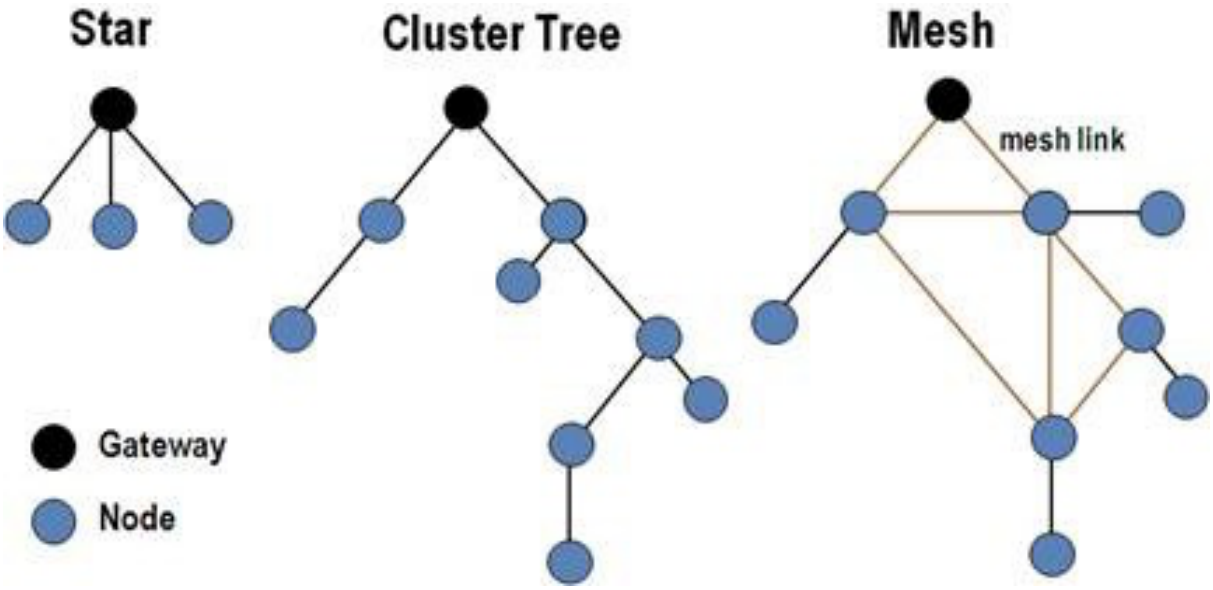


FIGURA 2.4: Topologías de red más comunes. Fuente: National Instruments Corp. [4]

2.2.3. XBee

Los XBee son pequeños dispositivos fabricados por la empresa Digi que transmiten y reciben información mediante radiofrecuencia. Son altamente configurables y pueden soportar múltiples protocolos, esto te permite escoger la tecnología adecuada a tu aplicación. El módulo es el mostrado en la Figura 2.5. Algunos ejemplos de dónde se puede usar los XBee son los siguientes:

FIGURA 2.5: Dispositivo XBee. Fuente: Digi International Inc.

Podemos enumerar algunas de las aplicaciones donde un XBee es utilizado:

- Controlar un robot de manera remota.
- Crear wearables sin cables que estorben su utilización.

- Dentro de la industria, los RF XBee pueden ser usados dentro de los nodos para monitorear temperatura, nivel del agua en tanques, humedad o presión en determinados ambientes de la fábrica.

FIGURA 2.6: Aplicaciones XBee. Fuente: Digi International Inc.

- **Comunicación XBee** La comunicación entre dispositivos XBee es inalámbrica. Los XBee solo se encargan de enviar y recibir los mensajes pero no en manejarlos. Sin embargo, es posible comunicarse con dispositivos que hagan este trabajo, como microcontroladores o computadoras, mediante una conexión serial. Un XBee transmite datos que recibe mediante una entrada serial, los transmite inalámbricamente, otro XBee recibe el mensaje y lo envía a una salida serial como es mostrado en la Figura 2.7. De esta manera, se puede tener un control de lo que se envía y de lo que se recibe mediante el serial.

FIGURA 2.7: Aplicaciones XBee. Fuente: Digi International Inc.

Para que los dispositivos XBee sean capaces de transportar información toman en cuenta los siguientes conceptos, relevantes también para la configuración de la red.

- **Direccionamiento.** Las direcciones de XBee son similares a direcciones de postal o direcciones email, algunas pueden ser únicas pero otras, como cuando se

comparte una dirección postal, pueden ser iguales. Cada XBee es conocido por muchas direcciones, cada una con un propósito específico.

- **Dirección 64-bits**. Es un valor único para identificarlo de otros dispositivos y prever información duplicada. Esta dirección también es llamada MAC y es asignada por la IEEE, así dos módulos no tienen la misma dirección. Este valor se encuentra impreso en el XBee como se muestra en la Figura 2.8 y obtenido si leemos los parámetros Serial Number High (SH) y el Serial Number Low (SL) en cualquier módulo XBee.

FIGURA 2.8: MAC XBee. Fuente: Digi International Inc.

- **Dirección 16-bits**. Un dispositivo recibe una dirección aleatoria de 16 bits cuando se une a una red ZigBee, esta dirección se denomina "dirección en la red". Esta dirección solo puede cambiar si el dispositivo se elimina de la red y es agregada nuevamente. La lectura del parámetro Network Address (MY) nos da esta dirección. Para el Coordinator Node es reservada la dirección 0000.
- **Identificador de Nodo**. Este identificador es una cadena de caracteres que puede ser editada por el usuario, otorgando un nombre relacionado a su función dentro de la red y sea más fácil de identificar. Se puede leer el parámetro Node Identifier (NI) para obtener este valor.

- **Dirección de una Red de Área Personal**. Es otro nombre que se le da a una red ZigBee o PAN(Personal Area Network). Un único *PAN ID* define a cada red y el identificador es común en todos los dispositivos en una misma red.

Algunos dispositivos ZigBee están preconfigurados con un PAN ID para unirse o pueden descubrir redes cercanas y seleccionar una a cual unirse. Este valor puede ser configurado mediante el parámetro *ID*. Si este valor es 0, el XBee automáticamente selecciona el PAN ID, así podremos leer el PAN ID actual con el parámetro Operating PAN ID (OP).

- **Canales**. Para que los dispositivos se puedan comunicar, deben operar bajo la misma frecuencia, los XBees soportan todos los 16 canales definidos en la capa física del estándar 802.15.4 y el XBee PRO, soporta 14 de los 16 canales. Para determinar el canal de operación en el XBee se debe leer el parámetro Operating Channel (CH). A diferencia del 802.15.4 el parámetro CH no puede ser escrito en una aplicación ZigBee. Sin embargo, se puede seleccionar el CH configurando un bit simple en el parámetro Scan Channel (SC). Ese bit simple fuerza al Coordinator Node operar sobre ese canal especificado por ese bit; además, previene que Router Nodes o End Nodes se unan a redes en otro canal que no tenga el especificado en SC.

Un XBee puede operar de manera autónoma o puede ser embebido a un dispositivo inteligente, como un microcontrolador o una computadora. Cuando esto sucede, la interfaz serial del XBee envía información a la interfaz serial del XBee para transmitirla o el XBee recibe información y esta es enviada al dispositivo inteligente mediante la interfaz serial. Los XBee utilizan Universal Asynchronous Receiver Transmitter (UART) para la comunicación serial (ver Figura 2.9).

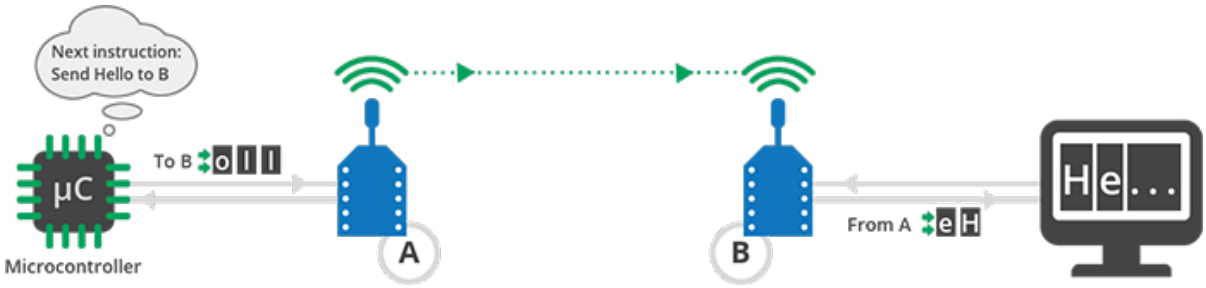


FIGURA 2.9: Conexión serial UART. Fuente: Digi International Inc.

Los XBee pueden usar su conexión serial en diferentes formas, sirve para establecer cómo un XBee se comunica con un dispositivo (Microcontrolador o Computadora) a través de la interfaz serial. Estos se conocen como los modos de operación.

Modo Transparente	Modo API
No puede enviar a múltiples destinos.	Envía datos a múltiples destinos.
No se puede configurar el XBee de forma remota.	El XBee puede ser configurado de forma remota.
Solo recibe datos de un XBee.	Puede recibir paquetes de datos de múltiples XBees e identifica de dónde son enviados.
No recibe muestras de entrada/salida de XBees.	Recibe muestras de entrada/salida de XBees.
No soporta múltiples End Nodes, clusters o perfiles (ZigBee).	Debe soportar múltiples End Nodes, clusters y perfiles(ZigBee).
No usa objetos ZigBee.	Usa servicios de objetos ZigBee(ZigBee Device Objects).

TABLA 2.2: Características principales de los diferentes modos de operación del XBee. Fuente: Digi International Inc.

- **Modo Transparente:** Es llamado así porque el XBee envía la información tal y como la recibe. Toda la información recibida por el Xbee es enviado inalámbricamente a un XBee remoto y cuando el otro módulo recibe el paquete, lo envía por el puerto serial tal como lo recibió.

- **Modo Interfaz de Programación para Aplicación** (API, por sus siglas en ingles): Un protocolo determina la forma en que la información es intercambiada. La comunicación es enviada por paquetes, comúnmente llamados *frames*. Este modo permite construir redes más extensas y es la más apropiada para crear redes de sensores que realizan tareas de recolectar información desde múltiples nodos.

La elección del modo de operación del XBee depende del perfil y requerimientos de la aplicación a desarrollar. La Tabla 2.2 describe las principales características de cada modo, mientras que la Tabla 2.3 y Tabla 2.4 hacen mención de las ventajas y desventajas de cada modo, respectivamente. De acuerdo a nuestro objetivo y dadas las características de los distintos modos de operación, el modo bajo el que se realizará el trabajo es el modo API.

A partir del 27 de enero del 2015 Digi International Inc., la empresa que comercializa los XBee, puso a disposición su librería Java para dispositivos XBee. El cual soporta el protocolo ZigBee además de otros protocolos propios de la empresa como DigiMesh and Point-to-Multipoint [5].

Modo Transparente	Modo API
Provee una interfaz simple que lo hace fácil comenzar con dispositivos XBee.	Se puede configurar o leer la configuración de un XBee remoto en la red.
Es fácil de mantener, porque lo que se envía es lo que recibe exactamente el otro módulo y viceversa.	Puede transmitir a uno o más destinos, esto es más rápido que el modo transparente donde la configuración debe ser actualizada para establecer una nueva dirección.
Funciona muy bien en comunicación bidireccional entre XBees.	La información recibida incluye la dirección del que la envió.
	Información recibida incluye detalles de transmisión y razones de éxito o falla.
	Muchos más características avanzadas, como diagnósticos de red y actualizaciones de firmware.

TABLA 2.3: Ventajas de los diferentes modos de operación XBee. Fuente: Digi International Inc.

Modo Transparente	Modo API
No puede configurar o leer la configuración de un XBee remoto en la red.	La interface es más compleja, los datos están estructurados en paquetes con un formato específico.
Se debe actualizar primero la configuración para establecer un nuevo destino y transmitir.	Es más difícil de mantener, las transmisiones son estructuradas en paquetes que necesitan ser parceados(cuando se recibe datos) o creados(cuando se envia datos).
No se puede identificar la fuente de la información recibida.	Información enviada y recibida no son idénticas, los paquetes recibidos incluyen unos datos de control e información extra.
La información recibida no incluye detalles de transmisión o las razones de éxito o fallo.	
No ofrece características avanzadas del modo API, incluyendo diagnósticos avanzados de red o actualizaciones de firmware.	

TABLA 2.4: Desventajas de los de los diferentes modos de operación XBee. Fuente: Digi International Inc.

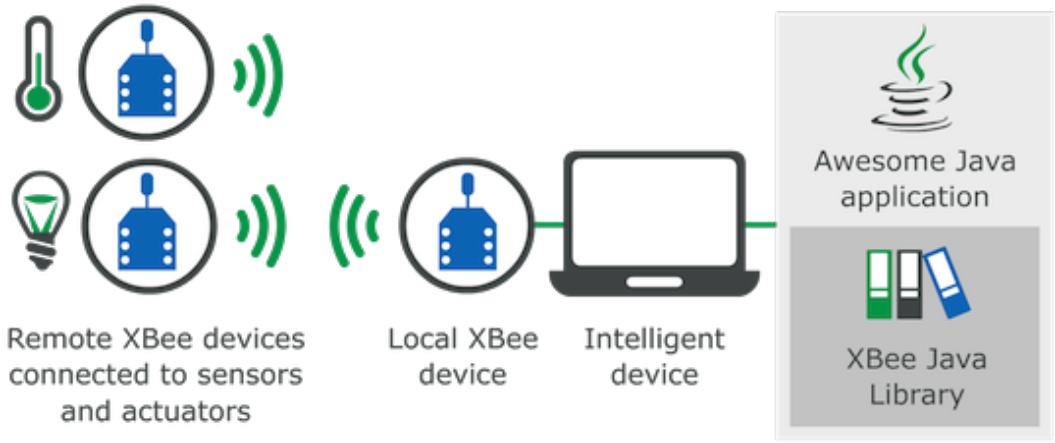


FIGURA 2.10: XBee Java Library. Fuente: Digi International Inc. [5]

La librería permite realizar lo siguiente:

- Configuración de dispositivos XBee local y remoto.
  - Configuración de parámetros comunes con *getters* y *setters*.
  - Configurar cualquier otro parámetro con métodos genéricos.
  - Ejecutar comandos AT.
  - Aplicar cambios de configuración.
  - Guardar cambios de configuración.
  - Resetear el dispositivo.
- Transmitir información a todos los dispositivos de la red o a un dispositivo específico.
- Recepción de datos de dispositivos remotos.
  - Análisis de datos.
  - *Callback* de recepción de datos.
- Recepción de estatus de la red relacionado al XBee local.
- Administración de input/Output.
  - Configurar puertos input/output.

- Configurar valores de input/output.
- Leer valores input/output.
- Recibir valores de prueba desde XBees remotos.

2.2.4. Estándar ZigBee

ZigBee es un estándar abierto para comunicación inalámbrica de bajo costo, bajo consumo de energía, alta confiabilidad y seguridad basado en el estándar IEEE 802.15.4. ZigBee. Es desarrollado por un consorcio de compañías denominada ZigBee Alliance, compuesta por más de 300 miembros permitiendo una implementación global.

Gracias a su naturaleza mesh y enrutamiento, ZigBee permite una transmisión de datos a largas distancas. Además hace posible la configuración de múltiples topologías de red permitiendo hasta 65000 nodos por red.

Una red mesh es una topología donde cada nodo de la red está conectado a otros nodos cercanos a él. Cada nodo coopera en la transmisión de información. Una topología de tipo mesh nos da los siguientes beneficios.

- **Enrutamiento.** Técnica donde el mensaje es propagado a través de una ruta buscando nodo a nodo hasta que alcance su destino final.
- **Creación de una red ad-hoc.** Proceso automático que crea una red entera de nodos sin alguna intervención humana.
- **Auto recuperación.** Es el proceso por el cual la red descubre si existe algún nodo deficiente para iniciar el proceso de reconfiguración de rutas.

En una red mesh, no importa la distancia entre nodos mientras existan nodos que hagan posible el paso de mensaje a través de ellos. Cuando un nodo quiere comunicarse con otro, la red automáticamente calcula la mejor ruta. Una red tipo mesh es muy confiable y ofrece redundancia, esto quiere decir que si un nodo deja de operar o es removido de la red, los otros nodos aún tienen la capacidad de comunicarse entre ellos, de manera directa o mediante enrutamiento. El punto débil en una red de tipo

mesh es que usan más ancho de banda para la administración, haciendo que exista menos memoria disponible para los mensajes enviados.

La mayoría de protocolos de red usan el concepto de capas para separar diferentes componentes y funciones en diferentes módulos que pueden ser ensamblados en distintas maneras. ZigBee está construido en la capa física y en la sub-capa de Control de Acceso Medio (MAC) bajo el estándar IEEE 802.15.4 que manejan operaciones de red a bajo nivel y envío y recepción de mensajes.

La especificación ZigBee define la capa de red (NWK) y el framework para la capa de aplicación. La capa de red se encarga de la estructura de red, enrutamiento y seguridad. El framework de la capa de aplicación consiste en Application Support Sub-Layer (APS), los dispositivos ZigBee (ZDO) y aplicaciones definidas por el usuario que determinan la función del dispositivo. Estas capas se pueden apreciar en la Figura 2.11.

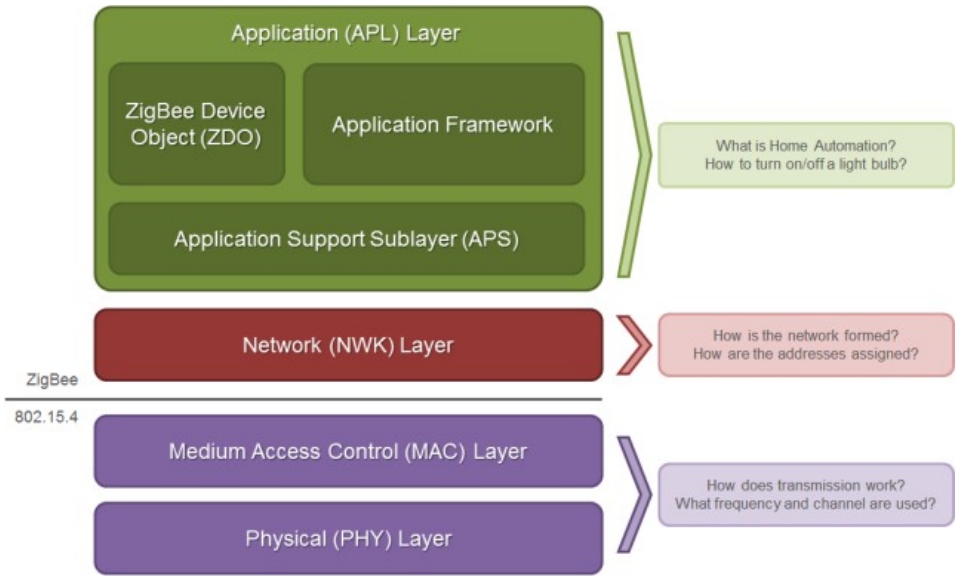


FIGURA 2.11: Capas ZigBee. Fuente: ZigBee Alliance.

ZigBee define tres tipos de dispositivos, Coordinator Node (a partir de ahora, C), Router Node (a partir de ahora, R) y End Node (a partir de ahora, E).

- **Coordinator Node**. Toda red ZigBee debe tener solo un Coordinator Node. Las características de este dispositivo son la siguientes.
  - Inicia la red, seleccionando un canal y un PAN ID.

- Distribuye direcciones, permitiendo a los Router Node y End Node adherirse a la red.
- Empaqueta datos a enviar a los End Nodes en *sleep mode*.
- Administra otras funciones que definen la red, manteniéndola segura y estable. Este tipo de nodo no puede entrar en modo *sleep* y debe tener una fuente de energía continua.

- **Router Node**. Un Router Node es un nodo ZigBee. Este dispositivo tiene las siguientes características.
  - Puede unirse a redes existentes, enviar, recibir y enrutar información. Un Router Node actúa como un mensajero para comunicación.
  - Puede almacenar paquetes de datos para nodos en estado dormido para ser enviados. Permite otros Router Node y End Nodes a la red.
  - No puede entrar en modo sleep y debe tener una fuente de energía de manera continua.
  - Pueden existir muchos Router Node en una red.

- **End Nodes**. Un End Node es básicamente una versión reducida de un router. Este nodo tiene las siguientes características:
  - Puede unirse a redes existentes, enviar y recibir información pero no puede actuar como un mensajero entre otros dispositivos.
  - No puede permitir otros dispositivos unirse a la red.
  - Utiliza menos recursos de hardware, se puede apagar por sí mismo intermitentemente, ahorrando energía temporalmente entrando a un modo *sleep*.
  - Siempre necesita un Router Node o Coordinator Node para ser su nodo padre. Este nodo padre ayuda al nodo terminar unirse a la red y almacenar mensajes para ellos cuando estén en modo *sleep*.

Una red ZigBee puede tener cualquier cantidad de End Nodes. Una red puede estar compuesta por un Coordinator Node, muchos End Nodes y ningún Router Node. Un ejemplo de una red ZigBee se muestra en la figura 2.12

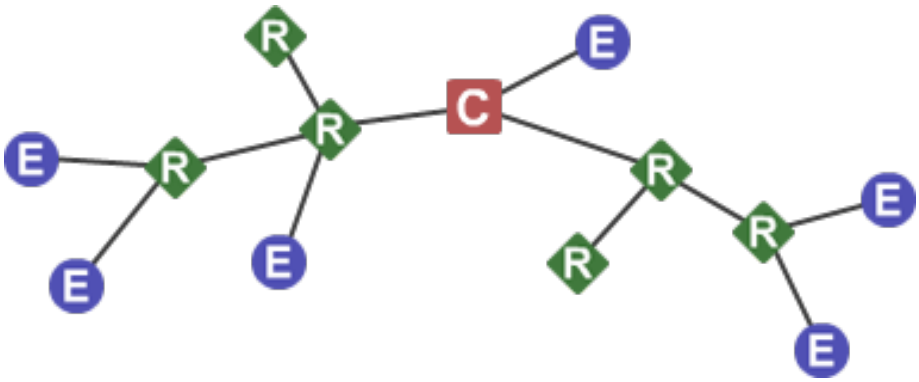


FIGURA 2.12: Ejemplo de Red ZigBee. Fuente: ZigBee Alliance.

2.2.5. Protocolo MQTT

El protocolo Message Queue Telemetry Transport (MQTT) es un protocolo ligero de mensajería que provee una red de recursos limitados con una manera simple de distribuir información. Este protocolo utiliza un patrón de comunicación de cliente/suscriptor, el cual es usado para comunicación máquina a máquina(machine to machine M2M). Fue creado por el Dr. Andy Standford-Clark de IBM y Nipper en 1999.

La forma de comunicación mediante MQTT es simple, unos dispositivos envían o publican información con una etiqueta o tópico a un servidor que funciona como broker. El broker envía la información a todos los clientes que se hayan suscrito al tópico referido.

MQTT es una buena opción para la WSN que experimentan distintas niveles de latencia debido a los límites del ancho de banda o a conexiones no confiables. Los mensajes dirigidos a un suscriptor, que se encuentra en modo offline, deben ser almacenados hasta que este vuelva a tener una conexión. Aplicaciones de mensajería en la actualidad utilizan una versión modificada del protocolo MQTT para su intercambio de información.

Una sesión en MQTT tiene 4 etapas: conexión, autenticación, comunicación y terminación.

1. **Conexión:** un cliente inicia creando una conexión TCP/IP con el broker usando el puerto 1883 por defecto o un puerto elegido por el usuario. Cuando se conecta es necesario que reconozca que probablemente el servidor continúa una sesión anterior si es que se reusa la identidad del cliente.

2. **Autenticación:** el puerto estándar a usar es el 1883 para comunicación no encriptada y el puerto 8883 para comunicación encriptada usando SSL/TLS. Durante el handshake de SSL/TLS el cliente valida el certificado del servidor para autenticar el servidor. El cliente tiene la posibilidad de enviar un certificado al broker durante el handshake que puede ser utilizado para autenticar al cliente.

3. **Comunicación:** MQTT es un protocolo ligero de comunicación porque todos los mensajes tienen una pequeña huella. Cada mensaje consiste en 2 bytes, como cabecera opcional y variable, el contenido del mensaje limitado a 256MB, y un nivel de Calidad de Servicio (QoS). Existen tres niveles de calidad de servicio, mientras más alto el nivel, más confiable la comunicación, pero también hay mayor latencia y ancho de banda necesario así los clientes pueden especificar el más alto QoS que ellos quieran recibir.

   El servicio más simple es el no admitido (QoS=0), en este nivel el publicador envía un mensaje al broker solo una vez y el broker pasa el mensaje a los suscriptores solo una vez sin asegurarse que estos hayan sido correctamente entregados, es por ello que este servicio también es llamado **dispara y olvida**. El segundo nivel es el admitido (QoS=1), en este nivel el publicador envía un mensaje al broker especificando el QoS, luego el broker se encarga de enviar este mensaje a los clientes suscritos verificando si el paquete ha sido entregado y si no ha sido, volverlo a enviar hasta asegurar su entrega. Sin embargo esto podría generar multiples copias del mismo mensaje, por ello es que este servicio es llamado **al menos uno**. El tercer nivel (QoS=2), también conocido como **exactamente uno** entrega dos tipos de paquetes que, a pesar de los números

de intentos realizados para enviar el mensaje, este solo se enviará una vez. Estos servicios están explicados de manera gráfica en la Figura 2.13.

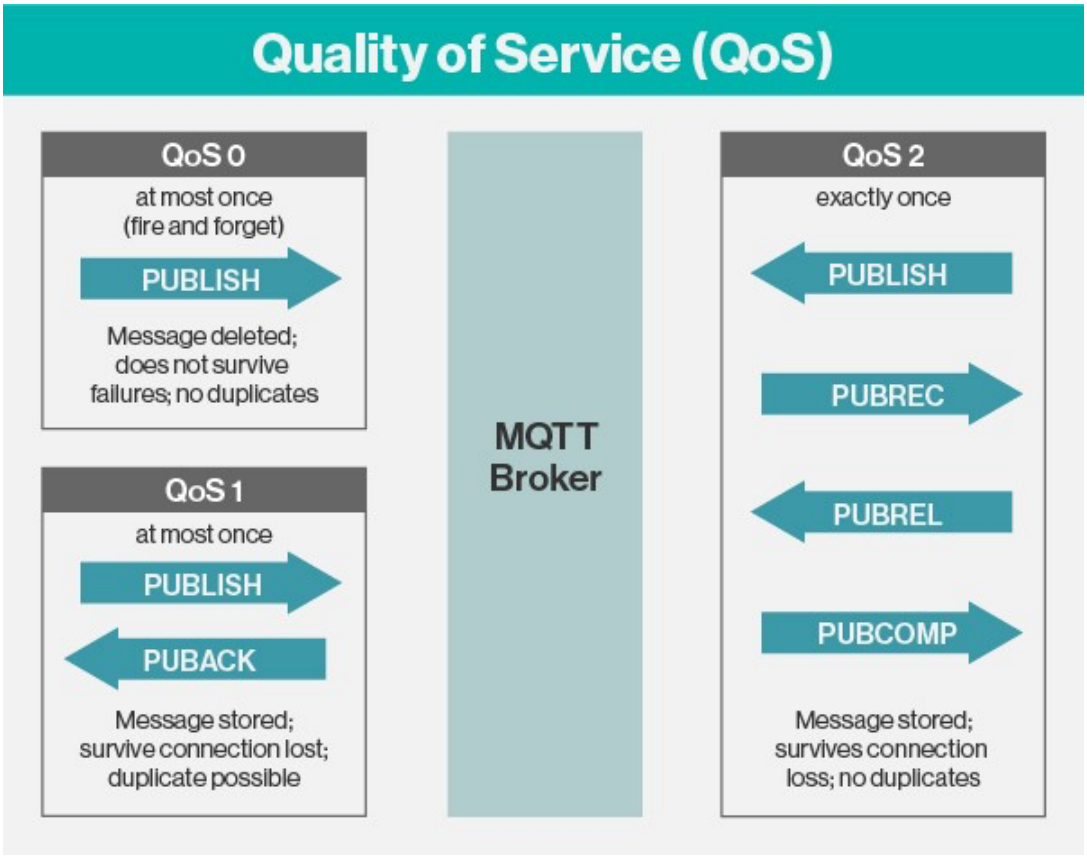


FIGURA 2.13: Tipos de QoS. Fuente: TechTarget.

Durante la fase de comunicación, un cliente puede realizar operaciones como publicar, suscribir, desuscribir y operaciones ping. Una operación de tipo publicar envía un bloque binario de información a un tópico definido por el publicador, MQTT soporta hasta 256MB de información. Las suscripciones son hechas usando el paquete binario *SUBSCRIBE/SUBACK*. Desuscribirse usa un paquete similar que es *UBSUBSCRIBE/UNSUBACK*.

Los tópicos de los mensajes enviados tienen una estructura de árbol usando un delimitador diagonal(/). Un cliente puede suscribirse o desuscribirse a múltiples ramas mediante el uso de caracteres especiales como '+', comodín de nivel simple y '#', comodin de multinivel. Un caracter especial de tópico es el '$' que excluye un topico de cualquier comodin de suscripciones. Usualmente el '$' es usado para transportar mensajes especifícos del sistema.

La otra operación que un cliente puede realizar es hacer un ping al broker usando un paquete *PINGREQ/PINGRESP* que su única función es mantener una conexión estable y que no haya sido cerrada por una puerta de enlace(gateway). Todos los tipos de mensajes en MQTT y su descripción se muestran en la Figura 2.14.

MQTT Message	Description
CONNECT	Client request to connect to server
CONNACK	Connect acknowledgement
PUBLISH	Publish message
PUBACK	Publish acknowledgement
PUBREC	Publish received
PUBREL	Publish release
PUBCOMP	Publish complete
SUBSCRIBE	Client subscribe request
SUBACK	Subscribe acknowledgement
UNSUBSCRIBE	Unsubscribe request
UNSUBACK	Unsubscribe acknowledgement
PINGREQ	PING request
PINGRESP	PING response
DISCONNECT	Client is disconnecting

FIGURA 2.14: Tipos de mensajes MQTT. Fuente: TechTarget.

4. **Terminación**, cuando un cliente desea terminar una sesión MQTT, este envía un mensaje *DISCONNECT* al broker y luego cierra la conexión. Esto se denomina **elegante cierre de sesión** porque el cliente es capaz de reconectarse mediante

el envío de su identificador y retomar su actividad de cuando se desconectó. En caso el cliente por alguna razón se desconecta con el broker sin enviar el mensaje *DISCONNECT*, el broker envía a los clientes suscriptores un mensaje del publicador, este mensaje se llama **último testamento** que provee información de qué hacer si el publicador se desconecta inesperadamente.

A pesar de su facilidad de manejo, MQTT puede tener muchas desventajas como el hecho de que su estructura de árbol en los tópicos puede formar fácilmente un árbol tan grande, en otras palabras, no sería escalable pues mientras el árbol de tópicos crece, la complejidad también. Además de ese problema, la autenticación en MQTT es en texto plano y si se quiere una versión segura, se debe usar SSL/TLS que no es un protocolo ligero. Otro problema es que un cliente recibe un mensaje sin saber quién lo ha enviado a menos que ese dato se incluya en el mensaje.

Aún con estos problemas, MQTT está jugando un rol muy importante en IoT facilitando cosas como control de inventario, telemática automotriz y monitoreo de recursos. Actualmente el protocolo ahora acepta WebSockets, otro protocolo que permite la comunicación en doble sentido entre clientes y brokers en tiempo real.

2.3. Métodos de ahorro de energía en una red de sensores

Las diferentes propuestas para reducir el consumo energético intentan abordar distintos enfoques. Algunos se concentran en el reducir el número de transmisiones al nodo central mientras que otros proponen fuentes de energía alternativas para incrementar el tiempo de vida. No existe una recomendación estándar ya que cada alternativa debe tomarse en cuenta según el entorno en el que se desarrolla.

2.3.1. Agregación de datos

El enfoque de agregación de datos (*data aggregation* en inglés) propone minimizar el número de paquetes a enviar entre nodos, eliminando paquetes redundantes. Los diversos algoritmos descritos en [15], buscan encontrar una selección óptima de nodos donde aplicar *data aggregation*. Un subconcepto de *data aggregation* es *data fussion* que

consiste en reducir el tamaño del paquete de transmisión. El trabajo en [6] presenta un mecanismo llamado diferenciación de datos locales (LDD, por sus siglas en inglés), que se describe en la Figura 2.15, consiste en enviar únicamente la diferencia entre las nuevas lecturas de los sensores y el promedio de los datos ya enviados.

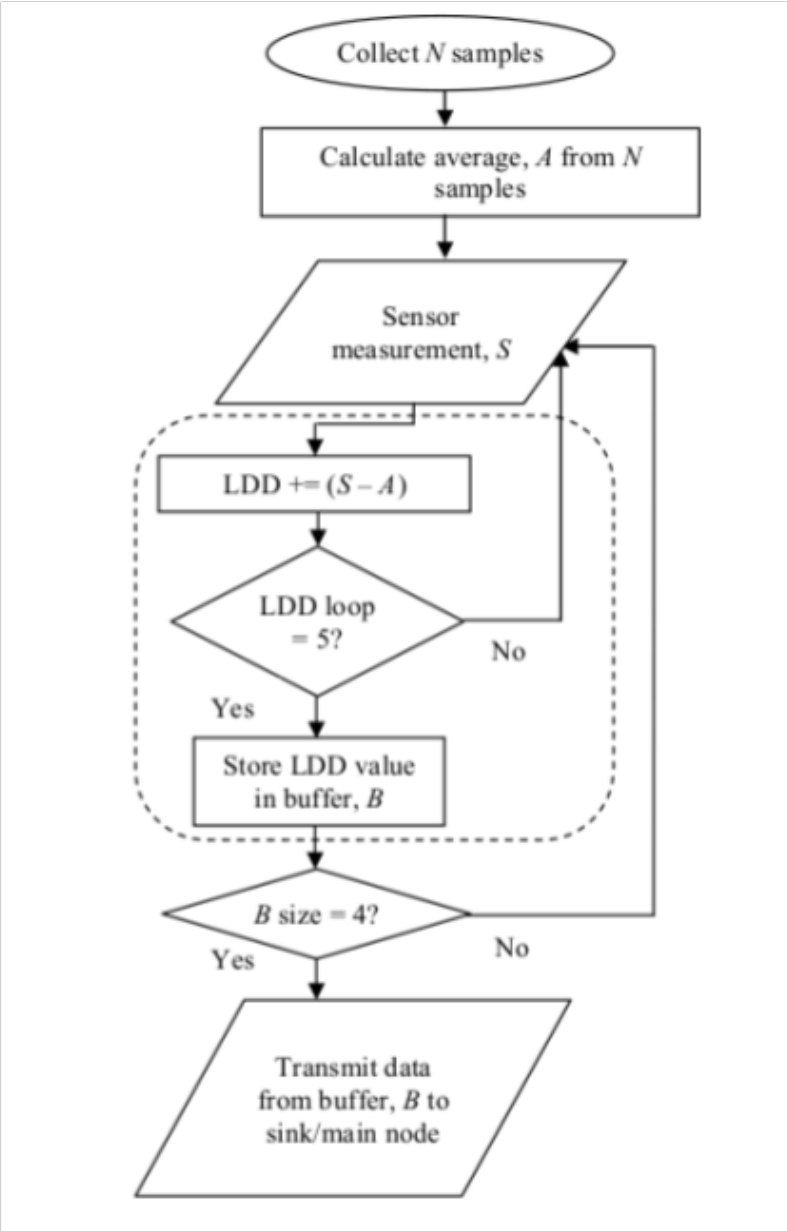


FIGURA 2.15: Diagrama de flujo del LDD. Fuente: ICCOINS [6]

2.3.2. Fuentes de energía alternativas

Cuando una WSN es desplegado en una ciudad o campo agrícola, se deberá realizar un estudio previo determina cuál será la fuente de energía de cada uno de los nodos que comprende esta red. Algunas alternativas consisten en extender la alimentación de los semáforos [16], otras soluciones optan por energía solar [7]. En el segundo caso, debido a que no se puede obtener una alimentación continua de energía, la propuesta es que la energía solar cargue una batería y esta regule la cantidad de energía que suministra al nodo. La Figura 2.16 muestra el flujo de energía obtenido por radiación solar, pasando por un recolector, se almacena en la batería para proveer de energía constantemente al Wise-Mote. El Wise-Mote, mostrado en la Figura 2.17, está compuesto por una fuente de energía, sensores, un microcontrolador ARM, actuadores y dispositivos de comunicación XBee.

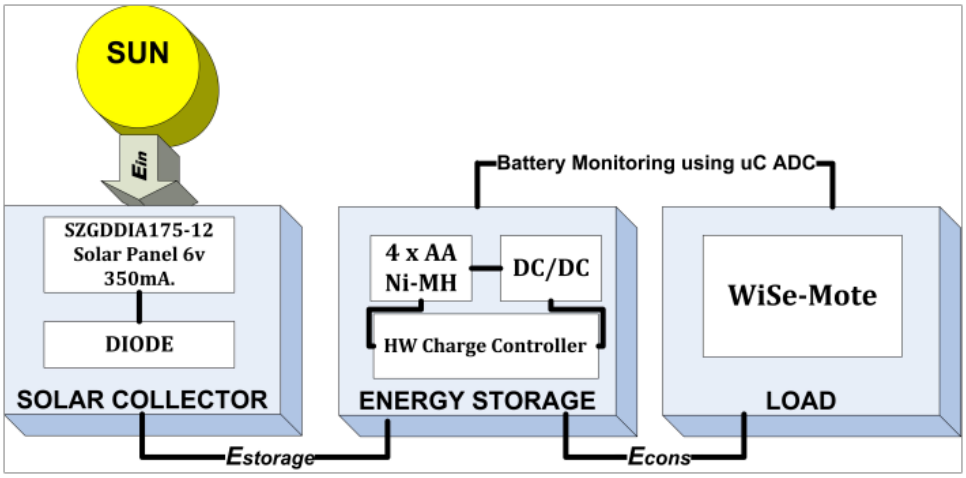


FIGURA 2.16: Modelo de un sistema alimentado por energía solar. Fuente: ICCOINS [7]

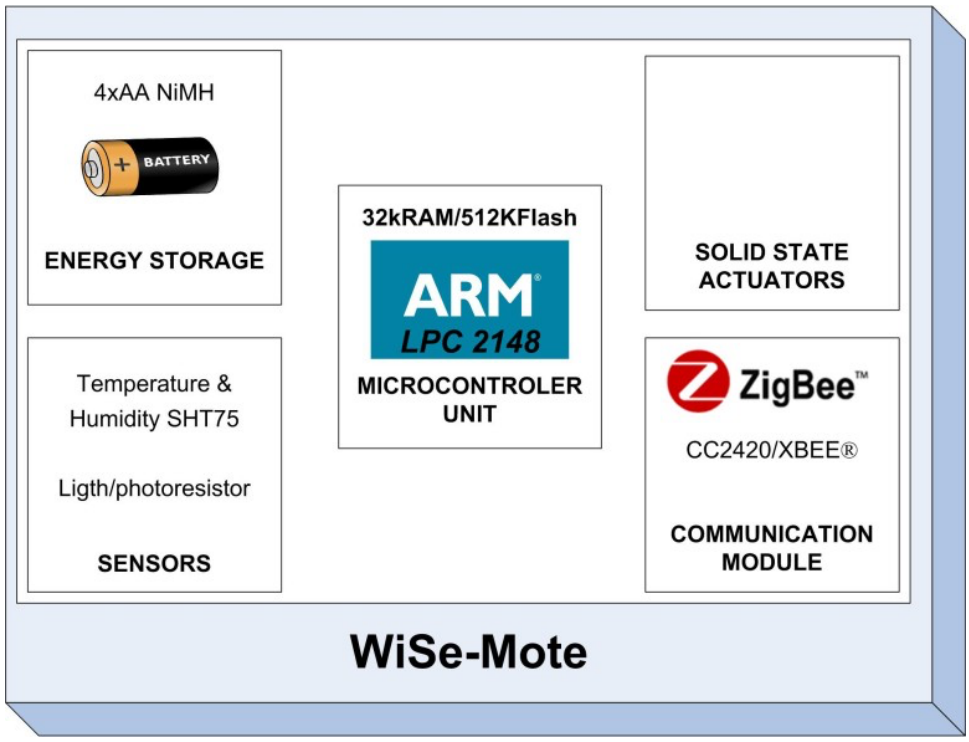


FIGURA 2.17: Modelo de un nodo en una WSN. Fuente: Sensors [7]

2.3.3. Control de componentes

Cada vez que un nodo recibe una cantidad determinada de energía, esta energía se distribuye para diferentes tareas como medición de parámetros, procesamiento, control o comunicación de la información [8]. Esta última tarea es la que más energía consume en cada nodo; por ese motivo es que en muchos trabajos se proponen alternativas de protocolos de enrutamiento como se muestra en la Figura 2.18, por ejemplo, perfiles de *sleep/wake* [17]. Otros proponen un menor envío de paquetes como en *data aggregation* [15]. Algunas alternativas proponen apagar el emisor mientras no tenga información para enviar, si este proceso se realiza sin un control adecuado luego de un determinado periodo resulta con un mayor consumo a que si se hubiera dejado encendido todo ese tiempo [18].

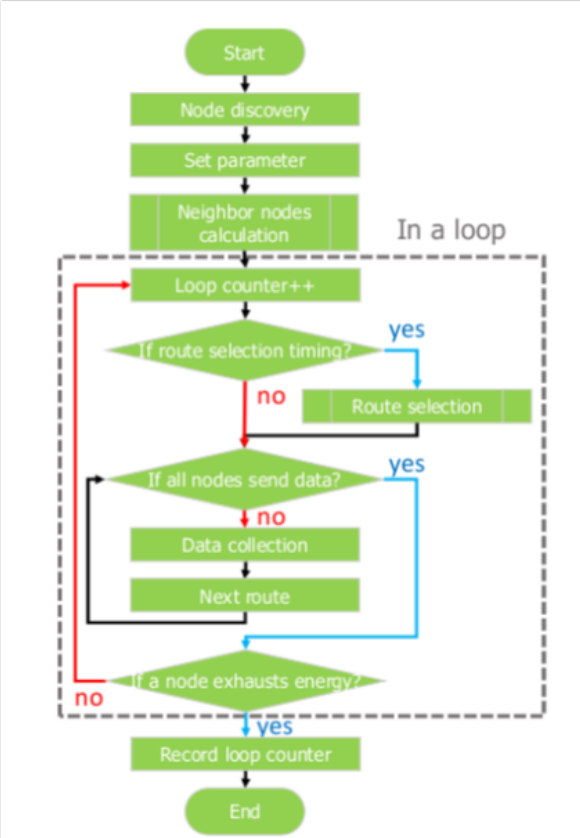


Fig. 7. Simulation flow chart.

FIGURA 2.18: Algoritmo de selección de enrutamiento ECORS. Fuente: INDIN [8]

2.4. Conclusiones

Después de hacer un análisis de los distintos métodos que existen para maximizar el tiempo de vida de una WSN, es posible observar que casi todas las investigaciones se enfocan en una optimización de rutas [15], tamaño de paquete [6], fuentes alternativas de energía [7, 16] o control de diversos componentes del nodo [8, 17].

Dadas estas observaciones, en el presente trabajo propone un control de energía a nivel de roles de acuerdo al estándar ZigBee. Este control asignará el rol que le corresponde a cada nodo, esto sin comprometer la calidad del servicio ni la comunicación de nodos asociados a este, además de garantizar el menor consumo de energía.

Capítulo 3

Diseño del estudio experimental

En el presente capítulo se describirán los procedimientos que justifican el desarrollo de este enfoque para maximizar el tiempo de vida de una WSN haciendo un juego de cambio de roles correspondientes al estándar ZigBee. Primero se define la topología base con la que se trabajó. Posteriormente, se describe cada tipo de nodo que existe en la WSN, sus componentes y formato de datos que transmiten y reciben.

Así, en este capítulo muestra el procedimiento realizado para la configuración inicial de los dispositivos XBee haciendo uso del software de configuración XCTU [19]. Una vez configurada la red, se muestran los resultados de la medición del consumo de energía de cada nodo actuando como *router* o *end-device* con dos tipos de batería. Una vez obtenida información del consumo de energía por cada rol, se determina el número mínimo empírico de Router Node de tal manera que el consumo total de energía de la WSN es mínimo. Por último, se describe el algoritmo con el que se hace control de los roles de la red y mantener un consumo mínimo de energía.

3.1. Topología

De acuerdo a la Figura 3.1, se observa una topología de tres niveles. Aún el objetivo sea reducir el consumo de energía, la conectividad de cada nodo de la red debe estar asegurado en una WSN. Dicho esto, este trabajo propone alternar el rol únicamente para los nodos en el segundo nivel manteniendo solo un Router Node que sirve como puente entre los nodos del tercer nivel con el Coordinator Node del primer nivel.

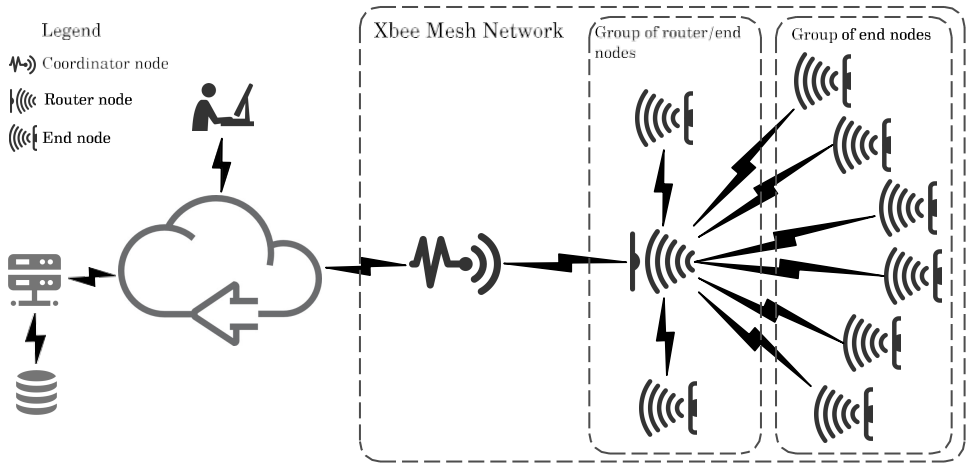


FIGURA 3.1: Topología de la red. Fuente: Journal of Sensors [2]

En la Figura 3.1 se puede observar que el Router Node tiene 8 End Nodes conectados a él, esta topología será utilizada para el análisis del consumo de energía con el rol de *router* y *end-node*. Además, en la Figura puede observarse los tres nodos en el segundo nivel que son los candidatos a cambiar de rol durante el tiempo de actividad de la red. En el siguiente apartado, se justificará el uso de un único Router Node en el segundo nivel que nos garantice la conectividad de todos los nodos de la WSN.

3.2. Coordinator Node

Dentro de la arquitectura ZigBee, para poder iniciar una red es necesario e imprescindible tener un XBee configurado con rol como Coordinator Node, que cumplirá las funciones descritas en la sección 2.2.4 del capítulo anterior. Mediante un adaptador USB-XBee se conecta al Raspberry Pi por el puerto USB como se muestra en la Figura 3.2. Así, cuando un paquete de datos llega al XBee, este puede manejarse mediante una librería de lectura serial en el mismo Raspberry Pi.

3.2.1. Especificaciones de Software

El Coordinator Node, un Raspberry Pi 3, está conectado al XBee de manera serial. El Raspberry tiene un sistema operativo Linux Raspbian con un script en Python ejecutándose leyendo, de esta manera, los valores que ingresan por el puerto USB de

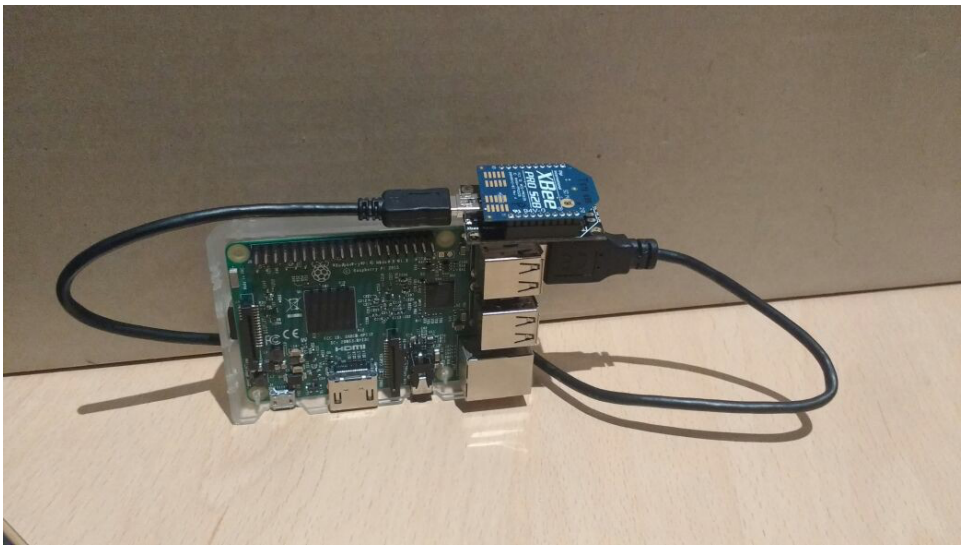


FIGURA 3.2: Nodo Master de la Red. Fuente: Elaboración propia

manera serial. Una vez un elemento es recibido por el script, este se encarga de hacer la publicación MQTT con el tópico correspondiente al nodo y tipo de dato enviado.

Se ha establecido Python, como lenguaje de programación, por su facilidad en la implementación sobre la lectura de datos desde el serial USB y, además, por su característica de poder ejecutar comandos del sistema haciendo un llamado a la librería/función os.system. El código del script utilizado es el siguiente:

```
import serial
import time
import os
import string

xbee = serial.Serial (
  port = '/dev/ttyUSB1',
  baudrate = 9600,
  timeout = 1 )

def transformData(s):
        fecharpi = time.strftime("%Y-%m-%d")
        horarpi = time.strftime("%H:%M:%S")
        hostrpi = "beagons.uni.edu.pe:8080"
        t = string.Template(s)
        return t.substitute(host=hostrpi,fecha=fecharpi, hora=horarpi)
```

```
17
18 while  1:
19   x = xbee.readline()
20   print x
21   os.system(transformData(x))
22   time.sleep(1)
```

El método transformData es usado para etiquetar con fecha y hora al dato recibido por el XBee e indicar el host del broker MQTT. Se ha resaltado el uso de la librería os de Python, ya que una vez que se tiene el dato, este ha pasado por el método transformData(s). Este string tiene una estructura de un comando de GNU/Linux que publica el dato indicado.

3.3. Router Node y End Node

Una vez definido el Coordinator Node, pueden agregarse los nodos que se estime oportuno a la red si es configurado el mismo PAN ID en el XBee (esa configuración se verá en la Sección 3.3.1). En cada uno de estos nodos está un módulo de sensores que son controlados por un arduino FIO, que es una tarjeta con un microcontrolador basado en ATmega328P que se alimenta con 3.3V. Esta tarjeta se encarga de la lectura y de darle el formato adecuado a los datos enviados, darle un formato JSON en este caso. Se hace uso de un modo *sleep* en el microcontrolador para ahorrar el consumo de energía. Cada nodo tiene los siguientes sensores:

- **BMP180:** es un sensor (ver Figura 3.3) con la capacidad de dar parámetros como temperatura, presión y altura.

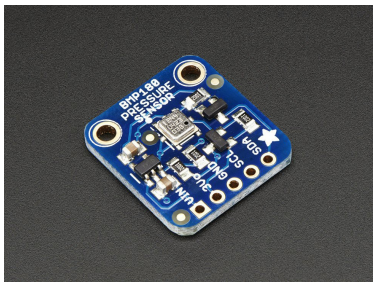


FIGURA 3.3: Sensor BMP180. Fuente: Adafruit Inc.

- **SHARP:** es un sensor (ver Figura 3.4) que envía información de la cercanía de otros objetos al nodo.

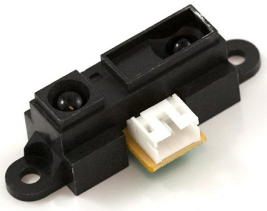


FIGURA 3.4: Sensor SHARP. Fuente: Adafruit Inc.

- **MQ131:** este sensor (ver Figura 3.5) se encarga de la lectura de información del nivel de ozono en el entorno del nodo.

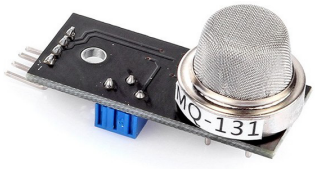


FIGURA 3.5: Sensor MQ131. Fuente: Adafruit Inc.

- **MQ135:** este sensor (ver Figura 3.6) lee valores de calidad de aire, como el amoniaco, sulfuro, benceno o humo.
- **MQ7:** este es un sensor (ver Figura 3.7) que procede a la lectura de los monóxido de carbono que hay en el ambiente.
- **Detector de sonido:** este sensor (ver Figura 3.8) lee el nivel de ruido alrededor del nodo.

Así mismo, para proceder a la integración de todos los dispositivos de censado se ha realizado una placa electrónica. El circuito encargado de manejar todos estos sensores es el mostrado en la Figura 3.9

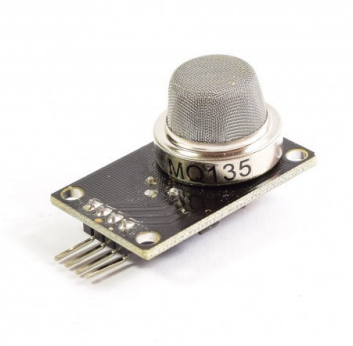


FIGURA 3.6: Sensor MQ135. Fuente: Adafruit Inc.

FIGURA 3.7: Sensor MQ7. Fuente: SparkFun Electronics.

Todos estos sensores establecidos en la placa electrónica es controlada mediante el microcontrolador Arduino FIO (ver Figura 3.10). Además, como puede verse en esta Figura se le ha añadido la parte mecánica, es decir, un recubrimiento para su protección, en filamento PLA mediante una impresora 3D. Así, todo el conjunto de sensores quedo empaquetado en el módulo mostrado en la Figura 3.10.

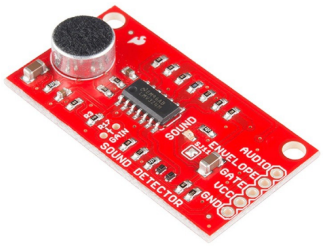


FIGURA 3.8: Detector de sonido. Fuente: SparkFun Electronics.

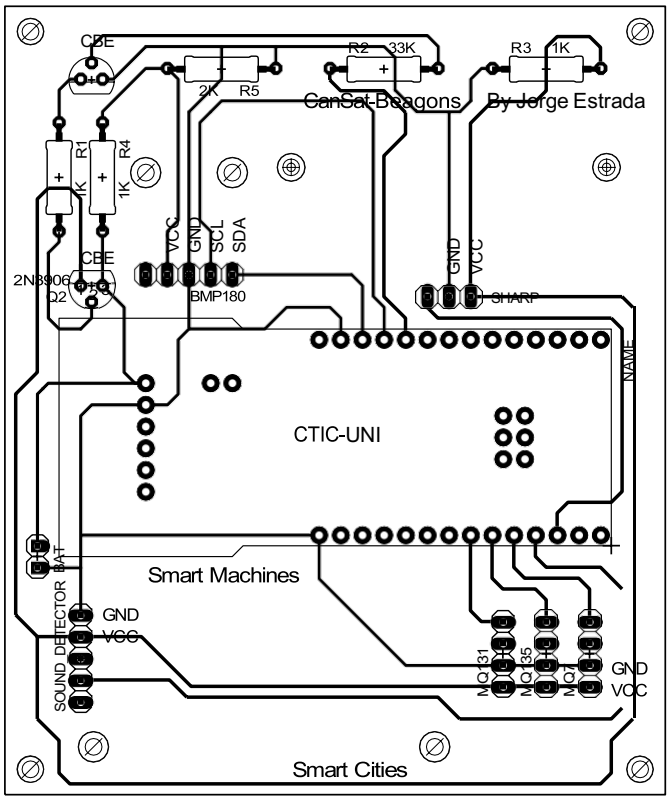


FIGURA 3.9: Circuito del nodo. Fuente: Elaboración propia

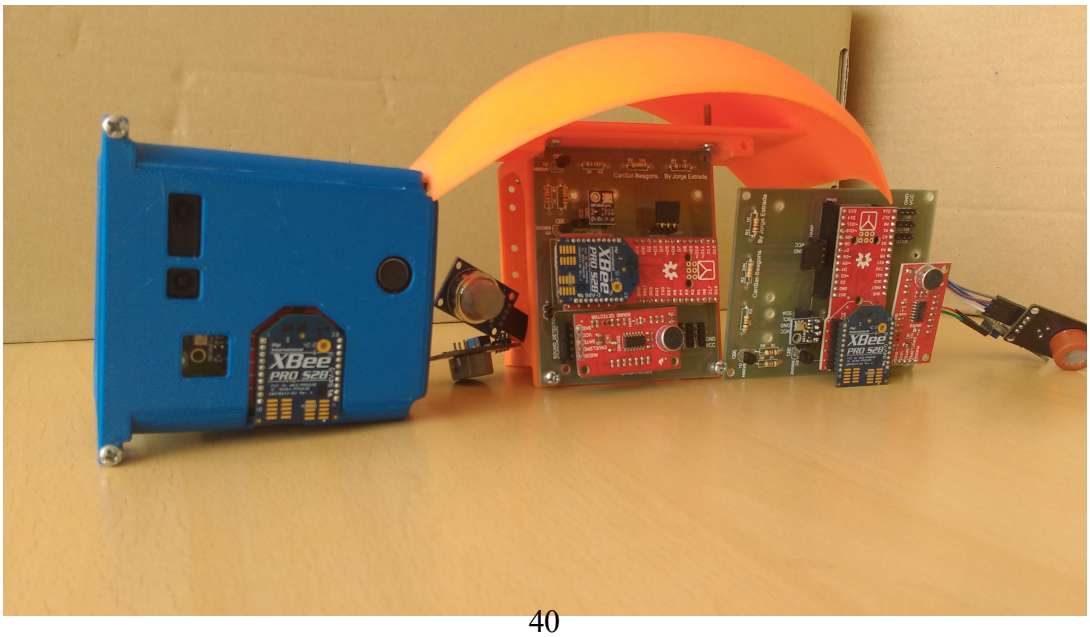


FIGURA 3.10: Router Node y End Node. Fuente: Elaboración propia

3.3.1. Formato de envío

Para enviar la información leída a través de la red al Coordinator Node, se debe definir un formato que el Coordinator Node lo conozca. Además, este procedimiento sirve para que al momento de desarrollar aplicaciones que se alimenten de estos sensores tengan bien definido el formato de datos.

Por tanto, cada nodo genera un comando a ejecutar, dentro del cual está definido el formato JSON a enviar al Broker. El formato JSON es el siguiente:

```
{"id_node": "idnode",
  "id_sensor": "idsensor",
  "value": value,
  "fecha": "fecha",
  "hora": "hora"
}
```

```
#include <Wire.h>
#include <Adafruit_BMP085.h>
#include <stdarg.h>
#include <string.h>
Adafruit_BMP085 bmp;
int sensorPin1 = A0;      //input pin
int sensorPin2 = A1;      //input pin
int ledPin = 13;        // pin LED
int sensorValue1 = 0;
int sensorValue2 = 0;
void setup() {
  Serial.begin(9600);
  pinMode(ledPin, OUTPUT);
  if (!bmp.begin()) {
    Serial.println("BMP sensor not found!");
    while (1) {
    }
  }
}

String jsonFormat(float value, String sensorType){
  String idcansat = "C1001";
```

```
23   String idsensor="S50000";
24   if(sensorType.equals("temperature")){
25       idsensor="S50001";
26     }
27   if(sensorType.equals("presure")){
28       idsensor="S51002";
29     }
30   if(sensorType.equals("altitute")){
31       idsensor="S55002";
32     }
33
34   if(sensorType.equals("ammonia")){
35       idsensor="S54002";
36     }
37
38   if(sensorType.equals("monoxide")){
39       idsensor="S52002";
40     }
41   if(sensorType.equals("dioxide")){
42       idsensor="S53002";
43     }
44
45   String sensorDate = "2016-08-08";
46   String sensorTime = "10:41:55";
47   String json="";
48   json = "{\"id_cansat\" : \""+idcansat+"\",";
49   json+= "\"id_sensor\" : \""+idsensor+"\",";
50   json+= "\"value\" : "+String(value)+",";
51   json+="\"date\" : \"sensorDate\",";
52   json+="\"time\" : \"sensorTime\"}";
53   return json;
54 }
55
56 void transmit(String topic, float value){
57     Serial.print(jsonFormat(value, topic));
58 }
59
```

```
60 void loop() {
61     float T=bmp.readTemperature();
62     float pressure = bmp.readPressure();
63     float altitude = bmp.readAltitude(101500);
64
65     transmit("temperatura", T);
66     transmit("presion", pressure);
67     transmit("altura", altitude);
68
69     delay(500);
70 }
```

3.4. Configuración XBee

Como se ha mencionado en capítulos anteriores, el protocolo de comunicación encargado de transmitir información entre nodos es el estándar IEEE 802.15.4, ZigBee.

3.4.1. XCTU

XCTU es el software gratuito que provee Digi, la empresa fabricante de XBee, para configurar los parámetros en cada dispositivo. Este es un software con una interfaz amigable que ayuda a configurar de manera muy fácil un dispositivo de DIGI. Esta no es la única forma de configurar los parámetros en un XBee, ya que mediante comandos AT pueden establecerse valores que se estimen oportunos. En este trabajo se utilizó el XCTU para configurar el Coordinator Node y los Router y End Node.

Algunas de las características que ofrece XCTU son las siguientes:

- Manejar módulos locales.
- Manejar módulos remotos si se encuentran en la misma red.
- Tiene un generador de paquetes para hacer pruebas.
- Hace una prueba de rango.
- Tiene la capacidad de actualizar el firmware.

- Habilita una consola para el uso de comandos AT.
- Provee una representación gráfica de la red.

La interfaz principal del XCTU se puede observar en la Figura 3.11. En esta figura podemos observar del lado izquierdo la lista de dispositivos conectados a la computadora que tiene instalada el XCTU. Es posible ver también los dispositivos que corresponden a la misma red. Del lado derecho se tienen las opciones de lectura y escritura de parámetros y configuración del firmware.

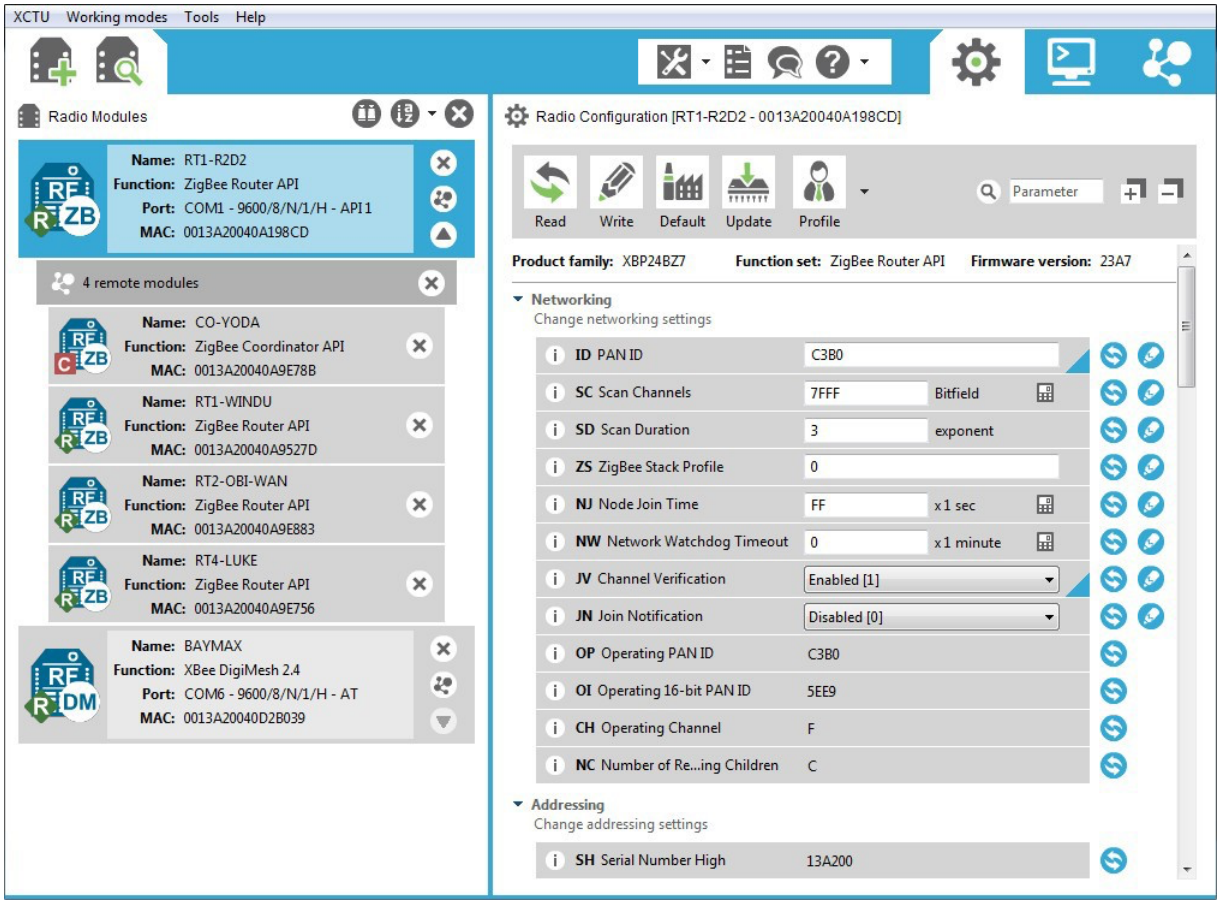


FIGURA 3.11: Layout XCTU. Fuente: Digi International Inc.

3.4.2. Configuración módulos XBee

Cada XBee tiene diferentes parámetros de configuración, para dar iniciada la red. Así, por tanto, es necesario configurar los dispositivos XBee con el mismo PAN ID. Así mismo, para el caso de uso del algoritmo de control del nivel de energía, es necesario

que todos los dispositivos estén en modo API. En este sentido, a continuación se exponen los parámetros más importantes:

- **Coordinator Enable (CE)**: es un parámetro que nos dice si el módulo es un coordinador o no.

- **Sleep Mode (SM)**: es el parámetro que determina si un módulo es un enrutador o un dispositivo terminal.

El Coordinator Node y los Router Node no pueden tener el estado *sleep*, así que el valor a establecer para este tipo de nodos en SM debe ser siempre 0 (desactivado).

Bajo este contexto, la configuración de los XBee que componen la red de sensores se muestra en la Tabla 3.1.

Rol	Configuración	Descripción
Coordinador	CE = 1, SM=0	Un módulo es un Coordinator Node si CE es igual a 1, el valor de SM no puede ser diferente a cero
Router Node	CE=0, SM=0	Un módulo XBee es un enrutador si CE es igual a 0 y el SM está desactivado
End Node	CE=0, SM=1	Un módulo XBee es dispositivo terminal si tiene algún SM habilitado.

TABLA 3.1: Configuración parámetros XBee. Fuente: Digi International Inc.

Una vez configurado cada dispositivo, debemos establecer los parámetros necesarios para que estos pertenezcan a una misma red ZigBee.

Cuando un XBee se enciende, realiza diferentes operaciones dependiendo del rol asignado. A continuación, se explican estas operaciones para cada tipo de nodo en la red.

- **Coordinador:** El coordinador es el único capaz de iniciar una red ZigBee, cada red debe tener uno y solo un coordinador. Es responsable de seleccionar un canal operativo, PAN ID, políticas de seguridad, y un perfil para la red. Para asegurar que el coordinador comience en un buen canal y PAN ID no utilizado, hace una serie de escaneo para encontrar actividad de radio-frecuencia

en diferentes canales, esto se llama escaneo de energía, y para descubrir PAN cercanos operables se llama escaneo activo.

Los siguientes comandos controlan el proceso de formación de la red.

- PAN ID (*ID*). Determina el PAN ID. Si tiene el valor de 0 (por defecto), el XBee selecciona un PAN ID aleatorio.
- Escaneo de canales (*Scan Channels SC*). Determina la máscara de bits de los canales escaneados y que el coordinador utiliza para formar la red.
- Duración de escaneo (*Scan Duration SD*). Establece la duración del proceso de escaneo, determina qué tanto el coordinador hace esa actividad en un canal dado.

Una vez que el coordinador ha iniciado la red, puede permitir dispositivos unirse a esta, con un máximo de hasta 20 dispositivos en la misma red, entre enrutadores y dispositivos terminales. Sin embargo, este número se puede configurar cuántos se pueden unir o si los deseamos unir solo por un corto tiempo.

- **Enrutador:** Estos dispositivos deben descubrir y unirse a redes ZigBee válidas, antes que puedan participar en ella. Para descubrir redes cercanas, un enrutador realiza un escaneo activo al igual que el coordinador. Cuando un enrutador se une a una red, recibe una dirección de 16 bits del dispositivo que le ha permitido la conexión. Una vez que el enrutador se ha unido a la red ZigBee, permanece conectada en el mismo canal y PAN ID tanto tiempo mientras no sea forzada de dejar la red. Una vez que el enrutador se ha unido a la red, este puede permitir unir hasta 20 dispositivos con la configuración de Node Join Time (NJ). Un enrutador también direcciona paquetes y mantiene comunicación con otros dispositivos de la red.
- **Dispositivo Terminal:** De manera similar a los Router Node, un End Node debe unirse a una red ZigBee válida antes de que pueda participar en ella. Los dispositivos terminales también pueden descubrir redes a las que se puede unir haciendo un escaneo activo, una vez unidos también reciben una dirección

de 16 bits. Ya que un dispositivo terminal puede entrar en estado dormido, el nodo que permitió su acceso a la red es el responsable de almacenar en un buffer sus mensajes hasta que el End Node esté disponible para recibir esos mensajes. Entonces ese nodo es llamado nodo padre. Cuando el End Node inicia su estado de encendido hace una petición a su padre por mensajes que le llegaron mientras estuvo dormido. Tanto coordinadores como enrutadores pueden permitir a dispositivos terminales unirse a su red; sin embargo, cada dispositivo tiene un límite de nodos hijo que pueden aceptar. Este procedimiento se configura en el parámetro *Number of Remaining Children* (NC). Luego de que un dispositivo terminal se ha unido a una red ZigBee, este se puede comunicar con otros dispositivos en esa red. Ya que los dispositivos terminales tienen la predisposición de ahorrar batería, ellos no pueden aceptar otros dispositivos unirse a la red.

Una vez configurados los nodos de la red, y haciendo uso del XCTU, es posible visualizar las conexiones de cada nodo en la red con información adicional como el perfil del nodo (Coordinator Node, Router Node o End Node), la dirección de 16 bits y la relación entre nodos padre/hijo. Esto se muestra en la Figura 3.12, una red con el Router Node conectado directamente al Coordinator Node y con 5 End Nodes conectados al Router Node.

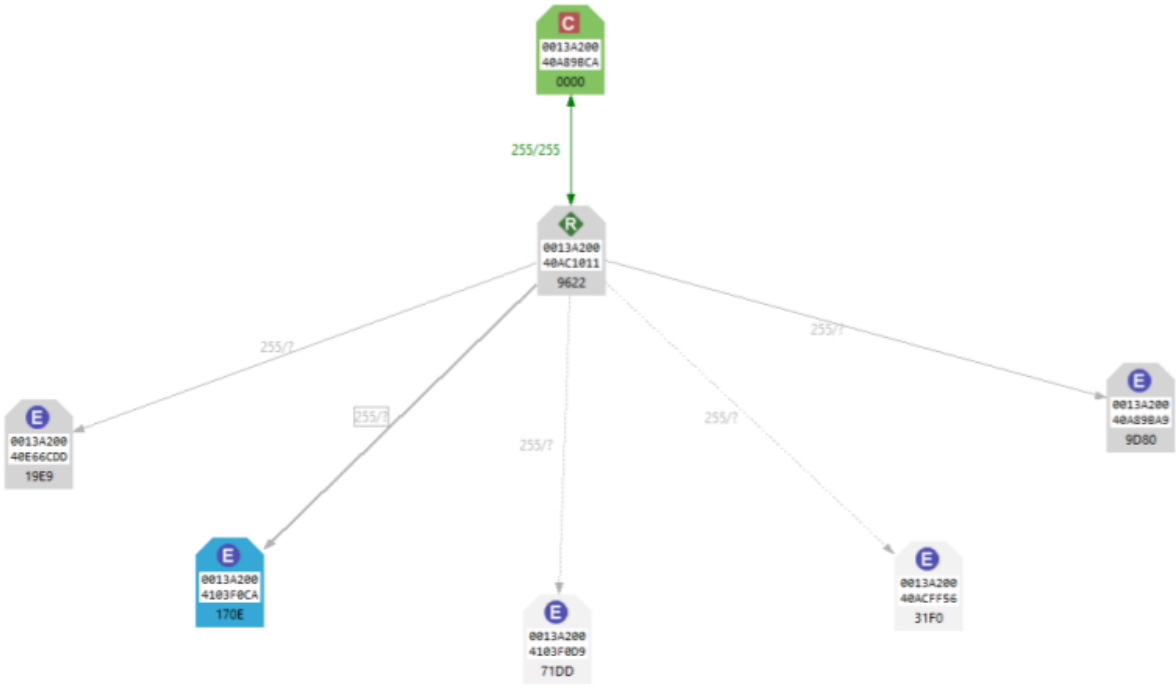


FIGURA 3.12: Red ZigBee en XCTU. Fuente: Elaboración propia

Capítulo 4

Proceso de optimización de consumo de energía

En el presente capítulo se describirán los procedimientos que justifican el desarrollo de este enfoque para maximizar el tiempo de vida de una WSN haciendo un juego de cambio de roles correspondientes al estándar ZigBee. Después de definida la topología en el capítulo anterior, se analiza cada tipo de nodo que existe en la WSN, sus componentes y el formato de datos que transmiten y reciben.

4.1. Medición del Consumo de energía

Antes de proponer el algoritmo de cambio de roles, se debe mostrar que los Router Nodes consumen más energía que los End Nodes. El experimento consiste en medir el nivel de batería en un mismo nodo, cambiando el rol entre Router y End Node luego de un determinado tiempo. Este experimento ha sido realizado 10 veces con un comportamiento similar para consumo de energía.

Cabe mencionar también que se utilizaron dos tipos de batería, una batería de ácido de plomo y una batería de Litio, las especificaciones de estas componentes se describen en la Tabla 4.1

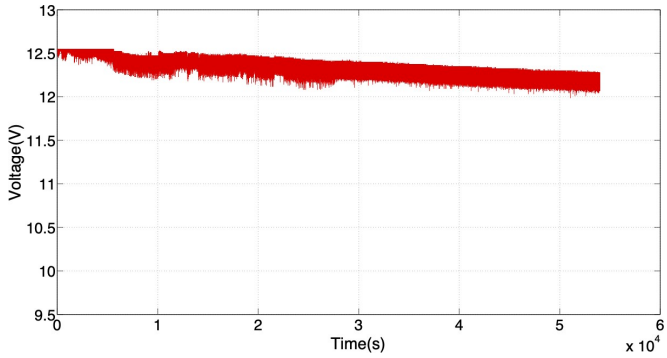
4.1.1. Caso 1: Batería de ácido de plomo

La Figura 4.1 representa el consumo de energía del nodo para cada rol en una batería de ácido de plomo. La Figura 4.1(a) muestra el consumo de energía para un

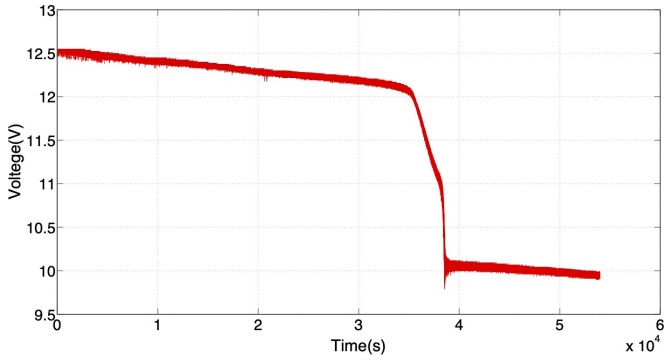
Especificación	Batería de ácido de plomo	Batería de Litio
Voltaje nominal	12V	11.1V
Voltaje por celda	-	3.7V
Número de celdas	-	3
Parámetro C	-	20C
Capacidad	4Ah	6400mAh
Longitud	8.90cm	13.50cm
Alto	10.00cm	4.23cm
Ancho	6.90cm	4.00
Peso	1.47Kg	0.5Kg

TABLA 4.1: Especificaciones técnicas de las baterías utilizadas en el experimento. Fuente: Elaboración propia.

End Node y la Figura 4.1(b) el consumo para un Router Node.



(a) Consumo de energía como End Node.

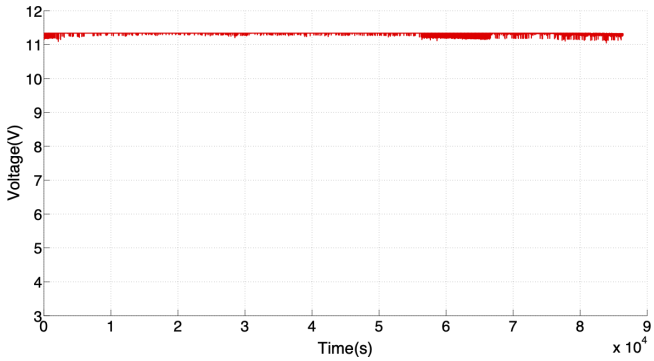


(b) Consumo de energía como Router Node.

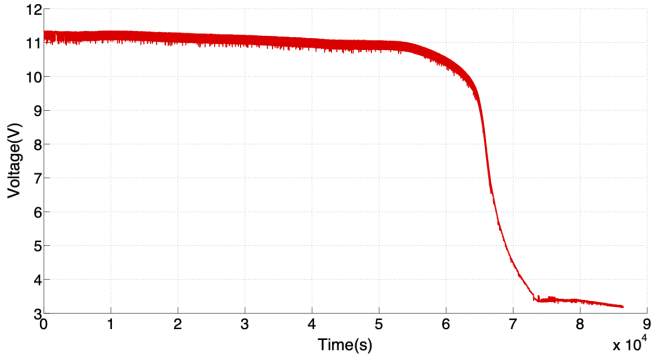
FIGURA 4.1: Consumo de energía con una batería de ácido de plomo para los dos tipos roles. Es posible observar un consumo constante de la energía en el End Node por un tiempo aproximado de 1 hora y 30 minutos. Mientras que en el Router Node, antes de la hora de uso se muestra un decaimiento significativo en su energía. Fuente: Elaboración propia.

4.1.2. Caso 2: Batería de Litio

La Figura 4.2 representa el consumo de energía del nodo para cada rol en una batería de Litio. La Figura 4.2(a) muestra el consumo de energía para un End Node y la Figura 4.2(b) el consumo para un Router Node.



(a) Consumo de energía como End Node.



(b) Consumo de energía como Router Node.

FIGURA 4.2: Medición del consumo de energía de un nodo en modo End Node. Se puede apreciar un consumo uniforme de la energía de la batería por un tiempo aproximado de 2 horas. Por el contrario el nivel de energía de un Router Node cae abruptamente en cierto instante entre la hora y media y 2 horas de funcionamiento

Tanto en la batería de Litio como en la de ácido de plomo se observa un mejor tiempo de vida del Router Node respecto al End Node. Esto debido a que al momento de transmitir información al Coordinator Node, es el Router Node quien debe realizar más transmisiones a comparación del End Node. Estas transmisiones por radiofrecuencia son las que consumen un excedente de energía.

4.2. Optimización objetiva de Router Node

En el apartado anterior se mostraron los resultados experimentales del consumo de energía para los módulos que se encuentran con perfil Router Node y End Node. Bajo este contexto, esta sección justifica que la existencia de tener una menor cantidad de Router Node significa un menor consumo de energía en la red, extiendo, por tanto, su tiempo de actividad. Se propone la Ecuación 4.1.

$$\begin{aligned} E_r(t) &= 100 - K_r * t - N_e \dot{M} \\ E_e(t) &= 100 - K_e * t \end{aligned} \tag{4.1}$$

Donde:

E_r: Nivel de energía del Router Node en un instante t.

E_e: Nivel de energía del nodo End Node en un instante t.

K_r: Velocidad de descarga de un Router Node.

N_e: Número de nodos conectados al Router Node.

K_e: Velocidad de descarga de un End Node.

M : Consumo de energía en un Router Node por cada nodo conectado a él.

Según nuestra configuración de la red en la Figura 3.1, se tienen nodos en el segundo nivel que pueden alternar entre Router Node y End Node. Sin embargo, aún debe determinarse cuántos de estos nodos deben ser Router Node para obtener un menor consumo de energía. Bajo este contexto, se obtienen las posibles combinaciones y su consumo de energía:

- 1 Router Node y 2 End Node: Todos los nodos del tercer nivel (End Node) están conectados al único Router Node en el segundo nivel para que no se pierda la conectividad según la Ecuación 4.2.

$$\begin{aligned} E_{nivel2}(t) &= 100 - K_r * t - 8 * M + 100 - K_e * t + 100 - K_e * t \\ &= 300 - t * (K_r + 2k_e) - 8M \end{aligned} \tag{4.2}$$

- 2 Router Node y 1 End Node: L nodos en el tercer nivel (End Node) están conectados a un Router Node del segundo nivel y $(8 - L)$ nodos están conectados al segundo Router Node en el segundo nivel para no perder conectividad según la ecuación 4.3

$$\begin{aligned} E_{nivel2}(t) &= 100 - K_r * t - L * M + 100 - K_r * t - (8 - L) * M + 100 - K_e * t \\ &= 300 - t * (2K_r + K_e) - 8M \end{aligned} \tag{4.3}$$

- 3 Router Node: Todos los nodos del segundo nivel actúan como Router Node. L_1 nodos conectados al primer Router Node, L_2 nodos conectados al segundo Router Node y L_3 nodos conectados al tercer Router Node del segundo nivel. Entonces la Ecuación del consumo de energía es 4.4.

$$\begin{aligned} E_{nivel2}(t) &= 100 - K_r * t - L_1 * M + 100 - K_r + t - L_2 * M + 100 - K_r * t - L_3 * M \\ &= 300 - t * (3K_r) - (L_1 + L_2 + L_3)M \\ &= 300 - t * (3K_r) - 8M \end{aligned} \tag{4.4}$$

En resumen, para cada posible configuración en el segundo nivel de la topología de red, tendremos el siguiente consumo de energía en un instante de tiempo:

- 1 Router Node + 2 End Node: $= 300 - t * (K_r + 2K_e) - 8M$
- 2 Router Node + 1 End Node: $= 300 - t * (2K_r + K_e) - 8M$
- 3 Router Node + 0 End Node: $= 300 - t * (3K_r) - 8M$

Es posible determinar cuál de estas configuraciones consume menor energía utilizando la Ecuación 4.5, que fue justificada en el apartado anterior.

$$\begin{aligned} &K_r > K_e \\ &K_r + 2K_e \quad < \quad 2K_r + K_e < 3K_r \end{aligned} \tag{4.5}$$

En conclusión, la configuración de 1 Router Node y 2 End Node en el segundo nivel es la que tiene menor consumo de energía según lo demostrado anteriormente.

4.3. Algoritmo de balance de energía

En las secciones anteriores se ha comprobado aspectos esenciales en una red bajo el estándar ZigBee. En resumen, extraemos las siguientes conclusiones:

- Un nodo con el rol de Router Node consume más energía que un nodo con el rol de End Node.
- El mínimo consumo de energía en la red es cuando se tiene únicamente 1 Router Node y 2 End Node en el segundo nivel.

Con la ayuda de la librería Java para XBee de Digi [5] es posible hacer un cambio de configuración en un XBee de manera remota en caso se cumpla cierta condición. Bajo este contexto, se propone un algoritmo para este propósito con el objetivo de mejorar la autonomía y tiempo de vida en la red. El control se hace desde el Coordinator Node y controla todos los nodos con el mismo PAN ID, siempre y cuando estos estén configurados en modo API.

Así mismo, debido a que el Coordinator Node controla todos los nodos en la red, se define un intervalo de tiempo para comprobar si el segundo nivel de la red cumple las condiciones para realizar el cambio de roles. Este procedimiento se describe en el Algoritmo 1.

Haciendo uso de este algoritmo, y tomando en cuenta que los nodos involucrados en este juego de cambio de roles están en el segundo nivel de la red según nuestra topología mostrada en la Figura 3.1, puede concluirse que el tiempo de vida de nuestra red de sensores se puede prolongar notablemente. En caso los nodos de la WSN que se ha desarrollado para este experimento hay que tener en cuenta que se alimentan por energía solar. Por lo que si, por ejemplo, uno de los paneles tiene un fallo y no funciona, entonces podemos cambiar la configuración de los nodos en la WSN de tal manera que la conectividad no se vea afectada y, por ende, mejorando la escalabilidad y calidad de servicio de nuestra red.

Algorithm 1 Game Role Changing Algorithm

```
1:  procedure ROLGAME
2:      periodLoop ← 1
3:      lookupRouter ← FALSE
4:      while TRUE do
5:          secondLevelDevices ← coordinator.buscarDispositivosConectados()
6:          indexDevice ← 0
7:          for indexDevice < secondLevelDevices.size() do
8:              remoteXBee ← secondLevelDevices[indexDevice]
9:              if remoteXBee.getRole() == ROUTER-NODE then
10:                 if remoteXBee.getBatteryLevel() <= 50 then
11:                     remoteXBee.setRole(END − NODE)
12:                     lookupRouter ← TRUE
13:                 end if
14:             else
15:                 if lookupRouter == TRUE then
16:                     if remoteXBee.getBatteryLevel() >= 70 then
17:                         remoteXBee.setRole(ROUTER − NODE)
18:                         lookupRouter ← FALSE
19:                     end if
20:                 end if
21:             end if
22:         end for
23:         sleep(periodLoop)
24:     end while
25: end procedure
```

Finalmente, comentar que este algoritmo no solo pretende incrementar el tiempo de vida de la WSN sino que, además, nunca comprometa la conectividad de esta, ni su calidad de servicio y rendimiento, en términos generales.

Capítulo 5

Conclusiones y Trabajo Futuro

En los capítulos anteriores se hizo descripción y análisis de los diversos parámetros que se involucran al querer optimizar los recursos energéticos en cada nodo de una WSN. Haciendo uso de estos conceptos y los recursos de los sensores XBee de Digi, se puede discutir las conclusiones que se han obtenido de este trabajo; así como establecer el trabajo a futuro.

5.1. Conclusiones

Como resultado del trabajo se puede concluir que la implementación, configuración y despliegue de la red de sensores ha sido exitosa. Así mismo, la aplicación para controlar los distintos roles de los nodos en la red ha sido implementada con la observación de la fragilidad de los dispositivos XBee a nivel hardware. Sin embargo, el control mencionado de los diferentes parámetros permiten mayor flexibilidad para el desarrollo de otras aplicaciones que son descritas en trabajos futuros a fin de mejorar el presente trabajo.

- **Una topología de red y protocolo definido:** Nuestra WSN tiene una topología y protocolo bien definido con el fin de cumplir el objetivo principal que es mejorar la escalabilidad y disminuir el consumo de energía de la red.
- **Elección del modo API para el dispositivo XBee:** Luego de un análisis de las distintas ventajas que da cada modo, se eligió el modo API porque permite un cambio de rol de manera remota.

- **Identificación de parámetros en un nodo:** Se detectaron y modificaron los parámetros que determinan el rol de un nodo, realizando un algoritmo que cambia dichos roles según nivel de batería.
- **Configuración inicial de los dispositivos XBee:** Se configuraron los dispositivos de manera inicial para dar iniciada la red y tener el control desde el Coordinator Node.
- **Desarrollo de la aplicación controlador de roles:** Mediante el uso de la librería de Digi y los parámetros identificados para el rol de un XBee, se desarrolló el algoritmo que controla el rol de los diversos nodos de la red. Esto permite maximizar el tiempo de vida de la red ya que el consumo de energía es menor.

Además de lo anterior, se pudo manejar conceptos de bajo consumo de energía y recursos de los dispositivos analizados, factores cruciales en el despliegue de una red de sensores ya que los recursos son limitados y los lugares donde se instalan pueden tener muchas limitaciones críticas respecto a la energía. Por tanto, este mecanismo desplegado y demostrado en el trabajo puede permitir que se adapten a ambientes más heterogéneos los cuales el consumo de energía sea un punto crítico en la red de sensores.

5.2. Trabajo Futuro

En este punto se mencionan las mejoras que podemos realizar a esta red de sensores. En este sentido, a continuación se describirán algunas aplicaciones útiles y optimización para continuar en la misma línea de investigación.

Implementación de estaciones sensoras

La librería de Digi, nos proporciona la posibilidad de implementar características tales como enviar información a un nodo móvil cuando este último se encuentre en la zona inalámbrica por un tiempo determinado. En esta aplicación, el nodo móvil puede ser un dron, un vehículo móvil o un robot que lleve consigo un dispositivo de radio

XBee. Este enfoque puede permitir crear misiones de recolección de información para el caso de que una red se encuentre aislada y no tenga conexión a Internet para el envío de información dando la posibilidad, finalmente, de continuar leyendo valores los cuales son recolectados por ese ente móvil.

Optimizar recursos mediante comunicación heterogénea

Si bien es cierto, los XBee son dispositivos que tienen un desempeño muy bueno, no son muy baratos y, si la distancia es pequeña entre nodos, no vale la pena comprarlos. En este tipo de escenario convendría utilizar otro tipo de tecnología mucho más barata pero igual de confiable para esa distancia como, por ejemplo, el Bluetooth o la novedosa L6PWAN como LoRa o SigFox. De esta manera definiremos métodos de comunicación dependiendo de la naturaleza de cada nodo.

Procesamiento de eventos complejos

Cuando se hace una red de sensores, lo usual es que se espere a almacenar los datos para poder emitir una alerta. Sin embargo, mediante la técnica de procesamiento de eventos complejos en cada nodo, se puede programar un conjunto de acciones a tomar cuando los valores leídos tienen anomalías o cambios bruscos.

5.3. Producción científica y tecnológica

El desarrollo de este trabajo de tesis ha dado la posibilidad de poder realizar contribuciones científicas sometidas a diferentes revistas y congresos internacionales indexados. Así, a continuación se exponen la producción científica y tecnológica desarrollada con la tesis.

5.3.1. Contribuciones científicas

- Development of a smartphone application for dental biofilm control in adolescents undergoing fixed orthodontic treatment. Behaviour & Information Technology. 2020. (IF: 1.781, Q2 - SJR: 0.64, Q1). *En revisión*

- SAVIA: Smart City Citizen Security Application based on Fog Computing Architecture. IEEE Latin American Transactions. 2019. DOI: 10.1109/TLA.2019.8931206. (IF: 0.804, Q3 – SJR: 0.33, Q2) [20].

- FROG: A Robust and Green Wireless Sensor Node for Fog Computing Platforms. Journal of Sensors. Vol. 2018, Article ID 3406858, 12 pages. 2018. [2] DOI:10.1155/2018/3406858. (IF: 2.057, Q2 – SJR: 0.37, Q2) [2].

- Ray: Smart indoor/outdoor routes for the blind using Bluetooth 4.0 BLE, Procedia Computer Science, Volume 83, 2016, Pages 690 – 694. Ed. Elsevier. DOI: 10.1016/j.prcocs.2016.04.153. (SJR: 0.34) [21].

5.3.2. Patente

- Dispositivo electrónico que se fija en el cuerpo para seguimiento y localización de personas o animales; con sensores de posicionamiento y rastreo, transmisión de datos y optimización de energía e información. Expediente: 002758-2016/DIN. Resolución: 000417-2019/DIN-INDECOPI. Título N.º: 1151. Perú. 2016. (Otorgada).

Bibliografía

[1] W. Kurschl and W. Beer, “Combining cloud computing and wireless sensor networks,” in *Proceedings of the 11th International Conference on Information Integration and Web-based Applications & Services*, pp. 512–518, 2009.

[2] M. Castillo-Cara, E. Huaranga-Junco, M. Quispe-Montesinos, L. Orozco-Barbosa, and E. A. Antúnez, “Frog: a robust and green wireless sensor node for fog computing platforms,” *Journal of Sensors*, vol. 2018.

[3] H.-J. Cha, H.-K. Yang, and Y.-J. Song, “A study on the design of fog computing architecture using sensor networks,” *Sensors*, vol. 18, no. 11, p. 3633, 2018.

[4] N. Instruments, “What is a wireless sensor network.” http://www.ni.com/white-paper/7142/en/, 2016. [Online; accessed 10-06-2017].

[5] D. I. Inc., “Introducing the official xbee java library.” Link:https://www.digi.com/blog/community/official-xbee-java-library/. [Online; accessed 25-June-2017].

[6] Y. F. Solahuddin and W. Ismail, “Data fusion for reducing power consumption in arduino-xbee wireless sensor network platform,” in *2014 International Conference on Computer and Information Sciences (ICCOINS)*, pp. 1–6, June 2014.

[7] A. González, R. Aquino, W. Mata, A. Ochoa, P. Saldaña, and A. Edwards, “Open-wise: A solar powered wireless sensor network platform,” *Sensors*, vol. 12, no. 6, pp. 8204–8217, 2012.

[8] T. Matsui and H. Nishi, “Ecors: Energy consumption-oriented route selection for wireless sensor network,” in *2016 IEEE 14th International Conference on Industrial Informatics (INDIN)*, pp. 1044–1049, July 2016.

[9] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, MCC '12, (New York, NY, USA), pp. 13–16, ACM, 2012.

[10] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: A survey on enabling technologies, protocols, and applications," *IEEE Communications Surveys Tutorials*, vol. 17, pp. 2347–2376, Fourthquarter 2015.

[11] M. B. A. P. Madumal, D. A. S. Atukorale, and T. M. H. A. Usoof, "Adaptive event tree-based hybrid cep computational model for fog computing architecture," in *2016 Sixteenth International Conference on Advances in ICT for Emerging Regions (ICTer)*, pp. 5–12, Sept 2016.

[12] A. G. de Prado, G. Ortiz, and J. Boubeta-Puig, "Collect: {COLLaborativE} context-aware service oriented architecture for intelligent decision-making in the internet of things," *Expert Systems with Applications*, vol. 85, pp. 231 – 248, 2017.

[13] M. Zennaro, *Radio Laboratory Handbook*. Radio Communication Unit, 2014.

[14] L. J. García Villalba, A. L. Sandoval Orozco, A. Triviño Cabrera, and C. J. Barenco Abbas, "Routing protocols in wireless sensor networks," *Sensors*, vol. 9, no. 11, pp. 8399–8421, 2009.

[15] J. N. Al-Karaki, R. Ul-Mustafa, and A. E. Kamal, "Data aggregation in wireless sensor networks - exact and approximate algorithms," in *2004 Workshop on High Performance Switching and Routing, 2004. HPSR.*, pp. 241–245, 2004.

[16] C. I. of Chicago, "Array of Things." https://arrayofthings.github.io, 2016. [Online; accessed 19-06-2017].

[17] Y. Wu, S. Fahmy, and N. B. Shroff, "Energy efficient sleep/wake scheduling for multi-hop sensor networks: Non-convexity and approximation algorithm," in *IEEE INFOCOM 2007 - 26th IEEE International Conference on Computer Communications*, pp. 1568–1576, May 2007.

[18] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, “A survey on sensor networks,” *IEEE Communications Magazine*, vol. 40, pp. 102–114, Aug 2002.

[19] D. I. Inc., “Xctu software.” Link:https://www.digi.com/support/ productdetail?pid=3352. [Online; accessed 25-June-2017].

[20] M. Castillo-Cara, G. Mondragón-Ruíz, E. Huaranga-Junco, E. A. Antúnez, and L. Orozco-Barbosa, “Savia: Smart city citizen security application based on fog computing architecture,” *IEEE Latin America Transactions*, vol. 17, no. 07, pp. 1171–1179, 2019.

[21] M. Castillo-Cara, E. Huaranga-Junco, G. Mondragón-Ruiz, A. Salazar, L. O. Barbosa, and E. A. Antúnez, “Ray: Smart indoor/outdoor routes for the blind using bluetooth 4.0 ble,” *Procedia Computer Science*, vol. 83, pp. 690–694, 2016.